UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

RICARDO FRANCISCO DE PIERRE SATIN

UM ESTUDO EXPLORATÓRIO SOBRE O USO DE DIFERENTES ALGORITMOS DE CLASSIFICAÇÃO, DE SELEÇÃO DE MÉTRICAS, E DE AGRUPAMENTO NA CONSTRUÇÃO DE MODELOS DE PREDIÇÃO CRUZADA DE DEFEITOS ENTRE PROJETOS

DISSERTAÇÃO – MESTRADO

CORNÉLIO PROCÓPIO

2016

RICARDO FRANCISCO DE PIERRE SATIN

UM ESTUDO EXPLORATÓRIO SOBRE O USO DE DIFERENTES ALGORITMOS DE CLASSIFICAÇÃO, DE SELEÇÃO DE MÉTRICAS, E DE AGRUPAMENTO NA CONSTRUÇÃO DE MODELOS DE PREDIÇÃO CRUZADA DE DEFEITOS ENTRE PROJETOS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Tecnológica Federal do Paraná – UTFPR como requisito parcial para a obtenção do título de "Mestre em Informática".

Orientador: Prof. Dr. Reginaldo Ré

CORNÉLIO PROCÓPIO

2016

Dados Internacionais de Catalogação na Publicação

S253 Satin, Ricardo Francisco de Pierre
Um estudo exploratório sobre o uso de diferentes algoritmos de classificação, de seleção de métricas, e de agrupamento na construção de modelos de predição cruzada de defeitos entre projetos / Ricardo Francisco de Pierre Satin. – 2015.
101 f. : il. ; 30 cm

Orientador: Reginaldo Ré.
Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Informática. Cornélio Procópio, 2015.
Bibliografia: p. 95-101.

1. Teoria da previsão. 2. Software – Desenvolvimento. 3. Falhas de sistemas de computação. 4. Informática – Dissertações. I. Ré, Reginaldo, orient. II. Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Informática. III. Título.

CDD (22. ed.) 004

Biblioteca da UTFPR, Câmpus Cornélio Procópio

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Programa de Pós-Graduação em Informática

Título da Dissertação Nº 06:

"Um Estudo Exploratório Sobre o Uso de Diferentes Algoritmos de Classificação, de Seleção de Métricas, e de Agrupamento na Construção de Modelos de Predição Cruzada de Defeitos Entre Projetos".

por

Ricardo Francisco de Pierre Satin

Orientador: **Prof. Dr. Reginaldo Ré**

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM INFORMÁTICA – Área de Concentração: Computação Aplicada, pelo Programa de Pós-Graduação em Informática – PPGI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Cornélio Procópio, às 09h30 do dia 18 de agosto de 2015. O trabalho foi APROVADO pela Banca Examinadora, composta pelos professores:

_____
Prof. Dr. Reginaldo Ré
(Presidente)

_____
Prof. Dr. José Augusto Fabri
(UTFPR-CP)

_____
Prof. Dr. Rogério Eduardo Garcia
(UNESP-SP)

Visto da coordenação:

_____
Carlos Nascimento Silla Junior
Coordenador do Programa de Pós-Graduação em Informática
UTFPR Câmpus Cornélio Procópio

A Folha de Aprovação assinada encontra-se na Coordenação do Programa.

Dedico a Deus, Márcia e Jonas.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus, que me fortaleceu, capacitou e conduziu fielmente até aqui.

Agradeço minha amada esposa Márcia, que soube entender meus momentos de ausência, como companheira que é sempre me incentivou.

Agradeço ao meu pequeno e amado filho Jonas, que com seu exemplo de luta e determinação, foi minha grande fonte de inspiração.

Agradeço ao meu orientador Prof. Dr. Reginaldo Ré, por todos os ensinamentos, acolhida, paciência, companheirismo e parceria em toda jornada. Espero poder continuar contando com sua preciosa amizade.

Agradeço ao prof. Dr. Igor Wiese, pelas importantes contribuições sobre algoritmos de classificação.

Agradeço ao prof. Dr. Diego Bertolini Gonçalves, pela importante contribuição sobre algoritmos de agrupamento.

Agradeço ao prof. Dr. Carlos Nascimento Silla Junior, pelo incentivo, compreensão e por todo cuidado percebido durante a condução do programa.

