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

JUSCELINO SEBASTIÃO AVELINO JÚNIOR

Uma abordagem de seleção dinâmica de classificadores para predição de defeitos de software

Recife

2022

JUSCELINO SEBASTIÃO AVELINO JÚNIOR

Uma abordagem de seleção dinâmica de classificadores para predição de defeitos de software

Trabalho apresentado à pós-graduação em ciência da computação do Centro de Informática da Universidade Federal de Pernambuco como requisito para obtenção do grau de mestre em ciência da computação.

Área de Concentração: Inteligência Computacional

Orientador: Prof. Dr. George Darmiton da Cunha Cavalcanti

Recife

2022

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

A949a Avelino Júnior, Juscelino Sebastião
Uma abordagem de seleção dinâmica de classificadores para predição de defeitos de software / Juscelino Sebastião Avelino Júnior. – 2022.
123 f.: il., fig., tab.

Orientador: George Darmiton da Cunha Cavalcanti.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.
Inclui referências e apêndices.

1. Inteligência computacional. 2. Predição de defeitos de software. 3. Predição de defeitos cruzada entre projetos. 4. Seleção dinâmica de classificadores. I. Cavalcanti, George Darmiton da Cunha (orientador). II. Título

006.31 CDD (23. ed.) UFPE - CCEN 2022 – 84

Juscelino Sebastião Avelino Júnior

“Uma abordagem de seleção dinâmica de classificadores para predição de defeitos de software”

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: 16/03/2022.

BANCA EXAMINADORA

_____

Prof. Dr. George Darmiton da Cunha Cavalcanti
Centro de Informática / UFPE
(**Orientador**)

_____

Prof. Dr. Rafael Menelau Oliveira e Cruz
Département de génie logiciel et des TI
École de Technologie Supérieure

_____

Prof. Dr. Araken de Medeiros Santos
Departamento de Ciências Exatas e
Tecnologia da Informação / UFERSA

RESUMO

A predição de defeitos é uma tarefa que visa alocar recursos e informações para predizer módulos de software propensos a defeitos. Contudo, devido a necessidade de quantidades suficientemente grandes de dados requeridas pelos modelos, pesquisadores tem se concentrado em pesquisas sobre *Cross-Project Defect Prediction* (CPDP). Essa abordagem envolve construir modelos usando um conjunto de treinamento composto por informações de diversos projetos externos. Diversas abordagens CPDP propostas na literatura utilizam abordagens tradicionais como, por exemplo, normalização/transformação dos dados através do logaritmo ou uso de algoritmos de aprendizagem de máquina. Entretanto, essas abordagens não possuem nenhum mecanismo capaz de selecionar um classificador ou um conjunto de classificadores mais apto em predizer uma determinada amostra de teste. Logo, a seleção dinâmica de classificadores é uma abordagem ao qual seleciona classificadores básicos em tempo real de acordo com cada amostra de teste a ser classificada. Neste contexto e considerando as limitações das abordagens CPDP tradicionais, propomos uma abordagem CPDP que, com base nos dados de treinamento, seleciona a melhor configuração de parâmetros (técnica de seleção dinâmica de classificadores $\times$ classificador básico $\times$ tamanho do *pool* de classificadores) para classificar as novas amostras de entrada (dados de teste). A abordagem proposta é composta por três etapas: Definição do Alvo, Superprodução e Avaliação do Modelo. Portanto, nessa dissertação são almejados quatro principais pontos. Primeiro, é conduzida uma análise experimental para investigar o comportamento da abordagem proposta na predição de defeitos de software. Segundo, são realizadas comparações entre a abordagem proposta versus abordagens CPDP da literatura. Neste contexto, foram investigados quais métodos apresentam melhor desempenho para os mesmos conjuntos de dados ou projetos de software. Terceiro, para verificar a precisão de classificação dos métodos CPDP, foi analisada a qualidade da performance dos métodos em relação à algumas escalas da área sob a curva ROC (ROC-AUC). Quarto, foi realizada uma análise experimental para verificar quando utilizar a abordagem proposta. Neste ponto, utilizamos um meta-classificador (árvore de decisão) que, através de regras de decisão, define quais características dos dados a abordagem proposta deve ser aplicada e, consequentemente, apresentar maior desempenho.

Palavras-chaves: predição de defeitos de software; predição de defeitos cruzada entre projetos; seleção dinâmica de classificadores.

ABSTRACT

Defect prediction is a task that aims to allocate resources and information to predict defect-prone software modules. However, due to the need for sufficiently large amounts of data required by the models, researchers have focused on research on *Cross-Project Defect Prediction* (CPDP). This approach involves building models using a training set composed of information from multiple external projects. Several CPDP approaches proposed in the literature use traditional approaches such as log normalization/transformation of data or use of machine learning algorithms, however, these approaches have no mechanism to select a classifier or a set of classifiers that are best able to predict a given test sample. Therefore, dynamic classifier selection is an approach that selects basic classifiers in real time according to each test sample to be classified. In this context and considering the limitations of traditional CPDP approaches, we propose a CPDP approach that, based on training data, selects the best parameter configuration (dynamic classifier selection technique $\times$ basic classifier $\times$ size of the *pool* of classifiers) to classify the new input samples (test data). The proposed approach is composed of three steps: Target Definition, Overproduction and Model Evaluation. Therefore, four main points are targeted in this dissertation. First, an experimental analysis is conducted to investigate the behavior of the proposed approach in predicting software defects. Second, comparisons are conducted between the proposed approach versus CPDP approaches from the literature. In this context, it was investigated which methods perform better for the same datasets or software designs. Third, to verify the classification accuracy of CPDP methods, the quality of the methods' performance against some scales of the area under the ROC curve (ROC-AUC) was analyzed. Fourth, an experimental analysis was performed to verify when to use the proposed approach. Here, we used a meta-classifier (decision tree) that, through decision rules, defines which characteristics of the data the proposed approach should apply and, consequently, present higher performance.

Keywords: software defect prediction; cross-project defect prediction; dynamic classifiers selection.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A confiabilidade do software é uma característica importante para garantir a qualidade dos produtos de software. Defeitos, falhas ou erros em software são problemas de programação que podem acontecer devido a descuidos no código-fonte ou nos requisitos, causando resultados inadequados e afetando negativamente a qualidade e confiabilidade do software (RAWAT; DUBEY, 2012). Portanto, *Software Defect Prediction* (SDP – Predição de Defeitos de Software) é um tópico de pesquisa de engenharia de software que visa auxiliar a alocação de dados/recursos de software e predizer, com antecedência, módulos de *softwares* propensos a defeitos.

A comunidade de pesquisa tem se dedicado em propor abordagens que, a partir de dados armazenados em repositórios de software (por exemplo, sistemas de detecção de defeitos (KIM; WHITEHEAD; ZHANG, 2008), constroem modelos para predição de defeitos capazes de predizer a propensão de software a defeitos (MALHOTRA, 2015). Logo, um grande número de métodos de predição de defeitos foram propostos que usam, por exemplo, algoritmos de aprendizado de máquina para prever módulos de software com ou sem defeitos, ou fornecer a probabilidade de um módulo apresentar falha futuramente. De forma que, uma instância ou módulo de software pode ser descrita(o) como um registro codificado de um componente de software ou parte de um programa que contém uma ou mais rotinas. Especificamente, Ghotra, McIntosh e Hassan (2015), Panichella, Oliveto e Lucia (2014) e Liu, Khoshgoftaar e Seliya (2010) em seus estudos realizaram experimentos usando diferentes classificadores de aprendizado de máquina e, com base nos resultados, relatam e sugerem o uso de uma determinada abordagem de aprendizado de máquina para construir modelos de predição a partir da combinação de vários classificadores — *ensemble learning* ou *ensemble techniques* (ZIMMERMANN; NAGAPPAN, 2009) — e sua aplicação à predição de defeitos (BASILI; BRIAND; MELO, 1996; BEZERRA; OLIVEIRA; MEIRA, 2007; OSTRAND; WEYUKER; BELL, 2005). No entanto, conforme destacado por Bowes, Hall e Petrić (2018), as abordagens tradicionais de *ensemble* erram predições de uma grande parte dos defeitos e, portanto, "as estratégias de conjunto de classificadores precisam ser aprimoradas para dar conta do sucesso de classificadores individuais em encontrar conjuntos específicos de defeitos". Diante disso, *Multiple Classifier Systems* (MCS — Sistemas de Múltiplos Classificadores) refere-se ao campo de aprendizado de máquina que usa vários modelos para resolver diversos problemas preditivos. Logo, diversas abordagens são utilizadas

para construir um MCS. Uma das abordagens mais promissoras é a *Dynamic Selection* (DS – Seleção Dinâmica), na qual os classificadores básicos são selecionados em tempo real, de acordo com cada nova amostra a ser classificada (CRUZ; SABOURIN; CAVALCANTI, 2018). Por exemplo, Rathore e Kumar (2021) realizaram um estudo usando a abordagem de seleção dinâmica em SPD.

Contudo, um dos principais pontos e desafios da predição de defeitos está relacionado a disponibilidade dos dados de treinamento, onde, de acordo com o estudo de Hall et al. (2011), o modelo utilizado para predizer os dados de teste requer uma quantidade suficientemente grande de dados disponível. Nesta perspectiva, pesquisadores tem se concentrado em pesquisas sobre *Cross-Project Defect Prediction* (CPDP – Predição de Defeitos Cruzada entre Projetos). Este tipo de pesquisa envolve construir um modelo usando dados de treinamento de outros projetos (projetos de origem) para predizer instâncias (módulos de softwares) defeituosas em um projeto específico (projeto alvo). Essa abordagem é útil quando a quantidade de dados históricos de defeitos é inapropriada ou insuficiente para compor o conjunto de treinamento. Em um estudo, Herbold, Trautsch e Grabowski (2017a) realizaram um mapeamento de 50 publicações que abordam, especificamente, aspectos de métodos CPDP. Alguns dos diferentes métodos CPDP comparados utilizam abordagens tradicionais de normalização/transformação de dados e, portanto, não implementam nenhuma abordagem para selecionar um classificador ou *ensemble* mais apto a predizer um determinado projeto.

Portanto, considerado essas limitações, nesse trabalho é proposto o método supervisionado para Predição de Defeitos Cruzada entre Projetos denominado Dynamic Selection Surpevised Cross-project (DSSC). O DSSC, com base nos dados de treinamento, seleciona a melhor configuração de parâmetros — técnica de seleção dinâmica, classificadores base e a quantidade de classificadores no *pool* (*ensemble*) — para classificar as novas amostras de entrada (dados de teste). A principal diferença em relação aos demais métodos CPDP, é que o DSSC utiliza diferentes parâmetros para definir o melhor modelo realizar a predição das nova entradas. Sendo assim, este trabalho tem como contribuições: análise sobre o comportamento de uma estrutura com métodos de seleção dinâmica na predição de defeitos e, principalmente, por este ser o primeiro estudo a propôr um método supervisionado para o cenário CPDP que tem como base técnicas de seleção dinâmica. A investigação das condições que a abordagem DSSC tem resultados melhores que outras abordagens para CPDP.

1.1 OBJETIVOS

O objetivo desse trabalho é propor uma abordagem para predição de defeitos entre projetos cruzados que tem como base técnicas de seleção dinâmica. A principal hipótese é que a seleção de um modelo preditivo com base nos dados de treinamento resulta em um melhor desempenho na predição de defeitos, visto que cenário CPDP requer grandes quantidades de informação, logo, o conjunto de treinamento contém informações dos mais diversos tipos de módulos de software para o aprendizado o modelo preditivo. Portanto, além do objetivo central, este trabalho busca responder os seguintes pontos levantados:

1. CPDP: comportamento da estrutura proposta (DSSC).
2. CPDP: abordagem proposta (DSSC) versus abordagens do estado da arte.
3. Qualidade dos métodos CPDP.
4. Quando usar o método DSSC.

1.2 ESTRUTURA

O restante deste trabalho está organizado da seguinte forma: No Capítulo 2 é apresentada a discussão sobre trabalhos relacionados e alguns aspectos sobre a seleção dinâmica na predição de defeitos. No Capítulo 3 é apresenta metodologia do método proposto. No Capítulo 4 são apresentados os protocolos experimentais. A seguir, no Capítulo 5, são discutidos os resultados, incluindo os pontos a serem respondidos por meio de comparações e análises estatísticas. Finalmente, concluímos o trabalho no Capítulo 6.

2 PROBLEMÁTICA E TRABALHOS RELACIONADOS

Este Capítulo apresentada aspectos gerais sobre a predição de defeitos de software, mais especificamente sobre o cenário CPDP, uma revisão de literatura sobre métodos CPDP, abordagens de predição de defeitos de modo geral, além da relação entre métodos de seleção dinâmica e predição de defeitos.

2.1 ABORDAGENS DE CROSS-PROJECT DEFECT PREDICTION (CPDP)

De acordo com Herbold, Trautsch e Grabowski (2018), a previsão de defeitos tem sido um tópico de pesquisa frequentemente abordado como uma forma de focar na garantia de qualidade do software, com primeiros trabalhos sendo realizados por Basili, Briand e Melo (1996).

De forma geral, o sucesso de um modelo de predição de defeitos de software requer dois fatores principais: (1) um conjunto de treinamento adequado e (2) aplicar um método de aprendizado de máquina adequado. Logo, em uma tarefa de classificação, o conjunto de treinamento pode ser representado por uma tabela de elementos (consultar Tabela 1). Nesta tabela, cada linha representa um elemento (parte, instância ou módulo de um software); os atributos (variáveis independentes) representam as características de cada exemplo; e a variável dependente (classe ou atributo de destino) representa a classe binária: defeituosa ou não defeituosa.

Tabela 1 – Exemplo de dados de treinamento para classificação.

Exemplos	Atributos				Classes
	A_1	A_2	...	A_m	
E_1	x_{11}	x_{12}	...	x_{1m}	c_1
E_2	x_{21}	x_{22}	...	x_{2m}	c_2
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
E_n	x_{n1}	x_{n2}	...	x_{nm}	c_n

Fonte: Elaborada pelo autor (2022)

Ainda, no contexto da predição de defeitos de software, é importante notar que, o aprendizado só é possível quando os dados utilizado para a predição de defeitos estão disponíveis.

Entretanto, de acordo com Kitchenham, Mendes e Travassos (2007), na prática, nem todas as empresas de software disponibilizam dados históricos de defeitos ou dados suficientes de projetos anteriores. Neste caso, o conjunto de treinamento pode ser composto por diversos projetos externos com informações de defeitos. Logo, a predição de defeitos cruzada entre projetos, ou seja, a predição de defeitos usando dados de outros projetos como treinamento, passou de um aspecto ignorado a um subtópico importante de previsão de defeitos de software em geral (HERBOLD; TRAUTSCH; GRABOWSKI, 2018).

Consequentemente, foram propostos diversos métodos supervisionados para melhorar o desempenho do cenário CPDP (HERBOLD; TRAUTSCH; GRABOWSKI, 2018; ZHOU et al., 2018). Trabalhos como (WATANABE; KAIYA; KAIJIRI, 2008; CRUZ; OCHIMIZU, 2009; TURHAN et al., 2009; MENZIES et al., 2011; MA et al., 2012; PETERS; MENZIES; MARCUS, 2013; XIA et al., 2016), se concentram no CPDP homogêneo, onde pressupõe que os projetos de origem e de destino tenham os mesmos conjuntos de recursos. Em contrapartida, trabalhos como (JING et al., 2015; NAM et al., 2017; LI et al., 2017; LI et al., 2018; Li et al., 2019), investigaram o CPDP heterogêneo, ao qual deduz que os projetos de origem e de destino tenham diferentes conjuntos de recursos. Boa parte dos métodos supervisionados constroem os modelos usando dados rotulados. Além disso, pesquisadores propuseram diversos métodos CPDP cada um deles com diferentes configurações experimentais, sejam por exemplo, métodos supervisionados, não supervisionados ou semi-supervisionados. A Tabela 2 lista um conjunto de métodos CPDP de trabalhos relacionados encontrados na literatura.

Tabela 2 – Métodos CPDP de trabalhos relacionados.

Abordagem CPDP	**Descrição**
Nam e Kim (2015)	Os autores propõem um método não supervisionado denominado CLAMI que consiste em dois processos: Clustering (C) e LAbeling (LA) para rotular os dados de treinamento para contar quantos valores de um atributo específico dos dados estão acima da mediana daquele atributo e a seleção de Métricas e Instâncias (MI) para selecionar um subconjunto de atributos e instâncias (módulos de software) que é consistente.

Continua na próxima página

Tabela 2 – Métodos CPDP de trabalhos relacionados. – continuação da página anterior

Abordagem CPDP	**Descrição**
Nam, Pan e Kim (2013)	Abordagem não supervisionada que realiza uma combinação entre abordagens de padronização de dados e a Análise de Componentes de Transferência (TCA), denominando de TCA+. Inicialmente, um algoritmo de padronização é selecionado, entre: nenhuma, normalização min-max, padronização Z-score, padronização Z-score onde todos os produtos (projetos) de software são padronizados com base nos dados de treinamento e padronização Z-score onde todos os produtos são padronizados com base nos dados de teste (destino). Em seguida, o TCA é aplicado para definir um mapeamento do treinamento e do teste.
Zhou et al. (2018)	Propuseram dois métodos não supervisionados: *ManualDown* e *ManualUp*. O primeiro considera que módulos de software maiores, isto é, módulos com maior número de linhas de código possuem maior propensão a defeitos, logo devem ser avaliados primeiro. Entretanto, o *ManualUp* considera um módulo menor como mais propenso, pois alguns estudos argumentam que um módulo menor é mais propenso a defeitos e, portanto, deve ser inspecionado primeiro (KORU et al., 2008a; KORU et al., 2008b; KORU et al., 2010).
Limsettho et al. (2018)	Propuseram a abordagem supervisionada *Class Distribution Estimation with Synthetic Minority Oversampling Technique* (CDE-SMOTE). Este método emprega a CDE para estimar a distribuição de classe do projeto-alvo e o SMOTE é utilizado para modificar a distribuição das classes dos dados de treinamento até que a distribuição se torne o contrário da distribuição de classe aproximada do projeto-alvo.

Continua na próxima página

Tabela 2 – Métodos CPDP de trabalhos relacionados. – continuação da página anterior

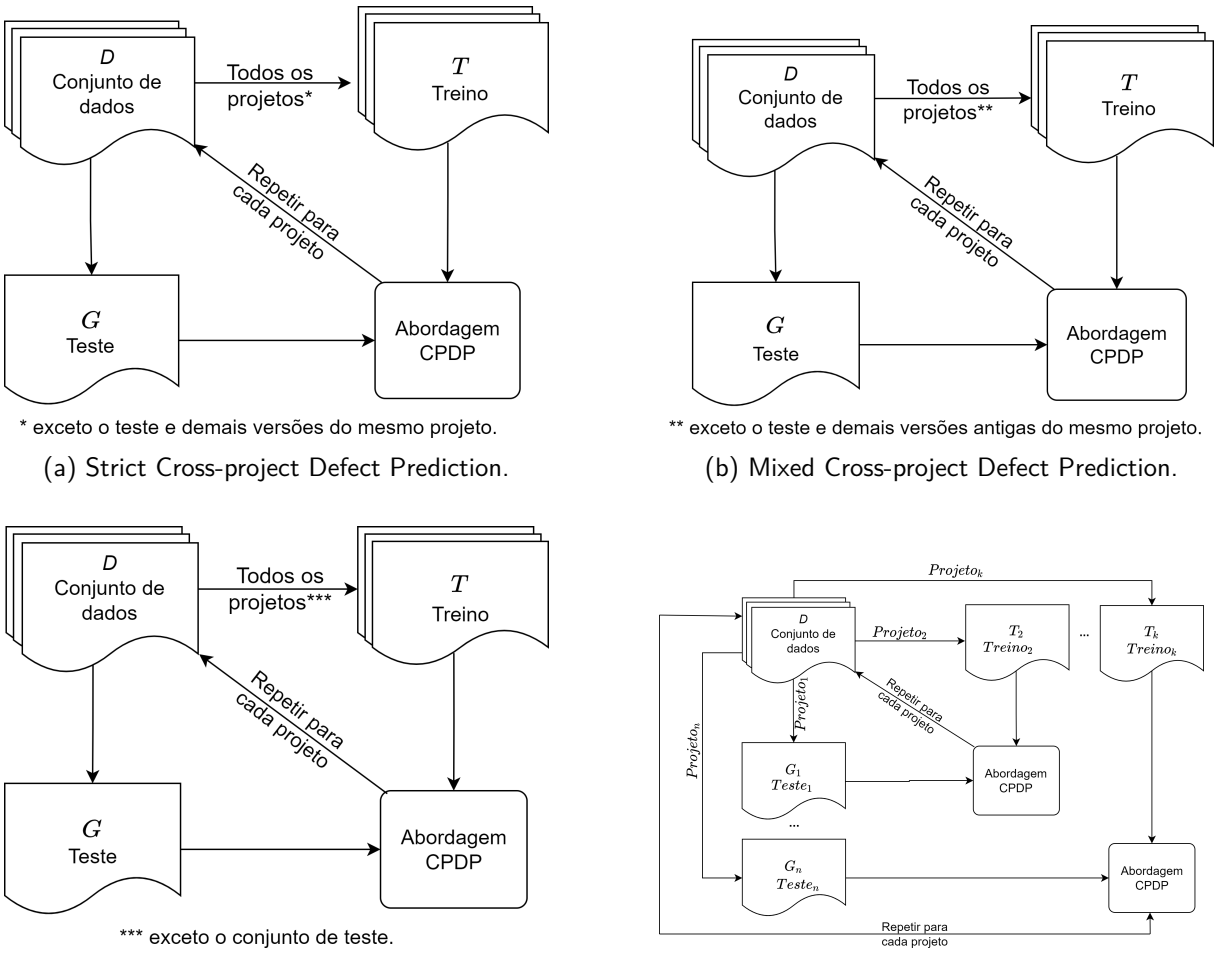
Abordagem CPDP	**Descrição**
Liu et al. (2019)	Propuseram um modelo supervisionado de aprendizagem de transferência em duas fases (TPTL – *two-phase transfer learning*). Este método utiliza um estimador de projeto fonte (SPE – *Source Project Estimator*) para selecionar automaticamente dois projetos de origem (dados de treinamento) com maior semelhança com um dos projetos-alvo candidatos. Tais projetos de origem são estimados para alcançar maiores valores de pontuação F1-score.
Sun et al. (2020a)	Propuseram uma abordagem semi-supervisionada de Aprendizagem de Recursos Contraditórios Discriminativos (DAFL – *Discriminative Adversarial Feature Learning*). Esta abordagem consiste em transformar os recursos para tentar gerar representação de recursos, aprender as informações discriminantes e preservar a estrutura intrínseca inferida a partir dos dados rotulados e não rotulados.
Sun et al. (2020b)	Para minimizar as diferenças de distribuição de dados entre diferentes projetos e prever instâncias-alvo não rotuladas (módulos do conjunto de teste), propuseram uma abordagem semi-supervisionada denominada Aprendizado Subespacial baseado em Pseudo-rotulagem Seletiva (SPSL – *Selective Pseudo-labeling based Subspace Learning*). Este método treina um subespaço comum usando instâncias de origem rotuladas (instâncias de treinamento) e instâncias de destino pseudo-rotuladas (instâncias de teste). A precisão da pseudo-rotulagem é promovida pela estratégia de pseudo-rotulagem seletiva iterativa, de forma que as instâncias pseudo-rotuladas do projeto-alvo são iterativamente atualizadas selecionando as instâncias com alta confiança de duas tecnologias de pseudo-rotulagem.

Fonte: Elaborada pelo autor (2022)

Em um estudo, Herbold (2017) fez um mapeamento de 50 publicações que abordam especificamente aspectivos sobre abordagens CPDP. De acordo com Herbold, Trautsch e Gra-

bowski (2018), há pelo menos quatro variantes de estudos de CPDP: *strict* CPDP (HERBOLD; TRAUTSCH; GRABOWSKI, 2017b), *mixed* CPDP, *Mixed-Project Defect Prediction* (MPDP) e *pair-wise* CPDP. A Figura 1 apresenta os fluxos de trabalho dos cenários CPDP.

Figura 1 – Diferentes tipos e fluxo de trabalho geral de métodos CPDP.



(a) Strict Cross-project Defect Prediction.

(b) Mixed Cross-project Defect Prediction.

(c) Mixed-project Defect Prediction.

(d) Pair-wise Cross-project Defect Prediction.

Fonte: Adaptado de Herbold, Trautsch e Grabowski (2018).

Tabela 3 – Exemplo de conjunto de dados com 3 projetos.

Projeto	No.	Instâncias
A-1.0	①	50
A-2.0	②	180
B	③	500

Notas: No.: Número do projeto

Fonte: Elaborada pelo autor (2022)

Conforme apresentado na Tabela 3, cada projeto possui um número específico de instâncias. De forma que, uma instância ou módulo de software pode ser definido como uma unidade de

software, um método, dados sobre um arquivo ou um registro de uma mudança de código, por exemplo: registro sobre a interface de usuário, sobre biblioteca de comunicação, etc. Logo, um módulo pode conter, por exemplo, informações de quantas linhas de código um determinado método possui, quantas linhas em branco ou se um dado método apresentou defeito ou não.

Considerando essas informações e o exemplo de conjunto de dados (D) apresentado na Tabela 3, os diferentes cenários CPDP atribuem diferentes projetos para os conjuntos de treinamento e de teste. Além disso, em cada cenário todos os projetos do conjunto de dados é definido como projeto-alvo (conjunto de teste) em sequência, mudando apenas quais projetos são atribuídos aos dados de treinamento.

- ***strict* CPDP**: um dos produtos no conjunto de dados D é atribuído como produto alvo $\mathcal{G}$ (conjunto de teste) e os demais projetos são usados como conjunto de treinamento $\mathcal{T}$. Entretanto, caso exista outras versões de $\mathcal{G}$, elas serão removidas de $\mathcal{T}$, de forma que nenhuma informação do mesmo contexto do projeto permaneça nos dados de treinamento. Cada versão em D é considerada como projeto de destino, em ordem.

Tabela 4 – Exemplo de iterações do cenário *strict* CPDP com os projetos da Tabela 3, onde cada produto é definido em ordem como projeto alvo (conjunto de teste, $\mathcal{G}$) e quais projetos são selecionados e desconsiderados (▲) para compôr o conjunto de treinamento ($\mathcal{T}$). A coluna *F1-score* representa a pontuação obtida após cada interação e a média dos resultados (μ).

Iterações	$\mathcal{G}$	$\mathcal{T}$	▲	*F1-score*
1ª	①	③	②	0,29
2ª	②	③	①	0,44
3ª	③	① ②	-	0,51
-	-	-	-	μ **= 0,41**

Fonte: Elaborada pelo autor (2022)

- ***mixed* CPDP**: somente as versões antigas de $\mathcal{G}$, além dos demais projetos diferente de $\mathcal{G}$ e suas possíveis versões, serão incluídas no conjunto de treinamento $\mathcal{T}$.

- ***Mixed-Project Defect Prediction (MPDP)***: quaisquer versões do projeto alvo são permitidas no treinamento.

Tabela 5 – Exemplo de iterações do cenário *mixed* CPDP com os projetos da Tabela 3, onde cada produto é definido em ordem como projeto alvo (conjunto de teste, $\mathcal{G}$) e quais projetos são selecionados e desconsiderados (▲) para compôr o conjunto de treinamento ($\mathcal{T}$). A coluna *F1-score* representa a pontuação obtida após cada interação e a média dos resultados (μ).

Iterações	$\mathcal{G}$	$\mathcal{T}$	▲	*F1-score*
1ª	①	③	②	0,29
2ª	②	① ③	-	0,73
3ª	③	① ②	-	0,51
-	-	-	-	μ **= 0,51**

Fonte: Elaborada pelo autor (2022)

Tabela 6 – Exemplo de iterações do cenário *Mixed-Project Defect Prediction* com os projetos da Tabela 3, onde cada produto é definido em ordem como projeto alvo (conjunto de teste, $\mathcal{G}$) e quais projetos são selecionados e desconsiderados (▲) para compôr o conjunto de treinamento ($\mathcal{T}$). A coluna *F1-score* representa a pontuação obtida após cada interação e a média dos resultados (μ).

Iterações	$\mathcal{G}$	$\mathcal{T}$	▲	*F1-score*
1ª	①	② ③	-	0,60
2ª	②	① ③	-	0,73
3ª	③	① ②	-	0,51
-	-	-	-	μ **= 0,61**

Fonte: Elaborada pelo autor (2022)

- ***pair-wise* CPDP**: os n projetos do conjunto de dados D são preditos em pares.

Tabela 7 – Exemplo de iterações do cenário *pair-wise* com os projetos da Tabela 3, onde cada produto é definido em ordem como projeto alvo (conjunto de teste, $\mathcal{G}$) e quais projetos são selecionados e desconsiderados (▲) para compôr o conjunto de treinamento ($\mathcal{T}$). A coluna *F1-score* representa a pontuação obtida após cada interação e a média dos resultados (μ).

