UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

SAMUEL DE PAULA

UM ESTUDO EXPLORATÓRIO SOBRE A CRIAÇÃO DE MODELOS DE PREDIÇÃO CRUZADA DE DEFEITOS APOIADA POR UMA MEDIDA DE CORRELAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO

2016

SAMUEL DE PAULA

UM ESTUDO EXPLORATÓRIO SOBRE A CRIAÇÃO DE MODELOS DE PREDIÇÃO CRUZADA DE DEFEITOS APOIADA POR UMA MEDIDA DE CORRELAÇÃO

Trabalho de Conclusão de Curso de graduação apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso de Bacharelado em Ciência da Computação do Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Reginaldo Ré

CAMPO MOURÃO

2016

Resumo

PAULA, Samuel. Um Estudo Exploratório Sobre a Criação de Modelos de Predição Cruzada de Defeitos Apoiada por uma Medida de Correlação. 2016. 47. f. Trabalho de Conclusão de Curso (Curso de Bacharelado em Ciência da Computação), Universidade Tecnológica Federal do Paraná. Campo Mourão, 2016.

Predizer defeitos em software é uma tarefa difícil, principalmente quando o projeto alvo está em fase inicial de desenvolvimento, pois nessa fase o projeto não possui uma base de dados histórica consolidada que possa ser utilizada para treinar um modelo de predição. Neste caso, é preciso encontrar projetos semelhantes ao projeto alvo para que, a partir de seus dados históricos, modelos de predição possam ser elaborados, essa técnica recebe o nome de predição cruzada de defeitos. Esse trabalho tem como objetivo analisar uma forma alternativa para o agrupamento de projetos. A proposta consiste em agrupar modelos de predição de forma que possam compartilhar seus conjuntos de dados de treinamento. Os agrupamentos foram criados pelo algoritmo de clusterização BSAS (do inglês, *Basic Sequential Algorithmic Scheme*), utilizando uma medida de correlação denominada MCC (do inglês, *Matthews correlation coefficient*) para cálculo de semelhança dos modelos. Uma vez agrupados, os dados de treinamento dos modelos podem formar um único conjunto de treinamento, dando origem a modelos de predição cruzada de defeitos. Os resultados obtidos através da análise dos valores de MCC obtidos pelos modelos de predição indicam que os modelos criados possuem baixo desempenho preditivo enquanto que os indicadores utilizados em trabalhos semelhantes indicam modelos com bom desempenho. Portanto, conclui-se que, a medida de correlação MCC é uma medida de desempenho mais robusta que as demais para análise de modelos de predição e que ela contribui para a redução do problema de desbalanceamento de classes.

Palavras-chaves: agrupamento por similaridade, predição cruzada de defeitos, predição de defeitos

Abstract

PAULA, Samuel. An Exploratory Study About Cross Defects Prediction Supported by a Correlation Measure. 2016. 47. f. Capstone Project (Computer Science), Federal University of Technology – Paraná. Campo Mourão – PR – Brazil, 2016.

Defect prediction in software is a difficult task, especially when the target project is in early stage of development, because in this stage the project does not have historical data that can be used to train a prediction model. In this case, it is necessary to find projects that matches to the target project in order to use their historical data. This aproach is named cross-project prediction. This work aims to analyze an alternative method to clustering the projects. Our proposal is to group prediction models so that they can share their sets of training data. The groups were created by clustering algorithm called Basic Sequential Algorithmic Scheme (BSAS), using a measure of correlation known as Matthews Correlation Coefficient (MCC) to compute similarity models. When grouped, the training data models results to a single training set, giving rise to models cross-project prediction. The results obtained by analyzing the MCC values computed by prediction models indicate that the created models have low predictive performance, while the indicators used in similar studies indicate models with good performance. Therefore, we can concluded that the MCC correlation measure is a more robust measure of performance than other prediction models, and it contributes to the reduction of the class imbalance problem.

Keywords: clustering by similarity, cross-project prediction, defect prediction

Lista de figuras

Lista de tabelas

Sumário

Capítulo 1

Introdução

1.1. Contextualização e Motivação

Predizer defeitos consiste em detectar a ocorrência de falhas antes que elas se manifestem durante a execução do software. A detecção pode ser feita com a criação, treino e teste de modelos de predição, que faz uso de dados históricos do projeto para realizar a predição. Predizer defeitos tem muitas vantagens, como por exemplo: permitir que a equipe de desenvolvimento direcione seus esforços para a detecção de defeitos; reduzir o tempo gasto pela equipe de desenvolvimento na detecção de defeitos no projeto; e, aumentar a percepção de qualidade do software ao fim de sua implementação (Catal; Diri, 2009; Hall *et al.*, 2012). Apesar das vantagens, realizar predição de defeitos em software não é uma tarefa trivial, principalmente em projetos que estão em fase inicial de desenvolvimento (Zimmermann *et al.*, 2009). Nessa etapa, os projetos não possuem dados históricos suficientes que possam ser utilizados para treinar um modelo de predição. A alternativa é utilizar dados históricos de outros projetos (Turhan *et al.*, 2009; Menzies *et al.*, 2011; Hall *et al.*, 2012) e, para isso, se faz necessário a seleção de projetos que sejam similares ao projeto que se deseja realizar a predição. Essa técnica recebe o nome de predição cruzada de defeitos.

Uma das principais dificuldades para utilização de predição cruzada de defeitos é encontrar uma forma que permita agrupar projetos que sejam semelhantes, no que diz respeito aos seus defeitos. O método mais utilizado para encontrar semelhanças entre projetos é utilizar as próprias métricas dos projetos, tornando necessário a seleção de projetos que contenham atributos que permitam essa comparação (Zimmermann *et al.*, 2009; Jureczko; Madeyski, 2010; Menzies *et al.*, 2011; Zhang *et al.*, 2014). Recentemente, uma nova abordagem foi proposta por Satin *et al.* (2015b), que consiste em agrupar modelos de predições que utilizam apenas a base de dados histórica do projeto alvo para realizar a predição. Para isso foi utilizada uma medida de desempenho denominada AUC (do inglês, *Area Under the Curve*) que é um indicador de desempenho que representa o grau de confiabilidade das predições realizadas (Satin *et al.*, 2015b). Os modelos de predição são agrupados por um algoritmo de clusterização que utiliza o valor de desempenho destes para cálculo de semelhança. Quando agrupados os modelos compartilham de suas bases de dados a fim de formar um único conjunto de dados para treinamento, formando-se assim modelos de predição cruzada de defeitos. Seguindo a mesma abordagem de Satin *et al.* (2015b), o presente trabalho analisou o desempenho de modelos de predição cruzada criados a partir do agrupamento de projetos tendo o valor de MCC (do inglês, *Matthews correlation*

coefficient) para cálculo de similaridade. Este valor é calculado através de uma predição prévia que utiliza apenas os dados históricos do projeto alvo.

Além dos métodos de agrupamento de projetos por algum tipo de semelhança, no caso específico deste estudo MCC, é necessária a seleção e a utilização de algoritmos de aprendizado de máquina que façam uso do conjunto de dados para "aprender" sobre fatores que levam a ocorrência de falhas ou ausência delas em projetos de software. Um fator relevante que pode influenciar no desempenho do aprendizado desses algoritmos é a utilização de um método que selecione um subconjunto de atributos que tenham maior influência na detecção de defeitos. Além de obter uma melhora no desempenho esse método também reduz a dimensionalidade do conjunto de dados. Contudo, a questão que envolve a escolha dos melhores algoritmos classificadores e algoritmos de seleção de métricas ainda é um assunto em aberto na literatura. Esse motivo levou a escolha de algoritmos diversos para criação de modelos de predição cruzada de defeitos.

1.2. Objetivos

O principal objetivo do trabalho é analisar o desempenho de modelos de predição cruzada criados a partir do agrupamento de projetos tendo uma medida de correlação denominada MCC para cálculo de similaridade. Este valor é calculado através de uma predição prévia que utiliza apenas os dados históricos do projeto alvo e sua utilização deve-se a dois fatores: MCC trata-se de uma medida de desempenho mais robusta frente as demais, pois seu cálculo leva em consideração os erros e acertos das predições realizadas para ambas as classes (contêm defeitos ou livre de defeitos); outro fator determinístico para sua utilização é que entre os trabalhos relacionados nenhum utiliza MCC para medir o desempenho de seus modelos, sendo assim, encontrou-se a possibilidade de enriquecer o estado da arte com análises sobre modelos que utilizam uma medida de desempenho ainda pouco utilizada.

O valor de MCC serviu de entrada para o algoritmo de agrupamento BSAS (do inglês, *Basic Sequencial Algorithmic Scheme*) (Kainulainen; Kainulainen, 2002), fazendo uso deste para cálculo de semelhança entre os modelos de predição, permitindo que os dados que formam o conjunto de treinamento de modelos presentes em um mesmo agrupamento possam ser utilizados como um único conjunto de dados para treinamento, dando origem aos modelos de predição cruzada de defeitos.

Sendo assim o objetivo geral deste trabalho está norteado pelos seguintes objetivos específicos:

- Analisar o desempenho de modelos de predição cruzada de defeitos criados pelo algoritmo de clusterização BSAS tendo o valor de MCC para cálculo de semelhança entre projetos de software;
- Analisar se existe uma redução do problema de desbalanceamento de classes com a utilização de MCC para cálculo de semelhança entre projetos de software;
- Analisar a influência de algoritmos classificadores no processo de criação de modelos de predição cruzada de defeitos;
- Analisar a influência de algoritmos de seleção de atributos no processo de criação de modelos de predição cruzada de defeitos;

1.3. Organização

Esse documento está organizado da seguinte forma: no Capítulo 2, são apresentados os conceitos gerais relacionados com aprendizado de máquina, conceitos específicos que levam a um melhor entendimento do trabalho em questão e uma breve descrição dos principais trabalhos relacionados; no Capítulo 3, são apresentados os passos para desenvolvimento do trabalho, bem como os métodos de avaliação empregados sobre os resultados alcançados; e por fim, o Capítulo 4 traz as principais observações sobre os resultados alcançados e também comparações com o estado da arte.

Capítulo 2

Fundamentação Teórica

2.1. Conceitos Básicos de Aprendizado de Máquina

2.1.1. Conjunto de Dados

Um conjunto de dados (do inglês, *dataset*) caracteriza objetos do mundo real a partir de um conjunto de atributos, também chamado de vetor de características (Jain *et al.*, 1999). Esses atributos representam características do objeto, por exemplo, um conjunto de dados composto por projetos de software contém atributos do tipo: número de linhas de código fonte, linguagens de programação utilizadas na implementação, grau de experiência dos programadores que participaram da implementação do projeto, entre outras características que poderiam caracterizar um projeto de software. Outro tipo de atributo é aquele que representa a qual classe o objeto pertence, chamado de atributo alvo ou rótulo. O valor desse atributo depende do vetor de característica e não é obrigatório que o conjunto de dados o contenha. O domínio do atributo alvo representa o tipo de problema que está sendo tratado (Alpaydin, 2010). Quando os valores são numéricos trata-se de um problema de regressão, quando os valores são rotulados trata-se de um problema de classificação.

2.1.2. Algoritmos de Aprendizado de Máquina

Algoritmos de aprendizado de máquina são utilizados na solução de vários problemas (Russell; Norvig, 2009; Alpaydin, 2010; Faceli *et al.*, 2011; Ghotra *et al.*, 2015). Entre os problemas que podem ser resolvidos com a utilização de algoritmos de aprendizado de máquina pode-se destacar: reconhecimento facial, reconhecimento de fala, reconhecimento de escrita, diagnósticos clínicos e também, predição de defeitos em software.

Os algoritmos podem resolver problemas de classificação, regressão, agrupamento (do inglês *clustering*) ou extração de regras (Alpaydin, 2010; Faceli *et al.*, 2011). O tipo de problema a ser tratado depende do domínio do atributo alvo presente no conjunto de dados utilizado pelo algoritmo. O problema será de classificação quando o valor do atributo alvo for nominal, por exemplo, classificar se um paciente pode ser

diagnosticado com uma determinada doença ou classificar se um software contém defeito ou está ausente de defeitos. Para problemas onde o objetivo é prever um determinado valor numérico o valor do atributo alvo deve ser numérico. Por exemplo, prever quais serão os números sorteados pela loteria. Quando o problema exige o agrupamento dos dados por algum tipo de semelhança é necessário que os dados contenham rótulos, um exemplo desse tipo de problema é agrupar clientes que tenham alto potencial para compra de um determinado veículo ou agrupar projetos de software que tem alto potencial para conterem defeitos. Outro tipo de algoritmo de aprendizado de máquina trabalham com extração de regras, eles são utilizados para se estabelecer algum tipo de relação entre os elementos, como por exemplo, verificar se clientes que compram um certo produto tendem a comprar outro produto, ou ainda, verificar se desenvolvedores que cometem um tipo de erro também cometem outro tipo de erro.

Os algoritmos possuem estilos de aprendizagem (Alpaydin, 2010): aprendizado supervisionado (do inglês, *supervisioned learning*); aprendizado não supervisionado (do inglês, *unsupervisioned learning*); aprendizado semi-supervisionado (do inglês, *semi-supervisioned learning*); e, aprendizado por reforço (do inglês, *reinforcement learning*). No aprendizado supervisionado os algoritmos são treinados com um conjunto de dados para que ele possa "aprender"sobre determinado problema, o conjunto de dados obrigatoriamente deve conter o atributo alvo. Após o treinamento as predições são realizadas para outros conjuntos de dados, os chamados conjunto de teste. Esse tipo de aprendizagem é bastante utilizado na resolução de problemas de classificação ou regressão.

No aprendizado não supervisionado, o conjunto de dados utilizado para treino do algoritmo não contém o atributo alvo, sendo assim o algoritmo faz deduções sobre o conjunto de dados que lhe está sendo apresentado. Esse estilo de aprendizagem é bastante utilizado na solução de problemas de agrupamento com extração de regras.

O aprendizado semi-supervisionado engloba os outros dois estilos de aprendizagem, onde atua sobre dados rotulados e dados não rotulados. Assim como na aprendizagem supervisionada, o algoritmo deve "aprender"com um conjunto de dados que lhe é fornecido como entrada. Assim como algoritmos de aprendizado supervisionado, é um estilo bastante utilizado em problemas de classificação e regressão.

Por fim, os algoritmos de aprendizagem apoiados por reforço recebem um conjunto de dados como entrada que representa estímulos, e o algoritmo deve responder e reagir de acordo com esses estímulos. Seu aprendizado ocorre através de punições e recompensas que o algoritmo recebe, e é bastante utilizado para solucionar problemas de robótica.

2.1.3. Algoritmos de Seleção de Atributos

A técnica de seleção de atributos permite selecionar os atributos mais relevantes do conjunto de dados durante do processo de predição (Guyon; Elisseeff, 2003; Khoshgoftaar; Gao, 2009; Muthukumaran *et al.*, 2015). Para tal finalidade, pode ser feita a combinação de algoritmos que possibilitem identificar os algoritmos mais relevantes.

De uma forma geral, a utilização da técnica de seleção de atributos com algoritmos de aprendizado de máquina para predição de defeitos tem o objetivo de otimizar o processo de predição, uma vez que ocorre uma redução da dimensionalidade dos atributos e o gasto de processamento computacional é reduzido. Também é objetivo da técnica de seleção de atributos melhorar o desempenho do modelo de predição, uma vez que os atributos selecionados são os que mais se relacionam com o atributo alvo.

A técnica de seleção de atributos pode ser utilizada de duas formas, como filtro ou como empacotamento. Os algoritmos de seleção de atributos que funcionam como filtro consideram que os atributos do conjunto de dados são independentes, ou seja, que não existe nenhuma relação entre eles para identificação do atributo alvo. Uma vantagem da utilização da técnica como filtro é que não existe dependência dos algoritmos de aprendizado de máquina para execução da técnica de seleção de atributos. Ao contrário da técnica de filtro, a técnica de empacotamento consiste na utilização do algoritmo de aprendizado de máquina em conjunto com algoritmos de seleção de atributos, criando-se diferentes combinações. Um dos grandes benefícios da utilização da técnica como empacotamento em relação ao método de filtro é a possibilidade de estabelecer dependência entre os atributos, contudo o método de empacotamento apresenta algumas desvantagens como: dependência entre algoritmo de aprendizado de máquina e algoritmo de seleção de atributos; e, aumento de custo de execução, se comparado com o método de filtro.

Existem duas outras formas para categorizar as técnicas de seleção de atributos, como ranqueadoras (do inglês, *ranking*) ou selecionadoras de subconjunto (do inglês, *subset selection*) (Gao; Khoshgoftaar, 2015). As ranqueadoras consistem em criar um ranking dos atributos, onde o primeiro colocado é o atributo com maior relação com o atributo alvo e o último colocado é o atributo que exerce menor influência sobre o atributo alvo. Já o método selecionadoras de subconjunto consiste em extrair do conjunto de atributos um subconjunto contendo os atributos que mais exercem influência sobre o atributo alvo.