Agradeço todos os professores do PPGI, pelo profissionalismo na condução do programa, especialmente aos professores: Luciano T. E. Pansanato, José A. Fabri, Elias C. Genvigir, Alexandre L´Erario, Fernando Barreto e Wanderley F. da Rosa.

Agradeço ao amigo Samuel de Paula, por suas importantes contribuições nas implementações com as bibliotecas do Weka.

Agradeço aos meus grandes amigos Wellington Carvalho (Ton), Devair Canedo (Jr) e Everton Gomede, pois sem vocês tudo teria sido muito mais difícil.

Agradeço a minha família, em especial ao meu pai Vivaldo (*in memorian*), pelos princípios que me ensinou e minha amada mãe Iracema, pela acolhida e apoio nos momentos mais difíceis.

Por fim, agradeço a UTFPR-CP pelo excelente programa e pela oportunidade.

Eu lhe darei tendões e músculos e os cobrirei de pele.
Porei respiração dentro de vocês e os farei viver de novo.
Aí vocês ficarão sabendo que eu sou o Senhor.
Ezequiel 37:6.

RESUMO

SATIN, Ricardo F. P.. UM ESTUDO EXPLORATÓRIO SOBRE O USO DE DIFERENTES ALGORITMOS DE CLASSIFICAÇÃO, DE SELEÇÃO DE MÉTRICAS, E DE AGRUPAMENTO NA CONSTRUÇÃO DE MODELOS DE PREDIÇÃO CRUZADA DE DEFEITOS ENTRE PROJETOS. 102 f. Dissertação – Mestrado – Programa de Pós-graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016.

Predizer defeitos em projetos de software é uma tarefa complexa, especialmente para aqueles projetos que estão em fases iniciais do desenvolvimento por, frequentemente, disponibilizarem de poucos dados para que modelos de predição sejam criados. A utilização da predição cruzada de defeitos entre projetos é indicada em tal situação, pois permite reaproveitar dados de projetos similares. Este trabalho propõe um estudo exploratório sobre o uso de diferentes algoritmos de classificação, seleção de métricas, e de agrupamento na construção de um modelo de predição cruzada de defeitos entre projetos. Esse modelo foi construído com o uso de uma medida de desempenho, obtida com a aplicação de algoritmos de classificação, como forma de encontrar e agrupar projetos semelhantes. Para tanto, foi estudada a aplicação conjunta de 8 algoritmos de classificação, 6 de seleção de atributos, e um de agrupamento em um conjunto de dados com 1283 projetos, resultando na construção de 61 584 diferentes modelos de predição. Os algoritmos de classificação e de seleção de atributos tiveram seus desempenhos avaliados por meio de diferentes testes estatísticos que mostraram que: o *Naive Bayes* foi o classificador de melhor desempenho, em comparação com os outros 7 algoritmos; o par de algoritmos de seleção de atributos que apresentou melhor desempenho foi o formado pelo avaliador de atributos CFS e método de busca *Genetic Search*, em comparação com outros 6 pares. Considerando o algoritmo de agrupamento, a presente proposta parece ser promissora, uma vez que os resultados obtidos mostram evidências de que as predições usando agrupamento foram melhores que as predições realizadas sem qualquer agrupamento por similaridade, além de mostrar a diminuição do custo de treino e teste durante o processo de predição.

Palavras-chave: Predição de Defeitos, Modelos de Predição de Defeitos, Modelos de Predição Cruzada de Defeitos.

ABSTRACT

SATIN, Ricardo F. P.. AN EXPLORATORY STUDY ON THE USE OF DIFFERENT CLASSIFICATION ALGORITHMS, OF SELECTION METRICS, AND GROUPING TO BUILD CROSS-PROJECT DEFECT PREDICTION MODELS. 102 f. Dissertação – Mestrado – Programa de Pós-graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016.

To predict defects in software projects is a complex task, especially for those projects that are in early stages of development by, often, providing few data for prediction models. The use of cross-project defect prediction is indicated in such a situation because it allows reuse data of similar projects. This work proposes an exploratory study on the use of different classification algorithms, of selection metrics, and grouping to build cross-project defect predictions models. This model was built using a performance measure, obtained by applying classification algorithms aim to find and group similar projects. Therefore, it was studied the application of 8 classification algorithms, 6 feature selection, and a cluster in a data set with 1283 projects, resulting in the construction of 61 584 different prediction models. The classification algorithms and feature selection had their performance evaluated through different statistical tests showed that: the *Naive Bayes* was the best performance classifier, as compared with other 7 algorithms; the pair of feature selection algorithms that performed better was formed by CFS attribute evaluator and search method *Genetic Search*, compared with 6 other pairs. Considering the clustering algorithm, this proposal seems to be promising, since the results shows evidence that the predictions were best grouping using the predictions performed without any similarity clustering, and shows the decrease in training cost and testing during the prediction process.