Iterações	$\mathcal{G}$	$\mathcal{T}$	▲	*F1-score*
1ª	①	②	③	0,31
2ª	①	③	②	0,29
3ª	②	①	③	0,58
4ª	②	③	①	0,49
5ª	③	①	②	0,44
6ª	③	②	①	0,36
-	-	-	-	μ **= 0,41**

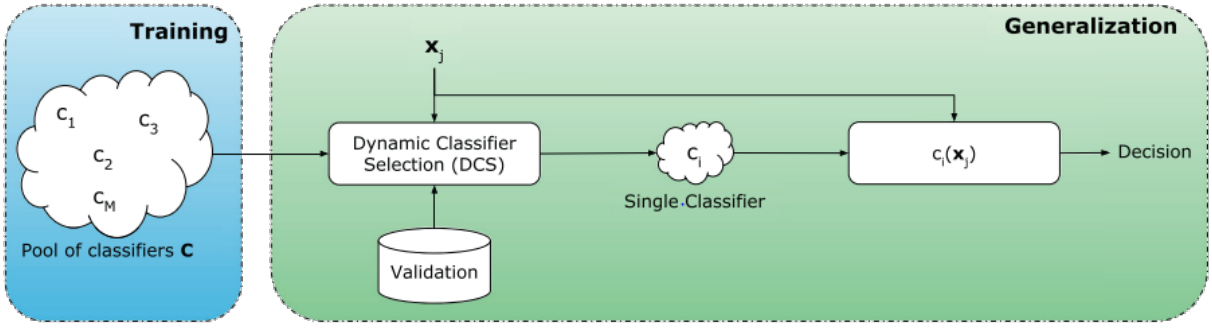
Fonte: Elaborada pelo autor (2022)

Conforme ilustrado nas Tabelas 4, 5, 6 e 7, cada produto no conjunto de dados é definido como projeto alvo ($\mathcal{G}$) e os demais são atribuídos aos dados de treinamento ($\mathcal{T}$), seguindo o critério de cada cenário CPDP. Durante cada interação, é obtida uma pontuação (*F1-score*,

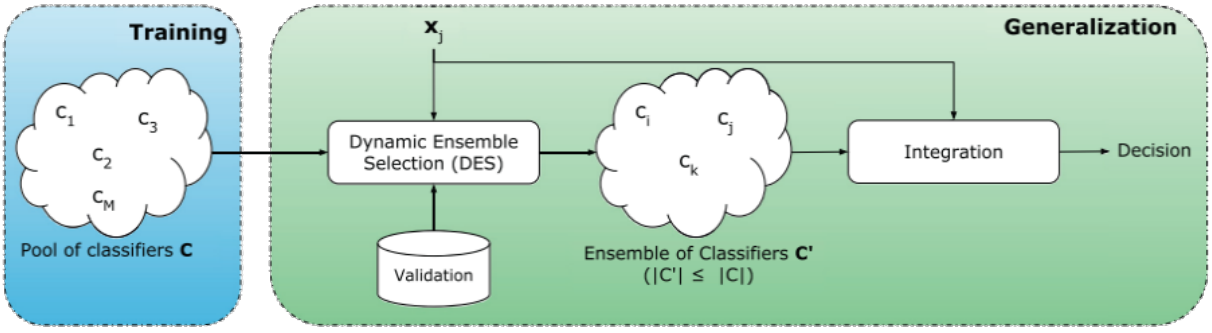
no exemplo) ao classificar um projeto p_i e, ao final, obtém-se a média geral (μ) do conjunto de dados.

Além dos diferentes métodos CPDP possuírem um fluxo de trabalho diferente, por exemplo o cenário *strict* CPDP ou *mixed* CPDP, os métodos CPDP também envolvem abordagens diversas, seja o processamento dos dados, uso de classificadores de aprendizado de máquina ou abordagens de *ensemble*. Entretanto, essas abordagens tradicionais por muitas vezes erram grande parte das predições, logo, requerendo abordagens aprimoradas para melhorar a precisão das predições (GHOTRA; MCINTOSH; HASSAN, 2015; PANICHELLA; OLIVETO; LUCIA, 2014; LIU; KHOSHGOFTAAR; SELIYA, 2010; BOWES; HALL; PETRIĆ, 2018).

Figura 2 – Diferença entre seleção dinâmica de classificadores (DCS) e seleção dinâmica de *ensemble* (DES).



(a) Seleção Dinâmica de Classificadores.



(b) Seleção Dinâmica de *Ensemble*.

Fonte: Adaptado de Cruz, Sabourin e Cavalcanti (2018)

Logo, além dessas abordagens tradicionais, os sistemas de múltiplos classificadores, mais especificamente a seleção dinâmica, escolhe os melhores classificadores (seleção dinâmica de classificadores) ou um conjunto de classificadores (seleção dinâmica de *ensemble*) durante a avaliação de um exemplo x_j de teste. Um aspecto vital da seleção dinâmica é o uso da regra de combinação se mais de um modelo for selecionado na região de competência; as regras de combinação usam a média ponderada dos modelos selecionados. Além disso, regras de combinação de ensemble, como o método de ensemble básico (BEM – *Basic Ensemble*

Method) e os métodos generalizados de ensemble, foram propostos por Sagi e Rokach (2018). Outra abordagem de combinação como *Bagging* (BREIMAN, 1996; BÜHLMANN, 2012) também foi proposta para a seleção dinâmica. As Figuras 2a e 2b apresentam os conceitos de seleção dinâmica de classificadores e seleção dinâmica de ensemble, respectivamente.

Neste contexto, Rathore e Kumar (2021) realizaram um estudo sobre a utilização da abordagem de seleção dinâmica para a predição de defeitos de software; o método do estudo consiste em predizer diversos produtos de software em um conjunto de dados específico separadamente, isto é, não segue nenhum cenário CPDP (ver Figura 1). Logo, destacando como principal limitação a quantidade de dados de treinamento utilizada, onde, de acordo com Hall et al. (2011), os modelos preditivos requerem que uma quantidade suficientemente grande de dados estejam disponíveis durante o treinamento do modelo para realizar a predição dos dados de teste.

Por fim, conforme apontado pelos estudos relatados, a seleção dinâmica tem o potencial de gerar melhores resultados em comparação às abordagens de *ensemble*, como também resultados superiores às abordagens que utilizam um único classificador base para realizar a predição. Diante disso, temos como motivação realizar um estudo de seleção dinâmica para a predição de defeitos, mais especificamente no cenário CPDP.

3 ABORDAGEM PROPOSTA

Nesta seção descreve a abordagem de predição de defeitos *Dynamic Selection Supervised Cross-Project* (DSSC), bem como a descrição de suas fases.

3.1 DYNAMIC SELECTION SUPERVISED CROSS-PROJECT (DSSC)

O princípio da abordagem está na hipótese de que, dado um conjunto de treinamento $\mathcal{T}$, ao gerar e selecionar modelos usando $\mathcal{T}$, o modelo terá mais informações sobre o contexto do problema e, consequentemente, será capaz de aprender sobre os mais variados tipos de módulos de software. Sendo assim, a abordagem proposta objetiva selecionar modelos preditivos com base nos dados de treinamento capazes o suficiente para classificar as novas amostras de entrada (dados de teste). Dessa forma, o método supervisionado de Predição de Defeitos entre Projetos com Seleção Dinâmica (*Dynamic Selection Supervised Cross-Project*) é dividido em três fases principais (Figura 3):

1. Na fase de Definição do Alvo, seguindo o cenário *strict* CPDP (ver Figura 1a), cada projeto do conjunto de dados é defino como projeto alvo ($\mathcal{G}$) e o restante dos projetos, exceto as demais versões de $\mathcal{G}$ (se houver), são atribuídas ao conjunto de treinamento ($\mathcal{T}$).

2. A fase de Superprodução consiste em definir um modelo preditivo competente para classificar uma dada amostra de teste $\mathbf{x}_{j,test}$. Portanto, n modelos preditivos são gerados usando técnicas de seleção dinâmica $\lambda = \{\lambda_1, ..., \lambda_n\}$ com um *pool* de classificadores $\mathcal{C} = \{c_1, ..., c_M\}$, composto por *M* classificadores base, usando o conjunto de treinamento $\mathcal{T}$ e, ao final, é selecionado o melhor modelo para predizer o conjunto de teste $\mathcal{G}$.

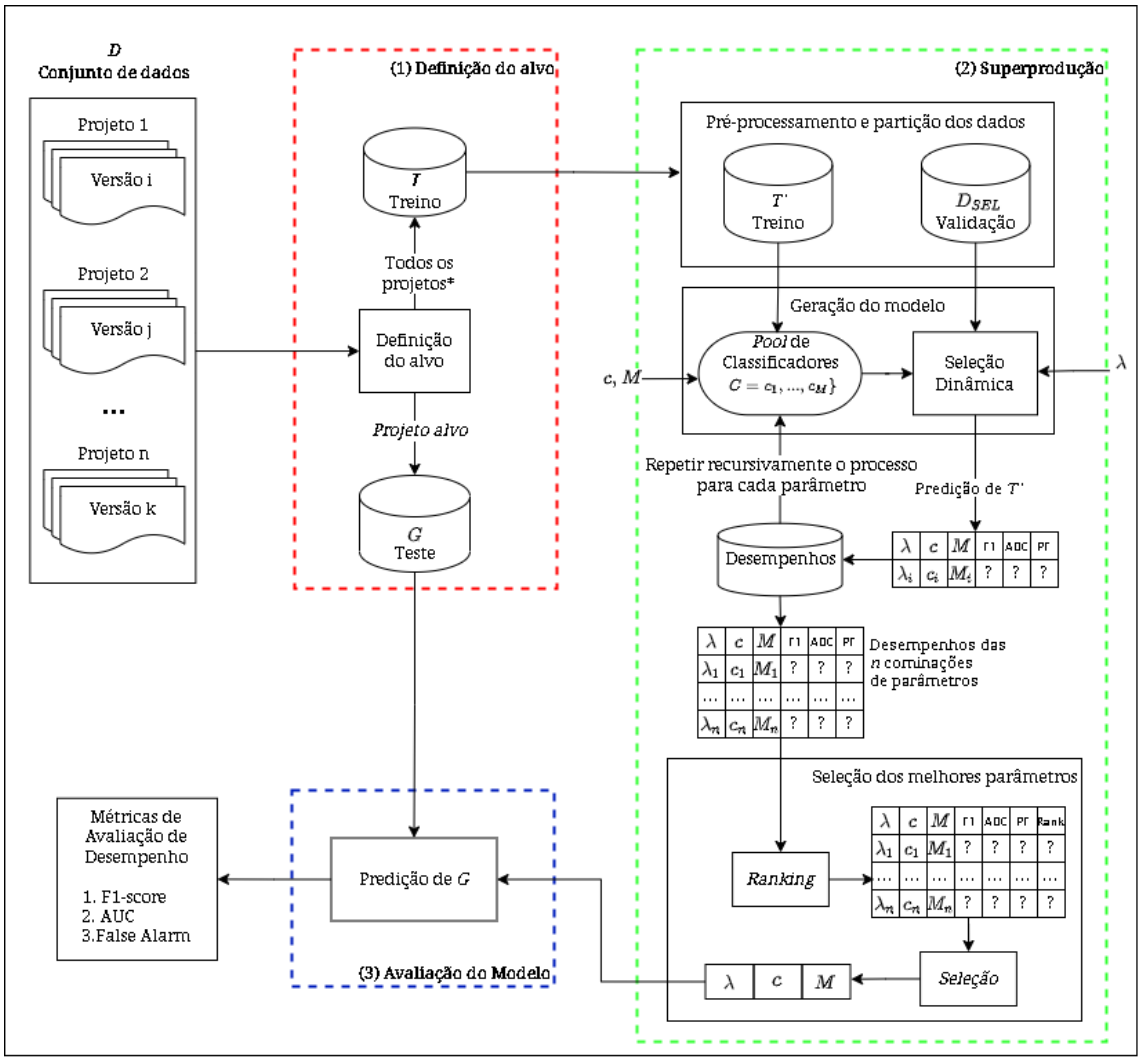
3. A Avaliação do Modelo realiza a predição e avaliação do conjunto de teste $\mathcal{G}$ usando o modelo selecionado na fase de Superprodução.

Em outras palavras, um modelo preditivo competente o suficiente para classificar um conjunto de teste $\mathcal{G}$ é gerado com base conjunto de treinamento $\mathcal{T}$. Assim, o sistema proposto difere das atuais abordagens CPDP, não apenas por usar múltiplos parâmetros (diferentes técnicas de seleção dinâmica, diferentes tipos de classificadores e variações no tamanho do

pool de classificadores), mas também porque a regra de seleção do modelo se dá pelo uso dos dados de treinamento.

A Figura 3 apresenta a visão geral do método DSSC. Conforme apresentado na figura, o termo "Todos os projetos*"indica que todos os projetos de um determinado conjunto de dados $\boldsymbol{D}$, exceto as versões de $\mathcal{G}$, podem fazer parte do conjunto de treinamento $\mathcal{G}$.

Figura 3 – Visão geral do método DSSC.



Fonte: Elaborada pelo autor (2022)

3.1.1 Definição do alvo

No cenário CPDP, o modelo de previsão é treinado usando dados dos diferentes projetos e, em seguida, é testado com dados de destino. Diante disso, nesta etapa cada projeto em um conjunto de dados (Figura 3), seguindo o cenário *strict* CPDP (HERBOLD; TRAUTSCH;

GRABOWSKI, 2017b) (Figura 1a), é definido, em ordem, como $\mathcal{G}$ (conjunto de teste) e todos os demais projetos, exceto as outras versões de $\mathcal{G}$ são atribuídas a $\mathcal{T}$ (conjunto de treinamento). O Algoritmo 1 apresenta o pseudocódigo da definição do projeto alvo a ser predito.

Algoritmo 1 : Definição do Alvo

Entrada:

D: Conjunto de dados com todos os projetos e n versões
projeto_alvo: Projeto de D, em sequência

1: $\mathcal{T}$ = Cópia de todos os projetos de D, exceto o *projeto_alvo* e suas versões
2: $\mathcal{G}$ = Cópia do projeto em D igual a *projeto_alvo*
3: **retorne** $\mathcal{T}$, $\mathcal{G}$

3.1.2 Superprodução

A fase de superprodução é divida em dois processos principais: geração do modelo e seleção. A primeira gera n modelos preditivos usando diferentes parâmetros fornecidos (técnicas de seleção dinâmica, classificadores base e tamanho do *pool*) e na segunda é selecionado o melhor modelo para predizer os dados de teste.

3.1.2.1 Geração do Modelo

Na fase de geração do modelo, primeiro, a estrutura gera diversos modelos preditivos usando o conjunto de treinamento, técnicas de seleção dinâmica, classificadores bases para gerar o *ensemble* com *Bagging* e variações do tamanho do *pool* de classificadores.

O Algoritmo 2 apresenta o pseudocódigo da geração do modelo. Primeiro é realizada a transformação dos dados (linha 6). Um aspecto importante sobre a transformação dos dados é que, geralmente, é um requisito para muitos algoritmos de aprendizado de máquina. Posteriormente, os dados em treinamento e conjunto de validação são definidos na linha 7. Na literatura de seleção dinâmica, o conjunto de validação é normalmente denominado conjunto de dados de seleção dinâmica (D_{SEL}), uma vez que esta partição é utilizada pelas técnicas de seleção dinâmica para selecionar os classificadores base (CRUZ; SABOURIN; CAVALCANTI, 2018). A próxima etapa é gerar o *pool* de classificadores; o *ensemble* pode ser homogêneo (todos os classificadores básicos são do mesmo tipo) ou heterogêneo (classificadores diferentes). Neste trabalho, é gerado um *pool* com variação de 10 em 10 e no máximo de 100 classificadores c usando a técnica *Bagging* (BREIMAN, 1996; BÜHLMANN, 2012) e, por sua vez, treinado com os

dados de treinamento (linha 8). É importante destacar que algumas técnicas de DS requerem que os classificadores base sejam capazes de estimar probabilidades. Em seguida, o modelo preditivo é construído usando uma técnica de seleção dinâmica λ a qual é inicializada com o *pool* de classificadores $\mathcal{C}$ e então treinada com D_{SEL} (linha 9). Em seguida, o desempenho do modelo é calculado com base nos dados de treinamento usando as métricas F1-score, AUC e *False Alarm* (linha 10-11). O melhor modelo é selecionado com o Algoritmo 3 (linha 15) e usado para predizer os dados de teste com o Algoritmo 4 (linha 16). Por fim, são obtidos os desempenhos do modelo (linha 17).

Algoritmo 2 : Geração do Modelo

Entrada:
$\mathcal{T}$: Conjunto de treinamento
λ_{list}: Uma lista [λ_1, λ_2, ..., λ_n] com técnicas de Seleção Dinâmica
C_{list}: Uma lista [c_1, c_2, ..., c_n] com classificadores bases para o *Bagging*

1: Inicializar *resultados* $= \phi$
2: Inicializar *scaler* $=$ Algoritmo de transformação de dados
3: **para** cada $\lambda \in \lambda_{list}$ **faça**
4: **para** cada $c \in C_{list}$ **faça**
5: **para** cada $m \in [10, 20, ..., 100]$ **faça**
6: $\mathcal{T} \leftarrow$ *scaler*$(\mathcal{T})$
7: $\mathcal{T}, D_{SEL} \leftarrow$ todas as amostras de $\mathcal{T}$
8: $\mathcal{C} \leftarrow$ gerar o *pool* com *Bagging*(c, m) e treinar com $\mathcal{T}$
9: $modelo \leftarrow$ Construir o preditor usando λ com $\mathcal{C}$ em D_{SEL}
10: *F1-score*, *AUC*, *PF* $\leftarrow$ predizer e calcular as métricas de avaliação com $\mathcal{T}$
11: Inserir em *resultados* $[modelo, z, m,$ *F1-score*$, AUC, PF]$
12: **fim para**
13: **fim para**
14: **fim para**
15: $modelo_{melhor} \leftarrow$ ***Seleção***(resultados)
16: *Desempenhos* $\leftarrow$ ***Avaliação do Modelo***$(modelo_{melhor}, scaler)$
17: **retorne** *Desempenhos*

3.1.2.2 Seleção

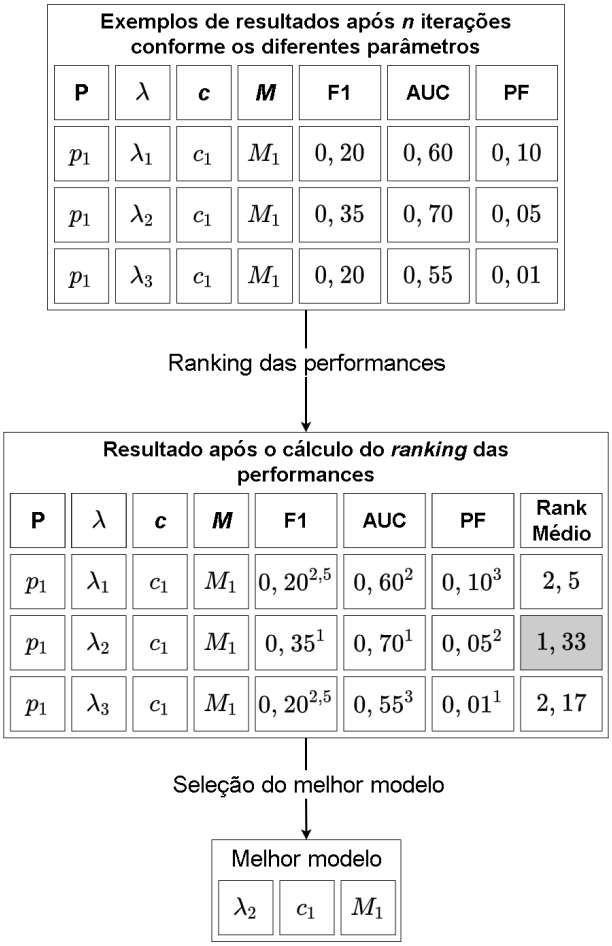
Nesta fase, é selecionado o modelo mais competente com base nos resultados de cada configuração experimental obtidos pelo Algoritmo 2. Portanto, usamos a função de classificação de dados *rankdata*, disponível na biblioteca **SciPy**[1] (VIRTANEN et al., 2020), para ranquear os desempenhos dos n modelos. Esta função classifica cada elemento dentro de uma lista, com

[1] <https://scipy.org/> — versão 1.4.1

as classificações iniciando em 1. Em seguida, são obtidas as classificações médias de cada modelo e então é selecionado o conjunto de parâmetros com o menor *ranking*.

Para melhor entender o processo de classificação dos modelos, a Figura 4 apresenta um exemplo básico com resultados após o experimento com 4 (quatro) técnicas de seleção dinâmica aleatórias $\lambda = [\lambda_1, \lambda_2, \lambda_3$ e $\lambda_4]$, 1 (um) tipo aleatório de classificador base (c_1) e 1 (uma) variação no tamanho de pool (M_1). Para cada combinação de parâmetros ($\lambda \times c \times M$), é exemplificado desempenhos com as métricas F1-score, AUC e *False Alarm* para um projeto aleatório p_1. A potência nos valores representa o *rank* do desempenho. Ainda, de acordo com o exemplo apresentado, em caso de empate entre dois ou mais elementos, ambos valores recebem a *rank* equivalente a soma de suas classificações dividido por k, sendo k a quantidade de elementos iguais (Tabela 8).

Figura 4 – Exemplo de cálculo do *ranking*.



Fonte: Elaborada pelo autor (2022)

Conforme a Tabela 8, o cálculo do *ranking* é um processo de duas etapas: ordem dos valores e cálculo da classificação. Na primeira parte, é atribuído um *rank* em ordem crescente

Tabela 8 – Exemplo de cálculo do *ranking* em caso de empate.

Valores	$0,20$	$0,35$	$0,20$
Ordem	2	1	3
Cálculo da classificação	$\frac{2+3}{2}$	$\frac{1}{1}$	$\frac{3+2}{2}$
Rank	$2,5$	1	$2,5$

Fonte: Elaborada pelo autor (2022)

aos valores, isto é, maiores valores recebem a menor classificação, caso a métrica seja de maximização, por exemplo F1-score e AUC, caso contrário os valores são ordenados em forma decrescente. No segundo processo, é realizado o cálculo da classificação usando a seguinte função:

$$Rank = \frac{\sum_{k=i}^{j} x_k}{h} \tag{3.1}$$

Onde:

- **k** representa a ordem do valor;
- **i** indica a i-ésima ordem da performance (índice inicial);
- **j** indica índice final; j-ésimo valor de acordo com as ordens dos desempenhos;
- **h** representa a número de desempenhos com o mesmo *rank*, caso existam empates;
- x_k representa o *rank* do desempenho.

Algoritmo 3 : Seleção

Entrada:

resultados: resultados cada configuração experimental

1: Inicializar *rankings* $= \phi$
2: **para** cada *métrica* $\in$ [F1, AUC, PF] **faça**
3: *valores* $\leftarrow$ todos os valores em *resultados* referente a *métrica*
4: *rankings[métrica]* $\leftarrow$ *rankdata*(valores)
5: **fim para**
6: *ranking_médio* $\leftarrow$ computar o *rank* médio de todas as combinações de parâmetros em *rankings*
7: *menor_rank* $\leftarrow$ Menor *rank* médio de *ranking_médio*
8: $\lambda_{melhor}, c_{melhor}, m_{melhor} \leftarrow$ Selecionar de *resultados* a DS, classificador e tamanho do *pool* com base no *menor_rank*
9: **retorne** $\lambda_{melhor}, c_{melhor}, m_{melhor}$

3.1.3 Avaliação do Modelo

O Algoritmo 4 apresenta o pseudocódigo da avaliação do modelo. Primeiro, é realizada a transformação dos dados, caso alguma abordagem de preprocessamento pré definida (linha 2). Por fim, são calculadas as medidas de desempenho (linha 3). Finalmente, são obtidas e retornadas o desempenho do modelo em relação às métricas de avaliação (linha 4).

Algoritmo 4 : Avaliação do Modelo

Entrada:

$\mathcal{G}$: Dados de teste
modelo: Modelo usado para predizer $\mathcal{G}$
z: Algoritmo de pré-processamento

1: Inicializar $Desempenho, F1, AUC, PF \leftarrow \phi$
2: $\mathcal{G} \leftarrow z(\mathcal{G})$ ▷ Transformação dos dados
3: $Predição \leftarrow$ Classificação do *modelo* com $\mathcal{G}$
4: $Probabilidade \leftarrow$ Calcular probabilidade das classes com *modelo* usando $\mathcal{G}$
5: $F1 \leftarrow$ Calcular F1-score $(\mathcal{G}, Predição)$
6: $AUC \leftarrow$ Calcular curva ROC-AUC $(\mathcal{G}, Probabilidade)$
7: $tn, fp, fn, tp \leftarrow$ Calcular matriz de confusão $(\mathcal{G}, Predição)$
8: $PF \leftarrow$ Calcular a probabilidade de *False Alarm* com $\frac{fp}{fp+tn}$
9: $Desempenho \leftarrow$ Anexar $F1, AUC$ e PF em *Desempenho*
10: **retorne** $Desempenho$

4 METODOLOGIA EXPERIMENTAL

Neste capítulo é descrito a configuração experimental do trabalho. Primeiro são apresentadas as características dos conjuntos de dados, as técnicas de seleção dinâmica, os classificadores base, método de avaliação (métricas e testes estatísticos) e, por fim, os métodos CPDP selecionados para comparação.

4.1 CONJUNTO DE DADOS

Os experimentos foram realizados em 4 (quatro) conjuntos de dados disponíveis publicamente. O primeiro conjunto de dados fornecido por Herbold, Trautsch e Grabowski (2017a), inclui 62 versões de produtos de 31 projetos, sendo referido como PROMISE. O segundo, denominado de RELINK, foi compartilhado por Wu et al. (2011) que contêm informações sobre defeitos de 3 projetos diferentes. O terceiro conjunto de dados contendo 12 projetos, é a versão pré-processada dos dados do NASA (MDP — *Metrics Data Program*) fornecidos por Shepperd et al. (2013). O quarto conjunto de dados foi compartilhado por D'Ambros, Lanza e Robbes (2010), no qual contêm dados sobre 5 produtos Java de projetos diferentes, sendo referido como AEEEM. Para mais detalhes sobre os conjuntos de dados, consultar o Apêndice A.

Tabela 9 – Distribuição de Instâncias e Defeitos no conjunto de dados AEEEM.

	Projeto	**Instâncias**	**Defeitos**	**%**
	eclipse	997	206	20,66
	equinox	324	129	39,81
	lucene	691	64	9,26
	mylyn	1,862	245	13,16
	pde	1,497	209	13,96
Total	**5**	**5,371**	**853**	**15,88**

Fonte: Elaborada pelo autor (2022)

Tabela 10 – Distribuição de Instâncias e Defeitos no conjunto de dados PROMISE.

Projeto	Instâncias	Defeitos	%
ant-1.3	125	11	8,8
ant-1.4	178	35	19,66
ant-1.5	293	29	9,9
ant-1.6	351	46	13,11
ant-1.7	745	93	12,48
arc	234	21	8,97
berek	43	4	9,3
camel-1.0	339	12	3,54
camel-1.2	608	99	16,28
camel-1.4	872	71	8,14
camel-1.6	965	101	10,47
ckjm	10	1	10
e-learning	64	3	4,69
forrest-0.7	29	2	6,9
ivy-1.1	111	21	18,92
ivy-1.4	241	15	6,22
ivy-2.0	352	28	7,95
jedit-3.2	272	32	11,76
jedit-4.0	306	36	11,76
jedit-4.1	312	37	11,86
jedit-4.2	367	27	7,36
jedit-4.3	492	10	2,03
kalkulator	27	5	18,52
log4j-1.0	135	21	15,56

Continua na próxima página

Tabela 10 – Distribuição de Instâncias e Defeitos no conjunto de dados PROMISE. – continuação da página anterior

Projeto	Instâncias	Defeitos	%
log4j-1.1	109	17	15,6
log4j-1.2	205	40	19,51
lucene-2.0	195	41	21,03
lucene-2.2	247	60	24,29
lucene-2.4	340	69	20,29
nieruchomosci	27	8	29,63
pbeans1	26	6	23,08
pbeans2	51	5	9,8
pdftranslator	33	10	30,3
poi-1.5	237	81	34,18
poi-2.0	314	35	11,15
poi-2.5	385	137	35,58
poi-3.0	442	161	36,43
redaktor	176	26	14,77
serapion	45	4	8,89
skarbonka	45	7	15,56
sklebagd	20	8	40
synapse-1.0	157	13	8,28
synapse-1.1	222	40	18,02
synapse-1.2	256	52	20,31
systemdata	65	4	6,15
szybkafucha	25	8	32
termoproject	42	10	23,81

Continua na próxima página

Tabela 10 – Distribuição de Instâncias e Defeitos no conjunto de dados PROMISE. – continuação da página anterior

	Projeto	Instâncias	Defeitos	%
	tomcat	858	55	6,41
	velocity-1.4	196	49	25
	velocity-1.5	214	65	30,37
	velocity-1.6	229	34	14,85
	workflow	39	19	48,72
	wspomaganiepi	18	4	22,22
	xalan-2.4	723	79	10,93
	xalan-2.5	803	297	36,99
	xalan-2.6	885	271	30,62
	xalan-2.7	909	189	20,79
	xerces-1.2	440	38	8,64
	xerces-1.3	453	27	5,96
	xerces-1.4	588	151	25,68
	xerces-init	162	77	47,53
	zuzel	29	6	20,69
Total	**61**	**17.681**	**6,062**	**34,29**

Fonte: Elaborada pelo autor (2022)

Tabela 11 – Distribuição de Instâncias, Defeitos e Esforço de Inspeção no conjunto de dados RELINK.

	Projeto	Instâncias	Defeitos	%
LOC				
	Apache2.0	194	98	50,52
	openintents	56	22	39,29
	zxing1.6	399	118	29,57
Total	**3**	**649**	**238**	**36,67**

Fonte: Elaborada pelo autor (2022)

Tabela 12 – Distribuição de Instâncias, Defeitos e Esforço de Inspeção no conjunto de dados NASA.

	Projeto	Instâncias	Defeitos	%
	CM1	344	42	12,21
	JM1	9,593	1,759	18,34
	KC1	2,096	325	15,51
	KC3	200	36	18
	MC1	9,277	68	0,73
	MC2	127	44	34,65
	MW1	264	27	10,23
	PC1	759	61	3,84
	PC2	1,585	16	1,01
	PC3	1,125	140	12,44
	PC4	1,399	178	12,72
	PC5	17,001	503	2,96
Total	**12**	**43,770**	**3,199**	**7,31**

Fonte: Elaborada pelo autor (2022)

4.2 TÉCNICAS DE SELEÇÃO DINÂMICA

As técnicas de seleção dinâmica foram: *K-Nearest Oracles Union* (KNORA-U), *K-Nearest Oracles Eliminate* (KNORA-E), *k-Nearest Output Profiles* (KNOP), *META-DES*, *Overall Local Accuracy* (OLA), *Local Class Accuracy* (LCA), *Multiple Classifier Behavior* (MCB), *Modified Classifier Rank* (Rank) (ver Tabela 13). Importante relatar que não foi utilizado nenhum critério para a escolha das técnicas de seleção dinâmica, de forma que foram utilizadas as técnicas comumente utilizadas em experimentos na biblioteca **DESlib**[1] (CRUZ et al., 2020). Elas são descritas a seguir:

- **KNORA-U** (KNU): seleciona todos os classificadores que predizem corretamente pelo menos uma amostra na região de competência da amostra de teste. Quanto mais amostras o classificador prediz corretamente, mais votos ele ganha.
- **KNORA-E** (KNE): seleciona todos os classificadores que classificam corretamente todas as amostras na região de competência de uma determinada amostra de teste. Apenas os classificadores com precisão perfeita são selecionados. Caso nenhum classificador seja selecionado, a região de competência é reduzida até pelo menos um classificador ser selecionado.