Independente da forma como as técnicas de atributos são categorizadas, seja como filtro, empacotamento, ranqueadora ou selecionadora de subconjunto, seu funcionamento depende da utilização de dois algoritmos em conjunto (Chandrashekar; Sahin, 2014; Gao; Khoshgoftaar, 2015). O primeiro tipo estabelece uma forma de como os atributos de maior influência sobre o atributo alvo serão pesquisados e encontrados, chamados de métodos de pesquisa (do inglês, *Search Method*). O segundo tipo são algoritmos que estabelecem uma relação entre os atributos que foram encontrados pelo algoritmo de método de pesquisa, aplicando uma análise sobre eles, esse tipo é chamado de algoritmo de seleção de atributos (do inglês, *attribute evaluators*).

Dentre os principais algoritmos de método de pesquisa pode-se citar (Hall; Smith, 1998; Hall; Holmes, 2003; Pitt; Nayak, 2007; Witten *et al.*, 2011; Satin *et al.*, 2015b):

- **Ranker:** Esse algoritmo analisa os atributos individualmente, e cria um Ranking em ordem crescente de influência dos atributos sobre o atributo alvo.
- **Best First:** Inicialmente contém um conjunto vazio de atributos, sendo preenchido a medida que a adição de um atributo melhora o processo preditivo. Esse método guarda um histórico contendo os estados do subconjunto de atributos, possibilitando voltar atrás em decisões tomadas.
- **Genetic Search:** Esse algoritmo baseia-se nos princípios naturais da biologia, utilizando conceitos de mutação a cruzamento de atributos. Mutação consiste em adicionar e remover atributos de subconjuntos que são utilizados para análise. Cruzamento consiste em agrupar pares de atributos distintos, também para processo de análise. O objetivo é encontrar um subconjunto de atributos que evoluem ao longo do tempo, gerando melhores predições.
- **Greedy Stepwise:** Analisa subconjuntos de atributos, buscando características que melhorem o processo de predição. Encerra o processo de análise apenas quando não se tem aumento no desempenho do modelo de predição. A busca ocorre percorrendo-se o conjunto de atributos, adicionando ou removendo atributos a outro subconjunto. A inserção de atributos no subconjunto pode ocorrer de duas formas, os atributos são adicionados a um subconjunto de forma que sua capacidade preditiva aumente, dá-se o nome de pesquisa para frente (do inglês, *forward*), ou o subconjunto contém inicialmente todos

os atributos, sendo que esses são removidos com a finalidade de melhorar a capacidade preditiva, dá-se o nome de pesquisa para trás (do inglês, *backward*).

Dentre os principais algoritmos para avaliação de atributos pode-se citar (Novakovic *et al.*, 2011; Witten *et al.*, 2011; Khoshgoftaar *et al.*, 2012; Shivaji *et al.*, 2013; Gao; Khoshgoftaar, 2015):

- **Information Gain:** Esse algoritmo estabelece uma medida que representa a quantidade de informação que cada atributo oferece para identificação do atributo alvo, de forma que as informações diminuam a medida de incerteza sobre o atributo alvo. Esse algoritmo tende a selecionar atributos com valores altos, mesmo que esses não influenciem o atributo alvo;
- **Gain Ratio:** Semelhante ao *Information Gain*, contudo, sua fórmula procura diminuir a influência dos valores altos sobre o atributo alvo, sendo assim, os atributos de valores baixos podem ter sua influência considerada.
- **OneR:** Esse algoritmo cria regras para cada atributo e aplica essas regras de forma a encontrar uma relação dos atributos com o atributo alvo. As regras criadas são treinadas e aquelas com um número menor de erros são utilizadas (Holte, 1993).
- **Chi-Square:** Aplica o cálculo estatístico de *Chi-Square* entre o atributo em análise e o atributo alvo, esse cálculo é feito com a finalidade de encontrar dependência entre eles. As hipóteses são: não há dependência entre o atributo em análise e o atributo alvo; ou, há dependência entre o atributo analisado e o atributo alvo (Liu; Setiono, 1995).
- **CFS (do inglês, *Correlation-based Feature Selection*):** Esse algoritmo considera que os atributos são independentes, mas analisa a capacidade preditiva de cada atributo estimando seu grau de redundância e intercorrelação com o atributo alvo (Witten *et al.*, 2011).

Para que um algoritmo de método de pesquisa seja utilizado em conjunto com um algoritmo de seleção de atributos, é necessário haver compatibilidade entre ambos. Por exemplo, o algoritmo de método de pesquisa *Ranker* pode ser utilizo em conjunto com todos os algoritmos de avaliação de atributos listados, exceto com o algoritmo *CFS*, devido a abordagem estatística de *Ranking* que o algoritmo *Rank* utiliza.

2.1.4. Validação Cruzada

Validação Cruzada é o nome da técnica que procura utilizar dados históricos do projeto alvo para realizar a predição, de forma que parte dos dados são utilizados para treino e parte é utilizado para teste do modelo de predição (Refaeilzadeh *et al.*, 2009).

A validação cruzada consiste em fragmentar o conjunto de dados em K partições (Refaeilzadeh *et al.*, 2009). O ideal é que cada uma das partições contenha o mesmo número de instâncias e que a quantidade dos atributos alvos em cada partição esteja o mais próximo possível de uma divisão exata. Após o particionamento ocorre K iterações, onde um fragmento é utilizado para testar o modelo de predição e os demais $K-1$ fragmentos são utilizados para treinar o modelo. A cada iteração ocorre uma troca do fragmento utilizado para teste, de forma que todos os fragmentos sejam utilizados para testar o modelo de predição. A análise ocorre através de medidas de desempenho que permitam a comparação de resultados dos algoritmos de aprendizado de máquina atuando cada um sobre um partição do conjunto de dados, possibilitando a escolha de um algoritmo com melhor desempenho (Kohavi, 1995). Outra importante vantagem da utilização de medidas de desempenho é a possibilidade de melhorar a performance de cada um dos algoritmos classificadores através da modificação dos parâmetros de configuração (Satin *et al.*, 2015b). A forma mais utilizada dessa técnica

consiste em dividir os dados históricos do projeto alvo em 10 partições, chamada de *10-fold cross-validation* (Kohavi, 1995).

A técnica de validação cruzada é uma possibilidade para realização de predição em projetos que contenham dados históricos, no entanto, essa prática se torna ineficiente diante de projetos que não possuem uma base de dados histórica consolidada (Refaeilzadeh *et al.*, 2009), esse cenário ocorre principalmente em projetos que estão em fase inicial de desenvolvimento.

2.1.5. Algoritmos de Agrupamento por Similaridade

A técnica de agrupamento tem como objetivo agrupar projetos que apresentem algum tipo de semelhança entre si (Jain; Dubes, 1988). Para realizar o processamento é necessário realizar algumas atividades, como ilustrado na Figura 2.1.

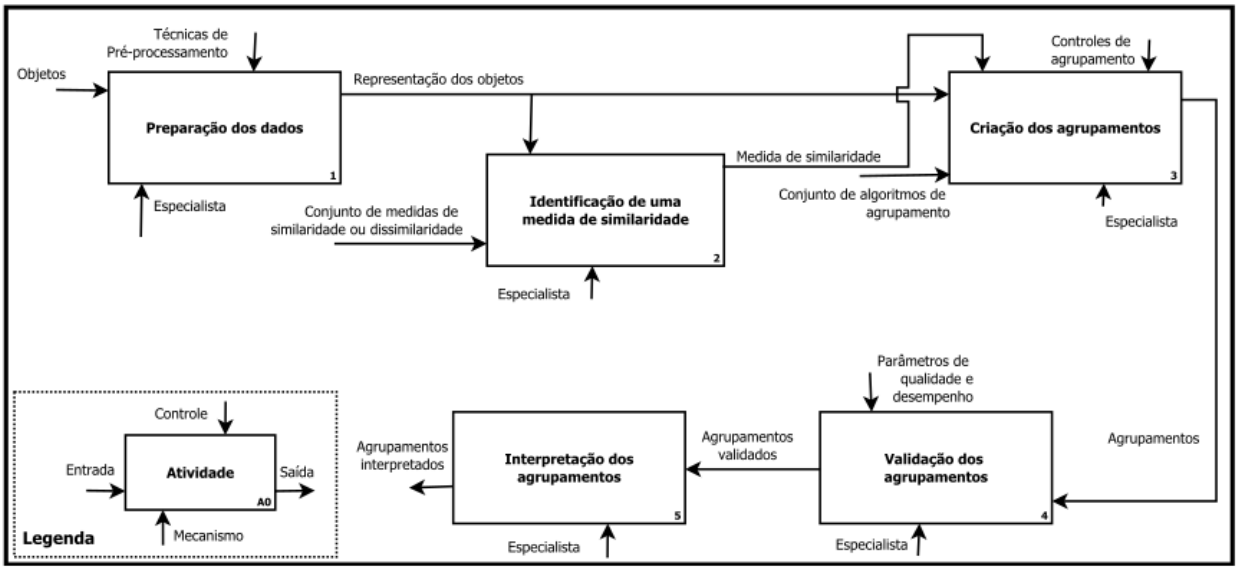

Figura 2.1. Atividades Necessárias para Criação de Agrupamentos
Fonte: Elaborado pelo Autor

A atividade "Preparação dos dados" tem como finalidade representar os objetos a serem agrupados, essa representação deve ser criada de forma apropriada para ser utilizada pelo algoritmo de agrupamento. As principais formas de se representar objetos utilizam matrizes de objetos, matrizes e grafos de similaridade/-dissimilaridade.

A atividade "Identificação de uma medida de similaridade" consiste em identificar o quão semelhantes ou diferentes são os objetos a serem agrupados (Jain; Dubes, 1988). A medida de similaridade é utiliza para que os agrupamentos sejam formados, sendo que uma das medidas mais utilizadas é a distancia euclidiana (Faceli *et al.*, 2011). Essa medida representa a distância entre dois objetos e seu cálculo é dado pela Equação 2.1, no qual:

$$d(pi, qi) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{2.1}$$

- p_i e q_i são os objetos que terão suas distâncias calculadas;

- i é o índice da dimensão para comparação entre os objetos;
- n é o número total de dimensões dos objetos a serem comparadas;
- d é a somatória das distâncias calculadas para todas as dimensões dos objetos p e q.

A atividade "Criação dos agrupamentos"consiste na utilização de algoritmos de agrupamento para a formação dos possíveis grupos (do inglês, *clusters*). Dentre os algoritmos mais utilizados está o algoritmo BSAS. Esse algoritmo recebe o conjunto de dados e faz o processamento desses dados uma única vez. Primeiramente o algoritmo calcula a distância de similaridade entre o objeto que se deseja agrupar e todos os agrupamentos já formados, em seguida analisa se a menor distância encontrada é menor que a distância de similaridade pré-estabelecida antes da execução do algoritmo, em caso verdadeiro o objeto é incluído no agrupamento que se obteve a menor distância, em caso negativo um novo agrupamento é formado para armazenar esse objeto. No entanto, para que um novo agrupamento seja formado a quantidade máxima de agrupamentos, também pré estabelecida antes da execução do algoritmo, não pode ter sido excedida (Theodoridis; Koutroumbas, 2008).

Após a formação dos agrupamentos se faz necessário a execução da atividade "Validação dos agrupamentos criados". Essa atividade analisa cada agrupamento de forma a identificar adequação frente ao conjunto de dados (Jain *et al.*, 1999). Alguns parâmetros são utilizados para se fazer a análise, entre eles: a estrutura dos agrupamentos, o grau de sobreposição entre os agrupamentos, a medida de similaridade utilizada para criação dos agrupamentos e a presença de objetos com valores que se diferenciam muito dos demais integrantes do grupo (*outliers*). E por fim, a atividade “Interpretação dos dados formados” consiste em oferecer meios que possam permitir a análise de significado, qualidade, e correlação de cada agrupamento. Ao final dessa atividade tem-se uma interpretação dos agrupamentos gerados.

2.2. Predição de Defeitos de Softwware

2.2.1. Predição de Defeitos

Predizer defeitos consiste em antecipar a ocorrência de defeitos que venham a ocorrer em versões futuras de um software, o que está intimamente relacionado a qualidade do software que está sendo produzido (Satin *et al.*, 2015b). Para realizar a predição, primeiramente é necessário selecionar um conjunto de dados com seus respectivos atributos (métricas), essa atividade pode ser feita através de técnicas de mineração de dados. Após mineração dos dados é necessário criar, treinar e testar modelos de predição, além de estabelecer indicadores de performance que permitam analisar a eficiência das predições realizadas. Todos esses passos tornam a criação de modelos de predição de defeitos uma tarefa complexa, pois não existe um padrão que possa ser seguido para realização de todas essas atividades (Hall *et al.*, 2012).

Apesar das inúmeras dificuldades, existem muitos benefícios que justificam a criação de modelos de predição (Moser *et al.*, 2008; Hall *et al.*, 2012). Entre essas vantagens pode ser destacada o aumento de produtividade da equipe de desenvolvimento de software, uma vez que o tempo para identificação de falhas será reduzido, fazendo com que a equipe tenha mais tempo para novas implementações. Um estudo mostra que 60% do tempo utilizado pela equipe para identificação de falhas em software pode ser reduzido com a utilização de predição de defeitos (Peters *et al.*, 2013). Outra vantagem é o aumento da percepção do usuário sobre a qualidade final do produto que está sendo desenvolvido, podendo atrair novos clientes (Moser *et al.*, 2008).

Para que a predição de defeitos ocorra é necessário que informações do software em desenvolvimento sejam armazenadas, resultando em um conjunto de dados para uso no processo de predição. Quanto maior o conjunto de dados, melhor para o preditor. No entanto, percebe-se facilmente que projetos em fase inicial de desenvolvimento apresentam poucas informações a respeito do seu ciclo de desenvolvimento, sendo assim, é eminente que existe uma dificuldade para criação de modelos de predição para tais projetos.

A abordagem para criação dos modelos de predição necessita de dois conjuntos de dados, um que é utilizado para treinar o modelo e outro que é utilizado para testar o modelo. O conjunto de treino, é normalmente formado por informações de versões anteriores do software que se deseja realizar a predição. Já o conjunto de teste é formado por informações da versão mais recente do software em desenvolvimento. Assim, é possível identificar que os modelos de predição utilizam dados de versões passadas para identificar erros que possam vir a ocorrer em versões futuras do software (Weyuker *et al.*, 2006; Bacchelli *et al.*, 2010).

Após a criação do conjunto de dados é necessário fazer um pré-processamento com o objetivo de tratar os atributos de forma que o conjunto de dados esteja apto para criação dos modelos de predição. Para criar um modelo de predição se faz necessário: selecionar um algoritmo de aprendizado de máquina; selecionar uma forma de tratar os atributos dos objetos do conjunto de dados, de forma a reduzir a dimensionalidade; e identificar indicadores de performance que permitam avaliar o modelo de predição (Russell; Norvig, 2009; Faceli *et al.*, 2011).

O algoritmo de aprendizado de máquina selecionado é treinado com o conjunto de dados designado para treino, com o treino o algoritmo procura reconhecer padrões, estabelecer correlações, e identificar qual a relação entre os atributos dos objetos com o atributo alvo. A última etapa consiste em avaliar a predição, essa análise é feita com base nos valores de desempenho obtidos através do teste do modelo de predição.

2.2.2. Medidas de Avaliação de Desempenho de Modelos de Predição

As predições realizadas pelos modelos de predição precisam ser avaliadas, e essa avaliação é feita com a utilização de uma medida de desempenho. Na literatura podem ser encontradas inúmeras medidas que podem ser utilizadas individualmente ou em conjunto para avaliação de modelos de predição (Ostrand; Weyuker, 2007; Bowes *et al.*, 2012). A descrição de cada medida é dada no contexto do presente trabalho:

- **Matriz de Confusão:** Trata-se de uma matriz com resultados obtidos durante etapa de teste do modelo, bastante utilizada em modelos que utilizam algoritmos de classificação. Conforme exibido na Tabela 2.1, essa matriz apresenta quatro conjuntos de dados. Na Tabela 2.1, TP retrata o valor de verdadeiro positivo (do inglês, *True Positive*), que é a quantidade de vezes que o modelo apontou haver defeitos, e após os testes se constatou que a predição estava correta; FP retrata o valor de falso positivo (do inglês, *False Positive*), que é a quantidade de vezes que o modelo apontou haver defeitos, e após os testes se constatou que a predição estava incorreta; FN retrata o valor de falso negativo (do inglês, *False Negative*), que é a quantidade de vezes que o modelo apontou não haver erros, e após os testes se constatou que a predição estava incorreta; e, TN retrata o valor de verdadeiro negativo (do inglês, *True Negative*), que é a quantidade de vezes que o modelo apontou não haver erros, e após os testes se constatou que a predição estava correta. Vale ressaltar que as próximas medidas de desempenho são calculadas a partir dos valores da matriz de confusão.
- **Acurácia (do inglês, *Accuracy*)**: Uma medida que representa o quão exato está ocorrendo as classificações. Seu valor varia de 0 à 1, e valores de acurácia mais próximos de 1 são indicadores de um modelo de predição com bom desempenho. Seu valor é obtido por: $Acuracia = \frac{(TP+TN)}{(TP+TN+FP+FN)}$;

Tabela 2.1. Matriz de Confusão

Previsão	Observado Verdadeiro	Observado Falso
Previsto Verdadeiro	TP	FP
Previsto Falso	FN	TN

- **Precisão (do inglês, *Precision*)**: Representa o quanto de projetos com defeitos se consegue classificar corretamente. Assim como o valor para acurácia, seu valor varia de 0 à 1, sendo que valores mais próximos de 1 são indicadores de um modelo de predição com bom desempenho. Seu valor é obtido por: $Precisao = \frac{(TP)}{(TP+FP)}$;
- **Sensibilidade (do inglês, *Recall*)**: Uma proporção das classificações erradas, onde se apontou haver ocorrência de defeitos. Assim como acurácia e precisão, seu valor varia de 0 à 1, sendo que valores mais próximos de 1 são indicadores de um modelo de predição com bom desempenho. Seu valor é obtido por: $Sensibilidade = \frac{(TP)}{(TP+FN)}$;
- **MCC (do inglês, *Matthews correlation coefficient*:** Trata-se de um coeficiente de correlação entre as classes dependentes que é utilizada como uma medida de qualidade. Ao contrário de acurácia, precisão e sensibilidade, seu valor varia de -1 à 1, onde valores mais próximos de -1 são indicadores de um modelo de predição ruim, valores iguais a 0 indicam que o modelo de predição é totalmente randômico e valores mais próximos de 1 são indicadores de um modelo de predição com bom desempenho. Seu valor é obtido por: $MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$.