Keywords: Fault Prediction, Defect Prediction Models, Cross-Project Defect Prediction.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO E MOTIVAÇÃO

O processo de predição de defeitos busca predizer a existência de defeitos antes que eles efetivamente se manifestem (DÁMBROS et al., 2010; HALL et al., 2012). O uso de seus resultados permite direcionar os esforços das equipes prioritariamente em diferentes tarefas, para, por exemplo: minimizar o tempo gasto com a correção de defeitos; reduzir os custos de manutenção; e, incrementar a percepção de qualidade final do produto (CATAL; DIRI, 2009; HALL et al., 2012).

Contudo, predizer defeitos que ocorrerão em um projeto de software alvo é uma tarefa complexa, especialmente nas fases iniciais do desenvolvimento (ZIMMERMANN et al., 2009). Nessas fases, geralmente, tais projetos alvo não têm dados históricos suficientes para que sejam construídos modelos próprios de predição com o uso, por exemplo, de dados de versões anteriores. Para resolver esse problema a predição cruzada de defeitos entre projetos (do inglês, *cross-project defect prediction*) pode ser utilizada, de maneira que projetos sem dados históricos suficientes usem dados de projetos semelhantes, e assim os modelos de predição possam ser construídos, treinados e testados (TURHAN et al., 2009; MENZIES et al., 2011; HALL et al., 2012). Entretanto, a utilização da predição cruzada entre projetos apresenta alguns desafios.

Um dos maiores desafios da predição cruzada de defeitos é encontrar projetos que tenham algum tipo de similaridade com o projeto alvo, de forma que os dados possam ser compartilhados. O uso de algoritmos de agrupamento permite encontrar padrões de similaridade entre projetos. Isso pode ser feito por meio de métricas disponíveis, que mensuram características do projeto alvo, que permitam o estabelecimento de uma relação com outros projetos, cuja base de dados históricos é maior e mais consolidada. Essa é uma das técnicas mais utilizadas em pesquisas anteriores que têm como objetivo a predição de defeitos entre projetos (ZIMMERMANN et al., 2009; JURECZKO; MADEYSKI, 2010; MENZIES et al., 2011; ZHANG et al., 2014), que acabam utilizando

implementações próprias de algoritmos de agrupamento ou implementações baseadas em algoritmos como o K-means (ALPAYDIN, 2010). As pesquisas que utilizam as métricas como medida de similaridade tentam criar modelos de predição cruzada genéricos, que podem ser usados para predizer defeitos em qualquer projeto alvo, desde que as técnicas empregadas pelos modelos encontrem semelhanças suficientes entre os projetos usados na construção do modelo e o projeto alvo (ZIMMERMANN et al., 2009; MENZIES et al., 2011; ZHANG et al., 2014).

Os algoritmos usados na criação dos modelos de predição também é um ponto que merece atenção e tem impacto na criação de modelos de predição cruzada. Geralmente, os modelos de predição de defeito são criados com o uso de classificadores e estudos sobre o real impacto desses algoritmos nos resultados dos modelos de predição são controversos (LESSMANN et al., 2008; GHOTRA et al., 2015). A controvérsia também atinge algoritmos, usados em conjunto com os classificadores para melhorar o poder de predição ou de custo de execução do modelo, especializados em selecionar um subconjunto de métricas mais adequado para a predição de defeitos (KHOSHGOFTAAR et al., 2012; GAO; KHOSHGOFTAAR, 2015; MALHOTRA, 2015).

Assim, de forma geral, o desempenho dos modelos de predição cruzada entre projetos é um ponto que apresenta lacunas a explorar. Alguns estudos mostram que a utilização de dados de diferentes projetos para a construção dos modelos de predição geram altos índices de taxa de falso positivo (TURHAN et al., 2009), enquanto outros estudos apresentam baixos índices de taxa de sucesso com a utilização de predição cruzada (ZIMMERMANN et al., 2009; HE et al., 2012). Indo mais além, uma terceira parte dos estudos simplesmente não conseguem mostrar evidências de quando a utilização de predição cruzada é recomendada (KITCHENHAM et al., 2007). Por fim, existem estudos que mostram evidências de sua utilização e que apresentam resultados positivos, tanto do ponto de vista de desempenho como de custo de execução (JURECZKO; MADEYSKI, 2010; ZHANG et al., 2014; SATIN et al., 2015).