[1] <https://github.com/Menelau/DESlib> — versão 0.3.5

- **KNOP**: funciona de forma semelhante à regra do KNU, com diferença que o KNORA-U trabalha no espaço de características, enquanto o KNOP trabalha o espaço de dimensão. Logo, seguindo a mesma regra do KNU, cada vez que um classificador base realiza uma predição correta para uma amostra de teste, ele ganha um voto. Ao final, os votos obtidos por todos os classificadores são agregados para obter a decisão do conjunto.

- **META-DES**: *framework* que utiliza meta-aprendizagem para selecionar dinamicamente o melhor conjunto de classificadores. Portanto, utiliza diferentes medidas para estimar a competência local de um classificador base C_i, a fim de decidir se o classificador é competente o suficiente para classificar uma determinada amostra de teste.

- **OLA**: avalia o nível de competência de cada classificador e seleciona o mais competente para predizer a classe de cada amostra de teste. A competência é determinada pela precisão de classificação na região de competência. Em caso de empate, isto é, mesmo nível de competência, o primeiro que foi avaliado é selecionado primeiro.

- **LCA**: avalia e seleciona o classificador com maior competência para predizer cada amostra de teste x. Por exemplo, seja um classificador C_i, que atribui uma amostra de teste como sendo da classe w_j, seu nível de competência é estimado pela porcentagem de amostras de treinamento localmente atribuídas como à classe w_j e previstas corretamente.

- **MCB**: avalia nível de competência de cada classificador levando em consideração sua precisão local na região de competência. Um único classificador é selecionado apenas se seu nível de competência for significativamente superior do que os demais classificadores no *pool*. Caso contrário, todos os classificadores no *pool* são combinados usando a regra de votação por maioria.

- **Rank**: estima o *ranking* de um classificador base C_i de acordo com o número de amostras classificadas corretamente na região de competência θ_j. É selecionado o classificador que classifica corretamente o maior número de amostras consecutivas.

Tabela 13 – Técnicas de seleção dinâmica consideradas no experimento.

Técnica	**Categoria**	**Abordagem**	**Referência**
K-Nearests Oracles Union (KNORA-U)	Oracle	DES	Ko, Sabourin e Jr (2008)
K-Nearests Oracles Eliminate (KNORA-E)	Oracle	DES	Ko, Sabourin e Jr (2008)
k-Nearest Output Profiles (KNOP)	Behavior	DES	Cavalin, Sabourin e Suen (2013)
META-DES	Meta-learning	DES	Cruz et al. (2015)
Overall Local Accuracy (OLA)	Accuracy	DCS	Woods, Kegelmeyer e Bowyer (1997)
Local Class Accuracy (LCA)	Accuracy	DCS	Woods, Kegelmeyer e Bowyer (1997)
Multiple Classifier Behavior (MCB)	Behavior	DCS	Giacinto e Roli (2001)
Modified Classifier Rank (Rank)	Ranking	DCS	Sabourin et al. (1993)

Fonte: Elaborada pelo autor (2022)

4.3 CLASSIFICADORES BASE

As técnicas de seleção dinâmica utilizam um conjunto de classificadores a fim de selecionar o melhor classificador ou um subconjunto com os melhores classificadores. Neste experimento, foi seguido o conceito de conjunto homogêneo, ou seja, um *pool* composto por n classificadores básicos semelhantes, como: *Decision Tree* (DT), *Logistic Regression* (LR), *Naive Bayes* (NB), *Random Forest* (RF) e *Support Vector Machine* (SVM). Foram utilizados os classificadores disponíveis na biblioteca scikit-learn (PEDREGOSA et al., 2011) com seus parâmetros padrões.

4.4 AVALIAÇÃO DO DESEMPENHO

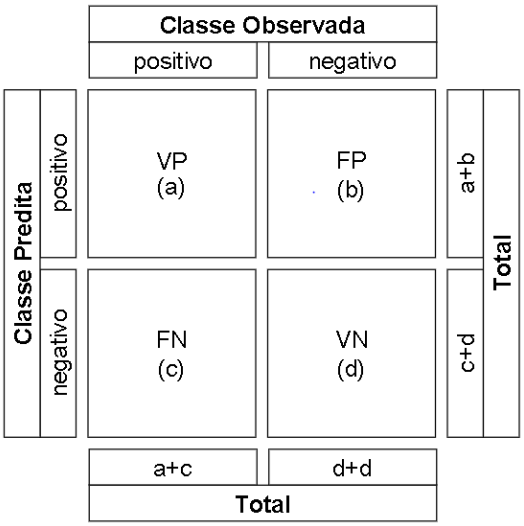
Há diversas métricas para avaliação do desempenho de modelos preditivos com base no paradigma de predição de defeitos de software, de forma que, essas métricas mensuram o desempenho das predições resultante do modelo ajustado, assim avaliando o quanto ele reproduz o valor real para a resposta de interesse.

Para modelos de predição por classificação, geralmente se realiza uma tabulação das classes observadas e preditas usando uma *matriz de confusão* apresentada na Figura 5.

A partir da Tabela 5 os valores observados e valores preditos podem ser descritos como:

- **True Positive** (TP – Verdadeiro Positivo): refere-se ao número de módulos de software defeituosos que são classificados corretamente como defeituosas;
- **True Negative** (TN – Verdadeiro Negativo): é o número de módulos de software não defeituosos que são classificados corretamente como não defeituoso;

Figura 5 – Exemplo de Matriz de Confusão para problemas de Classificação.



Fonte: Elaborada pelo autor (2022)

- **False Positive** (FP – *Falso Positivo*): denota o número de instâncias de software sem defeitos que são erroneamente classificadas como defeituosas;
- **False Negative** (FN – *Falso Negativo*): quantifica o número de instâncias de software com defeito que são erroneamente classificadas como não defeituosas.

Por fim, a partir da matriz de confusão (Tabela 5) e das descrições sobre seus valores, diversas métricas podem ser mensuradas, entre elas na Tabela 14 são apresentadas métricas comumente usadas para avaliar modelos de predição de defeitos.

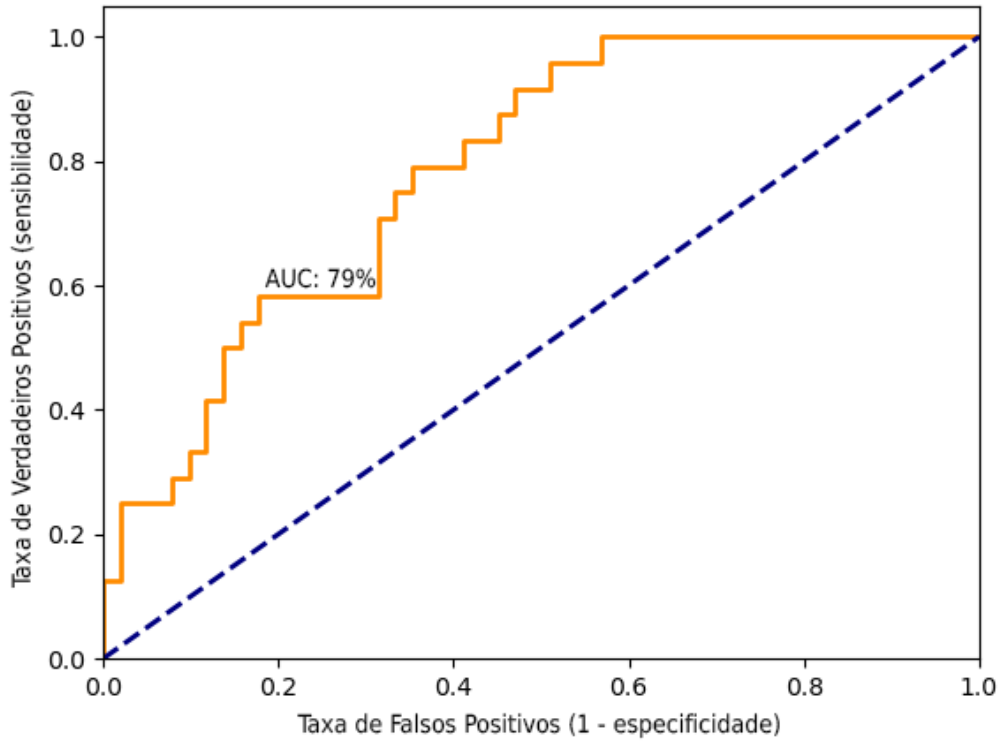
Além dessas métricas, uma medida aplicada para resumir o desempenho do modelo em diferentes contextos é a construção da Área Sob a Curva ROC (ROC-AUC) (HANLEY; MCNEIL, 1982; BRADLEY, 1997). Essa curva é uma ilustração 2D que calcula a área sob a curva da característica de operação do receptor (ROC – *Receiver Operating Characteristic*) a partir de TPR e FPR, onde ROC trata-se de uma curva de probabilidade e AUC representa o grau ou medida de separabilidade de forma que, quanto maior a AUC, melhor será o modelo em predizer i classes como 0 (não defeito) e j classes como 1 (defeito). Usamos a AUC por ser uma medida adequada para conjuntos de dados desequilibrados e, especialmente, para problemas binários (LÓPEZ et al., 2013). Importante ressaltar que essa métrica foi calculada a partir da função *predict proba*. A Figura 6 apresenta um exemplo de curva ROC-AUC para um determinado algoritmo de classificação.

Tabela 14 – Métricas de desempenho derivadas da matriz de confusão.

Métrica	Cálculo	Descrição
Precisão	$\frac{TP}{TP+FP}$	Proporção de predições positivas corretamente identificadas.
Sensibilidade; taxa de verdadeiro positivos; recall	$TPR = \frac{TP}{TP+FN}$	Proporção de positivos reais identificados corretamente.
Especificidade; taxa de verdadeiros negativos	$TNR = \frac{TN}{TN+FP}$	Proporção de verdadeiros negativos identificados corretamente.
Taxa de falsos positivos; Alarme Falso	$FPR = \frac{FP}{FP+TN}$	Proporção de observações negativas classificadas erroneamente como verdadeiras.
F-score (F1)	$\text{F1} = 2 \times \frac{Precisão \times Recall}{Precisão + Recall}$	Média harmônica entre precisão e o recall; com alguns estudos na literatura de detecção defeitos (KIM et al., 2011; LEE et al., 2011).

Fonte: Elaborada pelo autor (2022)

Figura 6 – Exemplo de uma curva ROC-AUC.



Fonte: Elaborada pelo autor (2022)

4.5 MÉTODOS CPDP

Como critério para seleção dos métodos para avaliar o desempenho em diferentes cenários, foram escolhidos os métodos que apresentam melhores resultados na literatura. Em particular, Herbold, Trautsch e Grabowski (2018) conduziram um experimento de larga escala onde fizeram uma comparação entre 24 métodos CPDP em 85 projetos de 5 conjunto de dados diferentes. Além dos quatro métodos selecionados, também comparamos o DSSC com os resultados replicados do método supervisionado EASC-NB (NI et al., 2020). A Tabela 15 apresenta os métodos CPDP selecionados para comparação experimental. Ambos os métodos supervisionados bem como seus desempenhos, com exceção do EASC-NB, foram verificados por Herbold, Trautsch e Grabowski (2017a) e Herbold (2018).

Tabela 15 – Métodos CPDP de trabalhos relacionados selecionados para comparação.

Abordagem CPDP	**Descrição**
Cruz e Ochimizu (2009) (CamargoCruz09-DT)	Esta abordagem busca padronizar os dados de teste e treinamento usando o logaritmo e a mediana do conjunto de treinamento como referência, ou seja, $\hat{m}_i = log(1+m_i(s))+median(log(1+m_i(S)))-median(log(1+m_i(S^train)))$. Os autores consideram um único produto de treinamento como referência.
Turhan et al. (2009) (Turhan09-DT)	O método CPDP primeiro transforma os dados métricos com o logaritmo e, em seguida, aplica um filtro de relevância aos dados de treinamento com base no *k*-Nearest Neighbor (KNN). Através do filtro de relevância, as *k* instâncias mais próximas de cada instância nos dados de teste são selecionadas. Os autores sugerem usar $k = 10$.
Menzies et al. (2011), Menzies et al. (2012) (Menzies11-RF)	Modelo local criado através do agrupamento dos dados de treinamento com o algoritmo WHERE para, posteriormente, classificar os resultados com o algoritmo de aprendizagem da regra WHICH. Regras de WHICH separadas são criadas para cada *cluster* criar modelos locais.
Watanabe, Kaiya e Kaijiri (2008) (Watanabe08-DT)	Método que padroniza os dados de destino, como $\hat{m}_i(s^*) = \frac{m_i \cdot mean(m_i(S))}{mean(m_i(S^*)}$ para melhorar a homogeneidade entre o treinamento e a nova entrada (dados de teste).
Ni et al. (2020) (EASC-NB)	Método supervisionado que constrói um modelo de predição que leva em conta o esforço necessário para inspecionar um determinado módulo de software. Este método é baseado em métricas de desempenho, tais como: medidas de desempenho sem consciência de esforço (NPM) e medidas de desempenho com consciência de esforço (EPM).

Fonte: Elaborada pelo autor (2022)

4.6 CONFIGURAÇÃO DOS EXPERIMENTOS

Os experimentos foram conduzidos utilizando a linguagem Python com a biblioteca **scikit-learn**[2] (PEDREGOSA et al., 2011) para o treinamento dos classificadores base. As técnicas de seleção dinâmica de classificadores (consultar Tabela 13) foram avaliadas usando a biblioteca **DESlib**[3] (CRUZ et al., 2020). Para ambas as técnicas de seleção dinâmica foram utilizados seus parâmetros padrões.

A abordagem de geração de classificadores escolhida para construir o *pool* de classificadores foi o *Bagging* (BREIMAN, 1996). Devido sua simplicidade e eficiência, este método seleciona aleatoriamente um subconjunto de amostras do conjunto de treinamento para treinar um classificador, de forma que, esse processo é repetido diversas vezes para construir novos classificadores para compôr o *pool*. A quantidade de iterações do *Bagging* corresponde ao tamanho do *pool* gerado. Em nosso experimento, o tamanho do *pool* foi variado entre os tamanhos $M = [10, ..., 100]$, com intervalos de 10 classificadores.

Para a avaliação em cada conjunto de dados D (AEEEM, NASA, PROMISE e RELINK), foi adotada a abordagem *strict* CPDP (ver Figura 1a), de forma que, para as n execuções com base na quantidade de projetos no conjunto de dados, é obtido a média e variância para cada conjunto de dados. Como as técnicas de seleção dinâmica requerem um conjunto de validação D_{SEL}, atribuímos a D_{SEL} os mesmos dados contidos no conjunto de treinamento $\mathcal{T}$.

Para as diferentes comparações, foram utilizadas as mesmas configurações experimentais, isto é, conjunto de dados e métricas de avaliação. Com exceção do EASC-NB, os resultados dos demais métodos CPDP usados para comparação foram obtidos a partir do *kit*[4] replicação disponibilizado por Herbold, Trautsch e Grabowski (2017a). Quanto ao EASC-NB, utilizamos os resultados obtidos através da replicação do método seguindo as orientações e configurações conforme apresentados por Ni et al. (2020). Um ponto importante sobre o EASC-NB, é sua limitação quanto aos dados em que o método pode ser aplicado, ou seja, requer que um conjunto de dados específico ofereça informações sobre a quantidade de linhas que determinado módulo de software possui. De forma que, esta informação é uma variável independente no conjunto de dados que define a quantidade de linhas de código (*Lines of Code* – LOC), na qual se refere e/ou está relacionada ao esforço/tempo necessário para inspecionar uma dada

[2] <https://scikit-learn.org/> — versão 1.0.2
[3] <https://github.com/Menelau/DESlib> — versão 0.3.5
[4] <https://github.com/sherbold/replication-kit-tse-2017-benchmark>

instância (módulo de software).

Os testes estatísticos foram realizados usando a bilbioteca **scipy.stats**[5]. Logo, para comparação em pares foi conduzido o teste Wilcoxon Signed Rank (WILCOXON, 1992) para determinar se há diferença estatisticamente e significativamente diferente e o teste não paramétrico Cliff's delta (δ) (CLIFF, 2014). O Wilcoxon Signed Rank apenas informa se há ou não diferença, logo usamos o Cliff's delta pois quantifica a diferença entre o desempenho de dois métodos. O intervalo do delta é $[-1, 1]$, para δ igual a 1 indica que o desempenho de um método supera o de outro. Para δ igual a 0 significa que não há diferença entre os dois métodos. Diante disso, utilizamos os intervalos propostos por Cliff (2014), onde considera-se δ que são menores que 0.147, entre 0.147 e 0.33, entre 0.33 e 0.474 e acima de 0.474 como diferença "Insignificante (Negligible — N)", "Pequena (Small — S)", "Média (Medium — M)" e "Grande (Large — L)", respectivamente. Por fim, a Tabela 16 sumariza os diversos componentes utilizadas nos experimentos.

Tabela 16 – Componentes utilizados nos experimentos.

Conjuntos de Dados	AEEEM, NASA, PROMISE e RELINK
Método de Geração do *pool*	*Bagging*
Tamanhos do *pool*	10, 20, 30, 40, 50, 60, 70, 80, 90, 100
Classificadores Bases	*Decision Tree, Logistic Regression, Naive Bayes*, *Random Forest e Support Vector Machine* **Fonte:** Elaborada pelo autor (2022)
Métodos de Seleção Dinâmica	KNORA-U,KNORA-E, METADES, KNOP, OLA, LCA, MCB e Rank **Fonte:** Elaborada pelo autor (2022)
Métodos CPDP	EASC-NB, CamargoCruz09-DT, Turhan09-DT, Menzies11-RF e Watanabe08-DT **Fonte:** Elaborada pelo autor (2022)
Métricas de Avaliação	F1-score, AUC e False Alarm
Testes Estatísticos	Wilcoxon Signed Rank e Cliff's delta **Fonte:** Elaborada pelo autor (2022)

Fonte: Elaborada pelo autor (2022)

[5] <https://scipy.org/> — versão 1.4.1

4.6.1 Entendendo os resultados pré-análise

Considerando que o experimento é uma seleção e avaliação de n iterações com base em uma determinada configuração de componentes, sejam: técnicas de seleção dinâmica, classificadores bases e a variação do tamanho do *pool* de classificadores. Então, foram realizados no total de 32.800 experimentos (82 projetos $\times$ 8 técnicas de seleção dinâmica $\times$ 5 classificadores base $\times$ 10 variações no tamanho do *pool* — tamanho de 10 a 100, com intervalos de 10). O tratamento dos resultados são separados em duas partes: geração e seleção do modelo, e predição do projeto alvo (dados de teste).

No primeiro momento, ao rodar os experimentos, é obtido um conjunto de resultados das diferentes configurações (DS, classificador e tamanho do *pool*) e, em seguida, é realizado o cálculo de *ranking* para cada configuração com base nas métricas F1-score, AUC e PF (Alarme Falso). A Tabela 17 apresenta uma visão dos resultados obtidos durante o processo de geração do modelo (Algoritmo 2) e após calcular o *ranking* de cada configuração (Algoritmo 3).

Tabela 17 – Exemplo da tabela de resultados com o conjunto de configurações para um dado projeto de um conjunto de dados. *Decision Tree* (DT), *Logistic Regression* (LR), tamanho do *pool* de classificadores (M).

Projeto Alvo	**Método DS**	**Classificador**	***M***	**F1-score**	**AUC**	**PF**	***Rank***
Apache2.0	KNORAU	DT	10	0,373	0,437	0,801	179,000
Apache2.0	KNORAU	DT	20	0,000	0,413	0,000	222,833
Apache2.0	KNORAU	DT	30	0,000	0,391	0,000	243,833
Apache2.0	KNORAU	DT	40	0,000	0,387	0,000	228,500
Apache2.0	KNORAU	DT	50	0,000	0,416	0,000	222,167
...	...	...	...	...	...	...	...
Apache2.0	Rank	LR	60	0,000	0,604	0,000	98,500
Apache2.0	Rank	LR	70	0,000	0,561	0,012	173,500
Apache2.0	Rank	LR	80	0,056	0,536	0,012	124,833
Apache2.0	Rank	LR	90	0,029	0,566	0,000	64,167

Continua na próxima página

Tabela 17 – Exemplo da tabela de resultados com o conjunto de configurações para um dado projeto de um conjunto de dados. *Decision Tree* (DT), *Logistic Regression* (LR), tamanho do *pool* de classificadores (M). – continuação da página anterior

Projeto Alvo	**Método DS**	**Classificador**	***M***	**F1-score**	**AUC**	**PF**	***Rank***
Apache2.0	Rank	LR	100	0,029	0,556	0,006	123,167

Fonte: Elaborada pelo autor (2022)

Tabela 18 – Exemplo da tabela após calcular o *ranking*, seleção e predição do projeto alvo. *Logistic Regression* (LR), tamanho do *pool* de classificadores (M).

Projeto alvo	**Método DS**	**Classificador**	**M**	**F1-score**	**AUC**	**PF**
Apache2.0	KNORAE	LR	30	0,746	0,781	0,240
openintents	KNORAU	LR	80	0,581	0,775	0,000
zxing1.6	MCB	LR	60	0,301	0,660	0,103

Fonte: Elaborada pelo autor (2022)

A configuração com o melhor *rank* (menor sendo melhor) é utilizada para realizar a predição do projeto alvo (dados de teste). Portanto, é obtida uma segunda tabela contendo os resultados referentes a cada projeto. Por fim, a partir do exemplo de resultados apresentados na Tabela 18 são realizadas as análises conforme os pontos levantados na seção 1.1, isto é: (1) comportamento da estrutura proposta no cenário CPDP; (2) DSSC versus abordagens do estado da arte; (3) qualidade dos métodos CPDP; e (4) quando usar a abordagem proposta. Para as análises foram utilizadas técnicas estatísticas apresentadas na Tabela 16.

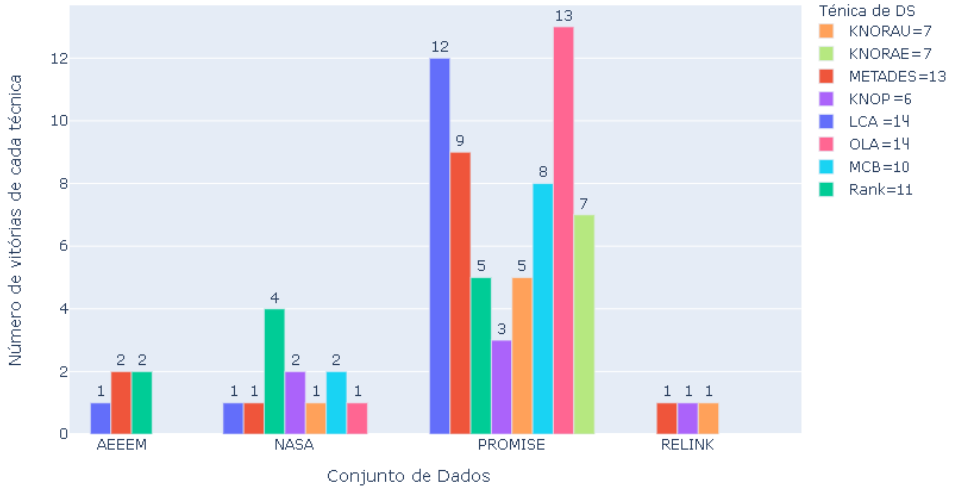
5 RESULTADOS

Nesta seção são mostradas as análises realizadas para responder as questões de pesquisa levantadas. Comparamos os resultados obtidos pelo DSSC × cinco métodos CPDP encontrados na literatura (CRUZ; OCHIMIZU, 2009; TURHAN et al., 2009; MENZIES et al., 2012; WATANABE; KAIYA; KAIJIRI, 2008; NI et al., 2020). Devido ao escopo do DSSC, cenário CPDP, bem como aos dados coletados, diferentes análises podem ser feitas, por exemplo, avaliação da performance dos métodos em relação a cada métrica de avaliação, para cada conjunto de dados separadamente, etc. Portanto, para facilitar esses *insights* adicionais, é fornecido nos Apêndices A e B um conjunto de resultados experimentais.

5.1 CPDP: COMPORTAMENTO DA ESTRUTURA PROPOSTA (DSSC)

Inicialmente, para entender melhor como a estrutura proposta com seleção dinâmica se comporta, a Figura 7 apresenta o número de vezes que cada técnica de seleção dinâmica foi escolhida, com base no conjunto de treinamento, para realizar a predição de um determinado conjunto de teste $\mathcal{G}$. Na figura é mostrado quantas vezes (número acima da barra) cada técnica de seleção dinâmica foi escolhida para predizer $\mathcal{G}$; no canto superior direito apresenta a totalidade de cada uma das técnicas com base no número total de projetos (82). Por fim, de acordo com os número totais de vitórias, ambas técnicas LCA (14) e OLA (14) obtiveram a melhor pontuação, seguindo do METADES (13), enquanto KNOP (6) obtém a pior.

Figura 7 – Quantidade de vezes em que cada técnica de seleção dinâmica foi selecionada com base nos 82 projetos.



Fonte: Elaborada pelo autor (2022)

Outra característica do DSSC é a possibilidade de utilizar diferentes classificadores básicos durante a construção do *pool* de classificadores a ser utilizado pelas DS. Portanto, de forma semelhante à escolha das técnicas de seleção dinâmica, durante as n iterações realizadas pelo DSSC, também é possível definir qual o classificador a compôr o *pool*. Na Tabela 19 são apresentadas as pontuações de cada classificador base. Na primeira coluna apresenta os conjunto de dados utilizados e a segunda coluna lista os classificadores utilizados no DSSC com suas respectivas pontuações.

Tabela 19 – Número vitórias/empates dos classificadores básicos no processo de seleção no DSSC em quatro conjuntos de dados. *Decision Tree* (DT), *Logistic Regression* (LR), *Naive Bayes* (NB), *Random Forest* (RF) e *Support Vector Machine* (SVM).

	Classificadores				
Dados	NB	LR	RF	DT	SVM
RELINK	0	3	0	0	0
AEEEM	0	0	5	0	0
NASA	0	0	12	0	0
PROMISE	33	29	0	0	0
Total	**33**	32	17	0	0

Fonte: Elaborada pelo autor (2022)

No geral, nossa análise sugere que modelos usando o classificador SVM apresentam desempenho inferior. Isso pode ocorrer porque as configurações padrões do SVM não são ideais para o contexto aplicado, logo sendo necessário a otimização de parâmetros para melhorar o desempenho (BIRD et al., 2009). Além disso, visto o desequilíbrio dos dados, supõe-se que modelos com *Decision Tree* apresentam desempenho inferior, pois este tipo de classificador aparentou ser sensível a esses tipos de dados.

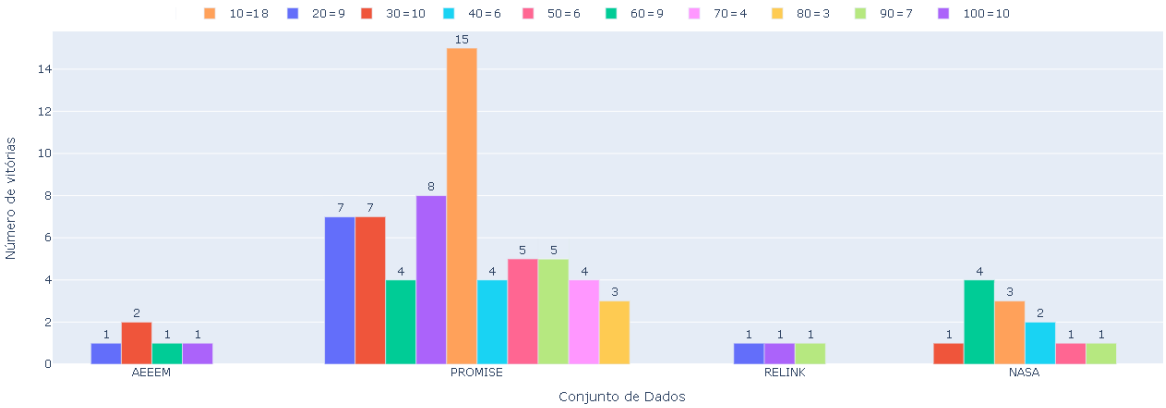
Além disso, nossa análise sugere que classificadores simples de aplicar e de compreender, como *Naive Bayes* e *Logistic Regression*, apresentam bons desempenhos para conjunto de dados constituídos de *features* geradas a partir da combinação de variáveis independentes, como PROMISE e RELINK. Outro ponto é a caracterização desses dados, onde os dois conjuntos apresentam são baseados em informações (métricas de software) coletadas em nível de classe, isto é, número de métodos por classe, quantidade de linhas, etc.

Por mim, não foi possível identificar a relação do classificador RF ter sido utilizado para predizer todos os projetos dos conjuntos de dados da NASA e AEEEM. Sendo que, enquanto os dados da NASA contém informações de várias métricas de código estático (tamanho, legibilidade, complexidade, etc.) que estão relacionadas à qualidade do software, os dados

AEEEM consistem em métricas de código-fonte baseadas em métricas de mudança, entropia, etc. Evidentemente, esses conjuntos de dados são baseados em diferentes recursos de software. Portanto, é necessário mais estudos para melhor analisar e relatar esse contexto mais detalhado. Consultar o Apêndice A para melhor entender as características dos conjuntos de dados.

No DSSC, além de ser possível utilizar diferentes técnicas de seleção dinâmica bem como diferentes classificadores básicos, é possível também definir e selecionar o melhor tamanho do *ensemble* entre os intervalos de 10 a 100, com variações de 10 (ver seção 3.1.2). Portanto, na Figura 8 apresenta a análise do número de vitórias referente ao melhor tamanho do *pool* de classificadores selecionado com base no conjunto de treinamento para predizer um determinado conjunto de teste.

Figura 8 – Quantidade de vezes em que um número específico de classificadores no *pool* foi selecionada com base nos 82 projetos.