2.2.3. Predição Cruzada entre Projetos de Software

A predição cruzada é uma alternativa para a realização de predição em projetos que não contém um base de dados histórica consolidada que, como visto na Seção 2.2.1, é um problema que ocorre principalmente em projetos que estão em fase inicial de desenvolvimento (Turhan *et al.*, 2009; Zimmermann *et al.*, 2009; Jureczko; Madeyski, 2010; Menzies *et al.*, 2011; Zhang *et al.*, 2014). A alternativa é realizar a predição desses projetos utilizando dados históricos de outros projetos que contenham uma base de dados consolidada. Para que isso ocorra é necessário (Satin *et al.*, 2015b): estabelecer uma medida de similaridade entre projetos; encontrar projetos que sejam similares ao projeto alvo; e, realizar um tratamento dos dados de forma que o conjunto de treino seja compatível com o conjunto de teste.

A possibilidade de se utilizar dados de outros projetos para criação de modelos de predição foi confirmada em alguns trabalhos presentes na literatura (Menzies *et al.*, 2007; Zimmermann *et al.*, 2009; Hall *et al.*, 2012; Zhang *et al.*, 2014). O primeiro desafio é estabelecer medidas que permitam encontrar projetos que sejam semelhantes, permitindo o compartilhamento de dados. A forma mais utilizada consiste em utilizar o vetor de métricas dos projetos para calcular o quão semelhantes eles são. Inclusive, alguns trabalhos apontam quais características são melhores preditoras, uma delas são as características de contexto (Hall *et al.*, 2012; Zhang *et al.*, 2014), no entanto, quais exatamente devem ser utilizadas para identificar semelhança entre os projetos ainda é um assunto em aberto (Ostrand *et al.*, 2005; Nagappan *et al.*, 2006). Vale destacar que cada métrica pode possuir um domínio próprio, o que consequentemente deve levar a uma forma adequada para comparação desses valores.

Após estabelecer as medidas de desempenho os modelos de predição cruzada podem ser criados. Não existe uma regra que diz como o conjunto de dados dos projetos similares devem ser utilizados. O compartilhamento de dados históricos pode ocorrer através da formação de pares de projetos similares (Zimmermann *et al.*, 2009), ou, através da formação de agrupamentos dos dados de projetos similares (Satin *et al.*, 2015b). Independente da forma que o conjunto de dados é utilizado, os dados referentes aos projetos encontrados será utilizado

para treinar o modelo de predição cruzada e o conjunto de dados do projeto alvo será utilizado para testar o modelo de predição cruzada.

A principal dificuldade está em criar modelos de predição cruzada que possam ser transferíveis para projetos que não foram utilizados na construção desses modelos. Alguns estudos apontam que é inviável a criação de modelos de predição genéricos (Bell *et al.*, 2006), contudo, outros estudos apresentam possibilidades de se criar modelos genéricos, inclusive apresentando resultados satisfatórios (Nagappan *et al.*, 2006; Zhang *et al.*, 2014).

2.3. Trabalhos Relacionados

Buscou-se encontrar na literatura trabalhos semelhantes que utilizassem abordagens diferentes para a criação e validação de modelos de predição cruzada de defeitos. Uma das maiores dificuldades é encontrar trabalhos na literatura que possibilitem uma comparação dos resultados. Um dos principais fatores que levam a essa dificuldade é a utilização de base de dados distintas, pois um modelo de predição pode se mostrar eficiente diante de uma base de dados e apresentar um rendimento totalmente oposto quanto avaliado com outras bases de dados. Contudo, foi possível encontrar na literatura trabalhos que utilizaram a mesma base de dados que a utilizada no presente trabalho e com possibilidade de comparação de resultados devido ao método de validação utilizado. E apesar das dificuldades, trabalhos que utilizaram outras bases de dados também foram selecionados, pois utilizaram os mesmos algoritmos que os utilizados no trabalho em desenvolvimento, além de propor métodos de validação. Sem deixar de citar trabalhos que não serão utilizados para possíveis comparações de resultados mas que oferecem um grande embasamento teórico a respeito de predição cruzada de defeitos.

Os trabalhos relacionados são: Os trabalhos de Turhan *et al.* (2009), Rahman *et al.* (2012), Jureczko e Madeyski (2010), Zimmermann *et al.* (2009), Menzies *et al.* (2011), Zhang *et al.* (2014) e Satin *et al.* (2015b).

O trabalho de Turhan *et al.* (2009) fez uso de 12 projetos extraídos do repositório da NASA. Esse trabalho mostrou vantagens da utilização de predição cruzada de defeitos, apesar dos resultados obtidos por modelos que utilizaram apenas os dados históricos dos próprios projetos para realizar a predição terem sido melhores. Esse estudo também apontou para um risco, que é a alta taxa de falso positivo obtida pela predição cruzada, tornando a transferência do modelo para outros projetos inviável. O autor utilizou algoritmos de aprendizado de máquina baseados no vizinho mais próximo, e aplicou a distância euclidiana sobre as próprias métricas dos projetos para obter o grau de semelhança entre eles.

O trabalho proposto por Rahman *et al.* (2012) utilizou um conjunto de dados contendo 39 *releases* de 9 projetos de software livre. Seu objetivo era verificar se predições realizadas a partir do agrupamento de projetos obtinham resultados superiores que as predições feitas utilizando apenas os dados históricos do próprio projeto alvo. Seus resultados apontaram que utilizar apenas os dados históricos do projeto alvo era mais vantajoso, pois foi o método que obteve resultados superiores. No entanto, pode-se verificar que com a utilização de agrupamento de projetos e compartilhamento de dados históricos houve uma redução no custo para realização da predição.

O trabalho de Jureczko e Madeyski (2010) utilizou uma base de dados que continha 92 *releases* de 38 projetos. Esses projetos foram agrupados com a utilização de algoritmos de agrupamento hierárquicos de clusterização e K-means. Os resultados obtidos foram superiores aos resultados obtidos por modelos que utilizaram os próprios dados históricos do projeto alvo.

Em seu trabalho, Zimmermann *et al.* (2009) utilizou 12 projetos de grande porte, seu método consistiu em agrupar esses projetos em pares de forma que pudessem compartilhar dados históricos entre eles. Os resultados obtidos não foram bons, sendo que apenas 3,40% dos pares formados conseguiram predizer defeitos entre si. Apesar do resultado, Zimmermann *et al.* (2009) forneceu contribuições importantes, sendo uma delas apontar valores de desempenho que indicam se um modelo de predição cruzada de defeitos pode ou não ser transferido para outros projetos, esses valores de desempenho foram obtidos a partir da produção de uma árvore de decisão criada para os valores de precisão, sensibilidade e acurácia. Seu estudo apontou que modelos de predição cruzada com valores de precisão, sensibilidade e acurácia acima de 0,75 são modelos transferíveis para outros projetos. Outra importante contribuição foi indicar qual projeto deve ser utilizado para testar o modelo. Segundo seu estudo o projeto com maior número de instâncias fornece mais informações para predição em outros projetos, e por esse motivo deve ser o escolhido para testar o modelo de predição cruzada.

Menzies *et al.* (2011) sugeriu que modelos de predição cruzada criados a partir do agrupamento de projetos deve obter resultado superior no caso em que não é utilizado agrupamento, ou seja, o modelo é treinado com todos os projetos do conjunto de dados.

A proposta de Zhang *et al.* (2014) é criar um modelo de predição que possa ser utilizado para predizer defeitos em qualquer projeto de software. Para atingir esse objetivo ele utilizou uma técnica para padronizar os atributos dos projetos. A padronização ocorre da seguinte forma: primeiramente busca-se identificar a inexistência de diferença estatística relevante entre os valores dos atributos de cada par de projetos, caso haja diferença, que a diferença não seja significativa; Os projetos que não passam por essa regra são descartados para a segunda atividade, que é padronizar os valores de seus atributos. A padronização é feita com a divisão dos valores em 10 partes iguais e conversão para outra escala, de forma que o valor fique entre 0 e 10. Seguindo esse processo, para utilizar os dados históricos dos projetos da base de dados para predição basta realizar a mesma conversão de valores para o projeto alvo.

Zhang *et al.* (2014) utilizou o critério de Zimmermann *et al.* (2009) para estabelecer quais modelos poderiam ser utilizados para predizer defeitos em projetos não contidos na base de dados. Dos modelos criados, os que obtiveram valores de precisão, sensibilidade e acurácia superior a 0,75, segundo o critério de Zimmermann *et al.* (2009), foram 3% dos modelos. Apesar da taxa de modelos transferíveis para outros projetos ter sido consideravelmente baixa, Zhang *et al.* (2014) obteve resultados superiores que as predições realizadas com apenas dados históricos do próprio projeto alvo.

O principal trabalho encontrado foi o trabalho de Satin *et al.* (2015b). Essa importância deve-se ao fato de utilizar o mesmo conjunto de dados que o que será utilizado pelo presente trabalho, além de propor um novo método para criação de modelos de predição cruzada de defeitos. Satin *et al.* (2015b) utilizou o algoritmo de agrupamento BSAS que fez uso de uma medida de desempenho denominada AUC (do inglês, *Area Under a Curve*) para cálculo de semelhança entre os projetos. Com a utilização do algoritmo BSAS pôde-se agrupar modelos de predição treinados apenas com dados históricos do projeto alvo. Após agrupamento foi possível realizar compartilhamento do conjunto de dados de treinamento entre os modelos, fazendo com que um conjunto de modelos de predição contidos em um mesmo agrupamento fossem transformados em um único modelo de predição cruzada de defeitos. Dos modelos de predição cruzada criados por Satin *et al.* (2015b), 31,58% poderiam ser utilizados para predizer defeitos em qualquer outro projeto, segundo o critério proposto por Zimmermann *et al.* (2009). Apesar de obter um resultado satisfatório se comparado com outros trabalhos, Satin *et al.* (2015b) apontou haver problemas de desbalanceamento de classes nos agrupamentos, o que pode ter influenciado negativamente seus resultados.

Capítulo

3

Criação dos Modelos de Predição Cruzada de Defeitos

Neste capítulo, apresentam-se as atividades que foram realizadas durante a condução do trabalho e os resultados obtidos em cada uma delas. O processo é baseado na proposta de Satin *et al.* (2015b). No entanto, foi utilizada uma medida de correlação entre as classes para formação dos agrupamentos, descrito na Seção 3.3, seguido pela aplicação de testes estatísticos distintos para avaliação dos resultados. A condução do trabalho consisti na realização de cada atividade apresentada na Figura 3.1.

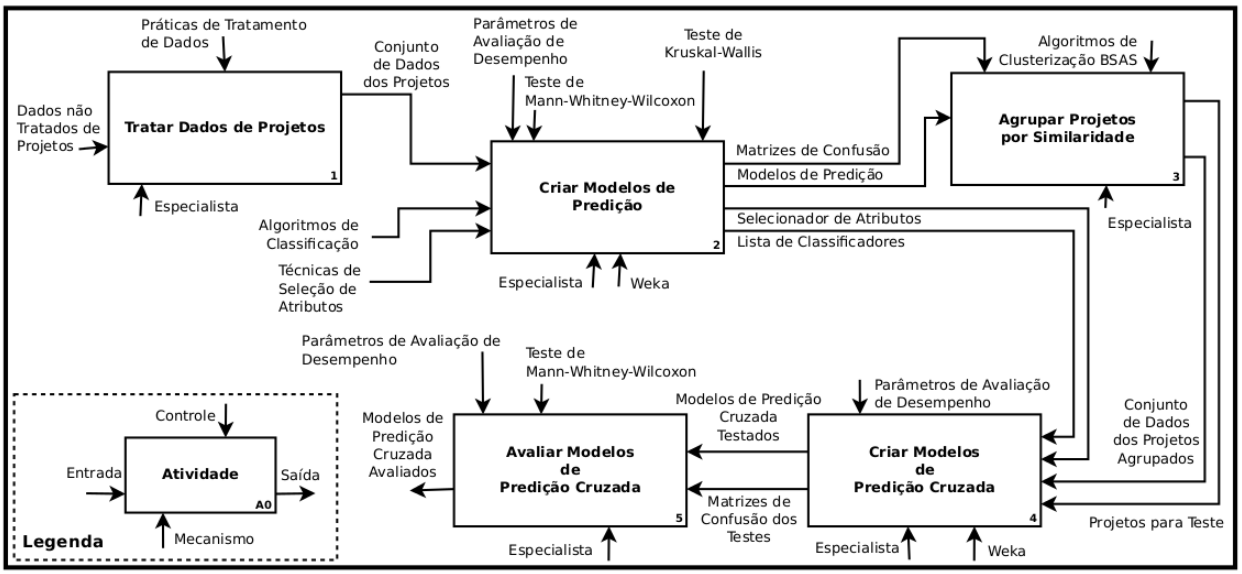

Figura 3.1. Atividades para Criação dos Modelos de Predição Cruzada de Defeitos.
Fonte: (Satin *et al.*, 2015b)

3.1. Tratar Dados de Projetos

Essa atividade tem como finalidade tratar o conjunto de dados de forma que as informações contidas neste sejam proveitosas para a condução deste trabalho. A base de dados é a mesma utilizada por Zhang *et al.* (2014) e Satin *et al.* (2015b), e é formada por 1382 projetos de software livre minerados a partir dos repositórios *Source Forge* [1] e *Google Code* [2]. Cada projeto possui 72 métricas de código, sendo que um desses atributos é a variável dependente, que pode assumir valores *Buggy* ou *Clean*. Esse atributo dita se determinada instância possui erros no seu código fonte ou se não contém erros no seu código fonte.

Os primeiros esforços foram voltados para o tratamento adequado dos projetos de forma que pudessem ser processados pelos algoritmos de classificação e seleção de atributos sem interferência nos resultados. Basicamente foram realizadas três etapas, a primeira foi reordenar os atributos para que não houvesse conflito no momento da validação cruzada entre os projetos, por exemplo, em certos casos um determinado projeto P_1 continha os atributos A, B e C, já o projeto P_2 continha os atributos A, C e B, sendo assim, era necessário realizar uma reordenação dos atributos de forma que em todos os projetos aparecessem na mesma sequência. As duas etapas posteriores consiste na aplicação das técnicas de pré-processamento propostas por Faceli *et al.* (2011): eliminação manual de atributos e redução manual de dimensionalidade. A eliminação manual de atributos consiste em eliminar da base de dados atributos que são irrelevantes para que um determinado algoritmo classificador consiga predizer a ocorrência ou ausência de defeitos em projetos de software. Nessa atividade foram removidos atributos como, o nome do projeto e *links* de repositórios onde o código fonte do projeto pode ser encontrado, resultando em uma redução da dimensionalidade dos dados uma vez que a quantidade de atributos foi reduzida de 72 para 68. Já a terceira e última etapa, a redução manual de dimensionalidade, foi remover os projetos que não podiam ser processados pelo método *ten-fold croos-validation*, melhor detalhado na Seção 2.1.4, que obrigatoriamente requer que um projeto contenha no mínimo 10 instância para processamento. Seguindo essa regra, projetos com uma quantidade de instâncias inferior a 10 foram descartados. Essa última etapa resultou em uma redução na quantidade de projetos da base de dados, passando de um total de 1382 para 1283 projetos.

3.2. Criar Modelos de Predição

Essa atividade consiste na criação dos modelos de predição de defeitos que utilizam apenas os dados históricos do projeto alvo, ou seja, nesta etapa não deve existir compartilhamento de dados históricos entre projetos durante o processo de treino e teste dos modelos de predição. A condução dessa atividade resultou na especificação de um conjunto de algoritmos, composto por: algoritmo classificador e par de algoritmos selecionadores de atributos, sendo este formado por um algoritmo de busca de atributos e um algoritmo avaliador de atributos. Todo o processo consiste, basicamente, na criação dos modelos de predição e análise dos resultados obtidos por esses de tal forma que possibilitou classificar entre todos os pares de algoritmos selecionadores de atributos utilizados aquele que apresentou melhor performance diante dos demais algoritmos. Para concluir esta atividade fez-se necessário: a implementação de um especialista que em conjunto com a ferramenta Weka [3] pôde gerar os dados necessários para análise; definir parâmetros de avaliação de desempenho nos modelos de predição; e, selecionar testes estatísticos por meio das técnicas de Anderson e Darling (1952), Kruskal e Wallis (1952), Jelihovschi *et al.* (2014) e Cohen's d Cohen (1977).