Fonte: Elaborada pelo autor (2022)

5.2 CPDP: ABORDAGEM PROPOSTA (DSSC) VERSUS ABORDAGENS DO ESTADO DA ARTE

Nesta seção serão realizadas comparações entre métodos CPDP usando as mesmas configurações experimentais e medidas de desempenho. Foram comparados os seguintes métodos CPDP: (1) o método proposto *Dynamic Selection Surpevised Cross-Project* (DSSC), (2) *Effort-Aware Supervised Cross-project* (EASC-NB), (3) *CamargoCruz09-DT* (CC09-DT), (4) *Turhan09-DT* (T09-DT), (6) *Menzies11-RF (M11-RF)* e (6) *Watanabe08-DT (W08-DT)* (ver Tabela 15). Dado que os diferentes métodos selecionados para comparação seguem o mesma configuração experimental, ou seja, mesmos conjuntos de dados, métricas de avaliação e classificadores básicos, as análises experimentais serão comparadas em relação a cada projeto,

cada conjunto de dados, como também de forma geral.

5.2.1 Comparação com *F1-score*.

Na Tabela 20 apresenta resultado em forma de média±variância dos métodos CPDP em cada conjunto de dados individualmente. Para cada conjunto de dados, os melhores resultados são destacados em negrito e os resultados significativamente diferentes e inferiores ao DSSC são marcados com $\circ$ (*p-valor* $\leq 0,05$), onde o *p-valor* é obtido pelo teste Wilcoxon Signed Rank. Conforme apresentado, não há como definir uma abordagem CPDP que se sobressaia em relação as demais. Entretanto, o método EASC-NB se destaca com melhores resultados nos conjuntos de dados AEEEM e NASA. Quanto ao DSSC, para AEEEM obteve $\frac{4}{5}$ dos melhores resultados e para RELINK $\frac{3}{5}$.

Tabela 20 – Comparação entre DSSC e métodos CPDP supervisionados em quatro conjuntos de dados em termos de F1-score na forma de média±variância.

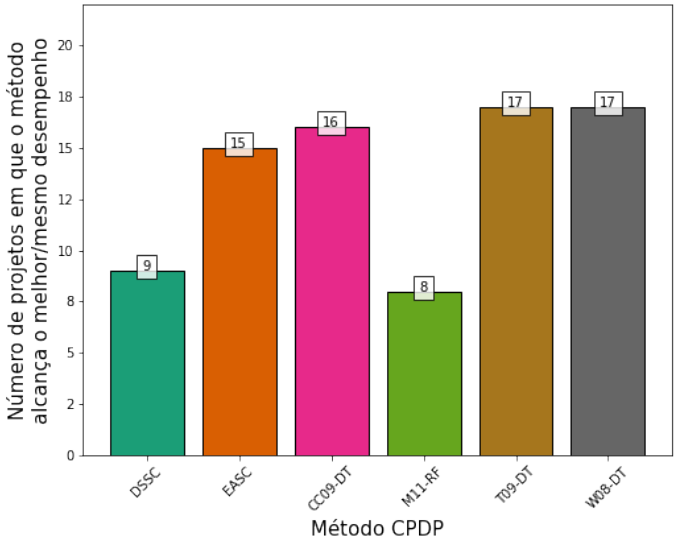
	AEEEM	NASA	PROMISE	RELINK
DSSC	0,347±0,014	0,136±0,003	0,278±0,020	0,542±0,051
EASC-NB	**0,371±0,010**	**0,216±0,014**	0,285±0,024	0,239±0,115
CC09-DT	0,312±0,007	0,089±0,007	0,367±0,023	0,543±0,003
M11-RF	0,269±0,008	0,142±0,003	0,328±0,016	**0,555±0,009**
T09-DT	0,271±0,004	0,161±0,007	0,359±0,029	0,528±0,057
W08-DT	0,301±0,000	0,109±0,002	**0,368±0,014**	0,489±0,022

Fonte: Elaborada pelo autor (2022)

Ainda, na Figura 9 mostra a quantidade total de vitórias/empates de cada método CPDP para os 82 projetos. Para cada vez que uma abordagem CPDP obtiver o melhor resultado, isto é, para cada um dos 82 projetos, esta recebe 1 ponto; em caso de empate entre dois ou mais métodos, cada um deles recebem a pontuação de $\frac{1}{n}$, onde n representa o número de métodos CPDP que obtiveram o maior desempenho. Conforme os resultados, o DSSC obteve 9 pontos contra 17 do T09-DT e W08-DT.

Além de computar e analisar o número de vitória/empates, também avaliamos os resultados dos métodos CPDP para cada projeto. Então, para cada projeto, associamos a melhor F1-score obtida entre os seis métodos CPDP. Essa associação dos melhores métodos para cada projeto é apresentada na Tabela 21. Os resultados gerais, isto é, desempenho de cada método $\times$ projeto é apresentado com mais detalhes no Apêndice C.1.

Figura 9 – Gráfico de barras mostrando o número de projetos em que cada abordagem CPDP apresentou o maior/mesmo F1-score.



Fonte: Elaborada pelo autor (2022)

Tabela 21 – Melhor valor F1-score e respectivo melhor método CPDP por projeto.

Conjunto de Dados	Projeto	Melhor F1-score	Melhor(es) Método(s)
AEEEM	eclipse	0,555	DSSC
	equinox	0,434	EASC
	lucene	0,302	EASC
	mylyn	0,316	W08-DT
	pde	0,342	DSSC
	ant-1.3	0,381	T09-DT
	ant-1.4	0,357	W08-DT
	ant-1.5	0,389	DSSC
	ant-1.6	0,491	CC09-DT
	ant-1.7	0,468	T09-DT
	arc	0,293	CC09-DT
	berek	0,857	EASC
	camel-1.0	0,308	EASC

Continua na próxima página

Tabela 21 – Melhor valor F1-score e respectivo melhor método CPDP por projeto. – continuação da página anterior

Conjunto de Dados	Projeto	Melhor F1-score	Melhor(es) Método(s)
	camel-1.2	0,360	T09-DT
	camel-1.4	0,337	CC09-DT
	camel-1.6	0,345	CC09-DT
	ckjm	0,500	W08-DT
	e-learning	0,128	W08-DT
	forrest-0.7	0,571	W08-DT
	ivy-1.1	0,467	CC09-DT
	ivy-1.4	0,350	DSSC
	ivy-2.0	0,424	DSSC
	jedit-3.2	0,585	CC09-DT
	jedit-4.0	0,500	CC09-DT
	jedit-4.1	0,601	EASC
	jedit-4.2	0,481	DSSC
	jedit-4.3	0,094	DSSC
	kalkulator	0,545	CC09-DT
	log4j-1.0	0,528	T09-DT
	log4j-1.1	0,571	T09-DT
	log4j-1.2	0,371	W08-DT
	lucene-2.0	0,492	T09-DT
	lucene-2.2	0,368	T09-DT
	lucene-2.4	0,426	T09-DT
	nieruchomosci	0,480	W08-DT
	pbeans1	0,687	W08-DT

Continua na próxima página

Tabela 21 – Melhor valor F1-score e respectivo melhor método CPDP por projeto. – continuação da página anterior

Conjunto de Dados	Projeto	Melhor F1-score	Melhor(es) Método(s)
	pbeans2	0,500	CC09-DT
	pdftranslator	0,741	CC09-DT
	poi-1.5	0,490	T09-DT
	poi-2.0	0,291	W08-DT
	poi-2.5	0,531	T09-DT
	poi-3.0	0,681	T09-DT
	redaktor	0,429	CC09-DT
	serapion	0,556	CC09-DT
	skarbonka	0,421	M11-RF
	sklebagd	0,783	CC09-DT
	synapse-1.0	0,367	M11-RF
	synapse-1.1	0,468	CC09-DT
	synapse-1.2	0,461	M11-RF
	systemdata	0,462	EASC
	szybkafucha	0,435	CC09-DT
	termoproject	0,710	T09-DT
	tomcat	0,384	EASC
	velocity-1.4	0,483	W08-DT
	velocity-1.5	0,474	W08-DT
	velocity-1.6	0,453	W08-DT
	workflow	0,486	W08-DT
	wspomaganiepi	0,556	T09-DT
	xalan-2.4	0,385	DSSC

Continua na próxima página

Tabela 21 – Melhor valor F1-score e respectivo melhor método CPDP por projeto. – continuação da página anterior

Conjunto de Dados	Projeto	Melhor F1-score	Melhor(es) Método(s)
	xalan-2.5	0,441	T09-DT
	xalan-2.6	0,483	W08-DT
	xalan-2.7	0,450	T09-DT
	xerces-1.2	0,293	M11-RF
	xerces-1.3	0,440	T09-DT
	xerces-1.4	0,521	CC09-DT
	xerces-init	0,426	W08-DT
	zuzel	0,571	M11-RF
RELINK	Apache2.0	0,746	DSSC
	openintents	0,698	W08-DT
	zxing1.6	0,473	M11-RF
NASA	CM1	0,318	EASC
	JM1	0,259	EASC
	KC1	0,131	M11-RF
	KC3	0,269	EASC
	MC1	0,148	W08-DT
	MC2	0,393	EASC
	MW1	0,269	T09-DT
	PC1	0,284	EASC
	PC2	0,140	M11-RF
	PC3	0,247	EASC
	PC4	0,101	EASC
	PC5	0,384	EASC

Continua na próxima página

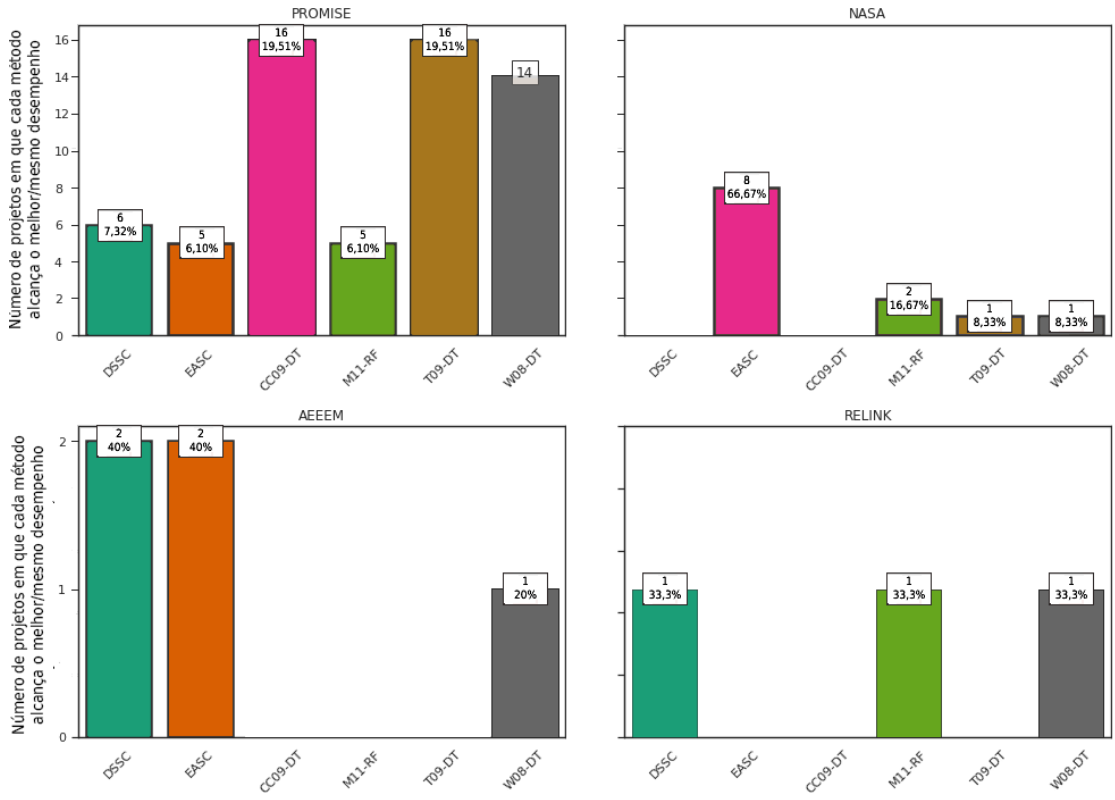
Tabela 21 – Melhor valor F1-score e respectivo melhor método CPDP por projeto. – continuação da página anterior

Conjunto de Dados	Projeto	Melhor F1-score	Melhor(es) Método(s)

Fonte: Elaborada pelo autor (2022)

Além disso, com base nos resultados apresentados na Tabela 21, a Figura 10 mostra, separadamente por conjunto de dados, a quantidade de projetos que cada método CPDP obteve a maior F1-score. Nesta análise, em cada subfigura apresentada conta com números totais de projetos e também a porcentagem referente a quantidade total de projetos por conjunto de dados. Por exemplo, para o conjunto dados PROMISE o DSSC obteve em 6 projetos a pontuação máxima (F1-score), equivalente à 7,32% dos 82 projetos ao todo pertencente ao PROMISE; a contagem não leva em conta empates, portanto, há divisão de pontuação como apresentado na Figura 9. Para mais detalhes sobre quais projetos cada método obteve o melhor desempenho, no Apêndice B, apresenta os resultados utilizados para computar os gráficos.

Figura 10 – Gráfico de barras mostrando o número de projetos por conjuntos de dados em que cada abordagem CPDP apresentou o maior/mesmo F1-score.



Fonte: Elaborada pelo autor (2022)

5.2.2 Comparação com AUC.

Na Tabela 22 apresenta os resultado em forma de média±variância e da comparação do DSSC com os demais métodos CPDP para cada conjunto de dados em relação a AUC. Os melhores resultados são destacados em negrito e os resultados significativamente diferentes e inferiores ao DSSC são marcados com um ∘ (*p-valor* $\leq 0,05$), onde o *p-valor* é obtido pelo teste Wilcoxon Signed Rank. Além disso, é apresentado a diferença de acordo com o Cliff's delta para os casos significativamente inferiores ao DSSC. Conforme a Tabela 22, o DSSC obtém os melhores resultados nos conjuntos de dados AEEEM e PROMISE. Entre as cinco comparações, isto é, DSSC versus cinco métodos CPDP selecionados, em NASA e RELINK a abordagem proposta obtém $\frac{4}{5}$ dos melhores resultados médios. Para os casos com diferença estatística comprovadas pelo teste Wilcoxon (*p-valor* $\leq$ 0,05), pode-se notar que, de acordo com o Cliff's delta (letra em parênteses), há uma larga/média diferença para os métodos inferiores ao DSSC, logo, mostrando uma clara superioridade do DSSC.

Tabela 22 – Comparação entre DSSC e métodos CPDP supervisionados em quatro conjuntos de dados em termos de AUC na forma de média±variância.

	AEEEM	NASA	PROMISE	RELINK
DSSC	**0,747±0,003**	0,739±0,008	**0,724±0,013**	0,739±0,005
EASC-NB	0,693±0,006	**0,760±0,005**	0,716±0,013	**0,744±0,006**
CC09-DT	0,604±0,005	0,697±0,013	0,582±0,010∘(L)	0,648±0,001
M11-RF	0,578±0,001∘(L)	0,541±0,001∘(L)	0,574±0,005∘(L)	0,653±0,001
T09-DT	0,530±0,000∘(L)	0,622±0,004∘(L)	0,588±0,012∘(L)	0,635±0,006
W08-DT	0,588±0,004∘(L)	0,669±0,009∘(M)	0,593±0,008∘(L)	0,602±0,011

Notas: (1) ∘: indica desempenho significativamente inferior ao DSSC (*p-value* $\leq 0,05$).

(2) L/M/S: Large/Medium/Small diferença de acordo com o Cliff's delta.

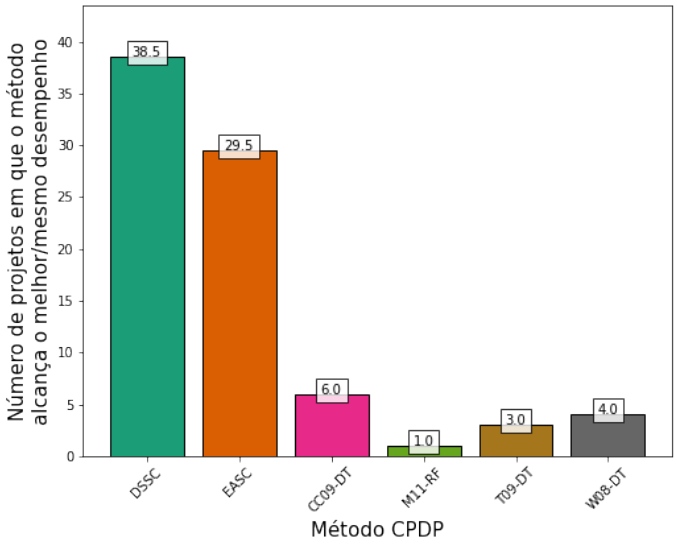
Fonte: Elaborada pelo autor (2022)

Além das análises e comparações estatísticas em pares em cada conjunto de dados, a Figura 11 apresenta a quantidade de projetos em que cada método obteve a melhor/mesma performance. Para cada vitória em um dos 82 projetos, este método recebe 1 ponto; em caso de empate entre dois ou mais métodos, cada um deles recebem a pontuação de $\frac{1}{n}$, onde n representa o número de métodos CPDP que obtiveram o maior desempenho. Conforme os resultados, o DSSC obtém a maior pontuação (38,5), equivalente a 38 vitórias e 1 empate correspondendo a 47,56% dos 82 projetos, logo, superando as demais abordagens.

Também avaliamos o desempenho dos métodos para cada um dos 82 projetos. Então,

associamos a cada projeto o método CPDP que obteve a melhor AUC. Essa avaliação é apresentada na Tabela 23. Podemos extrair algumas informações dessa tabela. Primeiro, o melhor método CPDP para uma determinada versão do projeto não é necessariamente o mesmo para todas as versões do mesmo projeto. Segundo, mais de uma abordagem CPDP pode obter a melhor performance AUC para o mesmo projeto. E, terceiro, para todas as 82 versões: 62 (75,61%) resultados alcançou AUC $\geq$ 0,7 — equivalendo a bons e/ou excelentes desempenhos de acordo com as escalas de valores AUC abordados por Jr, Lemeshow e Sturdivant (2013); e 20 (24,39%) resultados com AUC menor que 0,7 (dado como ruim e/ou muito ruim). Além disso, o DSSC apresentou 31 (maior quantidade) dos melhores resultados com AUC $\geq$ 0,7 e o menor número de projetos com AUC $<$ 0,7 (5 projetos).

Figura 11 – Gráfico de barras mostrando o número de projetos em que cada abordagem CPDP apresentou o maior/mesmo AUC.



Fonte: Elaborada pelo autor (2022)

Tabela 23 – Melhor valor AUC e respectivo melhor método CPDP por projeto. Em negrito são destacados os resultados com AUC $\geq$ 0,7.

Conjunto de Dados	Projeto	Melhor F1-score	Melhor(es) Método(s)
AEEEM	eclipse	**0,829**	DSSC
	equinox	**0,738**	EASC
	lucene	**0,770**	DSSC
	mylyn	0,689	DSSC

Continua na próxima página

Tabela 23 – Melhor valor AUC e respectivo melhor método CPDP por projeto. Em negrito são destacados os resultados com AUC $\geq$ 0,7. – continuação da página anterior

Conjunto de Dados	Projeto	Melhor F1-score	Melhor(es) Método(s)
	pde	**0,747**	DSSC
NASA	CM1	**0,745**	DSSC
	JM1	0,694	EASC
	KC1	**0,793**	EASC
	KC3	**0,725**	W08-DT
	MC1	**0,894**	DSSC
	MC2	**0,708**	EASC
	MW1	**0,793**	DSSC
	PC1	**0,755**	EASC
	PC2	**0,894**	CC09-DT
	PC3	**0,749**	EASC
	PC4	0,660	EASC
	PC5	**0,902**	DSSC
	ant-1.3	**0,762**	DSSC
	ant-1.4	0,556	W08-DT
	ant-1.5	**0,768**	EASC
	ant-1.6	**0,791**	DSSC
	ant-1.7	**0,779**	DSSC, EASC
	arc	0,690	CC09-DT
	berek	**0,986**	DSSC
	camel-1.0	**0,802**	EASC
	camel-1.2	0,585	T09-DT
	camel-1.4	0,685	DSSC

Continua na próxima página

Tabela 23 – Melhor valor AUC e respectivo melhor método CPDP por projeto. Em negrito são destacados os resultados com AUC $\geq$ 0,7. – continuação da página anterior

Conjunto de Dados	Projeto	Melhor F1-score	Melhor(es) Método(s)
	camel-1.6	0,621	EASC
	ckjm	**0,880**	DSSC
	e-learning	**0,803**	DSSC
	forrest-0.7	**0,812**	T09-DT
	ivy-1.1	**0,730**	DSSC
	ivy-1.4	**0,752**	DSSC
	ivy-2.0	**0,800**	EASC
	jedit-3.2	**0,780**	DSSC, EASC
	jedit-4.0	**0,749**	DSSC
	jedit-4.1	**0,789**	DSSC
	jedit-4.2	**0,821**	EASC
	jedit-4.3	0,593	DSSC
	kalkulator	**0,726**	CC09-DT
	log4j-1.0	**0,796**	EASC
	log4j-1.1	**0,797**	DSSC
	log4j-1.2	0,616	EASC
	lucene-2.0	**0,721**	DSSC
	lucene-2.2	0,630	DSSC, EASC
	lucene-2.4	**0,714**	EASC
	nieruchomosci	**0,918**	EASC
	pbeans1	**0,842**	EASC
	pbeans2	**0,763**	EASC
	pdftranslator	**0,900**	DSSC

Continua na próxima página

Tabela 23 – Melhor valor AUC e respectivo melhor método CPDP por projeto. Em negrito são destacados os resultados com AUC $\geq$ 0,7. – continuação da página anterior

Conjunto de Dados	Projeto	Melhor F1-score	Melhor(es) Método(s)
	poi-1.5	**0,747**	EASC
	poi-2.0	0,622	EASC
	poi-2.5	**0,765**	DSSC
	poi-3.0	**0,805**	DSSC
	redaktor	**0,720**	CC09-DT
	serapion	**0,843**	DSSC, EASC
	skarbonka	**0,728**	DSSC, EASC
	sklebagd	**0,880**	W08-DT
	synapse-1.0	**0,818**	DSSC
	synapse-1.1	**0,700**	EASC
	synapse-1.2	**0,742**	EASC
	systemdata	**0,760**	T09-DT
	szybkafucha	0,662	EASC
	termoproject	**0,897**	DSSC
	tomcat	**0,811**	DSSC
	velocity-1.4	0,612	CC09-DT
	velocity-1.5	0,669	EASC
	velocity-1.6	**0,711**	DSSC
	workflow	**0,795**	DSSC
	wspomaganiepi	**0,903**	DSSC
	xalan-2.4	**0,741**	DSSC
	xalan-2.5	0,624	CC09-DT
	xalan-2.6	0,651	W08-DT

Continua na próxima página

Tabela 23 – Melhor valor AUC e respectivo melhor método CPDP por projeto. Em negrito são destacados os resultados com AUC $\geq$ 0,7. – continuação da página anterior

Conjunto de Dados	Projeto	Melhor F1-score	Melhor(es) Método(s)
	xalan-2.7	**0,749**	DSSC
	xerces-1.2	0,577	M11-RF
	xerces-1.3	**0,781**	EASC
	xerces-1.4	**0,734**	EASC
	xerces-init	0,520	DSSC
	zuzel	**0,853**	DSSC
RELINK	Apache2.0	**0,781**	DSSC
	openintents	**0,840**	EASC
	zxing1.6	0,660	DSSC

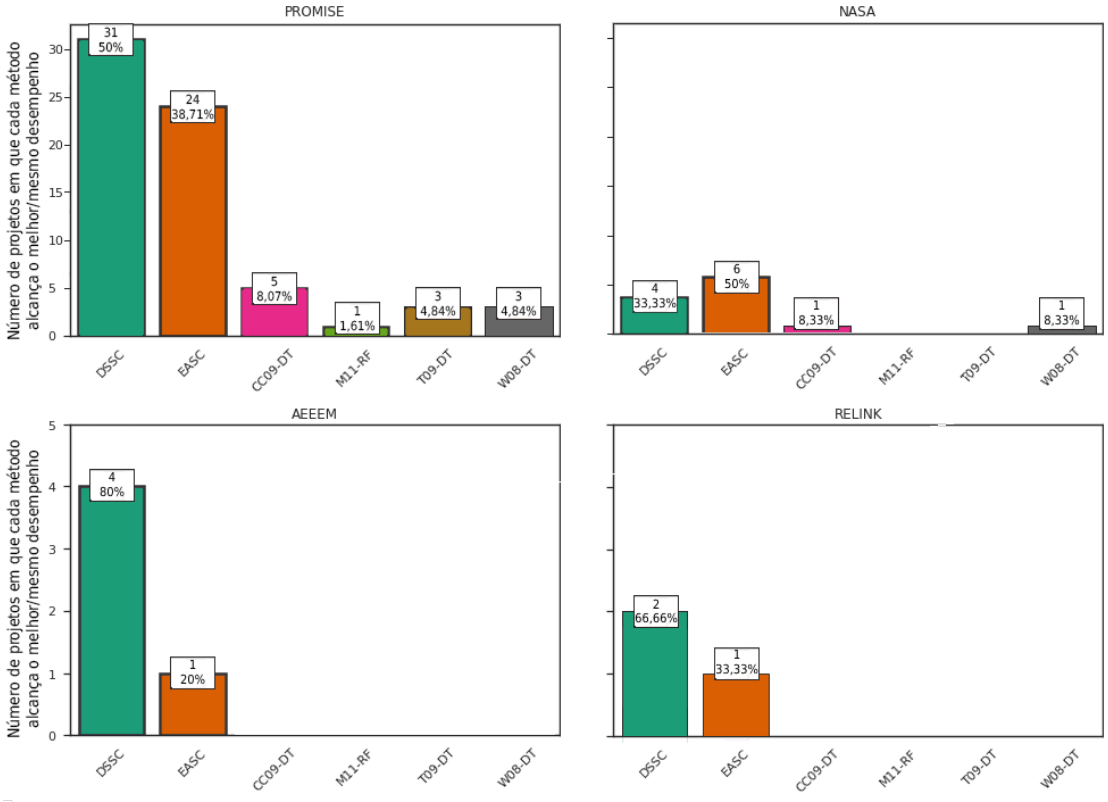
Fonte: Elaborada pelo autor (2022)

Para melhor ilustrar os resultados apresentados na Tabela 23, a Figura 12 mostra, por conjunto de dados, a distribuição de projetos em que cada método CPDP obteve a maior AUC (desconsiderando a divisão de pontuação em caso de empate). De acordo com os valores totais e as porcentagens apresentadas, o DSSC obteve a maior pontuação em relação aos demais métodos, com 31 projetos do PROMISE (50%), 4 do AEEEM (80%) e 2 do RELINK (66,66%). Para o conjunto de dados NASA obteve a segunda melhor classificação com 4 projetos dos 12 (33,33%). Para mais detalhes, no Apêndice C.2 são apresentados os resultados de cada método CPDP $\times$ projetos.

5.2.3 Comparação com *False Alarm*.

Na Tabela 24 apresenta os resultados em forma de média$\pm$variância dos métodos CPDP nos conjuntos de dados em relação a *False Alarm*. Para cada conjunto de dados, os melhores resultados são destacados em negrito e os resultados significativamente diferentes e inferiores ao DSSC são marcados com $\circ$ (*p-valor* $\leq 0,05$), onde o *p-valor* é obtido pelo teste Wilcoxon Signed Rank. Conta também com o índice de diferença (Cliff's delta) entre os resultados que

Figura 12 – Gráfico de barras mostrando o número de projetos para cada conjunto de dados em que cada abordagem CPDP apresentou o maior/mesmo AUC.



Fonte: Elaborada pelo autor (2022)

são estatisticamente inferiores ao DSSC. Conforme os resultados, o DSSC obtém os melhores resultados nos conjuntos de dados AEEEM, PROMISE e RELINK. Para NASA obtém $\frac{4}{5}$ entre os melhores índices de *False Alarm*. Para os casos com diferença estatística comprovadas pelo teste Wilcoxon (*p-valor* $\leq 0{,}05$), pode-se notar que, de acordo com o Cliff's delta (letra em parênteses), há uma larga diferença entre o DSSC e os métodos significativamente inferiores, logo, mostrando uma clara superioridade do DSSC.

Na Figura 13 mostra a quantidade total de vitórias/empates de cada método CPDP, com 82 pontos ao total de acordo com a quantidade de projetos. Com uma diferença de pontos claramente superior, o DSSC obteve a melhor pontuação (57,18) equivalente à 46 vitórias e 27 empates, o que totaliza 89,02% de todos os projetos (73 projetos de 82). Os resultados para essa análise são apresentados na Tabela 25.

Tabela 24 – Comparação entre DSSC e métodos CPDP supervisionados em quatro conjuntos de dados em termos de *False Alarm* na forma de média±variância.

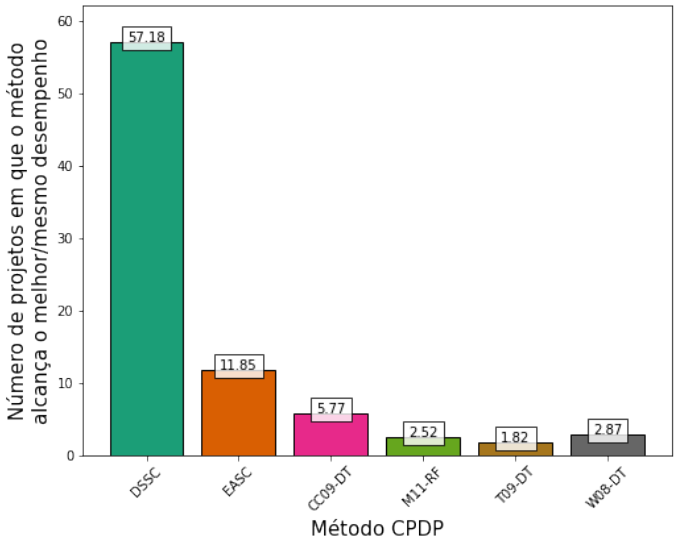
	AEEEM	NASA	PROMISE	RELINK
DSSC	**0,021±0,000**	0,016±0,000	**0,050±0,002**	**0,114±0,010**
EASC-NB	0,176±0,021∘(L)	0,045±0,001∘(L)	0,070±0,004∘(L)	0,181±0,065
CC09-DT	0,063±0,001	**0,013±0,000**	0,203±0,012∘(L)	0,212±0,006
M11-RF	0,047±0,002	0,034±0,001	0,171±0,008∘(L)	0,211±0,002
T09-DT	0,131±0,009∘(L)	0,052±0,002∘(L)	0,178±0,001∘(L)	0,167±0,015
W08-DT	0,109±0,002∘(L)	0,019±0,000	0,262±0,019∘(L)	0,168±0,004

Notas: (1) ∘: indica desempenho significativamente inferior ao DSSC (*p-value* $\leq 0,05$).

(2) L/M/S: Large/Medium/Small diferença de acordo com o Cliff's delta.

Fonte: Elaborada pelo autor (2022)

Figura 13 – Gráfico de barras mostrando o número de projetos em que cada abordagem CPDP apresentou o maior/mesmo *False Alarm*.



Fonte: Elaborada pelo autor (2022)

Tabela 25 – Melhor valor de *False Alarm* e respectivo melhor método CPDP por projeto. Em negrito são destacados os melhores resultado quando obtidos pelo DSSC.

Conjunto de Dados	**Projeto**	**Melhor *False Alarm***	**Melhor(es) Método(s)**
AEEEM	eclipse	**0,029**	DSSC
	equinox	**0,005**	DSSC, CC09-DT
	lucene	**0,002**	DSSC
	mylyn	0,014	M11-RF

Continua na próxima página

Tabela 25 – Melhor valor AUC e respectivo melhor método CPDP por projeto. Em negrito são destacados os resultados com AUC $\geq$ 0,7. – continuação da página anterior

Conjunto de Dados	Projeto	Melhor *False Alarm*	Melhor(es) Método(s)
	pde	**0,037**	DSSC
NASA	CM1	0,007	CC09-DT
	JM1	0,006	CC09-DT
	KC1	0,007	EASC
	KC3	**0,006**	DSSC, CC09-DT
	MC1	**0,004**	DSSC
	MC2	0,000	W08-DT
	MW1	**0,000**	DSSC
	PC1	**0,011**	DSSC, CC09-DT
	PC2	**0,003**	DSSC
	PC3	0,012	CC09-DT
	PC4	**0,006**	DSSC
	PC5	**0,004**	DSSC, T09-DT
	ant-1.3	**0,067**	DSSC
	ant-1.4	**0,109**	DSSC
	ant-1.5	**0,100**	DSSC
	ant-1.6	**0,085**	DSSC
	ant-1.7	**0,064**	DSSC
	arc	**0,048**	DSSC
	berek	**0,000**	DSSC, EASC
	camel-1.0	**0,031**	DSSC
	camel-1.2	**0,033**	DSSC
	camel-1.4	**0,073**	DSSC

Continua na próxima página

Tabela 25 – Melhor valor AUC e respectivo melhor método CPDP por projeto. Em negrito são destacados os resultados com AUC $\geq$ 0,7. – continuação da página anterior

Conjunto de Dados	Projeto	Melhor *False Alarm*	Melhor(es) Método(s)
PROMISE	camel-1.6	**0,068**	DSSC
	ckjm	**0,000**	DSSC, EASC, CC09-DT, M11-RF, T09-DT
	e-learning	**0,000**	DSSC, EASC
	forrest-0.7	**0,000**	DSSC, EASC
	ivy-1.1	**0,000**	DSSC, EASC
	ivy-1.4	**0,076**	DSSC
	ivy-2.0	**0,087**	DSSC
	jedit-3.2	**0,082**	DSSC
	jedit-4.0	**0,113**	DSSC
	jedit-4.1	**0,094**	DSSC
	jedit-4.2	**0,157**	DSSC
	jedit-4.3	**0,187**	DSSC
	kalkulator	**0,095**	DSSC, CC09-DT, W08-DT
	log4j-1.0	**0,010**	DSSC
	log4j-1.1	**0,014**	DSSC, EASC
	log4j-1.2	**0,062**	DSSC, EASC, M11-RF
	lucene-2.0	**0,000**	DSSC
	lucene-2.2	**0,019**	DSSC
	lucene-2.4	**0,036**	DSSC
	nieruchomosci	**0,000**	DSSC, EASC, T09-DT
	pbeans1	**0,000**	DSSC, EASC, CC09-DT, M11-RF, T09-DT

Continua na próxima página

Tabela 25 – Melhor valor AUC e respectivo melhor método CPDP por projeto. Em negrito são destacados os resultados com AUC $\geq$ 0,7. – continuação da página anterior

Conjunto de Dados	Projeto	Melhor *False Alarm*	Melhor(es) Método(s)
	pbeans2	**0,024**	DSSC, EASC
	pdftranslator	**0,000**	DSSC, EASC, M11-RF, T09-DT
	poi-1.5	**0,031**	DSSC
	poi-2.0	**0,072**	DSSC
	poi-2.5	**0,073**	DSSC
	poi-3.0	**0,050**	DSSC, EASC
	redaktor	**0,034**	DSSC
	serapion	**0,000**	DSSC, EASC
	skarbonka	**0,028**	DSSC
	sklebagd	**0,000**	DSSC, EASC, W08-DT
	synapse-1.0	**0,028**	DSSC
	synapse-1.1	**0,012**	DSSC
	synapse-1.2	**0,035**	DSSC
	systemdata	**0,018**	DSSC, EASC
	szybkafucha	**0,000**	DSSC, EASC, M11-RF
	termoproject	**0,000**	DSSC, EASC
	tomcat	**0,105**	DSSC
	velocity-1.4	**0,143**	DSSC
	velocity-1.5	**0,042**	DSSC
	velocity-1.6	**0,026**	DSSC
	workflow	**0,000**	DSSC, EASC, T09-DT
	wspomaganiepi	**0,000**	DSSC, EASC, CC09-DT

Continua na próxima página

Tabela 25 – Melhor valor AUC e respectivo melhor método CPDP por projeto. Em negrito são destacados os resultados com AUC $\geq$ 0,7. – continuação da página anterior

Conjunto de Dados	Projeto	Melhor *False Alarm*	Melhor(es) Método(s)
	xalan-2.4	**0,135**	DSSC
	xalan-2.5	**0,115**	DSSC
	xalan-2.6	**0,089**	DSSC
	xalan-2.7	**0,000**	DSSC, EASC, CC09-DT, M11-RF, W08-DT
	xerces-1.2	**0,092**	DSSC
	xerces-1.3	**0,060**	DSSC
	xerces-1.4	0,020	EASC
	xerces-init	**0,129**	DSSC
	zuzel	**0,000**	DSSC, EASC
RELINK	Apache2.0	0,083	W08-DT
	openintents	**0,000**	DSSC, EASC
	zxing1.6	0,000	EASC

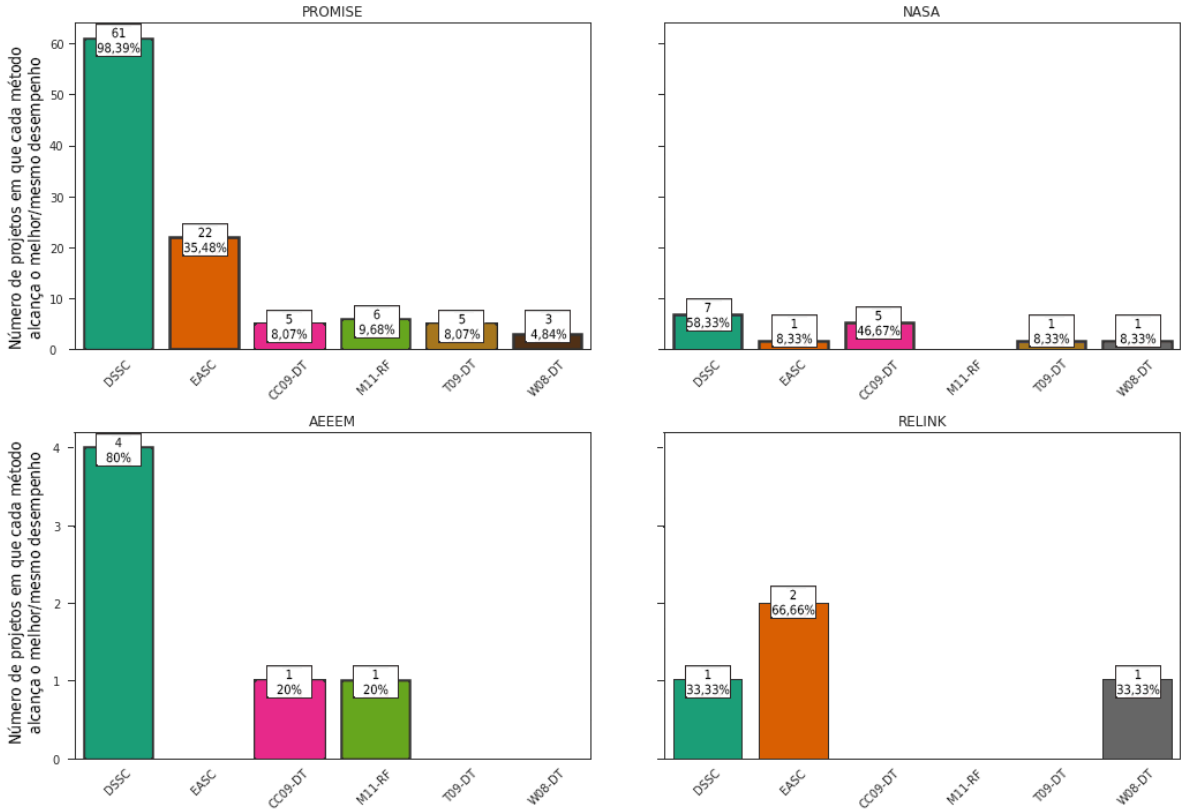
Fonte: Elaborada pelo autor (2022)

Ainda sobre a Tabela 25 observa-se alguns destaques em relação aos desempenhos do DSSC: apresentou $\frac{4}{5}$ dos melhores índices de *False Alarm* (menor valor) para o conjunto de dados AEEEM, $\frac{61}{62}$ melhores desempenhos no PROMISE, obteve $\frac{18}{19}$ dos melhores índices de *False Alarm*, isto é, $PF = 0$, além disso, obteve $\frac{74}{82}$ dos melhores resultados.

Por fim, para melhor visualizar a distribuição de vitórias por conjunto de dados, a Figura 14 apresenta a quantia dos projetos em que cada método CPDP obteve o melhor índice de *False Alarm* (desconsiderando a divisão de pontuação em caso de empate). Os resultados mostram a superioridade do DSSC em relação aos demais métodos para os conjuntos de dados AEEEM, PROMISE e RELINK. Portanto, para o problema mais significativo associado à detecção de defeitos em softwares, concluí-se que a estrutura proposta utilizando métodos de seleção dinâmica(DSSC), apresenta baixos índices de alarmes falsos, isto é, quantidade de

vezes que a abordagem erroneamente classifica projetos não defeituosos como defeituosos. Os resultados gerais, por método CPDP, utilizados para esta análise estão disponíveis no Apêndice C.3, lá são apresentados os resultados de cada método CPDP $\times$ projeto.

Figura 14 – Gráfico de barras mostrando o número total de projetos por conjunto de dados em que cada abordagem CPDP obteve o melhor índice de *False Alarm*.



Fonte: Elaborada pelo autor (2022)

5.3 QUALIDADE DOS MÉTODOS CPDP

Para calcular e comparar a precisão de classificação dos métodos CPDP, os resultados dos métodos em relação à área sob a curva ROC AUC (HANLEY; MCNEIL, 1982) foram analisados. A curva ROC AUC é interpretada através da representação gráfica da taxa de falsos positivos (1 - especificidade, onde especificidade é a proporção de predições incorretas) e a taxa de verdadeiros positivos (referida como sensibilidade ou a proporção de predições corretas). Uma classificação perfeita terá uma AUC de 1,0 (sem falsos positivos e falsos negativos), enquanto 0,5 indica um modelo falho. Jr, Lemeshow e Sturdivant (2013) propuseram algumas escalas para os valores de AUC que descrevem a qualidade do modelo em geral. Portanto, os valores de AUC nunca devem ser iguais ou menores que 0,5. Por exemplo, se uma AUC de 0,0 for obtida

como resultado de uma classificação (ou seja, falho ou perfeitamente ruim), significa que o modelo está sempre predizendo incorretamente 100% das vezes. Para AUC $\leq$ 0,5 representa aleatoriedade durante a classificação, ou seja, o modelo classifica aleatoriamente uma amostra como sendo da classe 0 ou 1. Para 0,5 $<$ AUC $<$ 0,7 indica baixa capacidade de discriminação porque a taxa de sensibilidade não é muito maior do que a taxa de falsos positivos. Para 0,7 $\leq$ AUC $<$ 0,8 o modelo possui boa discriminação. Valores entre 0,8 e 0,9 indicam que o modelo possui capacidade de discriminação muito boa e taxas AUC superiores a 0,9 indicam discriminação excelente devido a taxa de sensibilidade ser alta em relação à taxa de falsos positivos. A Figura 15 representa as curvas e áreas que correspondem aos "rótulos de qualidade AUC". Os modelos com suas respectivas taxas AUC e áreas podem ser descritos da seguinte forma:

- AUC $\leq$ 0,5: corresponde a um modelo muito ruim, sem discriminação (área sob a linha tracejada vermelha, área E);
- 0,5 $<$ AUC $<$ 0,7: significa pouca discriminação, modelo ruim (área D);
- 0,7 $\leq$ AUC $<$ 0,8: define um modelo com bom desempenho (área C);
- 0,8 $\leq$ AUC $<$ 0,9: modelo com discriminação muito boa (área B);
- AUC $\geq$ 0,9: modelo com classificação excelente (área A);

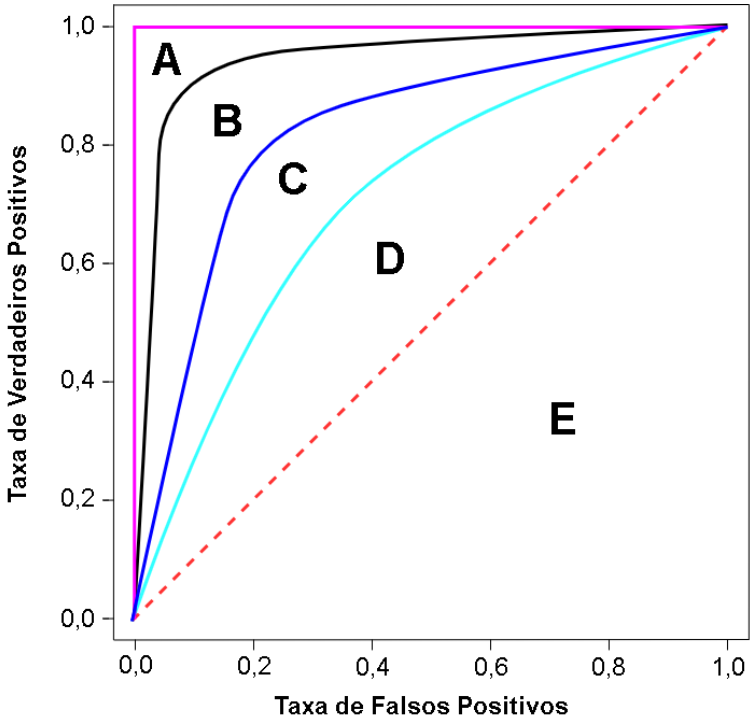
Os desempenhos dos método CPDP foram classificados de acordo com cada escala de qualidade. A Tabela 26 mostra a contabilidade dos desempenhos dos métodos CPDP para 82 projetos conforme às categorias de qualidade.

Tabela 26 – Contabilidade de qualidade para os métodos CPDP. As colunas representam os totais de desempenhos que foram categorizados de acordo com o escala AUC.

Método CPDP	Muito Ruim	Ruim	Bom	Muito bom	Excelente
DSSC	3	26	36	13	4
EASC-NB	3	24	40	13	2
CC09-DT	15	54	9	4	0
M11-RF	9	71	1	1	0
T09-DT	13	58	9	2	0
W08-DT	7	66	7	2	0

Fonte: Elaborada pelo autor (2022)

Figura 15 – Escalar de qualidade de curvas AUC.



Fonte: Elaborada pelo autor (2022)

Conforme apresentado, o DSSC apresenta alguns destaques em relação aos demais métodos: obteve a menor quantidade de resultados classificados como "Muito Ruim"(junto com o EASC-NB), apenas $\frac{3}{82}$, equivalente a 3,66% de todos os projetos; o segundo menor índice de desempenhos ruins, sendo apenas 26 de 82 projetos (31,71%) com $0{,}5 < \text{AUC} < 0{,}7$; a segunda maior quantidade de projetos com desempenho classificado como bom, 43,90% dos projetos; melhor número de desempenhos classificadores como "Muito Bom", 15,85% dos projetos com $\text{AUC} > 0{,}9$; e o maior índice de projetos com excelentes desempenhos.

Além disso, podemos extrair algumas informações dos resultados médios apresentados na Tabela 22 (seção 5.2.2) ao analisar em conjunto com os conceitos/escalas de qualidade, logo:

- O melhor caso é apresentado pelo DSSC, ao qual obteve médias ROC-AUC acima de 0,7 para todos os conjuntos de dados, classificando-o como um modelo preditivo com boa discriminação;
- O segundo melhor caso é apresentado pelo EASC-NB, com $\frac{3}{4}$ resultados médios categorizados como desempenho bom e 1 como ruim;
- Para os métodos CC09-DT, M11-RF, T09-DT e W08-DT, ambos apresentam os piores desempenhos médios (AUC entre 0,530 e 0,697) resultando em modelos com pouca

discriminação (modelos ruins).

5.4 QUANDO USAR O MÉTODO DSSC

Construímos um meta-classificador para verificar quais características/descrições dos projetos são mais discriminantes para a escolha do método DSSC. Cada exemplo (consultar tabelas disponíveis no Apêndice C), é representado por cinco descrições (*meta-features*), tais como: número de instâncias, número de dimensões, porcentagem de instâncias com defeito e o desbalanceamento de classes calculada pela Equação 5.1 (ALMEIDA, 2014):

$$Desbalanceamento = \sum_{\omega \in \mathbf{\Omega}} \left| \frac{1}{L} - \frac{n_{\omega_l}}{N} \right| \tag{5.1}$$

sendo ω_l uma determinada classe do projeto $\mathbf{\Omega}$ com L classes, n_{ω_l} o total de instâncias da classe ω_l e N o total de instâncias do projeto.

Para cada meta-feature dos projetos foi calculada sua importância no desempenho dos métodos CPDP com relação ao parâmetro de interesse (métrica de avaliação). A descoberta da importância de cada descrição foi obtida através de um meta-classificador. Para o meta-classificador foi utilizada a árvore de decisão por ser um dos modelos mais utilizados para extração de conhecimento através das regras de classificação. Na Tabela 27 apresenta o resultado da análise de importância de cada característica dos projetos com base nas métricas de avaliação. Para as células tracejadas (—) representam descrições com importância igual a 0.

Tabela 27 – Análise da importância de cada descrição.

	Importância por Métrica		
Descrição dos Projetos	**F1-score**	**AUC**	**PF**
Número de instâncias	—	—	0,674
Número de dimensões	0,520	—	—
Porcentagem de instâncias com defeito	0,480	0,498	0,326
Desbalanceamento de classes	—	0,503	—

Fonte: Elaborada pelo autor (2022)

A partir da Tabela 27 obversa-se que as descrições mais relevantes são: porcentagem de instâncias com defeito, número de instâncias, número de dimensões e desbalanceamento de classes. Além disso, nota-se que a porcentagem de instâncias com defeito há influência no desempenho dos métodos para todas as métricas de avaliação. Essas descrições são os

principais fatores que definem qual método CPDP deve ser utilizado, pois estão relacionados com o desempenho e com determinada métrica de avaliação, e portanto, influenciando no resultado final dos métodos CPDP.

Foram geradas árvores de decisão para ilustrar a classificação da melhor abordagem CPDP a ser utilizada a partir das descrições dos projetos. No Apêndice C são listadas diversas tabelas contendo os dados utilizados para gerar as árvore. Em cada tabela são apresentados os valores de cada meta-feature com base nos dados originais dos conjuntos de dados. Além disso, as tabelas possuem uma coluna denominada "Classe"na qual o rótulo para a classificação do problema é a melhor abordagem CPDP, isto é, classe 1 corresponde ao método DSSC, e classe 0 (zero) não DSSC. A Figura 16 mostra as árvores de decisão e suas respectivas regras de decisão.

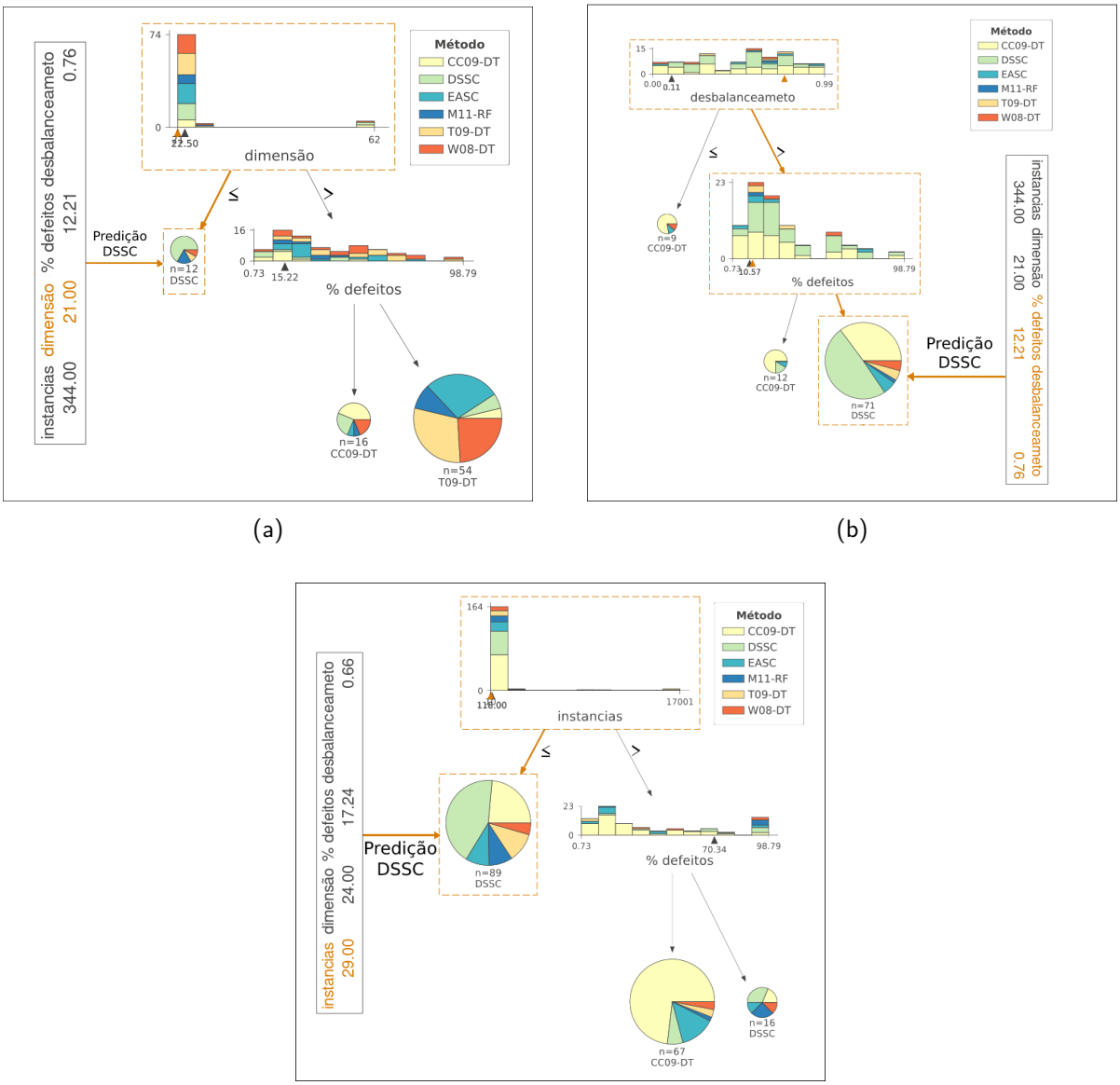
De acordo com a análise do meta-classificador (Figura 16), algumas informações podem ser extraídas: em cada nó, podemos ver um histograma empilhado do recurso (descrição) que é utilizado para dividir as observações, colorido por rótulo (métodos CPDP). O pequeno triângulo preto (▲) com o valor no eixo x representa o ponto de divisão. Na Figura 16a, primeiro histograma, podemos ver claramente todas as observações de cada classe. Os ramos direito da árvore indicam a seleção dos valores maiores ou iguais ao valor de divisão, enquanto o ramo esquerdo é inferior ao valor de divisão. Os nós folhas são representados como gráficos de pizza, que mostram a fração das observações dentro da folha pertence a uma determinada classe. Dessa forma, é possível ver facilmente qual classe é a maioria, assim como a predição do modelo. Outra característica importante é o destaque de um determinado ponto de observação (triângulo laranja). Esse ponto representa o valor indicado de uma dada descrição. Além disso, observa-se o destaque em laranja mostrando qual o caminho a análise seguiu. No final, são apresentados os valores de todas as características, sendo as utilizadas para decisão destacadas em laranja.

Por fim, conforme ilustrado na Figura 16, as árvores apresentadas sugerem regras para decidir qual método deve ser escolhido, logo:

- Para F1-score, apenas uma característica foi utilizada para predizer que a observação pertence à classe DSSC, isto é, número de dimensões. Logo, o DSSC é melhor nos projetos que possuem em média número de dimensões menores que 22,50, além disso, o ponto ideal é dado como sendo 21 ou menos dimensões.

- Para AUC, duas características influenciam positivamente em caso de valores superiores

Figura 16 – Meta-classificador e regras de decisão para escolha da melhor abordagem CPDP. (a) regras de decisão em relação à F1-score, (b) em relação à AUC e (c) em relação à False Alarm.



(a) (b)

(c)

Fonte: Elaborada pelo autor (2022)

aos pontos de divisão das descrições: desbalanceamento de classes e porcentagem de instâncias com defeito. Ainda, é possível notar com clareza a diferença de vitórias caso o DSSC seja aplicado usando essas regras de decisão.

- Para *False Alarm* (PF), a regra de decisão escolhida é aplicar o DSSC em projetos que possuem menos de 29 instâncias.

- Por fim, para todas as métricas, há relação entre a porcentagem de instâncias defeituosas no desempenho do DSSC, tendo maior impacto quando avaliado usando AUC.

5.5 CONSIDERAÇÕES FINAIS

Neste Capítulo foi apresentada a metodologia de pesquisa e o resultado experimental com o objetivo de responder os pontos levantados.

No primeiro ponto, comportamento da estrutura proposta (DSSC) no cenário CPDP, foram realizadas análises e uso de ilustrações com objetivo de verificar qual a melhor configuração de parâmetros ou quais componentes que mais influenciam na abordagem proposta durante a fase de geração e seleção do modelo, tais como: métodos de seleção dinâmica, classificadores base e tamanho do *pool* de classificadores. Na primeira análise sobre as técnicas de seleção dinâmica, foi concluído que há não há diferença clara entre a quantidade de vezes que cada técnica foi selecionada. Quanto aos classificadores base, foi visto que a escolha do classificador está diretamente ligada aos tipos de dados usados no treinamento, e portanto, o *Naive Bayes* apresentou maior influência no conjunto de dados PROMISE sendo selecionado 33 vezes, seguido da *Logistic Regression* com 29 seleções no PROMISE e 3 no RELINK (total 32), *Random Forest* selecionada para todos os projetos de NASA e de AEEEM, enquanto *Decision Tree* e *Support Vector Machine* não foram escolhidas nenhuma vez. Além disso, também foi analisado quais foram as melhores quantidades de classificadores para gerar o *pool* de classificadores, e concluiu-se que não há um tamanho ideal do *pool*. Portanto, neste primeiro ponto, dado que não há como definir uma configuração (técnica de seleção dinâmica $\times$ classificador base $\times$ tamanho do *pool* de classificadores) que maximize o desempenho da abordagem DSSC em todos os tipos de dados, conclui-se que utilizar as n configurações possíveis fará com que o DSSC use uma configuração de parâmetros que melhor se adéqua às características dos dados de treinamento.

No segundo ponto, DSSC versus abordagens do estado da arte, foi conduzido o teste Wilcoxon Signed Rank em conjunto com cálculo do Cliff's delta com objetivo de comparar o DSSC com diferentes abordagens CPDP nos 82 projetos dos 4 conjuntos de dados avaliados. Além disso, nesta análise também computamos as vitórias/empates de cada método em relação a cada métrica de avaliação. Nesta análise foi concluído que a abordagem DSSC superou as demais abordagens nos conjuntos de dados AEEEM e PROMISE em relação a AUC e *False Alarm*, com diferença estatística em boa parte dos métodos, e com média/larga diferença (segundo o Cliff's delta) para resultados significativamente inferiores. Quanto as vitórias/empates, o DSSC mostrou pontuações superiores aos demais métodos em relação a AUC e *False Alarm*. Portanto, foi concluído que o DSSC é superior às abordagens CPDP analisadas para

a maioria das comparações com as métricas AUC e *False Alarm*, por exemplo, de 20 comparações (DSSC $\times$ 5 métodos CPDP $\times$ 4 conjuntos de dados), o DSSC apresentou $\frac{18}{20}$ dos melhores resultados para AUC e $\frac{19}{20}$ para *False Alarm*.

Posteriormente, no terceiro ponto, foi analisado a qualidade dos métodos CPDP com objetivo de verificar o grau/capacidade de discriminação dos métodos. Nesta análise, usamos a área sob a curva (ROC-AUC) e definimos algumas escalas ou intervalos para valores de AUC que descrevem a qualidade do modelo. Primeiro, categorizamos os 492 resultados (referente aos desempenhos de 82 projetos $\times$ 6 métodos CPDP) conforme cada escala AUC. Por fim, conforme a categorização dos desempenhos de cada método, foi visto que o DSSC apresentou os melhores índices discriminatórios: menores quantidade de desempenhos classificados como muito ruim, maior número de projetos classificados com desempenho excelente e como excepcional. Em seguida, foram analisados os resultados em relação à média para cada conjunto de dados, e portanto, o DSSC apresentou médias ROC-AUC acima de 0,7 para todos os conjuntos de dados, classificando-o como o único modelo com boa discriminação para todos os conjuntos de dados.