[1] http://www.sourceforge.net
[2] https://code.google.com
[3] http://www.cs.waikato.ac.nz/ml/weka

Para um melhor detalhamento de todos os passos necessários para a conclusão desta atividade, ela foi dividida em sub-atividades, como pode ser observado na Figura 3.2. De forma geral, as atividades que detalham a criação dos modelos de predição seguem a proposta de Hall *et al.* (2012): desenvolvimento, treino, teste e avaliação de desempenho.

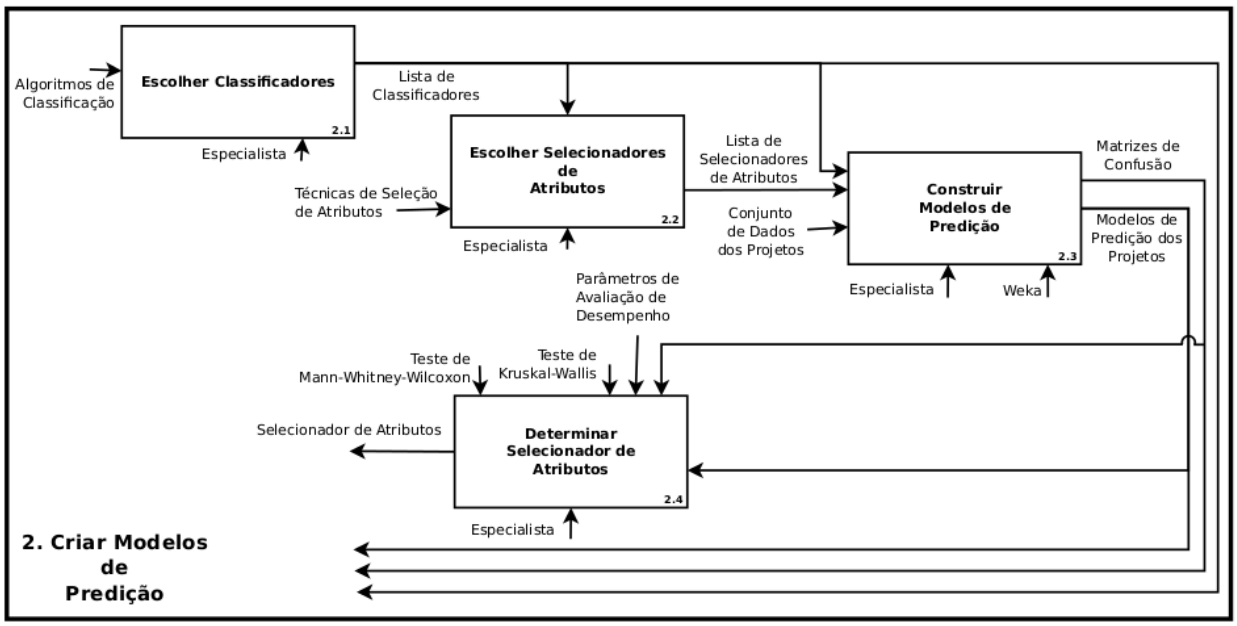

Figura 3.2. Atividades Necessárias para Criação dos Modelos de Predição
Fonte: (Satin *et al.*, 2015b)

3.2.1. Escolher Classificadores

O objetivo desta atividade foi selecionar na literatura especializada os classificadores utilizados por trabalhos semelhantes e que também pudessem ser utilizados na condução deste trabalho. A escolha dos classificadores adequados para a criação de modelos de predição é um assunto que ainda traz discussões, pois estudos são controversos quanto a influência dos algoritmos de classificação no desempenho dos modelos de predição (Lessmann *et al.*, 2008; Ghotra *et al.*, 2015).

Sendo assim, buscou-se encontrar na literatura os algoritmos de classificação mais utilizados em trabalhos semelhantes e que utilizam abordagens diferentes, com a finalidade de analisar sua influência no processo de criação de modelos de predição cruzada. Dentre os classificadores mais utilizados em trabalhos similares (Zimmermann *et al.*, 2009; Menzies *et al.*, 2011; Herbold, 2013; Ghotra *et al.*, 2015; Lessmann *et al.*, 2008; Satin *et al.*, 2015b) os selecionados foram:

- **Naive Bayes:** Com atuação estatística;
- **Simple Logistic:** Com atuação estatística;
- **J48:** Baseado em árvore de decisão;
- **Random Forest:** Baseado em árvore de decisão;
- **Decision Table:** Baseado em regra de decisão;
- **OneR:** Baseado em regra de decisão;
- **Multilayer Perceptron:** Baseado em redes neurais artificiais;

3.2.2. Escolher Selecionadores de Atributos

O objetivo dessa atividade foi selecionar algoritmos para identificação das métricas mais relevantes para um modelo de predição composto por um algoritmo de classificação específico, bem como diminuir a dimensionalidade dos dados. A seleção de métricas (do inglês, *feature selection*) pode melhorar o desempenho dos classificadores e possibilita a transferência dos modelos de predição para outros projetos (Hall *et al.*, 2012).

Alguns trabalhos desenvolvem seus próprios algoritmos para a seleção de métricas (Menzies *et al.*, 2007; Liu; Setiono, 1995; Zhang *et al.*, 2014), método que foi descartado por este trabalho pois pretende-se avaliar o desempenho dos algoritmos de seleção de métricas quando executados juntamente com algoritmos classificadores.

Assim como na escolha dos classificadores, procurou-se selecionar os algoritmos de seleção de atributos mais utilizados em trabalhos similares (Malhotra, 2015), levando em consideração a possibilidade da atuação conjunta dos algoritmos de seleção de atributos com os algoritmos classificadores (Gao; Khoshgoftaar, 2015), selecionados na atividade "Escolher Classificadores" da Figura 3.2.

Sendo assim, foram selecionados dois tipos de algoritmos de seleção de atributos: ranqueadores e selecionadores de subconjunto, Subseção 2.1.3. Cada método de seleção de atributos é composto por: um algoritmo de método de pesquisa e um algoritmo avaliador de atributo, formando um empacotamento (Gao; Khoshgoftaar, 2015). Os algoritmos de método de pesquisa selecionados foram:

- **Best First**;
- **Genetic Search**;
- **Greedy Stepwise**;
- **Ranker**.

Os algoritmos de seleção de atributos selecionados foram:

- **CFS**;
- **Information Gain**;
- **Gain-Ratio**;
- **Chi-Square**.

Como retratado na Seção 2.1.3, deve haver compatibilidade para que um algoritmo de método de busca possa ser utilizado em conjunto com um algoritmo de seleção de atributos. Sendo assim, a Tabela 3.1 apresenta as possíveis combinações desses algoritmos.

Tabela 3.1. Combinação dos Algoritmos Classificadores com os Algoritmos de seleção de Métricas para a Criação dos Modelos de Predição

Classificador	**CFS + BestFirst**	**CFS + GeneticSearch**	**CFS + GreedyStepwise**	**Ranker + InfoGain**	**Ranker + GainRatio**	**Ranker + ChiSquare**
Naive Bayes	Cenário 01	Cenário 02	Cenário 03	Cenário 04	Cenário 05	Cenário 06
Random Forest	Cenário 07	Cenário 08	Cenário 09	Cenário 10	Cenário 11	Cenário 12
Decision Table	Cenário 13	Cenário 14	Cenário 15	Cenário 16	Cenário 17	Cenário 18
Simple Logistic	Cenário 19	Cenário 20	Cenário 21	Cenário 22	Cenário 23	Cenário 24
J48	Cenário 25	Cenário 26	Cenário 27	Cenário 28	Cenário 29	Cenário 30
OneR	Cenário 31	Cenário 32	Cenário 33	Cenário 34	Cenário 35	Cenário 36
Multilayer Perceptron	Cenário 37	Cenário 38	Cenário 39	Cenário 40	Cenário 41	Cenário 42

3.2.3. Construir Modelos de Predição

O objetivo dessa atividade é desenvolver, treinar e testar modelos de predição de defeitos. Essa atividade tem como entrada o conjunto de dados, os algoritmos classificadores e os algoritmos de seleção de atributos, todos gerados por atividades anteriores.

Primeiramente foram criados 42 cenários para o desenvolvimento dos modelos de predição, conforme pode ser visto na Tabela 3.1. Cada um dos 42 cenários foi aplicado individualmente em cada um dos 1283 projetos do conjunto de dados, totalizando a criação de 53886 modelos de predição de defeitos. Cada modelo de predição foi criado com a aplicação da técnica *10-fold cross-validation* (Kohavi, 1995), e após o teste dos modelos foi calculada a matriz de confusão contendo os valores de verdadeiro positivo, falso positivo, verdadeiro negativo e falso negativo.

Com a conclusão desta atividade foi gerada uma lista contendo informações específicas de cada modelo de predição: projeto utilizado para treino e teste do modelo; algoritmo de classificação; algoritmo de método de pesquisa; algoritmo de seleção de atributos; métricas de avaliação do modelo; e, a matriz de confusão gerada para o respectivo modelo de predição.

3.2.4. Determinar Selecionador de Atributos

O objetivo dessa atividade foi selecionar o algoritmo de método de pesquisa e o algoritmo de seleção de atributos que, em conjunto, obtiveram melhores resultados diante da prática proposta por este trabalho. Ser o melhor método de seleção de atributos implica que, os melhores modelos de predição são aqueles compostos pelo algoritmo de método de pesquisa e pelo algoritmo de seleção de atributos apontados por esta atividade.

Conforme descrito na Subseção 3.2.3, a etapa de criação resultou em um total de 53886 modelos de predição de defeitos. Para cada um deles foi possível obter a matriz de confusão gerada na etapa de teste dos modelos de predição, e consequentemente, obter o cálculo do valor de MCC para cada uma das predições realizadas. Para poder afirmar que existe uma diferença estatística entre o desempenho dos modelos de predição que utilizam um ou outro par de algoritmos de seleção de atributos, e indo além, poder afirmar que os modelos que utilizam um determinado par de algoritmos selecionadores de atributos obtém melhor desempenho foram aplicados três testes estatísticos, passando os modelos de predição agrupados por algoritmos de seleção de atributos e seus respectivos valores de MCC como entrada para realização dos testes.

O primeiro teste aplicado foi o de Kruskal-Wallis, que possibilitou observar se existe diferença estatística relevante entre os algoritmos de seleção de atributos utilizados. Para esse teste considerou-se duas hipóteses, a primeira, chamada de hipótese nula, diz que não existe diferença estatística entre os resultados obtidos pelos algoritmos analisados, já a segunda hipótese afirma que há diferença estatística relevante entre os resultados. O teste de Kruskal-Wallis resultou em $p = 0,02991$, ou seja, $< 0,05$, indicando que deve-se ignorar a hipótese nula e seguir com a hipótese que nos diz que existe diferença estatística relevante entre os algoritmos de seleção de atributos utilizados.

Podendo concluir que existe diferença estatística entre os algoritmos de seleção de atributos foi possível analisar a magnitude dessa diferença e ranquear os algoritmos de acordo com seu desempenho, e consequentemente, apontar aquele que melhor contribuí com o algoritmo classificador no processo de predição. Para alcançar esses resultados foram aplicados os testes estatísticos de Scott-Knott e *effect size* de Cohen's *d*. O teste de Kruskal-Wallis faz uma análise hierárquica de agrupamentos para fazer dentro destes uma classificação de forma estática (Kruskal; Wallis, 1952), o par de algoritmos vencedor será aquele que se

encontrar unicamente em um agrupamento, ou seja, separado dos demais pares. Já o teste *effect size* de Cohen's d funciona da seguinte forma: faz uma comparação entre dois cenários, um contém o conjunto de amostras de controle e o outro é formado pelas amostras de tratamento. O valor resultante desta comparação pode ser $< 0; > 0$; ou, $= 0$. Quando o valor resultante for igual a zero significa não haver diferença entre as amostras, quando o valor resultante for < 0 significa que a amostra de tratamento obteve valores inferiores que a amostra de controle, e quando o valor resultante for > 0 significa que a amostra de tratamento obteve valores superiores que a amostra de controle. Esse valor também representa a magnitude da diferença entre as amostras (Cohen, 1977).

Os resultados obtidos pelo teste estatístico de Scott-Knott são apresentados na Tabela 3.2. A coluna "Valor MCC Médio" corresponde ao valor MCC médio obtido pelos modelos de predição que utilizaram o algoritmo avaliador de atributos e o algoritmo de método de pesquisa referentes as colunas "Avaliador de Atributos" e "Método de Pesquisa". Igualmente para as colunas "Valor MCC Min." e "Valor MCC Máx." que, respectivamente, apresenta o menor e maior valor de MCC obtido pelos modelos de predição. O principal resultado obtido com a aplicação do teste estatístico de Scott-Knott é apresentado na coluna "Agrup." da Tabela 3.2, nela é possível observar à qual agrupamento pertence cada um dos pares de algoritmos. Analisando a Tabela 3.2 percebe-se que não é possível concluir, através do teste estatístico de Skott-Knott, qual par de algoritmos selecionadores de atributos obteve melhor desempenho, uma vez que para tal, seria necessário que um dos agrupamentos fosse formado por um único par de algoritmos, o que não ocorreu, já que apenas um agrupamento foi formado (Agrupamento 1) contendo todos os algoritmos de seleção de atributos. No entanto, a aplicação do teste estatístico de Skott-Knott possibilitou observar que apesar de existir uma diferença estatística relevante entre os algoritmos essa diferença é muito pequena, uma vez que o teste não conseguiu dividir as amostras em agrupamentos distintos.

Tabela 3.2. Comparação dos algoritmos de seleção de atributos pelo teste estatístico de *Scott Knott*

Par	Avaliador de Atributos	Método de Pesquisa	Valor MCC Médio	Valor MCC Mín.	Valor MCC Máx.	Agrup.
Par 02	CFS	Genetic Search	0,1675198	-0,845	1,000	1
Par 04	*Chi-Square*	*Ranker*	0,1657046	-0,845	1,000	1
Par 06	*Information Gain*	*Ranker*	0,1648661	-0,845	1,000	1
Par 05	*Gain-Ratio*	*Ranker*	0,1646114	-0,845	1,000	1
Par 01	CFS	*Best First*	0,1620232	-1,000	1,000	1
Par 03	CFS	*Greedy Stepwise*	0,1612042	-1,000	1,000	1

Apesar de existir uma diferença pequena entre os algoritmos selecionadores de atributos foi possível obter a magnitude dessa diferença e ainda classificar o par de algoritmos com melhor desempenho através da aplicação do teste estatístico *effect size* de Cohen's *d*. Os resultados obtidos com a aplicação do teste são apresentados na Tabela 3.3. Nela as linhas representam as amostras de tratamento e as colunas representam as amostras de controle. Valores $= 0$ implicam que não existe diferença alguma entre as duas amostras. Valores < 0 implicam que a amostra de tratamento obteve um desempenho inferior que a amostra de controle. Valores > 0 implicam que a amostra de tratamento obteve desempenho superior que a amostra de controle, essa medida também representa o tamanho da diferença entre as amostras. Para uma melhor visualização dos dados, valores < 0 são destacados em negrito. Analisando a Tabela 3.3 percebe-se que os resultados obtidos pelo teste estatístico de Cohen's d reforçam os resultados obtidos pelo teste estatístico de Scott Knott, de que a diferença entre os algoritmos selecionadores de atributos é de fato muito pequena, uma vez que, a maior diferença obtida foi de 0,024693 para os algoritmos *CFS + Genetic Search* e *CFS + Greedy Stepwise*. Outra informação relevante é a do par de algoritmos com melhor desempenho, sendo *CFS + Genetic Search* os algoritmos que melhor contribuem com os algoritmos classificadores no processo de predição, uma vez que

obteve valores > 0 sempre que confrontado com os demais algoritmos. Seguido de *CFS + Genetic Search* tem-se: *Chi-Square + Ranker*, *Information Gain + Ranker*, *Gain Ratio + Ranker*, *CFS + Best First*, *CFS + Greedy Stepwise*, em ordem de classificação.