Por fim, no quarto ponto, sobre quando usar o método DSSC, construímos um meta-classificador (árvore de decisão) para verificar quais características dos projetos que mais influenciam na escolha da abordagem proposta (DSSC). Nesta análise, calculamos a importância de uma dada descrição dos dados com relação ao parâmetro de interesse: métrica de avaliação, ou seja, F1-score, AUC e *False Alarm*. Como resultados, foram obtidas as descrições dos projetos mais relevantes para o desempenho dos métodos: porcentagem de instâncias com defeito, número de instâncias, número de dimensões e desbalanceamento de classes. Em seguida, foram geradas árvores de decisão (1 para cada métrica de avaliação) para ilustrar e identificar quando utilizar a abordagem DSSC a partir das descrições dos dados. No meta-classificador foi confirmada e fortalecida a hipótese de que a porcentagem de instâncias com defeito, número de instâncias, número de dimensões e desbalanceamento de classes são as principais descrições para a escolha da abordagem CPDP.

6 CONCLUSÕES

Diferentes abordagens foram propostas para predizer com antecedência módulos de software propensos a defeitos, tanto abordagens utilizando apenas classificadores básicos de aprendizado de máquina como também técnicas de combinação de classificadores (*ensemble*). Entretanto, durante a predição, essas abordagens tradicionais costumam errar grande parte dos rótulos (classes). A principal hipótese é que modelos construídos usando apenas um classificador não são propensos a detectar defeitos de forma abrangente. Portanto, é de interesse que abordagens atuais sejam re-consideradas. Entre essas abordagens, os sistemas de múltiplos classificadores, mais especificamente a seleção dinâmica de classificadores, buscam selecionar em tempo real quais classificadores são mais aptos para resolver problemas preditivos.

Além disso, um dos principais desafios na predição de defeitos está relacionado a falta de informação (dados de treinamento), onde, de acordo com Hall et al. (2011), os modelos requerem uma quantidade suficientemente grande de dados disponíveis para realizar a predição. Nesta perspectiva, buscar novas abordagens para predição como também a quantidade de dados necessários para predição, pesquisadores se concentraram em pesquisas sobre a predição de defeitos entre projetos, ou seja, construir um modelo usando dados de treinamento de outros projetos para predizer os módulos de software de um projeto alvo.

Logo este trabalho, considerando as limitações de abordagens CPDP da literatura, por exemplo, métodos que utilizam apenas um classificador base para gerar o modelo, propomos o DSSC, um método CPDP supervisionado que, com base no conjunto de treinamento, seleciona a melhor configuração para um modelo, sejam técnicas de seleção dinâmica, o melhor classificador e o tamanho do *pool* de classificadores. O DSSC contém três fases principais: a definição do projeto alvo, a fase de superprodução e a fase de avaliação do modelo. Na primeira fase, seguindo o cenário *strict* CPDP (Figura 1a), cada projeto do conjunto de dados é definido como teste e os demais projetos (exceto as versões do projeto alvo) serão atribuídos aos dados de treinamento. Na segunda fase, durante n iterações diversos modelos são gerados com base no conjunto de treinamento e, posteriormente, o melhor modelo é selecionado (selecionado através de um cálculo de *rankings*) para predizer as novas entradas (dados de teste). Na última fase, é realizada a avaliação do modelo com o conjunto de teste. O desempenho foi avaliado usando as métricas F1-score, AUC e False Alarm.

A discussão se baseou nos seguintes pontos levantados: 1) CPDP: comportamento da

estrutura proposta (DSSC), 2) CPDP: abordagem proposta (DSSC) versus abordagens do estado da arte, 3) qualidade dos métodos CPDP, e 4) quando usar o método DSSC.

No primeiro ponto, foi analisado a influência dos métodos de seleção dinâmica, classificadores base e tamanho do *pool* de classificadores durante o treinamento dos dados. Neste ponto, foram analisadas quais técnicas de seleção dinâmica, classificadores e tamanho do *pool* que obtiveram mais pontuações (vitórias/empates), e conclui-se que não há uma configuração de parâmetros que maximize o desempenho do DSSC para todos os tipos de dados.

No segundo, considerando as mesmas configurações experimentais, o DSSC foi comparado com diferentes métodos CPDP da literatura. Durante a comparação em pares, os resultados mostram que o DSSC supera estatisticamente a maioria dos métodos CPDP em relação a AUC e PF. Em seguida, apresentamos uma análise das pontuações de cada método CPDP referente aos projetos em que obtiveram o melhor resultado. Conforme apresentado, o DSSC obteve maiores pontuações que os demais métodos quando comparados com AUC e *False Alarm*. Portanto, considerando as comparações entre os resultados médios (DSSC $\times$ 5 métodos CPDP $\times$ 4 conjuntos de dados $=$ 20 comparações), conclui-se que o DSSC é superior em 18 de 20 comparações para AUC e $\frac{19}{20}$ para *False Alarm*.

No terceiro ponto, foi analisado a qualidade dos métodos CPDP com base em algumas escalas dos valores de AUC. Primeiro, vimos que o DSSC apresentou os melhores resultados: menores quantidade de desempenhos categorizados como muito ruim, maior número de projetos classificados com desempenho excelente e como excepcional. Posteriormente, analisamos o desempenho dos métodos em relação à média para cada conjunto de dados, e portanto, constatou que o melhor caso foi apresentado pelo DSSC, onde apresentou médias ROC-AUC acima de 0,7 para todos os conjuntos de dados, classificando-o como sendo o único modelo com boa discriminação para todos os conjuntos de dados.

No quarto ponto, construímos um meta-classificador para identificar quando utilizar a abordagem DSSC a partir das descrições dos dados, e portanto, o impacto das descrições estão diretamente relacionadas à métrica de avaliação utilizada. Logo, conclui-se que a porcentagem de instâncias com defeito, número de instâncias, número de dimensões e desbalanceamento são as principais descrições para a escolha do método DSSC.

Por fim, os resultados obtidos pelo DSSC mostram-se satisfatórios, pois superam às abordagens CPDP na maioria dos casos estudados em relação a AUC e *False Alarm*, e que é possível avaliar previamente em quais projetos a abordagem DSSC pode ser melhor de acordo com suas descrições dos dados.

Extensões futuras interessantes podem incluir estudos sobre métodos estatísticos como a *Hellinger distance* (HELLINGER, 1909) para remover projetos com alto índice de semelhança do conjunto de treinamento. A hipótese para desconsiderar um projeto p_i é que, sejam dois projetos p_1 e p_2 que possuem alto índice de semelhança, um desses projetos será considerado como irrelevante, isto é, não fornece informações adicionais e/ou relevantes para compor o conjunto de treinamento e, posteriormente, ser utilizado pelo modelo. Investigar a influência e a relação entre as descrições dos dados durante a escolha (fase de seleção) dos diferentes parâmetros (técnicas de seleção dinâmica, classificadores base e tamanho do *pool* de classificadores).

REFERÊNCIAS

ALMEIDA, H. A. d. M. S. *Seleção dinâmica de classificadores baseada em filtragem e em distância adaptativa*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2014.

BANSIYA, J.; DAVIS, C. G. A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on software engineering*, IEEE, v. 28, n. 1, p. 4–17, 2002.

BASILI, V. R.; BRIAND, L. C.; MELO, W. L. A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on software engineering*, IEEE, v. 22, n. 10, p. 751–761, 1996.

BEZERRA, M. E.; OLIVEIRA, A. L.; MEIRA, S. R. A constructive rbf neural network for estimating the probability of defects in software modules. In: IEEE. *2007 International Joint Conference on Neural Networks*. [S.l.], 2007. p. 2869–2874.

BIRD, C.; NAGAPPAN, N.; GALL, H.; MURPHY, B.; DEVANBU, P. Putting it all together: Using socio-technical networks to predict failures. In: IEEE. *2009 20th International Symposium on Software Reliability Engineering*. [S.l.], 2009. p. 109–119.

BOWES, D.; HALL, T.; PETRIĆ, J. Software defect prediction: do different classifiers find the same defects? *Software Quality Journal*, Springer, v. 26, n. 2, p. 525–552, 2018.

BRADLEY, A. P. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, Elsevier, v. 30, n. 7, p. 1145–1159, 1997.

BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996.

BÜHLMANN, P. Bagging, boosting and ensemble methods. In: *Handbook of computational statistics*. [S.l.]: Springer, 2012. p. 985–1022.

CAVALIN, P. R.; SABOURIN, R.; SUEN, C. Y. Dynamic selection approaches for multiple classifier systems. *Neural computing and applications*, Springer, v. 22, n. 3, p. 673–688, 2013.

CHIDAMBER, S. R.; KEMERER, C. F. A metrics suite for object oriented design. *IEEE Transactions on software engineering*, IEEE, v. 20, n. 6, p. 476–493, 1994.

CLIFF, N. *Ordinal methods for behavioral data analysis*. [S.l.]: Psychology Press, 2014.

CRUZ, A. E. C.; OCHIMIZU, K. Towards logistic regression models for predicting fault-prone code across software projects. In: IEEE. *2009 3rd international symposium on empirical software engineering and measurement*. [S.l.], 2009. p. 460–463.

CRUZ, R. M.; HAFEMANN, L. G.; SABOURIN, R.; CAVALCANTI, G. D. Deslib: A dynamic ensemble selection library in python. *Journal of Machine Learning Research*, v. 21, n. 8, p. 1–5, 2020.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, Elsevier, v. 41, p. 195–216, 2018.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D.; REN, T. I. Meta-des: A dynamic ensemble selection framework using meta-learning. *Pattern recognition*, Elsevier, v. 48, n. 5, p. 1925–1935, 2015.

D'AMBROS, M.; LANZA, M.; ROBBES, R. An extensive comparison of bug prediction approaches. In: IEEE. *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*. [S.l.], 2010. p. 31–41.

GHOTRA, B.; MCINTOSH, S.; HASSAN, A. E. Revisiting the impact of classification techniques on the performance of defect prediction models. In: IEEE. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. [S.l.], 2015. v. 1, p. 789–800.

GIACINTO, G.; ROLI, F. Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition*, Citeseer, v. 34, n. 9, p. 1879–1882, 2001.

GYIMÓTHY, T.; FERENC, R.; SIKET, I. Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on Software engineering*, IEEE, v. 31, n. 10, p. 897–910, 2005.

HALL, T.; BEECHAM, S.; BOWES, D.; GRAY, D.; COUNSELL, S. A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering*, IEEE, v. 38, n. 6, p. 1276–1304, 2011.

HANLEY, J. A.; MCNEIL, B. J. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, v. 143, n. 1, p. 29–36, 1982.

HELLINGER, E. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, De Gruyter Berlin, New York, v. 1909, n. 136, p. 210–271, 1909.

HENDERSON-SELLERS, B. *Object-oriented metrics: measures of complexity*. [S.l.]: Prentice-Hall, Inc., 1995.

HERBOLD, S. A systematic mapping study on cross-project defect prediction. *arXiv preprint arXiv:1705.06429*, 2017.

HERBOLD, S. Benchmarking cross-project defect prediction approaches with costs metrics. *arXiv preprint arXiv:1801.04107*, 2018.

HERBOLD, S.; TRAUTSCH, A.; GRABOWSKI, J. A comparative study to benchmark cross-project defect prediction approaches. *IEEE Transactions on Software Engineering*, IEEE, v. 44, n. 9, p. 811–833, 2017.

HERBOLD, S.; TRAUTSCH, A.; GRABOWSKI, J. Global vs. local models for cross-project defect prediction. *Empirical software engineering*, Springer, v. 22, n. 4, p. 1866–1902, 2017.

HERBOLD, S.; TRAUTSCH, A.; GRABOWSKI, J. A comparative study to benchmark cross-project defect prediction approaches. *IEEE Transactions on Software Engineering*, v. 44, n. 9, p. 811–833, doi: <https://doi.org/10.1109/TSE.2017.2724538>, 2018.

JIANG, Y.; CUKI, B.; MENZIES, T.; BARTLOW, N. Comparing design and code metrics for software quality prediction. In: *Proceedings of the 4th international workshop on Predictor models in software engineering*. [S.l.: s.n.], 2008. p. 11–18.

JING, X.; WU, F.; DONG, X.; QI, F.; XU, B. Heterogeneous cross-company defect prediction by unified metric representation and cca-based transfer learning. In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. [S.l.: s.n.], 2015. p. 496–507.

JR, D. W. H.; LEMESHOW, S.; STURDIVANT, R. X. *Applied logistic regression*. [S.l.]: John Wiley & Sons, 2013. v. 398.

JURECZKO, M.; MADEYSKI, L. Towards identifying software project clusters with regard to defect prediction. In: *Proceedings of the 6th international conference on predictive models in software engineering*. [S.l.: s.n.], 2010. p. 1–10.

KIM, S.; WHITEHEAD, E. J.; ZHANG, Y. Classifying software changes: Clean or buggy? *IEEE Transactions on software engineering*, IEEE, v. 34, n. 2, p. 181–196, 2008.

KIM, S.; ZHANG, H.; WU, R.; GONG, L. Dealing with noise in defect prediction. In: IEEE. *2011 33rd International Conference on Software Engineering (ICSE)*. [S.l.], 2011. p. 481–490.

KITCHENHAM, B. A.; MENDES, E.; TRAVASSOS, G. H. Cross versus within-company cost estimation studies: A systematic review. *IEEE Transactions on Software Engineering*, IEEE, v. 33, n. 5, p. 316–329, 2007.

KO, A. H.; SABOURIN, R.; JR, A. S. B. From dynamic classifier selection to dynamic ensemble selection. *Pattern recognition*, Elsevier, v. 41, n. 5, p. 1718–1731, 2008.

KORU, A. G.; EMAM, K. E.; ZHANG, D.; LIU, H.; MATHEW, D. Theory of relative defect proneness. *Empirical Software Engineering*, Springer, v. 13, n. 5, p. 473–498, 2008.

KORU, A. G.; ZHANG, D.; EMAM, K. E.; LIU, H. An investigation into the functional form of the size-defect relationship for software modules. *IEEE Transactions on Software Engineering*, IEEE, v. 35, n. 2, p. 293–304, 2008.

KORU, G.; LIU, H.; ZHANG, D.; EMAM, K. E. Testing the theory of relative defect proneness for closed-source software. *Empirical Software Engineering*, Springer, v. 15, n. 6, p. 577–598, 2010.

LEE, T.; NAM, J.; HAN, D.; KIM, S.; IN, H. P. Micro interaction metrics for defect prediction. In: *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*. [S.l.: s.n.], 2011. p. 311–321.

Li, Z.; Jing, X.; Zhu, X.; Zhang, H.; Xu, B.; Ying, S. On the multiple sources and privacy preservation issues for heterogeneous defect prediction. *IEEE Transactions on Software Engineering*, v. 45, n. 4, p. 391–411, 2019.

LI, Z.; JING, X.-Y.; WU, F.; ZHU, X.; XU, B.; YING, S. Cost-sensitive transfer kernel canonical correlation analysis for heterogeneous defect prediction. *Automated Software Engineering*, Springer, v. 25, n. 2, p. 201–245, 2018.

LI, Z.; JING, X.-Y.; ZHU, X.; ZHANG, H. Heterogeneous defect prediction through multiple kernel learning and ensemble learning. In: IEEE. *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. [S.l.], 2017. p. 91–102.

LIMSETTHO, N.; BENNIN, K. E.; KEUNG, J. W.; HATA, H.; MATSUMOTO, K. Cross project defect prediction using class distribution estimation and oversampling. *Information and Software Technology*, v. 100, p. 87–102, 2018. ISSN 0950-5849. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584918300533>.

LIU, C.; YANG, D.; XIA, X.; YAN, M.; ZHANG, X. A two-phase transfer learning model for cross-project defect prediction. *Information and Software Technology*, Elsevier, v. 107, p. 125–136, 2019.

LIU, Y.; KHOSHGOFTAAR, T. M.; SELIYA, N. Evolutionary optimization of software quality modeling with multiple repositories. *IEEE Transactions on Software Engineering*, IEEE, v. 36, n. 6, p. 852–864, 2010.

LÓPEZ, V.; FERNÁNDEZ, A.; GARCÍA, S.; PALADE, V.; HERRERA, F. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information sciences*, Elsevier, v. 250, p. 113–141, 2013.

MA, Y.; LUO, G.; ZENG, X.; CHEN, A. Transfer learning for cross-company software defect prediction. *Information and Software Technology*, Elsevier, v. 54, n. 3, p. 248–256, 2012.

MALHOTRA, R. A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing*, Elsevier, v. 27, p. 504–518, 2015.

MARTIN, R. Oo design quality metrics. *An analysis of dependencies*, v. 12, n. 1, p. 151–170, 1994.

MCCABE, T. J. A complexity measure. *IEEE Transactions on software Engineering*, IEEE, n. 4, p. 308–320, 1976.

MENZIES, T.; BUTCHER, A.; COK, D.; MARCUS, A.; LAYMAN, L.; SHULL, F.; TURHAN, B.; ZIMMERMANN, T. Local versus global lessons for defect prediction and effort estimation. *IEEE Transactions on software engineering*, IEEE, v. 39, n. 6, p. 822–834, 2012.

MENZIES, T.; BUTCHER, A.; MARCUS, A.; ZIMMERMANN, T.; COK, D. Local vs. global models for effort estimation and defect prediction. In: IEEE. *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. [S.l.], 2011. p. 343–351.

NAM, J.; FU, W.; KIM, S.; MENZIES, T.; TAN, L. Heterogeneous defect prediction. *IEEE Transactions on Software Engineering*, IEEE, v. 44, n. 9, p. 874–896, 2017.

NAM, J.; KIM, S. Clami: Defect prediction on unlabeled datasets (t). In: IEEE. *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. [S.l.], 2015. p. 452–463.

NAM, J.; PAN, S. J.; KIM, S. Transfer defect learning. In: IEEE. *2013 35th international conference on software engineering (ICSE)*. [S.l.], 2013. p. 382–391.

NI, C.; XIA, X.; LO, D.; CHEN, X.; GU, Q. Revisiting supervised and unsupervised methods for effort-aware cross-project defect prediction. *IEEE Transactions on Software Engineering*, IEEE, 2020.

OSTRAND, T. J.; WEYUKER, E. J.; BELL, R. M. Predicting the location and number of faults in large software systems. *IEEE Transactions on Software Engineering*, IEEE, v. 31, n. 4, p. 340–355, 2005.

PADHY, N.; SATAPATHY, S.; SINGH, R. State-of-the-art object-oriented metrics and its reusability: a decade review. *Smart Computing and Informatics*, Springer, p. 431–441, 2018.

PANICHELLA, A.; OLIVETO, R.; LUCIA, A. D. Cross-project defect prediction models: L'union fait la force. In: IEEE. *2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*. [S.l.], 2014. p. 164–173.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

PETERS, F.; MENZIES, T.; MARCUS, A. Better cross company defect prediction. In: IEEE. *2013 10th Working Conference on Mining Software Repositories (MSR)*. [S.l.], 2013. p. 409–418.

RATHORE, S. S.; KUMAR, S. Software fault prediction based on the dynamic selection of learning technique: findings from the eclipse project study. *Applied Intelligence*, Springer, p. 1–16, 2021.

RAWAT, M. S.; DUBEY, S. K. Software defect prediction models for quality improvement: a literature study. *International Journal of Computer Science Issues (IJCSI)*, Citeseer, v. 9, n. 5, p. 288, 2012.

SABOURIN, M.; MITICHE, A.; THOMAS, D.; NAGY, G. Classifier combination for hand-printed digit recognition. In: IEEE. *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93)*. [S.l.], 1993. p. 163–166.

SAGI, O.; ROKACH, L. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley Online Library, v. 8, n. 4, p. e1249, 2018.

SHEPPERD, M.; SONG, Q.; SUN, Z.; MAIR, C. Data quality: Some comments on the nasa software defect datasets. *IEEE Transactions on Software Engineering*, IEEE, v. 39, n. 9, p. 1208–1215, 2013.

SUN, Y.; JING, X.-Y.; WU, F.; LI, J.; XING, D.; CHEN, H.; SUN, Y. Adversarial learning for cross-project semi-supervised defect prediction. *IEEE Access*, v. 8, p. 32674–32687, 2020.

SUN, Y.; JING, X.-Y.; WU, F.; SUN, Y. Selective pseudo-labeling based subspace learning for cross-project defect prediction. *IEICE TRANSACTIONS on Information and Systems*, The Institute of Electronics, Information and Communication Engineers, v. 103, n. 9, p. 2003–2006, 2020.

TANG, M.-H.; KAO, M.-H.; CHEN, M.-H. An empirical study on object-oriented metrics. In: IEEE. *Proceedings sixth international software metrics symposium (Cat. No. PR00403)*. [S.l.], 1999. p. 242–249.

TURHAN, B.; MENZIES, T.; BENER, A. B.; STEFANO, J. D. On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering*, Springer, v. 14, n. 5, p. 540–578, 2009.

VIRTANEN, P.; GOMMERS, R.; OLIPHANT, T. E.; HABERLAND, M.; REDDY, T.; COURNAPEAU, D.; BUROVSKI, E.; PETERSON, P.; WECKESSER, W.; BRIGHT, J.; van der Walt, S. J.; BRETT, M.; WILSON, J.; MILLMAN, K. J.; MAYOROV, N.; NELSON, A.

R. J.; JONES, E.; KERN, R.; LARSON, E.; CAREY, C. J.; POLAT, İ.; FENG, Y.; MOORE, E. W.; VanderPlas, J.; LAXALDE, D.; PERKTOLD, J.; CIMRMAN, R.; HENRIKSEN, I.; QUINTERO, E. A.; HARRIS, C. R.; ARCHIBALD, A. M.; RIBEIRO, A. H.; PEDREGOSA, F.; van Mulbregt, P.; SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, v. 17, p. 261–272, 2020.

WATANABE, S.; KAIYA, H.; KAIJIRI, K. Adapting a fault prediction model to allow inter languagereuse. In: *Proceedings of the 4th international workshop on Predictor models in software engineering*. [S.l.: s.n.], 2008. p. 19–24.

WILCOXON, F. Individual comparisons by ranking methods. In: *Breakthroughs in statistics*. [S.l.]: Springer, 1992. p. 196–202.

WOODS, K.; KEGELMEYER, W. P.; BOWYER, K. Combination of multiple classifiers using local accuracy estimates. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 19, n. 4, p. 405–410, 1997.

WU, R.; ZHANG, H.; KIM, S.; CHEUNG, S.-C. Relink: recovering links between bugs and changes. In: *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*. [S.l.: s.n.], 2011. p. 15–25.

XIA, X.; LO, D.; PAN, S. J.; NAGAPPAN, N.; WANG, X. Hydra: Massively compositional model for cross-project defect prediction. *IEEE Transactions on software Engineering*, IEEE, v. 42, n. 10, p. 977–998, 2016.

ZHOU, Y.; YANG, Y.; LU, H.; CHEN, L.; LI, Y.; ZHAO, Y.; QIAN, J.; XU, B. How far we have progressed in the journey? an examination of cross-project defect prediction. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, ACM New York, NY, USA, v. 27, n. 1, p. 1–51, 2018.

ZIMMERMANN, T.; NAGAPPAN, N. Predicting defects with program dependencies. In: IEEE. *2009 3rd international symposium on empirical software engineering and measurement*. [S.l.], 2009. p. 435–438.

APÊNDICE A – DETALHES DOS CONJUNTOS DE DADOS

Conjuntos de dados de defeitos de software podem ser vistos como registros de erros cometidos durante o desenvolvimento de sistemas de software. Esses erros são utilizados para predizer a ocorrência de defeitos ou o número de defeitos. De forma mais específica, dados de software possuem instâncias ou módulos de software que pode ou não ser propensos a defeitos. Esses dados consistem em um conjunto de informações do desenvolvimento do software, conhecidas principalmente como métricas de software. De acordo com Padhy, Satapathy e Singh (2018), essas métricas podem ser categorizadas em métricas de requisitos, métricas de produto ou métricas de processo.

Conforme essas informações, neste apêndice são apresentados os detalhes em relação às *features* (métricas) dos conjuntos de dados utilizados nos experimentos: AEEEM, PROMISE, NASA e RELINK.

A.1 MÉTRICAS PARA NASA

As métricas a seguir (Tabela 28) fazem parte do conjunto de dados da NASA. De forma que, são listadas um subconjunto de métricas contidas em todos os projetos no conjunto de dados da NASA. Cada projeto nesse conjunto de dados representa um sistema ou subsistema de software da NASA, que contém variáveis independentes (métricas de código estático) e variáveis dependentes (rótulos de dados de defeito). Além disso, cada módulo de software contém informações sobre, por exemplo, o tamanho e a complexidade do código. Para mais detalhes sobre as métricas utilizadas no conjunto de dados NASA, consultar o estudo de Shepperd et al. (2013).

Tabela 28 – Métricas do conjunto de dados NASA.

Métrica	**Símbolo**	**Definição ou fórmula**
PARAMETER_COUNT		Número de parâmetros para um determinado módulo/instância/método.

Continua na próxima página

Tabela 28 – Métricas do conjunto de dados NASA. – continuação da página anterior

Métrica	**Símbolo**	**Definição ou fórmula**
NUM_OPERATORS	N_1	Número de operadores contidos em um módulo.
NUM_OPERANDS	N_2	Número de operandos contidos em um módulo.
NUM_UNIQUE_OPERATORS	μ_1	Número de operadores únicos contidos em um módulo.
NUM_UNIQUE_OPERANDS	μ_2	Número de operandos únicos contidos em um módulo.
HALSTEAD_CONTENT	μ	*halstead content* de um módulo calculado por $\mu = \mu_1 + \mu_2$.
HALSTEAD_LENGTH	N	*halstead length* de um módulo calculada por $N = N_1 + N_2$.
HALSTEAD_LEVEL	L	*halstead level* de um módulo calculada por $L = \frac{2*\mu_2}{\mu_1*N_2}$.
HALSTEAD_DIFFICULTY	D	*halstead dificult* de um módulo calculada por $D = \frac{1}{L}$.
HALSTEAD_VOLUME	V	*halstead volume* de um módulo calculada por $V = N * Log_2(\mu_1 + \mu_2)$.
HALSTEAD_EFFORT	E	*halstead effort* de um módulo calculada por $E = \frac{V}{L}$.
HALSTEAD_PROG_TIME	T	Tempo de programação *halstead* de um módulo calculado por $T = \frac{E}{18}$.

Continua na próxima página

Tabela 28 – Métricas do conjunto de dados NASA. – continuação da página anterior

Métrica	**Símbolo**	**Definição ou fórmula**
HALSTEAD_ERROR_EST	B	O erro estimado de *halstead* calculado por $B = \frac{E^{\frac{2}{3}}}{1000}$.
NUMBER_OF_LINES		Número de linhas é um módulo de software.
LOC_BLANK		Número de linhas brancas em um módulo.
LOC_CODE_AND_COMMENT	$NCSLOC$	Número de linhas contendo código e comentário em um módulo.
LOC_COMMENTS		Número de linhas com comentários em um módulo.
LOC_EXECUTABLE		Número de linhas de código executáveis para um módulo (não em branco ou comentário).
PERCENT_COMMENTS		Porcentagem de código que são comentários.
LOC_TOTAL		Número total de linhas para um módulo.
EDGE_COUNT	e	Número de arestas encontradas em um determinado controle de um módulo para outro.
NODE_COUNT	n	Número de nós encontrados em um determinado módulo.
BRANCH_COUNT		Contagem de ramificações.

Continua na próxima página

Tabela 28 – Métricas do conjunto de dados NASA. – continuação da página anterior

Métrica	**Símbolo**	**Definição ou fórmula**
CALL_PAIRS		Número de chamadas para outras funções em um módulo.
CONDITION_COUNT		Número de condicionais em um determinado módulo.
CYCLOMATIC_COMPLEXITY	$v(G)$	A complexidade ciclomática de um módulo calculada por $v(G) = e - n + 2$.
DECISION_COUNT		Número de pontos de decisão em um determinado módulo.
DECISION_DENSITY		Densidade de decisão calculada por $\frac{Condition\_count}{Decision\_count}$.
DESIGN_COMPLEXITY	$iv(G)$	A complexidade do projeto de um módulo.
DESIGN_DENSITY		A densidade do projeto é calculada como: $\frac{iv(G)}{v(G)}$.
ESSENTIAL_COMPLEXITY	$ev(G)$	A complexidade essencial de um módulo.
ESSENTIAL_DENSITY		A densidade essencial é calculada como: $\frac{ev(G)-1}{v(G)-1}$.
MAINTENANCE_SEVERITY		A gravidade da manutenção é calculada como: $\frac{ev(G)}{v(G)}$.
MODIFIED_CONDITION_COUN		O efeito de uma condição afeta um resultado de decisão variando apenas essa condição.

Continua na próxima página

Tabela 28 – Métricas do conjunto de dados NASA. – continuação da página anterior

Métrica	**Símbolo**	**Definição ou fórmula**
MULTIPLE_CONDITION_COUNT		Número das condições existentes em um módulo.
PATHOLOGICAL_COMPLEXITY		Uma medida do grau em que um módulo contém construções extremamente não estruturadas.
NORMALIZED_CYLOMATIC_COMPLEXITY		$\frac{v(G)}{NUMBER\_OF\_LINES}$.
GLOBAL_DATA_COMPLEXITY	$gdv(G)$	A proporção de $v(G)$ da estrutura de um módulo para seu $PARAMETER\_COUNT$.
GLOBAL_DATA_DENSITY		A densidade global dos dados calculada por $\frac{gdv(G)}{v(G)}$.
CYCLOMATIC_DENSITY		$\frac{v(G)}{NCSLOC}$

Fonte: Adaptado de Jiang et al. (2008)

A.2 MÉTRICAS PARA PROMISE

Cada projeto do conjunto de dados PROMISE (JURECZKO; MADEYSKI, 2010) possui no total 20 métricas (*features* ou atributos), desconsiderando os do tipo "descritivo", isto é, *features* que definem, por exemplo, o nome do projeto e a versão. Esse conjunto de dados é extraído do software Apache escritos em linguagem de programação JAVA, de modo que, os módulos de software são baseados em métricas orientadas a objetos (OO) coletadas a nível de classe.

Entre esses atributos, Chidamber e Kemerer (1994) recomendaram os seis (6) seguintes:

- *Weighted Methods per Class* (WMC): quantidade ponderada de métodos em uma classe.
- *Depth of Inheritance Tree* (DIT): profundidade da árvore de herança para uma classe.