Tabela 3.3. Comparação dos algoritmos de seleção de atributos pelo teste estatístico *effect size* de Cohen's *d*

Par de Algoritmos	***CFS + Best First***	***CFS + Genetic Search***	***CFS + Greedy Stepwise***	***Chi-Square + Ranker***	***Gain-Ratio + Ranker***	***Information Gain + Ranker***
CFS + Best First		**-0.021471**	0.003157	**-0.014276**	**-0.010068**	**-0.011026**
CFS + Genetic Search	0.021471		0.024693	0.007139	0.011475	0.010439
CFS + Greedy Stepwise	**-0.003157**	**-0.024693**		**-0.017468**	**-0.013266**	**-0.014216**
Chi-Square + Ranker	0.014276	**-0.007139**	0.017468		0.004281	0.003274
Gain-Ratio + Ranker	0.010068	**-0.011475**	0.013266	**-0.004281**		**-0.000997**
Information Gain + Ranker	0.011026	**-0.010439**	0.014216	**-0.003274**	0.000997	

3.3. Agrupar Projetos por Similaridade

O objetivo dessa atividade é agrupar os modelos de predição criados na atividade "Criar Modelos de Predição" da Figura 3.1 de forma que a base de treino de cada um dos modelos possam ser compartilhadas entre si. Um especialista que foi implementado durante o desenvolvimento do trabalho fez uso do algoritmo de clusterização BSAS, que utilizou o valor de desempenho MCC de cada modelo para cálculo de similaridade. Uma questão importante a ser relatada é que, os agrupamentos foram formados individualmente para cada cenário da Tabela 3.1 que continha o melhor algoritmo de busca de atributos (*Genetic Search*) e o melhor algoritmo de seleção de atributos (*CFS*) obtidos na atividade "Escolher Selecionadores de Atributos" da Figura 3.2, os demais foram descartados.

Dois fatos levaram a escolha do algoritmo de agrupamento BSAS: o algoritmo BSAS é um dos mais comuns (Kainulainen; Kainulainen, 2002); além disso, outros algoritmos como o K-means (Alpaydin, 2010) necessitam da definição de um centroide inicial para que o processo de clusterização inicie, este centroide inicial tem influência direta nos agrupamentos formados ao fim da execução, pois a cada centroide inicial distinto a quantidade de agrupamentos e os elementos contidos nestes sofrem alterações. Esse problema se torna ainda mais complexo no trabalho em questão, pois a dificuldade de se selecionar um centroide inicial aumentaria devido a alta quantidade de modelos de predição (53886 modelos) criados (Theodoridis; Koutroumbas, 2008). Além disso, para que o presente trabalho seja de possível comparação com o trabalho de Satin *et al.* (2015b), faz-se necessário manter o mesmo algoritmo de agrupamento.

Um ponto negativo do algoritmo BSAS é a alta quantidade de agrupamentos formados por ele, incluindo alguns com poucos elementos. Esse problema pode ser controlado com a utilização de dois parâmetros, a quantidade máxima de clusters e uma medida de similaridade (Satin *et al.*, 2015b). A quantidade máxima de agrupamentos é um atributo que garante que a quantidade de clusters a serem criados não será maior que a quantidade estabelecida. A medida de similaridade vai indicar se determinado elemento vai ser adicionado a um agrupamento já existente ou se um novo agrupamento será formado. O algoritmo BSAS foi executado recebendo como parâmetro de entrada uma quantidade máxima de agrupamentos de 1000000, apenas para garantir recursos computacionais e uma distância de similaridade de 0,034, a mesma utilizada por Satin *et al.* (2015a). Essa distância garantiu um controle em relação ao número de agrupamentos formados, impedindo que

muitos clusters sejam formados com um único elemento, e assim reduzindo o número de projetos descartados no momento da predição cruzada. Ao final dessa atividade cada agrupamento formado estava apto a ser utilizado para a criação de um modelo de predição cruzada de defeitos.

3.4. Criar Modelos de Predição Cruzada

Essa atividade recebeu como entrada os agrupamentos formados na etapa anterior, o objetivo é transformar cada um dos *clusters* formados em um modelo de predição cruzada de defeitos. Para isso, foi necessário selecionar os projetos utilizados como treino e teste em cada agrupamento. Em seguida, o especialista implementado durante a condução do trabalho fez uso da ferramenta Weka para treinar e testar os modelos de predição cruzada de defeitos. Por fim, os resultados obtidos por cada modelo de predição cruzada foram analisados com a aplicação dos testes estatísticos de Kruskal-Wallis e Cohen's *d*.

A primeira tarefa desenvolvida nessa atividade foi definir quais projetos deveriam ser utilizados para treinar e testar o modelo de predição cruzada. Ao contrário da atividade "Criar Modelos de Predição" da Figura 3.1, este processo fará o treino do modelo de predição com o compartilhamento de dados históricos entre projetos. Para cada agrupamento formado foram selecionados os projetos para treino e teste do modelo de predição cruzada de defeitos, seguindo regras já pré estabelecidas. O projeto com maior número de instâncias no seu respectivo agrupamento foi utilizado como teste do modelo de predição cruzada. A opção de utilizar o projeto com maior número de instâncias foi baseada na análise de árvore de decisão proposta por Zimmermann *et al.* (2009), segundo sua avaliação, projetos com maior quantidade de instâncias fornecem melhores informações para os classificadores. As instâncias dos demais projetos serviram como base de treino do modelo de predição cruzada. É neste momento que ocorre o compartilhamento de dados históricos entre projetos ditos similares através da técnica proposta.

Após obter as instâncias para treino e teste basta fazê-los de fato. Um especialista utilizou a ferramenta Weka para treinar e testar os modelos de predição cruzada, sendo gerada a matriz de confusão para cada um deles. Com o cálculo da matriz de confusão foi possível obter o valor de MCC, que foi utilizado na avaliação dos modelos de predição cruzada, descrita detalhadamente na Seção 3.5.

Após treino e teste, foi realizada uma análise dos classificadores, com a finalidade de conseguir apontar o de melhor desempenho diante do método proposto. Para isso, foram aplicados os mesmos testes estatísticos que os utilizados na atividade "Escolher Selecionadores de Atributos" da Figura 3.2, com exceção do teste estatístico de Scott-Knott, pois para a aplicação desse teste é necessário garantir que as amostras contenham a mesma quantidade de elementos (Jelihovschi *et al.*, 2014), o que não pode-se garantir já que a quantidade de modelos de predição cruzada sofreu variação de classificador para classificador, conforme pode ser visto na Tabela 3.4: *Naive Bayes* com 23 agrupamentos, *Random Forest* com 20 agrupamentos, *Decision Table* com 24 agrupamentos, *Simple Logistic* com 24 agrupamentos, *OneR* com 22 agrupamentos, *Multilayer Perceptron* com 22 agrupamentos e *J48* com 23 agrupamentos.

A Tabela 3.4 ainda apresenta os valores de MCC calculados a partir da matriz de confusão obtida para o teste de cada modelo de predição cruzada de defeitos. Sendo que "MCC Cent." refere-se ao centroide do respectivo cluster e "MCC Teste" refere-se ao valor de MCC obtido com o teste do respectivo modelo de predição cruzada de defeitos. Primeiramente os valores da coluna "MCC Teste" serviram de entrada para o teste estatístico de Kruskal-Wallis com a finalidade de identificar a existência ou ausência de diferença estatística entre o desempenho dos classificadores. O teste retornou $p = 0,364$, ou seja, maior que $0,05$, impedindo o descarte da hipótese de que os algoritmos classificadores não se diferenciam quanto ao seu

desempenho. Apesar de não existir diferença estatística relevante entre os classificadores, os valores das colunas "MCC Teste" serviram de entrada para o teste estatístico *effect size* de Cohen's d que possibilitou obter uma correta interpretação dos resultados bem como indicar o algoritmo classificador com melhor desempenho. Os resultados obtidos pela aplicação do teste estatístico são apresentados na Tabela 3.5, no qual valores negativos são destacados em negrito para melhor visualização. A partir da mesma análise realizada na Subseção 3.2.4, valores $= 0$ implicam que não existe diferença alguma entre as duas amostras. Valores < 0 implicam que a amostra de tratamento obteve um desempenho inferior que a amostra de controle. Valores > 0 implicam que a amostra de tratamento obteve desempenho superior que a amostra de controle, essa medida também representa o tamanho da diferença entre as amostras. O teste apontou o classificador *Simple Logistic* como aquele que obteve melhor performance, pois obteve valores > 0 sempre que comparado com os demais classificadores, conforme Tabela 3.5. Na sequência, em ordem de melhor desempenho, aparecem *Decision Table*, *Random Forest*, *Naive Bayes*, *J48*, *Multilayer Perceptron* e por fim, *OneR*.

Tabela 3.4. Valores de MCC Centróide e MCC do teste realizado para os modelos de predição cruzada de defeitos

Agrup.	Naive Bayes			Random Forest			Decision Table			Simple Logistic			OneR			Multilayer Perceptron			J48		
	MCC Cent.	MCC Teste	AUC Teste	MCC Cent.	MCC Teste	AUC Teste	MCC Cent.	MCC Teste	AUC Teste	MCC Cent.	MCC Teste	AUC Teste	MCC Cent.	MCC Teste	AUC Teste	MCC Cent.	MCC Teste	AUC Teste	MCC Cent.	MCC Teste	AUC Teste
01	0,992	-0,243	0,149	1,000	-0,544	0,092	1,000	-0,161	0,746	1,000	-0,326	0,048	1,000	-0,068	0,493	1,000	-0,124	0,207	1,000	-0,097	0,619
02	0,182	0,112	0,732	0,708	0,273	0,693	0,672	0,140	0,401	0,708	0,294	0,655	0,866	-0,223	0,326	0,176	0,000	0,591	0,678	0,477	0,702
03	0,272	0,000	0,736	0,508	0,801	0,969	0,471	0,000	0,500	0,374	0,225	0,670	0,441	0,437	0,685	0,479	0,554	0,867	0,453	-0,100	0,668
04	0,016	-0,014	0,849	-0,404	-0,253	0,293	-0,201	0,000	0,483	-0,195	-0,026	0,520	-0,162	0,000	0,500	0,017	0,000	0,622	-0,161	-0,365	0,334
05	0,441	0,322	0,729	0,458	0,031	0,547	0,730	0,553	0,823	0,657	0,172	0,562	0,680	0,000	0,500	0,338	0,000	0,526	0,641	-0,050	0,485
06	-0,037	0,000	0,487	-0,004	-0,004	0,725	0,419	0,085	0,690	-0,065	0,173	0,573	0,384	0,000	0,500	0,127	0,000	0,543	0,377	0,020	0,463
07	0,374	0,359	0,713	0,373	0,047	0,614	0,003	0,000	0,500	0,142	0,081	0,578	0,329	0,000	0,500	0,252	0,114	0,588	0,308	0,147	0,555
08	0,327	0,132	0,059	0,431	0,258	0,649	0,245	0,157	0,560	0,299	0,055	0,705	0,510	0,000	0,500	0,406	0,070	0,597	0,187	0,089	0,601
09	0,102	0,000	0,765	0,597	-0,010	0,558	-0,039	0,041	0,530	0,061	0,000	0,789	0,077	0,000	0,500	0,586	0,099	0,355	0,248	0,187	0,608
10	-0,069	0,000	0,671	0,083	0,030	0,609	0,299	0,248	0,705	0,214	0,284	0,940	0,265	0,000	0,500	-0,116	0,000	0,445	0,002	0,000	0,579
11	0,625	0,351	0,711	0,139	0,036	0,523	0,188	0,004	0,663	0,001	0,000	0,518	0,140	0,000	0,500	0,931	0,341	0,625	-0,066	0,000	0,500
12	-0,123	0,000	0,048	0,209	0,076	0,654	0,132	0,171	0,586	0,512	0,170	0,793	-0,082	0,000	0,500	0,745	-0,050	0,577	0,558	0,225	0,586
13	0,503	0,290	0,762	0,295	0,131	0,577	-0,265	0,098	0,538	0,256	0,226	0,672	-0,005	0,000	0,500	-0,063	0,000	0,517	0,103	0,104	0,546
14	-0,253	0,030	0,528	-0,086	0,138	0,639	0,335	0,000	0,404	0,456	0,427	0,770	0,744	0,000	0,500	-0,240	0,000	0,435	-0,282	0,000	0,630
15	0,550	0,000	0,673	-0,170	-0,021	0,497	0,087	0,000	0,557	-0,137	0,000	0,465	0,204	0,000	0,500	-0,375	0,284	0,497	-0,354	0,246	0,651
16	-0,194	-0,060	0,577	-0,333	0,113	0,505	-0,076	0,000	0,500	-0,272	0,031	0,524	-0,448	0,289	0,646	-0,188	0,000	0,438	0,045	-0,056	0,463
17	0,745	0,339	0,846	-0,232	-0,097	0,483	-0,132	0,000	0,500	0,610	0,000	0,817	0,601	0,000	0,500	-0,291	0,241	0,667	0,879	-0,081	0,442
18	-0,553	0,000	0,530	0,803	0,188	0,624	0,602	0,000	0,500	0,861	0,659	0,949	0,558	0,000	0,500	0,682	0,061	0,540	0,841	0,085	0,562
19	0,702	0,097	0,588	-0,288	-0,115	0,494	0,532	0,000	0,500	-0,317	0,214	0,590	-0,571	-0,026	0,494	0,812	-0,252	0,333	0,761	-0,277	0,531
20	-0,424	0,051	0,503	0,929	0,452	0,843	0,368	0,000	0,500	0,562	0,000	0,823	-0,329	0,316	0,650	-0,596	0,000	0,720	0,724	-0,008	0,301
21	0,829	0,469	0,791	-	-	-	-0,373	0,189	0,630	0,773	0,009	0,719	-0,245	0,000	0,500	0,867	-0,140	0,461	0,789	0,471	0,748
22	-0,334	-0,166	0,384	-	-	-	-0,329	0,000	0,500	-0,498	0,000	0,600	0,796	-0,123	0,418	-0,476	0,120	0,514	-0,589	0,000	0,500
23	-0,502	-0,188	0,445	-	-	-	0,871	0,000	0,825	-0,537	0,148	0,565	-	-	-	-	-	-	-0,433	0,417	0,708
24	-	-	-	-	-	-	0,794	0,848	0,966	-0,367	0,000	0,683	-	-	-	-	-	-	-	-	-
MÉDIA	0,180	0,071	0,613	0,251	0,077	0,579	0,264	0,099	0,588	0,212	0,117	0,647	0,262	0,027	0,510	0,231	0,060	0,530	0,292	0,062	0,533

Tabela 3.5. Comparação dos algoritmos classificadores pelo teste estatístico *effect size* de Cohen's *d*

Classificadores	***Decision Table***	***J48***	***Multilayer Perceptron***	***Naive Bayes***	***OneR***	***Random Forest***	***Simple Logistic***
Decision Table		0.174357	0.202127	0.085581	0.396192	0.09451	**-0.092439**
J48	**-0.174357**		0.012645	**-0.097198**	0.194051	**-0.059518**	**-0.275062**
Multilayer Perceptron	**-0.202127**	**-0.012645**		**-0.120396**	0.205075	**-0.074602**	**-0.315410**
Naive Bayes	**-0.085581**	0.097198	0.120396		0.323398	0.023132	**-0.187471**
OneR	**-0.396192**	**-0.194051**	**-0.205075**	**-0.323398**		**-0.233153**	**-0.532358**
Random Forest	**-0.094512**	0.059518	0.074602	**-0.023132**	0.233153		**-0.179344**
Simple Logistic	0.092439	0.275062	0.315410	0.187471	0.532358	0.179344	

Ao final desta atividade apenas os modelos de predição cruzada de defeitos composto pelo melhor classificador e melhor par de algoritmos selecionadores de atributos foram mantidos para análise na atividade posterior. Sendo assim, conforme Figura 3.4, apenas 23 modelos de predição cruzada de defeitos foram mantidos para análise de desempenho, sendo esses compostos pelo algoritmo classificador *Simple Logistic*, algoritmo de seleção de atributos *CFS* e algoritmo de método de pesquisa *Genetic Search*, indicados como os de melhor desempenho em atividades anteriores.

3.5. Avaliar Modelos de Predição Cruzada

Quando um modelo faz uma predição, ele precisa ser avaliado para que se tenha conhecimento do seu desempenho (Ostrand; Weyuker, 2007; Bowes *et al.*, 2012). Sendo assim, a finalidade desta atividade foi avaliar a eficiência dos modelos de predição cruzada, criados a partir dos agrupamentos formados pelo algoritmo de clusterização BSAS tendo o valor de MCC para cálculo de similaridade entre os projetos de software utilizados na condução deste trabalho. Esta avaliação de desempenho foi realizada em duas etapas: a primeira foi realizar a aplicação de testes estatísticos que possibilitou comparar os modelos criados a partir dos agrupamentos formados com modelos que não realizam nenhum tipo de agrupamento, ou seja, utilizam toda a base de dados disponível para treinar o algoritmo classificador; e por fim, foram analisadas métricas de desempenho que, segundo a literatura especializada, são indicadores de um modelo de predição cruzada de defeitos eficiente.

A primeira avaliação foi feita a partir da comparação dos valores de MCC obtidos na predição local com os valores de MCC obtidos na predição global. Segundo Satin *et al.* (2015b), predição local é a predição realizada por um modelo de predição cruzada, treinado com instâncias de projetos agrupados por algum tipo de similaridade, para seu respectivo projeto de teste; já a predição global consiste na predição realizada sem qualquer tipo de agrupamento por similaridade, ou seja, o modelo é treinado com todos os projetos do conjunto de dados. Os resultados obtidos para as predições locais e globais podem ser observados na Tabela 3.6, bem como o número de instâncias de cada agrupamento formado e o número de instâncias do projeto selecionado para testar o modelo de predição cruzada, valores que se encontram, respectivamente, nas colunas "Instâncias do Cluster" e "Instâncias do Projeto". Os valores de MCC indicam que os modelos de predição locais e os modelos de predição globais obtiveram baixo desempenho. O modelo de predição cruzada formado através do agrupamento número 18 foi o que obteve o maior valor de MCC entre os demais, alcançando $0,659$, como mostra a Tabela 3.6. O modelo 17 obteve o maior MCC entre os modelos de predição cruzada que não utilizam agrupamento, alcançando $0,583$. Tanto o valor máximo de MCC obtido pelo modelo local quanto o alcançado pelo modelo global não são indicadores de bom preditores, uma vez que deveriam obter um valor de MCC superior à $0,8$ para serem considerados modelos com boa performance. Apesar dos valores de MCC não indicarem modelos com boa performance foi feita uma comparação de desempenho entre os modelos locais e globais. Para isso, os valores da coluna "MCC Predição Local" foram confrontados com os valores da coluna

"MCC Predição Global", permitindo avaliar se existe diferença estatística entre o desempenho dos modelos de predição cruzada criados a partir dos agrupamentos por similaridade e o desempenho das predições realizadas por modelos que não utilizam qualquer tipo de agrupamento por similaridade. A comparação entre os dois métodos foi realizada a partir da aplicação do teste estatístico com o uso da técnica Mann-Whitney-Wilcoxon (Sheskin, 2007). O teste retornou $p = 0,4523$, maior que $0,05$, apontando que não existe diferença estatística relevante entre o desempenho dos modelos que utilizam o valor de MCC para calculo de similaridade entre os projetos e o desempenho de modelos que não realizam nenhum tipo de agrupamento por similaridade.

Tabela 3.6. Resultado das predições realizadas localmente nos agrupamentos e globalmente com o algoritmo classificador *Simple Logistic*

Agrup.	Quantidade de Projetos	Instâncias do Cluster	Instâncias do Projeto	MCC Centróide	MCC Predição Local	MCC Predição Global	AUC Predição Local	AUC Predição Global
01	3	147	97	1,000	-0,326	-0,086	0,048	0,229
02	10	747	162	0,708	0,294	0,291	0,655	0,855
03	158	17591	876	0,374	0,225	0,000	0,670	0,553
04	44	1348	72	-0,195	-0,026	0,176	0,520	0,627
05	14	1058	317	0,657	0,172	-0,044	0,562	0,479
06	167	12966	460	-0,065	0,173	0,159	0,573	0,522
07	146	16611	1510	0,142	0,081	0,000	0,578	0,655
08	66	11081	1674	0,299	0,055	0,000	0,705	0,671
09	66	10131	3584	0,061	0,000	0,000	0,789	0,780
10	39	4048	503	0,214	0,284	0,388	0,940	0,912
11	277	31929	2959	0,001	0,000	0,000	0,518	0,770
12	51	5766	939	0,512	0,170	0,000	0,793	0,776
13	63	7799	605	0,256	0,226	0,232	0,672	0,689
14	59	6409	453	0,456	0,427	0,299	0,770	0,746
15	30	1081	90	-0,137	0,000	-0,131	0,465	0,473
16	14	302	42	-0,272	0,031	-0,021	0,524	0,602
17	16	1140	368	0,610	0,000	0,583	0,817	0,860
18	6	178	44	0,862	0,659	-0,390	0,949	0,371
19	6	108	40	-0,317	0,214	0,381	0,590	0,694
20	24	3552	829	0,562	0,000	0,131	0,823	0,756
21	13	1112	388	0,773	0,009	0,422	0,719	0,870
22	2	24	12	-0,498	0,000	0,000	0,600	0,443
23	3	53	31	-0,537	0,148	0,037	0,565	0,540
24	2	28	17	-0,367	0,000	-0,033	0,683	0,635

Para complementar a avaliação dos modelos de predição cruzada, foram calculadas medidas que são indicadores de desempenho do modelo: valores obtidos para precisão, sensibilidade e acurácia. Essas medidas podem ser calculadas a partir da matriz de confusão gerada para o teste de cada modelo de predição cruzada e sua escolha deve-se ao fato de permitir comparações com os trabalhos de Zimmermann *et al.* (2009), He *et al.* (2012), Watanabe *et al.* (2008), Zhang *et al.* (2014) e Satin *et al.* (2015b). Além disso, Zimmermann *et al.* (2009) e He *et al.* (2012) apresentam em seus respectivos trabalhos os valores mínimos de precisão, sensibilidade e acurácia que um modelo de predição cruzada precisa obter para que seja considerado transferível para projetos que não estão contidos na base de dados utilizada para a criação destes. Foram considerados os valores médios dessas medidas, por exemplo, para se calcular a precisão do modelo como um todo, foi calculado o valor de precisão obtido para a classe *Buggy* mais o valor obtido para a classe *Clean* e então dividido por dois, considerando-se assim a precisão das predições realizadas para ambas as classes (com defeito ou livre de defeito). O mesmo foi feito para as medidas de sensibilidade e acurácia. Essa abordagem foi utilizada com a finalidade de evitar que modelos que obtenham um bom rendimento preditivo para apenas uma das classes seja considerado eficiente. A Tabela 3.7 apresenta os valores de precisão, sensibilidade e acurácia obtidos para as predições locais e globais realizadas para os projetos selecionados para teste nos agrupamentos da coluna "Agrup.". Valores maiores que 0,75 encontram-se em negrito.

Tabela 3.7. Valores de precisão, sensibilidade e acurácia obtidos pelas predições realizadas localmente e globalmente nos agrupamentos formados pelo algoritmo classificador *Simple Logistic*

Agrup.	Precisão Pred. Local	Sensibilidade Pred. Local	Acurácia Pred.Local	Precisão Pred. Global	Sensibilidade Pred. Global	Acurácia Pred.Global
01	0,290	0,258	0,258	0,438	0,309	0,309
02	0,689	0,704	0,704	0,749	0,525	0,525
03	0,639	0,581	0,581	0,354	0,595	0,595
04	0,501	0,431	0,431	0,620	0,542	0,542
05	0,647	0,502	0,502	0,514	0,580	0,580
06	**0,791**	0,689	0,689	**0,783**	**0,796**	**0,796**
07	0,676	0,746	0,746	0,564	**0,751**	**0,751**
08	0,725	0,366	0,366	0,128	0,358	0,358
09	**0,919**	**0,959**	**0,959**	**0,919**	**0,959**	**0,959**
10	**0,952**	0,674	0,674	**0,943**	**0,907**	**0,907**
11	**0,782**	**0,884**	**0,884**	**0,782**	**0,884**	**0,884**
12	**0,810**	0,358	0,358	0,655	**0,809**	**0,809**
13	0,646	0,658	0,658	0,681	0,681	0,681
14	**0,757**	0,742	0,742	0,724	0,735	0,735
15	0,250	0,500	0,500	0,404	0,456	0,456
16	0,527	0,548	0,548	0,496	0,548	0,548
17	0,507	0,712	0,712	**0,832**	**0,837**	**0,837**
18	**0,861**	**0,795**	**0,795**	0,292	0,295	0,295
19	0,713	0,475	0,475	**0,751**	0,650	0,650
20	0,609	**0,780**	**0,780**	**0,832**	**0,785**	**0,785**
21	0,657	0,255	0,255	**0,832**	**0,820**	**0,820**
22	0,340	0,583	0,583	0,174	0,417	0,417
23	0,578	0,581	0,581	0,53	0,548	0,548
24	0,585	**0,765**	**0,765**	0,626	0,471	0,471

Segundo Zimmermann *et al.* (2009), um modelo de predição é transferível para outros projetos quando os valores para precisão, sensibilidade e acurácia são maiores que $0,75$. Sendo assim, os modelos de predição cruzada de defeitos formados a partir dos agrupamentos e representados por 09, 11 e 18 na coluna "Agrup." da Tabela 3.7 são modelos de predição, que segundo a proposta de Zimmermann *et al.* (2009), apresentam bom desempenho e poderiam estar sendo utilizados para predizer defeitos em projetos que não se encontram no conjunto de dados utilizado na condução deste trabalho. Estes três modelos de predição cruzada de defeitos representam juntos $12,5\%$ dos modelos de predição cruzada criados, $27,28\%$ dos projetos agrupados e $31,23\%$ das instâncias dos clusters.

Ao contrário de Zimmermann *et al.* (2009), He *et al.* (2012) conclui em seu trabalho de pesquisa que um modelo de predição cruzada de defeitos é transferível para outros projetos quando obtêm uma precisão maior que $0,5$ e sensibilidade maior que $0,7$. Sendo assim, os modelos de predição cruzada de defeitos representados por 02, 07, 09, 11, 14, 17, 18, 20 e 24 na coluna "Agrup." da Tabela 3.7 são aqueles que segundo a proposta de He *et al.* (2012) apresentam um bom desempenho e que poderiam estar sendo utilizados para predizer defeitos em projetos que não se encontram no conjunto de dados utilizado na condução deste trabalho. Os nove modelos de predição cruzada de defeitos representam juntos $37,5\%$ dos modelos, $47,38\%$ dos projetos agrupados e $52,3\%$ das instâncias dos clusters.

Capítulo 4

Discussão dos Resultados

4.1. Observações sobre os algoritmos selecionadores de atributos

Algoritmos de método de pesquisa e algoritmos de seleção de atributos foram utilizados para auxiliar os classificadores no processo de predição de defeitos. O desempenho de modelos de predição que utilizam apenas dados históricos do projeto alvo, mas especificamente os valores de MCC obtidos durante etapa de teste desses modelos, serviram como um parâmetro de avaliação de desempenho entre os algoritmos selecionadores de atributos, já que cada modelo é formado por um conjunto de treino e teste, um algoritmo classificador, um algoritmo de método de pesquisa e um algoritmo de seleção de atributos. Primeiramente, o teste estatístico de Kruskal-Wallis foi utilizado com a finalidade de se observar a existência de diferença estatística relevante de desempenho entre os algoritmos selecionadores. O valor obtido com a aplicação do teste foi de $p = 0,02991$, ou seja, $< 0,05$, indicando que o desempenho obtido pelos modelos de predição analisados não são estatisticamente iguais. O resultado obtido permitiu seguir com a hipótese de que existe diferença estatística relevante entre os algoritmos selecionadores, possibilitando analisar a magnitude dessa diferença e consequentemente apontar o par de algoritmos com desempenho superior aos demais.

A diferença de desempenho entre os algoritmos selecionadores de atributos foi obtida com a aplicação do teste estatístico *effect size* de Cohen's *d*. O teste apontou *CFS* como o melhor algoritmo de método de seleção de atributos e *Genetic Search* como melhor algoritmo de método de pesquisa. A magnitude da diferença de desempenho dos algoritmos *CFS* e *Genetic Search* foram superiores frente aos demais algoritmos: 0.022514 em relação a *CFS + Best First*; 0.026422 em relação a *CFS + Greedy Stepwise*; 0.010938 em relação a *Chi-Square + Ranker*; 0.014506 em relação a *Gain-Ratio + Ranker*; e, 0.013756 em relação a *Information Gain + Ranker*. Ordenando por melhor desempenho aparecem: *CFS + Genetic Search*, *Chi-Square + Ranker*, *Information Gain + Ranker*, *Gain-Ratio + Ranker*, *CFS + Best First*, e por fim, *CFS + Greedy Stepwise*.

Uma observação importante é que apesar de existir uma diferença de desempenho e essa ser estatisticamente relevante, a magnitude é muito pequena, sendo 0.026422 a maior diferença de desempenho encontrada. Os resultados obtidos com a aplicação do teste estatístico de *Scott Knott* reforçam esta afirmação, uma vez que a execução do teste gerou apenas um agrupamento contendo todos os algoritmos de seleção de atributos, como pode ser observado na Tabela 3.2.

4.2. Observações sobre os agrupamentos formados

Os agrupamentos foram formados pelo algoritmo de clusterização BSAS tendo o valor de MCC, obtido com o teste dos modelos de predição, para cálculo de similaridade. A etapa de clusterização foi executada individualmente para cada um dos sete classificadores utilizados, fazendo com que a quantidade de agrupamentos formados variasse de um para outro: *Naive Bayes* com 23 agrupamentos, *Random Forest* com 20 agrupamentos, *Decision Table* com 24 agrupamentos, *Simple Logistic* com 24 agrupamentos, *OneR* com 22 agrupamentos, *Multilayer Perceptron* com 22 agrupamentos e *J48* com 23 agrupamentos. A quantidade de agrupamentos formados é de se destacar, pois cada um deles resultou na criação de um modelo de predição cruzada de defeitos, totalizando 158 modelos de predição cruzada e aumentando a probabilidade de identificar modelos com bom desempenho. Outro fator que destaca a influência dos classificadores na criação dos agrupamentos foi o projeto selecionado para teste, que também sofreu variação. Por exemplo, o projeto selecionado para teste no agrupamento 01 formado com os valores de desempenho do classificador *Naive Bayes* foi o projeto **officefloor**, enquanto que para o mesmo agrupamento, porém formado com o classificador *Random Forest*, o projeto selecionado para teste foi o **smartgwt**. O agrupamento 01 foi o único que não sofreu variação do projeto selecionado para teste, sendo **ivef-sdk** o selecionado em todos eles, independentemente do classificador. Isso ocorreu pelo fato de que todos os modelos de predição obtiveram bom desempenho quando o projeto alvo era o **ivef-sdk**, fazendo com que este sempre ficasse posicionado em um agrupamento com centroide igual ou próximo de 1, e por possuir mais instâncias que os demais projetos do cluster, sempre foi selecionado como projeto de teste. A Tabela 4.1 exibe os projetos selecionados para teste em cada agrupamento segundo o algoritmo classificador utilizado.

Tabela 4.1. Projetos selecionados para teste em cada agrupamento segundo o algoritmo classificador

Agrup.	Naive Bayes	Random Forest	Decision Table	Simple Logistic	OneR	Multilayer Perceptron	J48
01	ivef-sdk	ivef-sdk	ivef-sdk	ivef-sdk	ivef-sdk	ivef-sdk	ivef-sdk
02	officefloor	smartgwt	lsb	uface	moast	jfreereport	lsb
03	lportal	wow-qrsk	uwom	reactos-mirror	emeraldemu	wow-qrsk	wow-qrsk
04	qtwin	freemarker	emite	red5	bacnet	unladen-swallow	gephex
05	decidr	t-2	smartgwt	rodin-b-sharp	lsb	lportal	rodin-b-sharp
06	massiv	qtwin	jbpm	bloodycore	wow-qrsk	mplayer-ce	brlcad
07	electric-core	lportal	lportal	brlcad	ltp	applet2app	lportal
08	brlcad	electric-core	applet2app	applet2app	baadengine	jitterbit	enlightenment
09	sepgsql	baadengine	flexpay	lportal	mplayer-ce	baadengine	mediaportal
10	zemljanka	atunes	brlcad	jasperreports	applet2app	panda3d	unladen-swallow
11	magicwars	mplayer-ce	vdsf	unladen-swallow	yale	arcanea-project	mesa3d
12	openmsx	officefloor	azureus	wow-qrsk	google-web-toolkit	smartgwt	uwom
13	wow-qrsk	applet2app	jstock	openwbem	lportal	v8	jfreereport
14	trackit	unladen-swallow	ltp	emeraldemu	customsagetv	fluidium	neostats
15	jitterbit	opennms	yale	betoffice	hibernate	hawkscope	fontforge
16	empyrean	moving-pictures	omseek	glsof	kinkatta	haphazard	google-web-toolkit
17	uface	array4j	google-web-toolkit	acdk	magicwars	qdvdauthor	valgrind
18	zoie	ngl	baadengine	arcanea-project	konversation	ngl	moast
19	uclmda	tcl	wow-qrsk	scintilla	ra-ajax	uface	smartgwt
20	fontforge	chabber	reactos-mirror	uwom	pure-data	webcompmath	uclmda
21	moast	-	hawkscope	smartgwt	libmesh	fleet-simulator	uface
22	cpcsdk	-	gfs	freeimage	co-ode-owl-plugins	freeimage	freeimage
23	cupsfilter	-	uclmda	ra-ajax	-	-	jsearchy
24	-	-	moast	bscweasel	-	-	-

Uma dos pontos negativos do algoritmo de clusterização BSAS é criar um número muito elevado de agrupamentos, aumentando a possibilidade de haver agrupamentos com um único elemento (Kainulainen; Kainulainen, 2002). Esta característica do algoritmo BSAS acaba se tornando um problema, pois agrupamentos contendo um único projeto precisam ser descartados pois não permitem que o compartilhamento de dados históricos ocorra. Sendo assim, é importante observar se o número de projetos e instâncias que estão sendo descartados não é um número muito elevado. Para o trabalho em questão este problema foi amenizado utilizando-se uma medida de similaridade de 0,034. Esta medida é um dos parâmetros de entrada do algoritmo BSAS e ela que vai definir se a inclusão de um determinado elemento será realizada em um agrupamento já existente ou se um novo agrupamento deverá ser criado. O resultado foi satisfatório pois o pior caso ocorreu com os agrupamentos baseados no algoritmo classificador *Decision Table* no qual precisou-se descartar 5

projetos por estarem contidos unicamente em um agrupamento, totalizando 66 instâncias, que representa apenas $0,04\%$ de todas as instâncias do conjunto de dados.