- *Number of Children* (NOC): número de filhos da classe.
- *Coupling Between Object* (CBO): número de classes acopladas a uma classe.
- *Response For a Class* (RFC): número de métodos diferentes que podem ser executados se uma determinada classe receber uma mensagem (chamada).
- *Lack of Cohesion in Methods* (LCOM): quantidade de métodos que não possuem relação através do compartilhamento de algum campo da classe.

Uma métrica sugerida por Henderson-Sellers (1995):

- *Lack of Cohesion in Methods 3* (LCOM3): métrica sugerida por Henderson-Sellers (1995) e calculada a partir da equação:

$$LCOM3 = \frac{\left(\frac{1}{a}\sum_{j=1}^{a}\mu(A_j)\right) - m}{1 - m} \tag{A.1}$$

  sendo m é o número de métodos da classe, a é o número de atributos da classe e $\mu(A)$ é o número de métodos que acessam o atributo A.

Além disso, cinco métricas sugeridas por Bansiya e Davis (2002), tais como:

- *Number of Public Methods* (NPM): número de métodos em uma determinada classe que são declarados como públicos, também conhecida como *Class Interface Size* (CIS).
- *Data Acess Metric* (DAM): razão entre a quantidade de atributos privados e o número total de atributos declarados na classe.
- *Measure of Aggregation* (MOA): quantidade de campos da classe cujos tipos são classes definidas pelo usuário.
- *Measure of Functional Abstration* (MFA): razão entre a quantidade de métodos herdados por uma classe e o número total de métodos acessíveis pelos métodos membros da classe.
- *Cohesion Among Methods of Class* (CAM): Soma entre os diferentes tipos de parâmetros de método em cada método dividido pela multiplicação da quantidade de diferentes tipos de parâmetros de método em toda a classe e o número de métodos totais.

Tang, Kao e Chen (1999) sugeriu três extensões orientadas para a qualidade do conjunto de métricas de Chidamber e Kemerer (1994), entre elas:

- *Inheritance Coupling* (IC): número de classes pai às quais uma determinada classe está acoplada.
- *Coupling Between Methods* (CBM): número total de métodos aos quais todos os métodos herdados estão acoplados.
- *Average Method Complexity* (AMC): mede o tamanho médio do método para cada classe, isto é, o número de códigos binários Java no método.

Duas métricas sugeridas por Martin (1994):

- *Afferent Couplings* (Ca): representa o número de classes que dependem da classe c_i.
- *Efferent Couplings* (Ce): número de classes das quais a classe c_i depende.

McCabe (1976) propôs uma métrica:

- *McCabe's Cyclomatic Complexity* (CC): esta métrica mede a quantidade de caminhos entre os métodos (funções). Onde, $CC = E - N + P$, sendo E o número de arestas no grafo, N o número de nós no grafo e P o número de componentes conectados. Além disso, duas métricas foram derivadas:
  - *Max(CC)*: maior valor de CC entre os métodos da classe investigada.
  - *Avg(CC)*: média aritmética do valor de CC na classe investigada.

Por fim, todas essas métricas foram complementadas com mais uma:

- *Lines Of Code* (LOC): calcula o número de linhas de código da classe sob investigação.

A.3 MÉTRICAS PARA RELINK

O conjunto de dados RELINK é baseado em informações de código-fonte sobre controle de versão do software. As seguintes métricas fazem parte do conjunto de dados RELINK:

- *AvgCyclomatic*: média ciclomática para todas as funções ou métodos relacionados.
- *AvgCyclomaticModified*: ciclomática modificada média para todas as funções e métodos aninhados.

- *AvgCyclomaticStrict*: complexidade média estrita para todas as funções ou métodos aninhados.
- *AvgEssential*: complexidade essencial média para todas as funções ou métodos aninhados. AvgLine: número médio de linhas.
- *AvgLineBlank*: número médio de linhas em branco.
- *AvgLineCode*: número médio de linhas contendo código-fonte.
- *AvgLineComment*: número médio de linhas contendo comentário.
- *CountLine*: quantidade total de linhas.
- *CountLineBlank*: quantidade de linhas em branco.
- *CountLineCode*: número de linhas contendo código-fonte.
- *CountLineCodeDecl*: número de linhas contendo código-fonte declarativo.
- *CountLineComment*: número de linhas contendo comentário.
- *CountSemicolon*: número de pontos e vírgulas.
- *CountStmt*: número de declarações.
- *CountStmtDecl*: número de declarações declarativas.
- *CountStmtExe*: número de instruções executáveis.
- *MaxCyclomatic*: complexidade ciclomática máxima das funções ou métodos aninhados.
- *MaxCyclomaticModified*: complexidade ciclomática modificada máxima das funções ou métodos aninhados.
- *MaxCyclomaticStrict*: complexidade ciclomática estrita máxima das funções ou métodos aninhados.
- *RaioCommentToCode*: proporção de comentários do código.
- *SumCyclomatic*: soma das complexidades ciclomáticas dos métodos ou funções aninhados.

- *SumCyclomaticModified*: Soma das complexidades ciclomáticas modificadas das funções ou métodos aninhados.
- *SumCyclomaticStrict*: Soma das complexidades ciclomáticas estritas das funções ou métodos aninhados.
- *SumEssential*: Soma das complexidades ciclomáticas essenciais dos métodos ou funções aninhados.

A.4 MÉTRICAS PARA AEEEM

O conjunto de dados AEEEM originalmente compartilhado por D'Ambros, Lanza e Robbes (2010) contém informações sobre cinco produtos Java diferentes. Diferentes métricas de software são consideradas, incluindo métricas de software estático, métricas de processo (número de defeitos em versões anteriores), métricas de entropia (alterações de código), entre outras. A Tabela 29 lista todas as métricas utilizadas e abordadas por D'Ambros, Lanza e Robbes (2010) em seu estudo.

Tabela 29 – Métricas do conjunto de dados AEEEM.

Tipo	Sigla	Descrição
CK	WMC	Quantidade ponderada de métodos em uma classe
CK	DIT	Profundidade da árvore de herança para uma classe
CK	RFC	Número de métodos diferentes que podem ser executados se uma determinada classe receber uma mensagem
CK	NOC	Número de filhos da classe
CK	CBO	Número de classes acopladas a uma classe
CK	LCOM	Quantidade de métodos que não possuem relação através do compartilhamento de algum campo da classe
OO	FanIn	Número de outras classes que fazem referência à classe
OO	FanOut	Número de outras classes referenciadas pela classe
OO	NOA	Número de atributos da classe

Continua na próxima página

Tabela 29 – Métricas do conjunto de dados AEEEM. – continuação da página anterior

Tipo	Sigla	Descrição
OO	NOPA	Número de atributos públicos
OO	NOPRA	Número de atributos privados
OO	NOAI	Número de atributos herdados
OO	LOC	Número de linhas de código
OO	NOM	Número de métodos em geral
OO	NOPM	Número de métodos públicos
OO	NOPRM	Número de métodos privados
OO	NOMI	Número de métodos herdados

Fonte: Adaptado de D'Ambros, Lanza e Robbes (2010)

Ainda, o conjunto AEEEM é composto por diversas *features* resultantes da combinação dos dois tipos de métricas (CK e OO), logo:

- **CK**: Métricas baseadas nas abordagens propostas por Chidamber e Kemerer (1994).
- **OO**: representa métricas orientadas à objetos.
- **CK+OO**: combinação dos dois tipos de métricas.
- **LOC**: é tratada como uma das métricas mais populares na predição de defeitos, conforme discutido por Gyimóthy, Ferenc e Siket (2005).

APÊNDICE B – RESULTADOS EXPERIMENTAIS DO DSSC

Neste apêndice consta os resultados da abordagem proposta (DSSC) para cada projeto nos conjuntos de dados. Para cada projeto $\times$ conjunto de dados é apresentada a melhor configuração (técnica de seleção dinâmica, classificador base e tamanho do *pool* de classificadores) definida na etapa de seleção através do ranqueamento (Algoritmo 3).

Tabela 30 – Resultados do DSSC para cada projeto do conjunto de dados AEEEM. Para cada projeto é apresentada a melhor configuração de parâmetros. Método de seleção dinâmica (DS), Classificador (CLF), tamanho do *pool* de classificadores (M), *Random Forest* (RF).

Projeto	DS	CLF	*M*	F1-score	AUC	PF
eclipse	METADES	RF	100	0,555	0,829	0,029
equinox	LCA	RF	30	0,255	0,698	0,005
lucene	METADES	RF	60	0,289	0,770	0,002
mylyn	Rank	RF	30	0,295	0,689	0,034
pde	Rank	RF	20	0,342	0,747	0,037

Fonte: Elaborada pelo autor (2022)

Tabela 31 – Resultados do DSSC para cada projeto do conjunto de dados NASA. Para cada projeto é apresentada a melhor configuração de parâmetros. Método de seleção dinâmica (DS), Classificador (CLF), tamanho do *pool* de classificadores (M), *Random Forest* (RF).

Projeto	DS	CLF	M	F1-score	AUC	PF
CM1	METADES	RF	10	0,188	0,745	0,053
JM1	OLA	RF	10	0,159	0,681	0,032
KC1	KNOP	RF	60	0,128	0,689	0,014
KC3	Rank	RF	10	0,150	0,688	0,006
MC1	KNORAU	RF	30	0,055	0,894	0,004
MC2	Rank	RF	60	0,192	0,628	0,036
MW1	Rank	RF	40	0,071	0,793	0,000
PC1	MCB	RF	60	0,184	0,728	0,011
PC2	LCA	RF	50	0,091	0,779	0,003
PC3	MCB	RF	60	0,128	0,687	0,021
PC4	Rank	RF	40	0,053	0,650	0,006
PC5	KNOP	RF	90	0,232	0,902	0,004

Fonte: Elaborada pelo autor (2022)

Tabela 32 – Resultados do DSSC para cada projeto do conjunto de dados RELINK. Para cada projeto é apresentada a melhor configuração de parâmetros. Método de seleção dinâmica (DS), Classificador (CLF), tamanho do *pool* de classificadores (M), *Logistic Regression* (LR).

Projeto	DS	CLF	M	F1-score	AUC	PF
Apache2.0	KNORAE	LR	30	0,746	0,781	0,240
openintents	KNORAU	LR	80	0,581	0,775	0,000
zxing1.6	MCB	LR	60	0,301	0,660	0,103

Fonte: Elaborada pelo autor (2022)

Tabela 33 – Resultados do DSSC para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentada a melhor configuração de parâmetros. Método de seleção dinâmica (DS), Classificador (CLF), tamanho do *pool* de classificadores (M), *Random Forest* (RF), GaussianNB (NB).

Projeto	DS	CLF	M	F1-score	AUC	PF
ant-1.3	MCB	NB	30	0,364	0,762	0,067
ant-1.4	OLA	NB	10	0,226	0,538	0,109
ant-1.5	METADES	NB	70	0,389	0,763	0,100
ant-1.6	MCB	NB	30	0,417	0,791	0,085
ant-1.7	METADES	NB	70	0,420	0,779	0,064
arc	LCA	NB	70	0,279	0,681	0,048
berek	Rank	LR	100	0,769	0,986	0,000
camel-1.0	LCA	NB	80	0,160	0,732	0,031
camel-1.2	Rank	LR	80	0,204	0,574	0,033
camel-1.4	MCB	LR	50	0,200	0,685	0,073
camel-1.6	LCA	NB	80	0,174	0,608	0,068
ckjm	OLA	LR	90	0,000	0,880	0,000
e-learning	Rank	LR	60	0,000	0,803	0,000
forrest-0.7	KNORAU	LR	10	0,000	0,700	0,000
ivy-1.1	METADES	NB	40	0,225	0,730	0,000
ivy-1.4	KNORAE	NB	20	0,350	0,752	0,076

Continua na próxima página

Tabela 33 – Resultados do DSSC para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentada a melhor configuração de parâmetros. Método de seleção dinâmica (DS), Classificador (CLF), tamanho do *pool* de classificadores (M), *Random Forest* (RF), GaussianNB (NB). – continuação da página anterior

Projeto	DS	CLF	M	F1-score	AUC	PF
ivy-2.0	LCA	NB	70	0,424	0,798	0,087
jedit-3.2	MCB	NB	60	0,478	0,780	0,082
jedit-4.0	METADES	NB	10	0,458	0,749	0,113
jedit-4.1	OLA	NB	20	0,597	0,789	0,094
jedit-4.2	KNOP	NB	100	0,481	0,818	0,157
jedit-4.3	LCA	NB	30	0,094	0,593	0,187
kalkulator	KNOP	LR	50	0,222	0,464	0,095
log4j-1.0	MCB	NB	30	0,250	0,786	0,010
log4j-1.1	KNORAE	NB	10	0,273	0,797	0,014
log4j-1.2	KNOP	NB	10	0,090	0,614	0,063
lucene-2.0	LCA	LR	100	0,180	0,721	0,000
lucene-2.2	MCB	NB	30	0,140	0,630	0,019
lucene-2.4	METADES	NB	40	0,245	0,702	0,037
nieruchomosci	OLA	LR	100	0,333	0,824	0,000
pbeans1	METADES	NB	20	0,261	0,808	0,000
pbeans2	METADES	NB	20	0,429	0,741	0,024
pdftranslator	OLA	LR	100	0,235	0,900	0,000
poi-1.5	METADES	LR	40	0,296	0,741	0,031
poi-2.0	LCA	LR	10	0,273	0,593	0,072
poi-2.5	KNORAE	LR	90	0,297	0,765	0,073
poi-3.0	OLA	NB	90	0,280	0,805	0,050

Continua na próxima página

Tabela 33 – Resultados do DSSC para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentada a melhor configuração de parâmetros. Método de seleção dinâmica (DS), Classificador (CLF), tamanho do *pool* de classificadores (M), *Random Forest* (RF), GaussianNB (NB). – continuação da página anterior

Projeto	DS	CLF	M	F1-score	AUC	PF
redaktor	OLA	LR	100	0,400	0,691	0,034
serapion	LCA	NB	10	0,200	0,843	0,000
skarbonka	KNORAE	LR	30	0,182	0,728	0,028
sklebagd	OLA	NB	100	0,400	0,740	0,000
synapse-1.0	OLA	LR	10	0,261	0,818	0,028
synapse-1.1	LCA	LR	100	0,229	0,666	0,012
synapse-1.2	OLA	LR	10	0,231	0,731	0,035
systemdata	MCB	NB	20	0,333	0,746	0,018
szybkafucha	OLA	LR	90	0,000	0,643	0,000
termoproject	Rank	LR	90	0,267	0,897	0,000
tomcat	LCA	NB	50	0,378	0,811	0,105
velocity-1.4	OLA	NB	10	0,099	0,347	0,143
velocity-1.5	KNORAU	NB	60	0,141	0,665	0,042
velocity-1.6	KNORAU	NB	60	0,237	0,711	0,026
workflow	METADES	LR	30	0,333	0,795	0,000
wspomaganiepi	LCA	NB	10	0,154	0,903	0,000
xalan-2.4	MCB	LR	40	0,385	0,741	0,135
xalan-2.5	Rank	LR	10	0,330	0,588	0,115
xalan-2.6	KNORAE	LR	10	0,365	0,639	0,089
xalan-2.7	KNORAE	LR	10	0,285	0,749	0,000
xerces-1.2	KNORAU	LR	20	0,264	0,464	0,092

Continua na próxima página

Tabela 33 – Resultados do DSSC para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentada a melhor configuração de parâmetros. Método de seleção dinâmica (DS), Classificador (CLF), tamanho do *pool* de classificadores (M), *Random Forest* (RF), GaussianNB (NB). – continuação da página anterior

Projeto	**DS**	**CLF**	**M**	**F1-score**	**AUC**	**PF**
xerces-1.3	LCA	LR	50	0,342	0,734	0,060
xerces-1.4	KNORAU	LR	20	0,204	0,695	0,026
xerces-init	KNORAE	LR	10	0,355	0,520	0,129
zuzel	OLA	NB	50	0,375	0,853	0,000

Fonte: Elaborada pelo autor (2022)

APÊNDICE C – RESULTADOS EXPERIMENTAIS DOS MÉTODOS CPDP

Neste apêndice será apresentado os resultados dos métodos para cada projeto nos conjuntos de dados. Para cada projeto × conjunto de dados é apresentada a melhor abordagem conforme a métrica de avaliação usada.

C.1 RESULTADOS COM F1-SCORE

Tabela 34 – Resultados dos métodos CPDP para cada projeto do conjunto de dados AEEEM. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a F1-score.

Projeto	Métodos CPDP						Melhor
	DSSC	EASC	CC09-DT	M11-RF	T09-DT	W08-DT	Método
eclipse	**0,555**	0,535	0,452	0,429	0,347	0,306	DSSC
equinox	0,255	**0,434**	0,333	0,275	0,326	0,282	EASC
lucene	0,289	**0,302**	0,284	0,259	0,213	0,290	EASC
mylyn	0,295	0,249	0,200	0,145	0,197	**0,316**	W08-DT
pde	**0,342**	0,337	0,289	0,236	0,273	0,309	DSSC

Fonte: Elaborada pelo autor (2022)

Tabela 37 – Resultados dos métodos CPDP para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a F1-score.

Projeto	Métodos CPDP						Melhor
	DSSC	EASC	CC09-DT	M11-RF	T09-DT	W08-DT	Método
ant-1.3	0,364	0,343	0,351	0,360	**0,381**	0,327	T09-DT
ant-1.4	0,226	0,209	0,308	0,141	0,217	**0,357**	W08-DT
ant-1.5	**0,389**	0,364	0,266	0,311	0,218	0,264	DSSC
ant-1.6	0,417	0,411	**0,491**	0,387	0,475	0,380	CC09-DT
ant-1.7	0,42	0,444	0,445	0,367	**0,468**	0,360	T09-DT
arc	0,279	0,267	**0,293**	0,272	0,290	0,237	CC09-DT

Continua na próxima página

Tabela 37 – Resultados dos métodos CPDP para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a F1-score. – continuação da página anterior

Projeto	Métodos CPDP						Melhor
	DSSC	EASC	CC09-DT	M11-RF	T09-DT	W08-DT	Método
berek	0,769	**0,857**	0,583	0,828	0,828	0,500	EASC
camel-1.0	0,160	**0,308**	0,075	0,188	0,099	0,157	EASC
camel-1.2	0,204	0,205	0,319	0,299	**0,360**	0,318	T09-DT
camel-1.4	0,200	0,213	**0,337**	0,252	0,295	0,279	CC09-DT
camel-1.6	0,174	0,206	**0,345**	0,218	0,303	0,343	CC09-DT
ckjm	0,000	0,000	0,333	0,333	0,000	**0,500**	W08-DT
e-learning	0,000	0,000	0,080	0,125	0,000	**0,128**	W08-DT
forrest-0.7	0,000	0,000	0,308	0,200	0,50	**0,571**	W08-DT
ivy-1.1	0,225	0,250	**0,467**	0,400	0,414	0,391	CC09-DT
ivy-1.4	**0,350**	0,293	0,222	0,154	0,154	0,169	DSSC
ivy-2.0	**0,424**	0,391	0,290	0,225	0,310	0,225	DSSC
jedit-3.2	0,478	0,531	**0,585**	0,526	0,515	0,482	CC09-DT
jedit-4.0	0,458	0,471	**0,500**	0,407	0,489	0,497	CC09-DT
jedit-4.1	0,597	**0,601**	0,547	0,464	0,526	0,470	EASC
jedit-4.2	**0,481**	0,457	0,395	0,377	0,379	0,358	DSSC
jedit-4.3	**0,094**	0,080	0,053	0,039	0,055	0,052	DSSC
kalkulator	0,222	0,200	**0,545**	0,400	0,000	0,222	CC09-DT
log4j-1.0	0,250	0,286	0,267	0,414	**0,528**	0,462	T09-DT
log4j-1.1	0,273	0,233	0,400	0,275	**0,571**	0,413	T09-DT
log4j-1.2	0,090	0,100	0,323	0,362	0,364	**0,371**	W08-DT
lucene-2.0	0,180	0,175	0,418	0,439	**0,492**	0,372	T09-DT

Continua na próxima página

Tabela 37 – Resultados dos métodos CPDP para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a F1-score. – continuação da página anterior

Projeto	Métodos CPDP						Melhor
	DSSC	EASC	CC09-DT	M11-RF	T09-DT	W08-DT	Método
lucene-2.2	0,140	0,173	0,367	0,270	**0,368**	0,276	T09-DT
lucene-2.4	0,245	0,287	0,410	0,414	**0,426**	0,349	T09-DT
nieruchomosci	0,333	0,000	0,308	0,353	0,462	**0,480**	W08-DT
pbeans1	0,261	0,261	0,667	0,261	0,095	**0,687**	W08-DT
pbeans2	0,429	0,429	**0,500**	0,400	0,471	0,353	CC09-DT
pdftranslator	0,235	0,125	**0,741**	0,421	0,421	0,500	CC09-DT
poi-1.5	0,296	0,274	0,310	0,412	**0,490**	0,338	T09-DT
poi-2.0	0,273	0,240	0,230	0,167	0,239	**0,291**	W08-DT
poi-2.5	0,297	0,250	0,304	0,521	**0,531**	0,428	T09-DT
poi-3.0	0,280	0,280	0,411	0,424	**0,681**	0,465	T09-DT
redaktor	0,400	0,273	**0,429**	0,114	0,142	0,233	CC09-DT
serapion	0,200	0,364	**0,556**	0,333	0,200	0,522	CC09-DT
skarbonka	0,182	0,154	0,375	**0,421**	0,400	0,261	M11-RF
sklebagd	0,400	0,400	**0,783**	0,444	0,143	0,500	CC09-DT
synapse-1.0	0,261	0,333	0,243	**0,367**	0,273	0,276	M11-RF
synapse-1.1	0,229	0,370	**0,468**	0,333	0,451	0,283	CC09-DT
synapse-1.2	0,231	0,419	0,428	**0,461**	0,328	0,346	M11-RF
systemdata	0,333	**0,462**	0,182	0,222	0,455	0,316	EASC
szybkafucha	0,000	0,000	**0,435**	0,133	0,222	0,286	CC09-DT
termoproject	0,267	0,143	0,125	0,190	**0,710**	0,400	T09-DT
tomcat	0,378	**0,384**	0,304	0,236	0,187	0,248	EASC

Continua na próxima página

Tabela 37 – Resultados dos métodos CPDP para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a F1-score. – continuação da página anterior

Projeto	**Métodos CPDP**						**Melhor**
	DSSC	**EASC**	**CC09-DT**	**M11-RF**	**T09-DT**	**W08-DT**	**Método**
velocity-1.4	0,099	0,120	0,400	0,237	0,212	**0,483**	W08-DT
velocity-1.5	0,141	0,164	0,279	0,243	0,344	**0,474**	W08-DT
velocity-1.6	0,237	0,250	0,328	0,291	0,305	**0,453**	W08-DT
workflow	0,333	0,261	0,357	0,370	0,400	**0,486**	W08-DT
wspomaganiepi	0,154	0,154	0,000	0,353	**0,556**	0,526	T09-DT
xalan-2.4	**0,385**	0,375	0,326	0,310	0,324	0,274	DSSC
xalan-2.5	0,330	0,405	0,423	0,349	**0,441**	0,419	T09-DT
xalan-2.6	0,365	0,467	0,475	0,377	0,338	**0,483**	W08-DT
xalan-2.7	0,285	0,402	0,361	0,351	**0,450**	0,379	T09-DT
xerces-1.2	0,264	0,240	0,164	**0,293**	0,232	0,213	M11-RF
xerces-1.3	0,342	0,374	0,270	0,284	**0,440**	0,332	T09-DT
xerces-1.4	0,204	0,278	**0,521**	0,366	0,375	0,492	CC09-DT
xerces-init	0,355	0,315	0,269	0,310	0,356	**0,426**	W08-DT
zuzel	0,375	0,375	0,455	**0,571**	0,545	0,333	M11-RF

Fonte: Elaborada pelo autor (2022)

Tabela 35 – Resultados dos métodos CPDP para cada projeto do conjunto de dados NASA. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a F1-score.

Projeto	Métodos CPDP						Melhor
	DSSC	EASC	CC09-DT	M11-RF	T09-DT	W08-DT	Método
CM1	0,188	**0,318**	0,044	0,200	0,250	0,040	EASC
JM1	0,159	**0,259**	0,035	0,179	0,238	0,100	EASC
KC1	0,128	0,101	0,069	**0,131**	0,035	0,112	M11-RF
KC3	0,150	**0,269**	0,103	0,098	0,240	0,190	EASC
MC1	0,055	0,068	0,010	0,096	0,094	**0,148**	W08-DT
MC2	0,192	**0,393**	0,200	0,246	0,185	0,044	EASC
MW1	0,071	0,067	0,000	0,129	**0,269**	0,067	T09-DT
PC1	0,184	**0,284**	0,110	0,127	0,095	0,150	EASC
PC2	0,091	0,100	0,122	**0,140**	0,121	0,098	M11-RF
PC3	0,128	**0,247**	0,026	0,100	0,145	0,140	EASC
PC4	0,053	**0,101**	0,048	0,057	0,020	0,085	EASC
PC5	0,232	**0,384**	0,303	0,203	0,243	0,139	EASC

Fonte: Elaborada pelo autor (2022)

Tabela 36 – Resultados dos métodos CPDP para cada projeto do conjunto de dados RELINK. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a F1-score.

Projeto	Métodos CPDP						Melhor
	DSSC	EASC	CC09-DT	M11-RF	T09-DT	W08-DT	Método
Apache2.0	**0,746**	0,718	0,604	0,691	0,725	0,394	DSSC
openintents	0,581	0,000	0,556	0,500	0,667	**0,698**	W08-DT
zxing1.6	0,301	0,000	0,470	**0,473**	0,192	0,374	M11-RF

Fonte: Elaborada pelo autor (2022)

C.2 RESULTADOS COM AUC

Tabela 39 – Resultados dos métodos CPDP para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a AUC.

Projeto	Métodos CPDP						Melhor
	DSSC	EASC	CC09-DT	M11-RF	T09-DT	W08-DT	Método
ant-1.3	0,762	0,753	0,607	0,625	0,646	0,685	DSSC

Continua na próxima página

Tabela 39 – Resultados dos métodos CPDP para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a AUC. – continuação da página anterior

Projeto	Métodos CPDP						Melhor
	DSSC	EASC	CC09-DT	M11-RF	T09-DT	W08-DT	Método
ant-1.4	0,538	0,536	0,521	0,468	0,426	0,556	W08-DT
ant-1.5	0,763	0,768	0,659	0,634	0,558	0,648	EASC
ant-1.6	0,791	0,778	0,639	0,586	0,646	0,601	DSSC
ant-1.7	0,779	0,779	0,625	0,590	0,618	0,617	DSSC, EASC
arc	0,681	0,670	0,690	0,610	0,609	0,529	CC09-DT
berek	0,986	0,958	0,708	0,856	0,853	0,631	DSSC
camel-1.0	0,732	0,802	0,471	0,662	0,573	0,687	EASC
camel-1.2	0,574	0,550	0,503	0,533	0,585	0,526	T09-DT
camel-1.4	0,685	0,682	0,573	0,546	0,579	0,577	DSSC
camel-1.6	0,608	0,621	0,611	0,519	0,543	0,576	EASC
ckjm	0,880	0,800	0,560	0,600	0,500	0,340	DSSC
e-learning	0,803	0,793	0,359	0,515	0,408	0,385	DSSC
forrest-0.7	0,700	0,583	0,675	0,517	0,812	0,796	T09-DT
ivy-1.1	0,730	0,711	0,499	0,549	0,634	0,583	DSSC
ivy-1.4	0,752	0,738	0,609	0,558	0,496	0,692	DSSC
ivy-2.0	0,798	0,800	0,582	0,563	0,604	0,613	EASC
jedit-3.2	0,780	0,780	0,701	0,651	0,583	0,650	DSSC, EASC
jedit-4.0	0,749	0,747	0,645	0,595	0,591	0,668	DSSC
jedit-4.1	0,789	0,786	0,709	0,635	0,668	0,617	DSSC
jedit-4.2	0,818	0,821	0,673	0,696	0,718	0,689	EASC
jedit-4.3	0,593	0,588	0,583	0,485	0,520	0,503	DSSC

Continua na próxima página

Tabela 39 – Resultados dos métodos CPDP para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a AUC. – continuação da página anterior

Projeto	Métodos CPDP						Melhor
	DSSC	EASC	CC09-DT	M11-RF	T09-DT	W08-DT	Método
kalkulator	0,464	0,615	0,726	0,607	0,194	0,405	CC09-DT
log4j-1.0	0,786	0,796	0,468	0,617	0,696	0,663	EASC
log4j-1.1	0,797	0,795	0,580	0,546	0,705	0,585	DSSC
log4j-1.2	0,614	0,616	0,439	0,580	0,405	0,550	EASC
lucene-2.0	0,721	0,707	0,540	0,602	0,620	0,640	DSSC
lucene-2.2	0,630	0,630	0,552	0,535	0,558	0,607	DSSC, EASC
lucene-2.4	0,702	0,714	0,546	0,574	0,546	0,533	EASC
nieruchomosci	0,824	0,918	0,429	0,532	0,544	0,588	EASC
pbeans1	0,808	0,842	0,767	0,575	0,475	0,688	EASC
pbeans2	0,741	0,763	0,689	0,626	0,637	0,682	EASC
pdftranslator	0,900	0,852	0,831	0,633	0,633	0,600	DSSC
poi-1.5	0,741	0,747	0,535	0,557	0,565	0,581	EASC
poi-2.0	0,593	0,622	0,538	0,472	0,585	0,600	EASC
poi-2.5	0,765	0,723	0,666	0,620	0,488	0,518	DSSC
poi-3.0	0,805	0,799	0,493	0,571	0,726	0,610	DSSC
redaktor	0,691	0,691	0,720	0,443	0,465	0,508	CC09-DT
serapion	0,843	0,843	0,642	0,583	0,503	0,696	DSSC, EASC
skarbonka	0,728	0,728	0,528	0,639	0,654	0,622	DSSC, EASC
sklebagd	0,740	0,740	0,750	0,542	0,677	0,880	W08-DT
synapse-1.0	0,818	0,790	0,599	0,696	0,679	0,560	DSSC
synapse-1.1	0,666	0,700	0,568	0,557	0,566	0,524	EASC

Continua na próxima página

Tabela 39 – Resultados dos métodos CPDP para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a AUC. – continuação da página anterior

Projeto	Métodos CPDP						Melhor
	DSSC	EASC	CC09-DT	M11-RF	T09-DT	W08-DT	Método
synapse-1.2	0,731	0,742	0,493	0,612	0,586	0,633	EASC
systemdata	0,746	0,744	0,502	0,549	0,760	0,692	T09-DT
szybkafucha	0,643	0,662	0,422	0,536	0,562	0,429	EASC
termoproject	0,897	0,790	0,340	0,473	0,760	0,536	DSSC
tomcat	0,811	0,807	0,609	0,605	0,666	0,675	DSSC
velocity-1.4	0,347	0,317	0,612	0,480	0,426	0,605	CC09-DT
velocity-1.5	0,665	0,669	0,516	0,459	0,536	0,506	EASC
velocity-1.6	0,711	0,700	0,520	0,550	0,485	0,581	DSSC
workflow	0,795	0,584	0,643	0,572	0,645	0,538	DSSC
wspomaganiepi	0,903	0,875	0,750	0,458	0,639	0,632	DSSC
xalan-2.4	0,741	0,720	0,679	0,593	0,588	0,570	DSSC
xalan-2.5	0,588	0,578	0,624	0,529	0,578	0,543	CC09-DT
xalan-2.6	0,639	0,533	0,597	0,551	0,587	0,651	W08-DT
xalan-2.7	0,749	0,717	0,496	0,606	0,694	0,698	DSSC
xerces-1.2	0,464	0,464	0,465	0,577	0,486	0,521	M11-RF
xerces-1.3	0,734	0,781	0,509	0,577	0,711	0,569	EASC
xerces-1.4	0,695	0,734	0,567	0,585	0,675	0,519	EASC
xerces-init	0,520	0,476	0,449	0,493	0,375	0,500	DSSC
zuzel	0,853	0,839	0,471	0,668	0,587	0,572	DSSC

Fonte: Elaborada pelo autor (2022)

Tabela 38 – Resultados dos métodos CPDP para cada projeto do conjunto de dados AEEEM. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a AUC.