Um problema que afeta negativamente o desempenho de modelos de predição é o desbalanceamento de classes. Por esse motivo foi feita uma análise da ocorrência desse problema nos agrupamentos formados através da medida de MCC para cálculo de similaridade. A análise foi feita apenas para os agrupamentos formados pelo classificador *Simple Logistic*, por ser o que obteve melhor performance. A Tabela 4.2 apresenta a taxa de instâncias rotuladas como *Buggy* e *Clean* para cada agrupamento. O resultado foi satisfatório, pois percebe-se que o desbalanceamento de classes não foi frequente. Apenas o agrupamento 01 obteve alta taxa de desbalanceamento, com mais de 80% das instâncias rotuladas como *Clean*. No entanto, este agrupamento contém apenas 3 projetos, totalizando 147 instâncias, que corresponde à $0,2\%$ dos projetos da base de dados e $0,1\%$ das instâncias. Os agrupamentos 02, 03, 04, 10, 12, 13, 14, 15, 16, 17, 18, 19 e 23 merecem destaque, pois obtiveram uma divisão bastante próxima da ideal. Sendo assim, conclui-se que a medida de correlação MCC contribuiu para contornar-se o problema de desbalanceamento de classes.

Tabela 4.2. Número de instâncias rotuladas como *Buggy* e *Clean* em cada um dos agrupamentos formados com base no classificador *Simple Logistic*

Agrup.	Instâncias Buggy (%)	Instâncias Clean (%)
01	12,00%	88,00%
02	49,00%	51,00%
03	43,89%	56,11%
04	45,69%	54,31%
05	57,76%	42,24%
06	33,98%	66,02%
07	37,51%	62,49%
08	37,33%	62,67%
09	33,91%	66,09%
10	44,54%	55,46%
11	28,64%	71,36%
12	50,40%	49,60%
13	44,15%	55,85%
14	44,31%	55,69%
15	48,13%	51,87%
16	45,00%	55,00%
17	49,35%	50,65%
18	49,25%	50,75%
19	52,94%	47,06%
20	35,44%	64,56%
21	57,60%	42,40%
22	41,67%	58,33%
23	50,00%	50,00%
24	27,27%	72,73%

4.3. Observações sobre os modelos de predição cruzada de defeitos

Inicialmente, apenas os modelos de predição cruzada formados pelo algoritmo classificador *Simple Logistic* foram considerados para análise, pois este foi o classificar apontado como sendo o que obteve melhor desempenho frente aos demais pelo teste estatístico de Cohen's *d*. A primeira análise consistiu em observar se o desempenho dos modelos de predição cruzada construídos é superior ao desempenho de modelos que não realizam nenhum tipo de agrupamento por similaridade para compartilhamento de dados históricos para treino dos modelos. Para alcançar este objetivo, o teste estatístico de Mann-Whitney-Wilcoxon (Sheskin, 2007) foi aplicado sobre os valores de *MCC Predição Local* e de *MCC Predição Global*, apresentados na

Tabela 3.6. A execução do teste estatístico retornou $p = 0,4523$, indicando que estatisticamente não houve diferença entre realizar as predições com os agrupamentos formados ou sem utiliza-los.

Mesmo não podendo afirmar que existe uma diferença estatística entre as predições locais e globais o desempenho de ambos foi avaliado com base na proposta de Zimmermann *et al.* (2009) e He *et al.* (2012). Zimmermann *et al.* (2009) apresenta em seu trabalho indicadores que classificam um modelo de predição como sendo de bom de desempenho. Segundo ele o teste de um modelo deve retornar valores de precisão, acurácia e sensibilidade maiores que $0,75$. Analisando as colunas "Precisão Pred. Local", "Sensibilidade Pred. Local" e "Acurácia Pred. Local" da Tabela 3.7 percebe-se que os modelos de predição cruzada com bom desempenho, segundo a proposta de Zimmermann *et al.* (2009), são os modelos 09, 11 e 18. Juntos representam $12,5\%$ dos modelos de predição cruzada criados, $27,28\%$ dos projetos agrupados e $31,23\%$ das instâncias dos clusters. Já os valores de precisão, sensibilidade e acurácia obtidos com a predição global podem ser observados nas colunas "Precisão Pred. Global", "Sensibilidade Pred. Global" e "Acurácia Pred. Global". A mesma análise foi realizada sobre esses valores, sendo os modelos 06, 09, 10, 11, 17, 20 e 21 são os com bom desempenho. Sendo assim, $29,16\%$ dos modelos globais obtiveram bom desempenho, segundo a proposta de Zimmermann *et al.* (2009).

Os indicadores de bom desempenho apresentados por He *et al.* (2012) também foram aplicados. Seu trabalho indica que valores de precisão maiores que 0,5 e sensibilidade maiores que 0,7 são indicadores de um modelo de predição com bom desempenho. Analisando a Tabela 3.7, percebe-se que os modelos de predição cruzada 02, 07, 09, 11, 14, 17, 18, 20 e 24 são os que obtiveram valores superiores que os propostos por He *et al.* (2012), representando $37,5\%$ dos modelos de predição cruzada criados, $47,38\%$ dos projetos agrupados e $52,3\%$ das instâncias dos clusters. A mesma avaliação foi feita para as predições realizadas pelos modelos de predição cruzada globais. Os modelos 06, 07, 09, 10, 11, 12, 14, 17, 20 e 21 são os que apresentam bom desempenho, representado $41,66\%$ dos modelos de predição cruzada globais.

A partir da análise dos testes estatísticos aplicados, da análise de desempenho proposta por Zimmermann *et al.* (2009) e da análise proposta por He *et al.* (2012) conclui-se que os modelos de predições cruzadas que não utilizam nenhum tipo de agrupamento por similaridade obtiveram resultados superiores aos modelos de predição cruzada criados a partir dos agrupamentos formados pelo algoritmo BSAS tendo o valor de MCC para cálculo de similaridade. No entanto percebe-se que o valor de MCC é um valor mais robusto para análise de desempenho. Por exemplo, os modelos de predição cruzada globais 06, 09, 10, 11, 17, 20 e 21 apresentados na Tabela 3.7, que segundo Zimmermann *et al.* (2009), são os que possuem um bom desempenho e que poderiam ser utilizados para predizer defeitos de projetos externos, obtiveram valores baixos de MCC quando testados: $0,159$, $0,000$, $0,388$, $0,000$, $0,583$, $0,131$ e $0,422$, respectivamente, como mostra a tabela Tabela 3.6. O mesmo ocorre com os modelos criados a partir dos agrupamentos. Os modelos indicados por Zimmermann *et al.* (2009) como sendo os de melhor desempenho, 09, 11 e 18, também obtiveram baixos valores de MCC, exceto o modelo 18 que obteve um valor de MCC de $0,659$. Os modelos 09 e 11 obtiveram um MCC igual a $0,000$. O mesmo ocorre com os modelos apontados por He *et al.* (2012) como sendo os de bom desempenho e que poderiam estar sendo utilizados para predizer defeitos em projetos externos, os valores de MCC obtidos por estes são baixos.

Uma mesma análise também pode ser realizada comparando os valores de MCC com os valores de AUC, este último bastante utilizado como indicador de desempenho em trabalhos semelhantes. A Tabela 3.6 apresenta esses valores para as predições locais e globais, nas colunas "MCC Pred. Local", "MCC Pred. Global", "AUC Pred. Local" e "AUC Pred. Global". Os modelos 09, 10, 12, 14, 17, 18 e 20 são aqueles que obtiveram valores de AUC maiores que $0,75$, considerado um indicador de modelos com bom desempenho. No entanto, o desempenho desses modelos pode ser questionado quando o valor de MCC é considerado para

análise de desempenho. Os mesmos modelos que obtiveram autos valores de AUC obtiveram $0,000$, $0,284$, $0,170$, $0,427$, $0,000$, $0,659$ e $0,000$ de MCC, respectivamente. Com exceção do modelo 18 que obteve um MCC maior que os demais, o restante obtiveram valores de MCC extremamente baixos.

4.4. Comparação dos resultados obtidos com o estado da arte

Os trabalhos de Zimmermann *et al.* (2009), He *et al.* (2012), Zhang *et al.* (2014) e Satin *et al.* (2015a) foram os encontrados na literatura com possibilidade de comparação dos resultados. Vale ressaltar que é difícil encontrar formas que possibilitem a comparação de resultados entre trabalhos semelhantes (Satin *et al.*, 2015a). Esta dificuldade resulta da diferença da técnica aplicada, diferença dos métodos de avaliação empregados e diferença no conjunto de dados utilizados. Sendo o conjunto de dados o maior fator que dificulta a realização de comparações de resultados (Hall *et al.*, 2012). Sendo assim, foram utilizadas as medidas de desempenho propostas por Zimmermann *et al.* (2009) e He *et al.* (2012), pois possibilita a comparação com os trabalhos listados.

O trabalho de Zimmermann *et al.* (2009) avaliou seus modelos de predição cruzada com base nos valores de precisão, sensibilidade e acurácia. Como já retratado em outras partes do texto, modelos que consigam obter precisão, sensibilidade e acurácia maior que $0,75$ apresentam um bom desempenho segundo sua abordagem. Zimmermann *et al.* (2009) obteve uma taxa de sucesso de $3,40\%$ enquanto que o presente trabalho obteve $12,5\%$ dos modelos formados pelo algoritmo classificador *Simple Logistic* pelo método sobre análise.

He *et al.* (2012) utilizou uma abordagem semelhante a de Zimmermann *et al.* (2009) no que diz respeito a estabelecer medidas que sejam indicadores de um modelo de predição com bom desempenho. He *et al.* (2012) propõe que os modelos de predição devem obter uma precisão maior que $0,5$ e sensibilidade maior que $0,7$ para que sejam considerados modelos com bom desempenho e que possam ser utilizados para predizer defeitos em projetos externos. Seguindo essa proposta, o trabalho em questão alcançou uma taxa de $37,5\%$ de modelos com bom desempenho enquanto que a abordagem de He *et al.* (2012) obteve $0,32\%$.

Os principais trabalhos para comparação são os propostos por Zhang *et al.* (2014) e Satin *et al.* (2015b). Ambos utilizaram o mesmo conjunto de dados que o utilizado no presente trabalho e os mesmos critérios de avaliação estabelecidos por Zimmermann *et al.* (2009) e He *et al.* (2012). Além disso, Satin *et al.* (2015b) também utilizou uma medida de desempenho para o cálculo de similaridade entre os projetos e o presente trabalho é fortemente baseado em sua abordagem. Aplicando o método de validação proposto por Zimmermann *et al.* (2009) e He *et al.* (2012), o trabalho de Zhang *et al.* (2014) obteve uma taxa de 3% e 14%, respectivamente. Já o presente trabalho obteve $12,5\%$ e $37,5\%$, taxas superiores às de Zhang *et al.* (2014). O trabalho de Satin *et al.* (2015a) é o que obteve maior desempenho entre os trabalhos relacionados, com uma taxa de $31,58\%$ para a abordagem de Zimmermann *et al.* (2009) e $42,11\%$ para a abordagem de He *et al.* (2012). Comparado esses valores com os obtidos pelo trabalho aqui proposto, percebe-se que o trabalho de Satin *et al.* (2015a) obteve um desempenho superior. No entanto, a medida de desempenho utilizada por Satin *et al.* (2015a) para agrupamento dos projetos e avaliação destes através de testes estatísticos foi AUC, que indicou o algoritmo *Naive Bayes* como sendo o que obteve melhor performance. Já o presente trabalho utilizou MCC para formação dos agrupamentos e também para validação através de testes estatísticos, apontando o classificador *Simple Logistic* como sendo o melhor entre os utilizados. Dessa forma, o algoritmo classificador *Naive Bayes* foi utilizado para fins de comparação, mesmo não sendo o que obteve melhor desempenho, através da análise do valor de MCC obtido por cada um dos classificadores durante a realização de predição cruzada. A Tabela 4.3 apresenta os valores de precisão, acurácia e sensibilidade para os modelos

de predição cruzada de defeitos pelo algoritmo classificador *Naive Bayes*. Mais uma vez, vale destacar, que esses valores representam os valores médios de precisão, sensibilidade e acurácia do modelo, calculados com base nos valores obtidos para ambas as classes (com defeito ou ausente de defeito).

Tabela 4.3. Valores de precisão, sensibilidade e acurácia obtidos pelas predições realizadas localmente nos agrupamentos formados para o algoritmo classificador *Naive Bayes* usando MCC

Agrup.	**Precisão**	**Sensibilidade**	**Acurácia**
01	0,382	0,289	0,289
02	**0,956**	**0,809**	**0,809**
03	**0,919**	**0,959**	**0,959**
04	**0,995**	**0,918**	**0,918**
05	0,698	0,613	0,613
06	**0,986**	**0,993**	**0,993**
07	**0,789**	**0,810**	**0,810**
08	0,715	0,403	0,403
09	**0,910**	**0,954**	**0,954**
10	0,002	0,043	0,043
11	0,719	0,729	0,729
12	0,167	0,409	0,409
13	**0,782**	**0,797**	**0,797**
14	0,559	0,438	0,438
15	0,527	0,726	0,726
16	0,491	0,374	0,374
17	**0,787**	0,537	0,537
18	0,250	0,500	0,500
19	**0,938**	**0,923**	**0,923**
20	0,547	0,473	0,473
21	**0,922**	**0,915**	**0,915**
22	0,292	0,516	0,516
23	0,412	0,040	0,040

Analisando a Tabela 4.3 pode-se perceber que $39,13\%$ dos modelos apresentam bom desempenho segundo a abordagem de Zimmermann *et al.* (2009), e que $47,82\%$ dos modelos apresentam bom desempenho segundo a abordagem de He *et al.* (2012). Essas taxas são superiores as obtidas por Satin *et al.* (2015a), de $31,58\%$ e $42,11\%$, para seus modelos de predição cruzadas formados a partir da utilização de AUC para cálculo de similaridade e utilizando o algoritmo classificador *Naive Bayes*. Esta é mais uma evidência de que MCC é uma medida de desempenho mais robusta, uma vez que *Naive Bayes* foi indicado como o quarto melhor classificador, segundo análise estatística aplicada sobre os valores de MCC obtidos pelos sete classificadores analisados.

Capítulo 5

Conclusão

Predizer defeitos em software é uma tarefa importante pois reduz o tempo que a equipe de desenvolvimento leva para identificar a ocorrência de defeitos no código fonte e aumenta a percepção da qualidade do software por parte do usuário. Apesar das vantagens, realizar predição de defeitos se torna uma tarefa difícil em um cenário em que os projetos de software estão em fase inicial de desenvolvimento, pois não possuem uma base de dados histórica consolidada que possa ser utilizada para treinar modelos de predição. Sendo assim, faz-se necessário encontrar projetos que tenham algum tipo de similaridade com o projeto alvo. Outros trabalhos utilizam algoritmos de agrupamento para encontrar padrões de similaridade através das métricas de software (Zimmermann *et al.*, 2009; Zhang *et al.*, 2014). O presente trabalho utilizou uma medida de correlação para cálculo de similaridade entres os projetos de software. Essa medida é um indicador de desempenho obtida através de modelos de predição que utilizam apenas os dados históricos do projeto alvo para realizar a predição.

Geralmente um modelo de predição cruzada de defeitos é formado por um algoritmo de agrupamento, um algoritmo classificador e um algoritmo de seleção de métricas. Os estudos sobre o impacto dos algoritmos de seleção de atributos (Khoshgoftaar *et al.*, 2012; Gao; Khoshgoftaar, 2015; Malhotra, 2015) e o impacto dos algoritmos classificadores (Lessmann *et al.*, 2008; Ghotra *et al.*, 2015) quanto ao desempenho dos modelos de predição são controversos. Sendo assim, este trabalho preocupou-se em estudar a diferença de desempenho de 6 pares de algoritmos selecionadores de atributos, cada um formado por um algoritmo de método de pesquisa e um algoritmo de seleção de atributos, e de 7 classificadores, cada um utilizando abordagens diferentes. Por meio da aplicação de testes estatísticos, pôde-se concluir que existe diferença estatística relevante entre o desempenho dos algoritmos selecionadores de atributos. Apesar de ser uma diferença pequena este era um dos objetivos específicos do trabalho, comparar o desempenho dos algoritmos selecionadores de atributos. O algoritmo de seleção de atributos *CFS* e o algoritmo de método de pesquisa *Genetic Search* foram apontados como os que melhor auxiliam o algoritmo classificador no processo de predição. Por outro lado, a aplicação de testes estatísticos apontou não haver diferença estatística relevante entre o desempenho dos algoritmos classificadores. Mesmo não havendo diferença estatística relevante, o desempenho de cada um foi analisado com a finalidade de indicar um com melhor desempenho. A análise dos valores de MCC através de um teste estatístico que mede a magnitude da diferença entre pares de amostras apontou o algoritmo classificador *Simple Logistic* como o que obteve melhor performance.

Outro ponto que merece destaque foi a análise sobre o desbalanceamento de classes, que afeta negativamente o desempenho dos modelos de predição. Pôde-se perceber que dos 24 modelos de predição cruzada de defeitos analisados apenas 1 apresentou o problema de desbalanceamento de classe de forma mais significativa, e a maioria se aproximou de uma divisão ideal das instâncias. O agrupamento que apresentou o problema de desbalanceamento de classe é formado por 3 projetos, totalizando 147 instâncias, que corresponde à $0,2\%$ dos projetos e $0,1\%$ de todas as instâncias do conjunto de dados. São taxas menores que as obtidas por Satin *et al.* (2015a), o qual obteve $2,416\%$ dos projetos e $1,04\%$ das instâncias contidas em agrupamentos desbalanceados. Sendo assim, pode-se concluir que MCC contribuiu para a formação de modelos de predição cruzada com bases de treino balanceadas com relação as classes.

Os modelos de predição cruzada de defeitos foram avaliados com base nos valores de MCC obtidos durante etapa de teste dos modelos e com base em indicadores de desempenho sugeridos pela literatura. As análises de desempenho com base nos valores de MCC indicaram que modelos criados sem nenhum tipo de agrupamento por similaridade obtém resultados superiores que os modelos criados a partir dos agrupamentos formados pelo algoritmo de clusterização BSAS tendo os valores de MCC para cálculo de similaridade. Esse resultado é relevante, uma vez que os modelos de predição cruzada criados obtiveram resultados superiores que modelos criados por outros trabalhos encontrados na literatura, seguindo indicadores de desempenho também utilizados por esses. Outro ponto importante e que merece destaque, é a análise realizada sobre os valores de MCC, AUC, precisão, sensibilidade e acurácia. Pôde-se perceber que MCC é uma medida mais robusta de desempenho, pois alguns dos modelos observados obtiveram baixos valores de MCC para autos valores de AUC, precisão, sensibilidade e acurácia. Sendo assim, um estudo mais aprofundado sobre a análise de desempenho de algoritmos selecionadores de atributos faz-se necessário.

5.1. Limitações do Trabalho

A primeira limitação deste trabalho a ser destacada está relacionada com a base de dados utilizada. Ela consiste na mesma base de dados utilizada por Zhang *et al.* (2014), sendo assim, além do pré processamento realizado considerou-se que os dados foram minerados e tratados de forma adequada por ele. Além disso, os resultados apresentados neste trabalho são válidos apenas para a base de dados utilizada, novos conjunto de dados desem ser estudados para criação, treino, teste e validação de modelos de predição cruzada de defeitos.

A análise realizada sobre os algoritmos classificadores e sobre os algoritmos de seleção de atributos são válidas diante do método sobre análise, além disso, novos algoritmos devem ser selecionados e estudados para contribuir com o estado da arte sobre sua real influência e desempenho. Também vale ressaltar que os algoritmos foram executados com a parametrização padrão da ferramenta Weka, um estudo que explore melhor seus atributos pode vir a ganhar em performance. O algoritmo de clusterização tem total influência nos agrupamentos formados, sendo assim, novos algoritmos classificadores além do BSAS devem ser utilizados, com possibilidade de comparação de desempenho entre algoritmos diversos.

5.2. Trabalhos Futuros

Estudos mais aprofundados podem trazer grandes contribuições para o estado da arte. Estudar um número maior de algoritmos classificadores e algoritmos selecionadores de métricas traria grandes benefícios, tanto para análise da existência de diferença de desempenho entre eles quanto para criação e análise de um número maior de modelos de predição cruzada de defeitos.

O presente estudo mostrou que MCC trata-se de uma medida de desempenho mais robusta que as demais, sendo assim, explorar mais essa medida de correlação pode trazer ganhos para a criação e avaliação dos modelos de predição. Talvez explorar a utilização de medidas de desempenho em conjunto, como por exemplo MCC, AUC, precisão, sensibilidade e acurácia, possa trazer grandes contribuições para a formação de agrupamentos que venham a se transformar em modelos de predição cruzada de defeitos eficientes.

O desempenho de algoritmos de clusterização também pode ser estudado, já que este tem influência direta na criação dos modelos de predição cruzada de defeitos. Métricas como *Davies–Bouldin index* podem ser utilizadas com o objetivo de medir o quão similar são os projetos contidos em um mesmo agrupamento, avaliando assim a eficiência do método de clusterização utilizado.

Capítulo 6

Referências

ALPAYDIN, Ethem. *Introduction to Machine Learning*. 2nd. ed. The MIT Press, 2010. ISBN 026201243X, 9780262012430. Disponível em: `http://stp.lingfil.uu.se/~santinim/ml/2014/Alpaydin2010_IntroductionToMl_2ed.pdf`.

ANDERSON, Theodore W; DARLING, Donald A. Asymptotic theory of certain"goodness of fit"criteria based on stochastic processes. *The annals of mathematical statistics*, JSTOR, p. 193–212, 1952.

BACCHELLI, Alberto; DÁMBROS, Marco; LANZA, Michele. Are popular classes more defect prone? In: *Proceedings of the 13th International Conference on Fundamental Approaches to Software Engineering*, 2010. (FASE'10), p. 59–73. ISBN 3-642-12028-8, 978-3-642-12028-2.

BELL, Robert M.; OSTRAND, Thomas J.; WEYUKER, Elaine J. Looking for bugs in all the right places. In: *Proceedings of the 2006 International Symposium on Software Testing and Analysis*, 2006. (ISSTA '06), p. 61–72. ISBN 1-59593-263-1.

BOWES, David; HALL, Tracy; GRAY, David. Comparing the performance of fault prediction models which report multiple performance measures: Recomputing the confusion matrix. In: *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*, 2012. (PROMISE '12), p. 109–118. ISBN 978-1-4503-1241-7. Disponível em: `http://doi.acm.org/10.1145/2365324.2365338`.

CATAL, Cagatay; DIRI, Banu. A systematic review of software fault prediction studies. *Expert Systems with Applications*, Elsevier Ltd, v. 36, n. 4, p. 7346–7354, maio 2009.

CHANDRASHEKAR, Girish; SAHIN, Ferat. A survey on feature selection methods. *Computers & Electrical Engineering*, v. 40, n. 1, p. 16 – 28, 2014.

COHEN, Jacob. *Statistical power analysis for the behavioral sciences (rev.* [S.l.]: Lawrence Erlbaum Associates, Inc, 1977.

FACELI, Katti; LORENA, Ana C; GAMA, João; CARVALHO, ACPLF. Inteligência artificial: Uma abordagem de aprendizado de máquina. *Livros Técnicos e Científicos*, p. 381, 2011.

GAO, Kehan; KHOSHGOFTAAR, Taghi M. Assessments of feature selection techniques with respect to data sampling for highly imbalanced software measurement data. *International Journal of Reliability, Quality and Safety Engineering*, World Scientific, v. 22, n. 02, p. 1550010, 2015.

GHOTRA, Baljinder; MCINTOSH, Shane; HASSAN, Ahmed E. Revisiting the impact of classification techniques on the performance of defect prediction models. *37th International Conference on Software Engineering (ICSE 2015)*, 2015.

GUYON, Isabelle; ELISSEEFF, André. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, JMLR.org, v. 3, p. 1157–1182, mar. 2003. ISSN 1532-4435.

HALL, M.A.; HOLMES, G. Benchmarking attribute selection techniques for discrete class data mining. *Knowledge and Data Engineering, IEEE Transactions on*, v. 15, n. 6, p. 1437–1447, Nov 2003. ISSN 1041-4347.

HALL, Mark A.; SMITH, Lloyd A. *Practical Feature Subset Selection for Machine Learning*. 1998.

HALL, Tracy; BEECHAM, Sarah; BOWES, David; GRAY, David; COUNSELL, Steve. A systematic literature review on fault prediction performance in software engineering. *IEEE Trans. Softw. Eng.*, IEEE Press, Piscataway, NJ, USA, v. 38, n. 6, p. 1276–1304, nov. 2012. ISSN 0098-5589.

HE, Zhimin; SHU, Fengdi; YANG, Ye; LI, Mingshu; WANG, Qing. An investigation on the feasibility of cross-project defect prediction. *Automated Software Engineering*, Springer US, v. 19, n. 2, p. 167–199, 2012. ISSN 0928-8910.

HERBOLD, Steffen. Training data selection for cross-project defect prediction. In: *Proceedings of the 9th International Conference on Predictive Models in Software Engineering*, 2013. (PROMISE '13), p. 6:1–6:10. ISBN 978-1-4503-2016-0.

HOLTE, RobertC. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, Kluwer Academic Publishers-Plenum Publishers, v. 11, n. 1, p. 63–90, 1993. ISSN 0885-6125.

JAIN, Anil K.; DUBES, Richard C. *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988. ISBN 0-13-022278-X.

JAIN, A K; MURTY, M N; FLYNN, P. J. *Data Clustering: A Review*. 1999.

JELIHOVSCHI, Enio G; FARIA, José Cláudio; ALLAMAN, Ivan Bezerra. Scottknott: a package for performing the scott-knott clustering algorithm in r. *TEMA (São Carlos)*, SciELO Brasil, v. 15, n. 1, p. 3–17, 2014.

JURECZKO, Marian; MADEYSKI, Lech. Towards identifying software project clusters with regard to defect prediction. In: *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, 2010. (PROMISE '10), p. 9:1–9:10. ISBN 978-1-4503-0404-7.

KAINULAINEN, Jukka; KAINULAINEN, Jan Jukka. *Clustering Algorithms: Basics and Visualization*. 2002.

KHOSHGOFTAAR, Taghi M.; GAO, Kehan. Feature selection with imbalanced data for software defect prediction. In: *Proceedings of the 2009 International Conference on Machine Learning and Applications*, 2009. (ICMLA '09), p. 235–240. ISBN 978-0-7695-3926-3.

KHOSHGOFTAAR, TAGHI M.; GAO, KEHAN; NAPOLITANO, AMRI. An empirical study of feature ranking techniques for software quality prediction. *International Journal of Software Engineering and Knowledge Engineering*, v. 22, n. 02, p. 161–183, 2012.

KOHAVI, Ron. The power of decision tables. Springer-Verlag, London, UK, UK, p. 174–189, 1995.

KRUSKAL, William H.; WALLIS, W. Allen. Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association*, American Statistical Association, v. 47, n. 260, p. 583–621, 1952. ISSN 01621459.

LESSMANN, S.; BAESENS, B.; MUES, C.; PIETSCH, S. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *Software Engineering, IEEE Transactions on*, v. 34, n. 4, p. 485–496, July 2008. ISSN 0098-5589.

LIU, H.; SETIONO, R. Chi2: feature selection and discretization of numeric attributes. *In: Tools with Artificial Intelligence, 1995. Proceedings., Seventh International Conference on*, p. 388–391, 1995. ISSN 1082-3409.

MALHOTRA, Ruchika. A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing*, v. 27, n. 0, p. 504 – 518, 2015. ISSN 1568-4946.

MENZIES, Tim; BUTCHER, Andrew; MARCUS, Andrian; ZIMMERMANN, Thomas; COK, David. Local vs. global models for effort estimation and defect prediction. In: *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, 2011. (ASE '11), p. 343–351. ISBN 978-1-4577-1638-6.

MENZIES, T.; GREENWALD, J.; FRANK, A. Data mining static code attributes to learn defect predictors. *Software Engineering, IEEE Transactions on*, v. 33, n. 1, p. 2–13, Jan 2007. ISSN 0098-5589.

MOSER, Raimund; PEDRYCZ, Witold; SUCCI, Giancarlo. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In: *Proceedings of the 30th International Conference on Software Engineering*, 2008. (ICSE '08), p. 181–190. ISBN 978-1-60558-079-1.

MUTHUKUMARAN, K.; RALLAPALLI, Akhila; MURTHY, N. L. Bhanu. Impact of feature selection techniques on bug prediction models. In: *Proceedings of the 8th India Software Engineering Conference*, 2015. (ISEC '15), p. 120–129. ISBN 978-1-4503-3432-7.

NAGAPPAN, Nachiappan; BALL, Thomas; ZELLER, Andreas. Mining metrics to predict component failures. ACM, New York, NY, USA, p. 452–461, 2006.

NOVAKOVIC, Jasmina; STRBAC, Perica; BULATOVIC, Dusan. Toward optimal feature selection using ranking methods and classification algorithms. *Yugoslav Journal of Operations Research*, v. 21, n. 1, p. 119–135, 2011. ISSN 0354-0243.

OSTRAND, Thomas J.; WEYUKER, Elaine J. How to measure success of fault prediction models. In: *Fourth International Workshop on Software Quality Assurance: In Conjunction with the 6th ESEC/FSE Joint Meeting*, 2007. (SOQUA '07), p. 25–30. ISBN 978-1-59593-724-7.

OSTRAND, Thomas J; WEYUKER, Elaine J; BELL, Robert M. Predicting the Location and Number of Faults in Large Software Systems. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, v. 31, n. 4, p. 340–355, 2005.

PETERS, Fayola; MENZIES, Tim; MARCUS, Andrian. Better cross company defect prediction. In: *Proceedings of the 10th Working Conference on Mining Software Repositories*, 2013. (MSR '13), p. 409–418. ISBN 978-1-4673-2936-1.

PITT, Ellen; NAYAK, Richi. The use of various data mining and feature selection methods in the analysis of a population survey dataset. In: *Proceedings of the 2Nd International Workshop on Integrating Artificial Intelligence and Data Mining - Volume 84*, 2007. (AIDM '07), p. 83–93.

RAHMAN, Foyzur; POSNETT, Daryl; DEVANBU, Premkumar. Recalling the "imprecision"of cross-project defect prediction. In: *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, 2012. (FSE '12), p. 61:1–61:11. ISBN 978-1-4503-1614-9.

REFAEILZADEH, Payam; TANG, Lei; LIU, Huan. Cross-validation. Springer US, p. 532–538, 2009.

RUSSELL, Stuart Jonathan; NORVIG, Peter. *Artificial intelligence: a modern approach (3rd edition)*. [S.l.]: Prentice Hall, 2009.

SATIN, Ricardo Francisdo de Pierre; WIESE, Igor Scaliante; RÉ, Reginaldo. Um estudo exploratório sobre a predição cruzada de defeitos entre projetos: impacto do uso de diferentes algoritmos de classificação e uma medida de desempenho na construção de modelos de predição. In: *Proceedings of XLI Conferência Latinoamericana de Informática*, 2015. (CLEI 2015). To appear.

SATIN, Ricardo F. P.; WIESE, Igor Scaliante; Ré, Reginaldo. An exploratory study about cross-project defect prediction: impact of using different classification algorithms and a measure of performance in building predictive models. In: CANCELA, Hector; CUADROS-VARGAS, Alex; CUADROS-VARGAS, Ernesto (Ed.). *2015 XLI Latin American Computing Conference (CLEI)*, 2015. p. 683–694. ISBN 978-1-4673-9143-6. Disponível em: `http://eventos.spc.org.pe/clei2015/pdfs/144463.pdf`.

SHESKIN, David J. *Handbook of Parametric and Nonparametric Statistical Procedures*. 4. ed. Chapman & Hall/CRC, 2007. ISBN 1584888148, 9781584888147. Disponível em: `http://library.mpib-berlin.mpg.de/toc/z2007_770.pdf`.

SHIVAJI, S.; WHITEHEAD, E. James; AKELLA, R.; KIM, Sunghun. Reducing features to improve code change-based bug prediction. *IEEE Transactions on Software Engineering*, IEEE Computer Society, Los Alamitos, CA, USA, v. 39, n. 4, p. 552–569, 2013. ISSN 0098-5589.

THEODORIDIS, Sergios; KOUTROUMBAS, Konstantinos. *Pattern Recognition, Fourth Edition*. 4th. ed. [S.l.]: Academic Press, 2008. ISBN 1597492728, 9781597492720.

TURHAN, Burak; MENZIES, Tim; BENER, Ayşe B.; STEFANO, Justin Di. On the relative value of cross-company and within-company data for defect prediction. *Empirical Softw. Engg.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 14, n. 5, p. 540–578, out. 2009. ISSN 1382-3256.

WATANABE, Shinya; KAIYA, Haruhiko; KAIJIRI, Kenji. Adapting a fault prediction model to allow inter languagereuse. In: *Proceedings of the 4th International Workshop on Predictor Models in Software Engineering*, 2008. (PROMISE '08), p. 19–24. ISBN 978-1-60558-036-4.

WEYUKER, Elaine J; OSTRAND, Thomas J; BELL, Robert M. Adapting a fault prediction model to allow widespread usage. In: *Proceedings of the Second International Promise Workshop*, 2006.

WITTEN, Ian H.; FRANK, Eibe; HALL, Mark A. *Data Mining: Practical Machine Learning Tools and Techniques*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN 0123748569, 9780123748560.

ZHANG, Feng; MOCKUS, Audris; KEIVANLOO, Iman; ZOU, Ying. Towards building a universal defect prediction model. ACM, New York, NY, USA, p. 182–191, 2014.

ZIMMERMANN, Thomas; NAGAPPAN, Nachiappan; GALL, Harald; GIGER, Emanuel; MURPHY, Brendan. Cross-project defect prediction: A large scale experiment on data vs. domain vs. process. ACM, New York, NY, USA, p. 91–100, 2009.