	Métodos CPDP						Melhor
Projeto	**DSSC**	**EASC**	**CC09-DT**	**M11-RF**	**T09-DT**	**W08-DT**	**Método**
eclipse	**0,829**	0,774	0,641	0,640	0,547	0,479	DSSC
equinox	0,698	**0,738**	0,699	0,574	0,543	0,629	EASC
lucene	**0,770**	0,680	0,579	0,578	0,500	0,589	DSSC
mylyn	**0,689**	0,554	0,492	0,536	0,511	0,664	DSSC
pde	**0,747**	0,721	0,609	0,562	0,551	0,577	DSSC

Fonte: Elaborada pelo autor (2022)

Tabela 40 – Resultados dos métodos CPDP para cada projeto do conjunto de dados NASA. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a AUC.

	Métodos CPDP						Melhor
Projeto	**DSSC**	**EASC**	**CC09-DT**	**M11-RF**	**T09-DT**	**W08-DT**	**Método**
CM1	**0,745**	0,742	0,585	0,543	0,568	0,550	DSSC
JM1	0,681	**0,694**	0,598	0,536	0,628	0,618	EASC
KC1	0,689	**0,793**	0,642	0,527	0,715	0,664	EASC
KC3	0,688	0,68	0,603	0,519	0,545	**0,725**	W08-DT
MC1	**0,894**	0,889	0,868	0,554	0,742	0,827	DSSC
MC2	0,628	**0,708**	0,570	0,543	0,610	0,629	EASC
MW1	**0,793**	0,747	0,712	0,533	0,583	0,758	DSSC
PC1	0,728	**0,755**	0,679	0,532	0,635	0,720	EASC
PC2	0,779	0,877	**0,894**	0,613	0,614	0,743	CC09-DT
PC3	0,687	**0,749**	0,676	0,516	0,576	0,701	EASC
PC4	0,650	**0,660**	0,652	0,506	0,558	0,648	EASC
PC5	**0,902**	0,828	0,881	0,567	0,684	0,447	DSSC

Fonte: Elaborada pelo autor (2022)

Tabela 41 – Resultados dos métodos CPDP para cada projeto do conjunto de dados RELINK. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a AUC.

	Métodos CPDP						Melhor
Projeto	**DSSC**	**EASC**	**CC09-DT**	**M11-RF**	**T09-DT**	**W08-DT**	**Método**
Apache2.0	**0,781**	0,734	0,676	0,701	0,729	0,522	DSSC
openintents	0,775	**0,840**	0,656	0,631	0,641	0,749	EASC
zxing1.6	**0,660**	0,657	0,612	0,625	0,534	0,534	DSSC

Fonte: Elaborada pelo autor (2022)

C.3 RESULTADOS COM *FALSE ALARM*

Tabela 42 – Resultados dos métodos CPDP para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a *False Alarm*.

Projeto	**Métodos CPDP**						**Melhor**
	DSSC	**EASC**	**CC09-DT**	**M11-RF**	**T09-DT**	**W08-DT**	**Método**
ant-1.3	0,067	0,086	0,390	0,200	0,295	0,200	DSSC
ant-1.4	0,109	0,145	0,428	0,188	0,304	0,210	DSSC
ant-1.5	0,100	0,119	0,352	0,169	0,253	0,280	DSSC
ant-1.6	0,085	0,108	0,297	0,208	0,228	0,220	DSSC
ant-1.7	0,064	0,083	0,316	0,206	0,193	0,287	DSSC
arc	0,048	0,058	0,208	0,300	0,155	0,444	DSSC
berek	0,000	0,000	0,037	0,037	0,037	0,185	DSSC, EASC
camel-1.0	0,031	0,061	0,273	0,138	0,196	0,248	DSSC
camel-1.2	0,033	0,051	0,268	0,161	0,181	0,179	DSSC
camel-1.4	0,073	0,077	0,180	0,191	0,150	0,276	DSSC
camel-1.6	0,068	0,084	0,184	0,169	0,198	0,292	DSSC
ckjm	0,000	0,000	0,000	0,000	0,000	0,800	DSSC, EASC, CC09-DT, M11-RF, T09-DT
e-learning	0,000	0,000	0,322	0,169	0,153	0,661	DSSC, EASC
forrest-0.7	0,000	0,000	0,250	0,167	0,292	0,208	DSSC, EASC
ivy-1.1	0,000	0,000	0,125	0,188	0,125	0,229	DSSC, EASC
ivy-1.4	0,076	0,084	0,249	0,196	0,142	0,169	DSSC
ivy-2.0	0,087	0,109	0,269	0,099	0,221	0,301	DSSC

Continua na próxima página

Tabela 42 – Resultados dos métodos CPDP para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a *False Alarm*. – continuação da página anterior

Projeto	**Métodos CPDP**						**Melhor**
	DSSC	**EASC**	**CC09-DT**	**M11-RF**	**T09-DT**	**W08-DT**	**Método**
jedit-3.2	0,082	0,099	0,236	0,198	0,187	0,198	DSSC
jedit-4.0	0,113	0,138	0,273	0,303	0,277	0,199	DSSC
jedit-4.1	0,094	0,120	0,253	0,300	0,275	0,318	DSSC
jedit-4.2	0,157	0,188	0,245	0,295	0,373	0,310	DSSC
jedit-4.3	0,187	0,227	0,358	0,393	0,339	0,289	DSSC
kalkulator	0,095	0,143	0,095	0,286	0,286	0,095	DSSC, CC09-DT, W08-DT
log4j-1.0	0,010	0,020	0,178	0,119	0,050	0,257	DSSC
log4j-1.1	0,014	0,014	0,153	0,097	0,111	0,181	DSSC, EASC
log4j-1.2	0,062	0,062	0,188	0,062	0,250	0,250	DSSC, EASC, M11-RF
lucene-2.0	0,000	0,029	0,144	0,115	0,067	0,135	DSSC
lucene-2.2	0,019	0,039	0,155	0,097	0,107	0,058	DSSC
lucene-2.4	0,036	0,044	0,190	0,139	0,109	0,212	DSSC
nieruchomosci	0,000	0,000	0,059	0,235	0,000	0,529	DSSC, EASC, T09-DT
pbeans1	0,000	0,000	0,000	0,000	0,000	0,167	DSSC, EASC, CC09-DT, M11-RF, T09-DT
pbeans2	0,024	0,024	0,268	0,049	0,073	0,098	DSSC, EASC

Continua na próxima página

Tabela 42 – Resultados dos métodos CPDP para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a *False Alarm*. – continuação da página anterior

Projeto	**Métodos CPDP**						**Melhor**
	DSSC	**EASC**	**CC09-DT**	**M11-RF**	**T09-DT**	**W08-DT**	**Método**
pdftranslator	0,000	0,000	0,111	0,000	0,000	0,333	DSSC, EASC, M11-RF, T09-DT
poi-1.5	0,031	0,042	0,062	0,177	0,135	0,219	DSSC
poi-2.0	0,072	0,105	0,144	0,462	0,292	0,206	DSSC
poi-2.5	0,073	0,080	0,146	0,139	0,168	0,387	DSSC
poi-3.0	0,050	0,050	0,106	0,149	0,143	0,242	DSSC, EASC
redaktor	0,034	0,074	0,188	0,262	0,523	0,430	DSSC
serapion	0,000	0,000	0,111	0,167	0,250	0,222	DSSC, EASC
skarbonka	0,028	0,083	0,111	0,167	0,306	0,306	DSSC
sklebagd	0,000	0,000	0,250	0,250	0,125	0,000	DSSC, EASC, W08-DT
synapse-1.0	0,028	0,064	0,348	0,170	0,156	0,418	DSSC
synapse-1.1	0,012	0,093	0,358	0,185	0,265	0,191	DSSC
synapse-1.2	0,035	0,094	0,359	0,182	0,065	0,141	DSSC
systemdata	0,018	0,018	0,196	0,125	0,143	0,125	DSSC, EASC
szybkafucha	0,000	0,000	0,364	0,000	0,182	0,364	DSSC, EASC, M11-RF
termoproject	0,000	0,000	0,069	0,207	0,241	0,379	DSSC, EASC
tomcat	0,105	0,128	0,201	0,218	0,137	0,182	DSSC
velocity-1.4	0,143	0,184	0,184	0,184	0,265	0,265	DSSC
velocity-1.5	0,042	0,056	0,083	0,236	0,097	0,264	DSSC

Continua na próxima página

Tabela 42 – Resultados dos métodos CPDP para cada projeto do conjunto de dados PROMISE. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a *False Alarm*. – continuação da página anterior

Projeto	**Métodos CPDP**						**Melhor**
	DSSC	**EASC**	**CC09-DT**	**M11-RF**	**T09-DT**	**W08-DT**	**Método**
velocity-1.6	0,026	0,040	0,126	0,106	0,146	0,411	DSSC
workflow	0,000	0,000	0,158	0,105	0,000	0,421	DSSC, EASC, T09-DT
wspomaganiepi	0,000	0,000	0,000	0,333	0,167	0,333	DSSC, EASC, CC09-DT
xalan-2.4	0,135	0,245	0,196	0,168	0,217	0,325	DSSC
xalan-2.5	0,115	0,233	0,171	0,197	0,166	0,303	DSSC
xalan-2.6	0,089	0,165	0,162	0,177	0,181	0,253	DSSC
xalan-2.7	0,000	0,000	0,000	0,000	0,091	0,000	DSSC, EASC, CC09-DT, M11-RF, W08-DT
xerces-1.2	0,092	0,106	0,322	0,171	0,241	0,285	DSSC
xerces-1.3	0,060	0,081	0,305	0,135	0,208	0,253	DSSC
xerces-1.4	0,026	0,020	0,093	0,060	0,060	0,053	EASC
xerces-init	0,129	0,165	0,459	0,247	0,235	0,353	DSSC
zuzel	0,000	0,000	0,250	0,125	0,188	0,125	DSSC, EASC

Fonte: Elaborada pelo autor (2022)

Tabela 43 – Resultados dos métodos CPDP para cada projeto do conjunto de dados AEEEM. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a *False Alarm*.

	Métodos CPDP						Melhor
Projeto	**DSSC**	**EASC**	**CC09-DT**	**M11-RF**	**T09-DT**	**W08-DT**	**Método**
eclipse	**0,029**	0,307	0,038	0,133	0,312	0,068	DSSC
equinox	**0,005**	0,041	**0,005**	0,015	0,077	0,056	DSSC, CC09-DT
lucene	**0,002**	0,042	0,091	0,016	0,072	0,152	DSSC
mylyn	0,034	0,388	0,088	**0,014**	0,053	0,115	M11-RF
pde	**0,037**	0,102	0,095	0,058	0,142	0,156	DSSC

Fonte: Elaborada pelo autor (2022)

Tabela 44 – Resultados dos métodos CPDP para cada projeto do conjunto de dados RELINK. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a *False Alarm*.

	Métodos CPDP						Melhor
Projeto	**DSSC**	**EASC**	**CC09-DT**	**M11-RF**	**T09-DT**	**W08-DT**	**Método**
Apache2.0	0,240	0,542	0,208	0,261	0,333	**0,083**	W08-DT
openintents	**0,000**	**0,000**	0,118	0,147	0,118	0,176	DSSC, EASC
zxing1.6	0,103	**0,000**	0,310	0,224	0,050	0,246	EASC

Fonte: Elaborada pelo autor (2022)

Tabela 45 – Resultados dos métodos CPDP para cada projeto do conjunto de dados NASA. Para cada projeto é apresentado uma ou mais abordagens que obtiveram o melhor resultado em relação a *False Alarm*.

	Métodos CPPD						Melhor
Projeto	**DSSC**	**EASC**	**CC09-DT**	**M11-RF**	**T09-DT**	**W08-DT**	**Método**
CM1	0,053	0,106	**0,007**	0,129	0,185	0,023	CC09-DT
JM1	0,032	0,044	**0,006**	0,046	0,107	0,023	CC09-DT
KC1	0,014	**0,007**	0,021	0,025	0,037	0,035	EASC
KC3	**0,006**	0,055	**0,006**	0,018	0,049	0,012	DSSC, CC09-DT
MC1	**0,004**	0,011	0,013	0,010	0,019	0,040	DSSC
MC2	0,036	0,06	0,012	0,072	0,060	**0,000**	W08-DT
MW1	**0,000**	0,008	0,004	0,008	0,076	0,008	DSSC
PC1	**0,011**	0,095	**0,011**	0,019	0,027	0,019	DSSC, CC09-DT
PC2	**0,003**	0,014	0,019	0,024	0,010	0,015	DSSC
PC3	0,021	0,081	**0,012**	0,031	0,026	0,020	CC09-DT
PC4	**0,006**	0,055	0,020	0,021	0,019	0,020	DSSC
PC5	**0,004**	0,008	0,023	0,008	**0,004**	0,008	DSSC, T09-DT

Fonte: Elaborada pelo autor (2022)

APÊNDICE D – RESULTADOS EXPERIMENTAIS CONSIDERANDO AS META-FEATURES

Neste apêndice será apresentado os meta-dados utilizados para verificar qual o melhor método CPDP de acordo com as meta-features definidas (consultar seção 5.4). Para cada exemplo, são listados e calculados os valores de cada meta-feature com base nas informações originais dos projetos nos conjuntos de dados. Por último, é atribuído a classe que corresponde ao método que apresentou o melhor desempenho de acordo com conjunto de informações das meta-features. Para melhor visualização e como o objetivo é identificar se DSSC deve ser o método a ser aplicado em determinado projeto conforme suas características (meta-features), logo, é atribuído o valor 1 à classe caso o DSSC apresente o melhor resultado; caso contrário, é atribuído o valor 0.

D.1 RESULTADOS COM F1-SCORE

Tabela 46 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados RELINK conforme a métrica F1-score.

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
eclipse	997	62	20,66	58,68	1
equinox	324	62	39,81	20,37	0
lucene	691	62	9,26	81,48	0
mylyn	1862	62	13,16	73,68	0
pde	1497	62	13,96	72,08	1

Fonte: Elaborada pelo autor (2022)

Tabela 47 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados PROMISE conforme a métrica F1-score.

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
ant-1.3	125	24	16,00	68,00	0
ant-1.4	178	24	22,47	55,06	0
ant-1.5	293	24	10,92	78,16	1
ant-1.6	351	24	26,21	47,58	0
ant-1.7	745	24	22,28	55,44	0
arc	234	24	11,54	76,92	0

Continua na próxima página

Tabela 47 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados PROMISE conforme a métrica F1-score. – continuação da página anterior

Projeto	**Instâncias**	***Features***	**Defeitos (%)**	**Desbalanceamento**	**Classe**
berek	43	24	37,21	25,58	0
camel-1.0	339	24	3,83	92,33	0
camel-1.2	608	24	35,53	28,95	0
camel-1.4	872	24	16,63	66,74	0
camel-1.6	965	24	19,48	61,04	0
ckjm	10	24	50,00	0,00	0
e-learning	64	24	7,81	84,38	0
forrest-0.7	29	24	17,24	65,52	0
ivy-1.1	111	24	56,76	13,51	0
ivy-1.4	241	24	6,64	86,72	1
ivy-2.0	352	24	11,36	77,27	1
jedit-3.2	272	24	33,09	33,82	0
jedit-4.0	306	24	24,51	50,98	0
jedit-4.1	312	24	25,32	49,36	0
jedit-4.2	367	24	13,08	73,84	1
jedit-4.3	492	24	2,24	95,53	1
kalkulator	27	24	22,22	55,56	0
log4j-1.0	135	24	25,19	49,63	0
log4j-1.1	109	24	33,94	32,11	0
log4j-1.2	205	24	92,20	84,39	0
lucene-2.0	195	24	46,67	6,67	0
lucene-2.2	247	24	58,3	16,60	0
lucene-2.4	340	24	59,71	19,41	0
nieruchomosci	27	24	37,04	25,93	0
pbeans1	26	24	76,92	53,85	0
pbeans2	51	24	19,61	60,78	0
pdftranslator	33	24	45,45	9,09	0
poi-1.5	237	24	59,49	18,99	0
poi-2.0	314	24	11,78	76,43	0
poi-2.5	385	24	64,42	28,83	0
poi-3.0	442	24	63,57	27,15	0
redaktor	176	24	15,34	69,32	0
serapion	45	24	20,00	60,0	0
skarbonka	45	24	20,00	60,0	0
sklebagd	20	24	60,00	20,0	0
synapse-1.0	157	24	10,19	79,62	0

Continua na próxima página

Tabela 47 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados PROMISE conforme a métrica F1-score. – continuação da página anterior

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
synapse-1.1	222	24	27,03	45,95	0
synapse-1.2	256	24	33,59	32,81	0
systemdata	65	24	13,85	72,31	0
szybkafucha	25	24	56,00	12,00	0
termoproject	42	24	30,95	38,10	0
tomcat	858	24	8,97	82,05	0
velocity-1.4	196	24	75,00	50,00	0
velocity-1.5	214	24	66,36	32,71	0
velocity-1.6	229	24	34,06	31,88	0
workflow	39	24	51,28	2,56	0
wspomaganiepi	18	24	66,67	33,33	0
xalan-2.4	723	24	15,21	69,57	1
xalan-2.5	803	24	48,19	3,61	0
xalan-2.6	885	24	46,44	7,12	0
xalan-2.7	909	24	98,79	97,58	0
xerces-1.2	440	24	16,14	67,73	0
xerces-1.3	453	24	15,23	69,54	0
xerces-1.4	588	24	74,32	48,64	0
xerces-init	162	24	47,53	4,94	0
zuzel	29	24	44,83	10,34	0

Fonte: Elaborada pelo autor (2022)

Tabela 48 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados RELINK conforme a métrica F1-score.

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
Apache2.0	194	27	50,52	1,03	1
openintents	56	27	39,29	21,43	0
zxing1.6	399	27	29,57	40,85	0

Fonte: Elaborada pelo autor (2022)

Tabela 49 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados NASA conforme a métrica F1-score.

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
CM1	344	21	12,21	75,58	0
JM1	9593	21	18,34	63,33	0
KC1	2096	21	15,51	68,99	0
KC3	200	21	18,00	64,00	0
MC1	9277	21	0,73	98,53	0
MC2	127	21	34,65	30,71	0
MW1	264	21	10,23	79,55	0
PC1	759	21	8,04	83,93	0
PC2	1585	21	1,01	97,98	0
PC3	1125	21	12,44	75,11	0
PC4	1399	21	12,72	74,55	0
PC5	17001	21	2,96	94,08	0

Fonte: Elaborada pelo autor (2022)

D.2 RESULTADOS COM AUC

Tabela 50 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados AEEEM conforme a métrica AUC.

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
eclipse	997	62	20,66	0.59	1
equinox	324	62	39,81	0.20	0
lucene	691	62	9,26	0.82	1
mylyn	1862	62	13,16	0.74	1
pde	1497	62	13,96	0.72	1

Fonte: Elaborada pelo autor (2022)

Tabela 51 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados PROMISE conforme a métrica AUC.

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
ant-1.3	125	24	16,00	68,00	1
ant-1.4	178	24	22,47	55,06	0
ant-1.5	293	24	10,92	78,16	0
ant-1.6	351	24	26,21	47,58	1

Continua na próxima página

Tabela 51 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados PROMISE conforme a métrica AUC. – continuação da página anterior

Projeto	**Instâncias**	***Features***	**Defeitos (%)**	**Desbalanceamento**	**Classe**
ant-1.7	745	24	22,28	55,44	0
arc	234	24	11,54	76,92	0
berek	43	24	37,21	25,58	1
camel-1.0	339	24	3,83	92,33	0
camel-1.2	608	24	35,53	28,95	0
camel-1.4	872	24	16,63	66,74	1
camel-1.6	965	24	19,48	61,04	0
ckjm	10	24	50,0	0,00	1
e-learning	64	24	7,81	84,38	1
forrest-0.7	29	24	17,24	65,52	0
ivy-1.1	111	24	56,76	13,51	1
ivy-1.4	241	24	6,64	86,72	1
ivy-2.0	352	24	11,36	77,27	0
jedit-3.2	272	24	33,09	33,82	0
jedit-4.0	306	24	24,51	50,98	1
jedit-4.1	312	24	25,32	49,36	1
jedit-4.2	367	24	13,08	73,84	0
jedit-4.3	492	24	2,24	95,53	1
kalkulator	27	24	22,22	55,56	0
log4j-1.0	135	24	25,19	49,63	0
log4j-1.1	109	24	33,94	32,11	1
log4j-1.2	205	24	92,20	84,39	0
lucene-2.0	195	24	46,67	6,67	1
lucene-2.2	247	24	58,30	16,60	0
lucene-2.4	340	24	59,71	19,41	0
nieruchomosci	27	24	37,04	25,93	0
pbeans1	26	24	76,92	53,85	0
pbeans2	51	24	19,61	60,78	0
pdftranslator	33	24	45,45	9,09	1
poi-1.5	237	24	59,49	18,99	0
poi-2.0	314	24	11,78	76,43	0
poi-2.5	385	24	64,42	28,83	1
poi-3.0	442	24	63,57	27,15	1
redaktor	176	24	15,34	69,32	0
serapion	45	24	20,0	60,00	0
skarbonka	45	24	20,0	60,00	0

Continua na próxima página

Tabela 51 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados PROMISE conforme a métrica AUC. – continuação da página anterior

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
sklebagd	20	24	60,0	20,0	0
synapse-1.0	157	24	10,19	79,62	1
synapse-1.1	222	24	27,03	45,95	0
synapse-1.2	256	24	33,59	32,81	0
systemdata	65	24	13,85	72,31	0
szybkafucha	25	24	56,0	12,00	0
termoproject	42	24	30,95	38,10	1
tomcat	858	24	8,97	82,05	1
velocity-1.4	196	24	75,0	50,00	0
velocity-1.5	214	24	66,36	32,71	0
velocity-1.6	229	24	34,06	31,88	1
workflow	39	24	51,28	2,56	1
wspomaganiepi	18	24	66,67	33,33	1
xalan-2.4	723	24	15,21	69,57	1
xalan-2.5	803	24	48,19	3,61	0
xalan-2.6	885	24	46,44	7,12	0
xalan-2.7	909	24	98,79	97,58	1
xerces-1.2	440	24	16,14	67,73	0
xerces-1.3	453	24	15,23	69,54	0
xerces-1.4	588	24	74,32	48,64	0
xerces-init	162	24	47,53	4,94	1
zuzel	29	24	44,83	10,34	1

Fonte: Elaborada pelo autor (2022)

Tabela 52 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados RELINK conforme a métrica AUC.

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
Apache2.0	194	27	50,52	1,03	1
openintents	56	27	39,29	21,43	0
zxing1.6	399	27	29,57	40,85	1

Fonte: Elaborada pelo autor (2022)

Tabela 53 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados NASA conforme a métrica AUC.

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
CM1	344	21	12,21	75,58	1
JM1	9593	21	18,34	63,33	0
KC1	2096	21	15,51	68,99	0
KC3	200	21	18,00	64,00	0
MC1	9277	21	0,73	98,53	1
MC2	127	21	34,65	30,71	0
MW1	264	21	10,23	79,55	1
PC1	759	21	8,04	83,93	0
PC2	1585	21	1,01	97,98	0
PC3	1125	21	12,44	75,11	0
PC4	1399	21	12,72	74,55	0
PC5	17001	21	2,96	94,08	1

Fonte: Elaborada pelo autor (2022)

D.3 RESULTADOS COM *FALSE ALARM*

Tabela 54 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados AEEEM conforme a métrica False Alarm.

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
eclipse	997	62	20,66	58,68	1
equinox	324	62	39,81	20,37	0
lucene	691	62	9,26	81,48	1
mylyn	1862	62	13,16	73,68	0
pde	1497	62	13,96	72,08	1

Fonte: Elaborada pelo autor (2022)

Tabela 57 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados PROMISE conforme a métrica False Alarm.

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
ant-1.3	125	24	16,00	68,00	1
ant-1.4	178	24	22,47	55,06	1
ant-1.5	293	24	10,92	78,16	1
ant-1.6	351	24	26,21	47,58	1

Continua na próxima página

Tabela 57 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados PROMISE conforme a métrica False Alarm. – continuação da página anterior

Projeto	**Instâncias**	***Features***	**Defeitos (%)**	**Desbalanceamento**	**Classe**
ant-1.7	745	24	22,28	55,44	1
arc	234	24	11,54	76,92	1
berek	43	24	37,21	25,58	0
camel-1.0	339	24	3,83	92,33	1
camel-1.2	608	24	35,53	28,95	1
camel-1.4	872	24	16,63	66,74	1
camel-1.6	965	24	19,48	61,04	1
ckjm	10	24	50,00	0,00	0
e-learning	64	24	7,81	84,38	0
forrest-0.7	29	24	17,24	65,52	0
ivy-1.1	111	24	56,76	13,51	0
ivy-1.4	241	24	6,64	86,72	1
ivy-2.0	352	24	11,36	77,27	1
jedit-3.2	272	24	33,09	33,82	1
jedit-4.0	306	24	24,51	50,98	1
jedit-4.1	312	24	25,32	49,36	1
jedit-4.2	367	24	13,08	73,84	1
jedit-4.3	492	24	2,24	95,53	1
kalkulator	27	24	22,22	55,56	0
log4j-1.0	135	24	25,19	49,63	1
log4j-1.1	109	24	33,94	32,11	0
log4j-1.2	205	24	92,20	84,39	0
lucene-2.0	195	24	46,67	6,67	1
lucene-2.2	247	24	58,30	16,60	1
lucene-2.4	340	24	59,71	19,41	1
nieruchomosci	27	24	37,04	25,93	0
pbeans1	26	24	76,92	53,85	0
pbeans2	51	24	19,61	60,78	0
pdftranslator	33	24	45,45	9,09	0
poi-1.5	237	24	59,49	18,99	1
poi-2.0	314	24	11,78	76,43	1
poi-2.5	385	24	64,42	28,83	1
poi-3.0	442	24	63,57	27,15	0
redaktor	176	24	15,34	69,32	1
serapion	45	24	20,00	60,00	0
skarbonka	45	24	20,00	60,00	1

Continua na próxima página

Tabela 57 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados PROMISE conforme a métrica False Alarm. – continuação da página anterior

Projeto	**Instâncias**	***Features***	**Defeitos (%)**	**Desbalanceamento**	**Classe**
sklebagd	20	24	60,00	20,00	0
synapse-1.0	157	24	10,19	79,62	1
synapse-1.1	222	24	27,03	45,95	1
synapse-1.2	256	24	33,59	32,81	1
systemdata	65	24	13,85	72,31	0
szybkafucha	25	24	56,00	12,00	0
termoproject	42	24	30,95	38,10	0
tomcat	858	24	8,97	82,05	1
velocity-1.4	196	24	75,00	50,00	1
velocity-1.5	214	24	66,36	32,71	1
velocity-1.6	229	24	34,06	31,88	1
workflow	39	24	51,28	2,56	0
wspomaganiepi	18	24	66,67	33,33	0
xalan-2.4	723	24	15,21	69,57	1
xalan-2.5	803	24	48,19	3,61	1
xalan-2.6	885	24	46,44	7,12	1
xalan-2.7	909	24	98,79	97,58	0
xerces-1.2	440	24	16,14	67,73	1
xerces-1.3	453	24	15,23	69,54	1
xerces-1.4	588	24	74,32	48,64	0
xerces-init	162	24	47,53	4,94	1
zuzel	29	24	44,83	10,34	0

Fonte: Elaborada pelo autor (2022)

Tabela 55 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados RELINK conforme a métrica False Alarm.

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
Apache2.0	194	27	50,52	1,03	0
openintents	56	27	39,29	21,43	0
zxing1.6	399	27	29,57	40,85	0

Fonte: Elaborada pelo autor (2022)

Tabela 56 – Meta-dados utilizados pelo meta-classificador em relação aos projetos do conjunto de dados NASA conforme a métrica False Alarm.

Projeto	Instâncias	*Features*	Defeitos (%)	Desbalanceamento	Classe
CM1	344	21	12,21	75,58	0
JM1	9593	21	18,34	63,33	0
KC1	2096	21	15,51	68,99	0
KC3	200	21	18,00	64,00	0
MC1	9277	21	0,73	98,53	1
MC2	127	21	34,65	30,71	0
MW1	264	21	10,23	79,55	1
PC1	759	21	8,04	83,93	0
PC2	1585	21	1,01	97,98	1
PC3	1125	21	12,44	75,11	0
PC4	1399	21	12,72	74,55	1
PC5	17001	21	2,96	94,08	0

Fonte: Elaborada pelo autor (2022)