1.2 OBJETIVOS

Os modelos de predição cruzada de defeitos entre projetos encontradas na literatura usam métricas dos projetos para encontrar semelhanças. Ao contrário de tais modelos, este trabalho usa uma forma alternativa de construção de modelos para a predição cruzada baseada em agrupamentos de projetos por similaridade. Ao invés de usar métricas dos projetos para encontrar similaridade, este trabalho utiliza uma medida

de desempenho de algoritmos de classificação denominada AUC (do inglês, *Area Under a Curve*), juntamente com um algoritmo de agrupamento por similaridade chamado BSAS (do inglês, *Basic Sequencial Algorithmic Scheme*), para construir modelos de predição cruzada de defeitos. No entanto, considerando a controvérsia a respeito do impacto do uso de diferentes algoritmos de classificação e de seleção de métricas, este trabalho também apresenta um estudo exploratório do desempenho de diferentes algoritmos de classificação e de seleção de métricas na construção de modelos de predição.

Portanto, o objetivo geral deste trabalho é realizar um estudo sobre o uso de diferentes algoritmos de classificação, de seleção de métricas e de agrupamento na construção de modelos de predição cruzada de defeitos. Para tanto, foi norteado pelos seguintes objetivos específicos:

- A condução de um estudo exploratório do uso de diferentes algoritmos selecionadores de métricas;
- A condução de um estudo exploratório do uso de diferentes algoritmos de classificação; e,
- A condução de um estudo exploratório do uso do algoritmo de agrupamento BSAS na construção de modelos de predição cruzada de defeitos.

1.3 ORGANIZAÇÃO DO TRABALHO

O restante deste trabalho está estruturado da seguinte forma: no Capítulo 2, é apresentado o referencial teórico que fundamenta a criação, execução e análise dos resultados obtidos com o presente trabalho. Também estão referenciados os trabalhos mais significativos, relacionados ao tema de pesquisa, com suas respectivas análises. No Capítulo 3, são apresentados os passos de construção dos modelos de predição cruzada de defeitos apoiada por similaridade entre projetos e os resultados iniciais encontrados com a execução dos passos de construção dos modelos. Já no Capítulo 4, são descritos os resultados finais e comparações com os resultados encontrados por trabalhos semelhantes. Por fim, as conclusões finais, as limitações da presente proposta e as sugestões para evolução desta pesquisa em trabalhos futuros são descritas no Capítulo 5.

2 CONCEITUALIZAÇÃO E FUNDAMENTAÇÃO TEÓRICA

2.1 CONSIDERAÇÕES INICIAIS

Esta seção aborda conceitos importantes para o entendimento do trabalho proposto sob três perspectivas: conceitos básicos de aprendizagem de máquina, descritos na Seção 2.2; detalhes da predição de defeitos em projetos de software, mostrados na Seção 2.3; e, trabalhos de predição cruzada de defeitos entre projetos, brevemente discutidos na Seção 2.4. Os conceitos básicos de aprendizado de máquina são abordados para que haja um entendimento individual dos elementos, necessários ou eletivos, usados na criação dos modelos de predição de maneira geral. Por sua vez, os detalhes da predição de defeitos são apresentados especificamente para que o processo de predição no âmbito dos modelos de predição de defeitos sejam conhecidos. Por fim, alguns trabalhos que representam os modelos de predição cruzada de defeitos são brevemente discutidos.

2.2 CONCEITOS BÁSICOS DE APRENDIZADO DE MÁQUINA

2.2.1 CONJUNTOS DE DADOS

O processo de desenvolvimento de um novo software gera um grande volume de dados. Esses dados gerados têm origem no próprio software que está sendo desenvolvido, no processo que está sendo aplicado, nas informações sobre os membros do projeto – inclusive os desenvolvedores – e no contexto no qual o software está sendo desenvolvido ou será utilizado. Todos esses dados podem ser consolidados a fim de que informações relevantes sejam extraídas. Tais informações podem ser utilizadas para melhora da qualidade dos produtos, para prevenção da ocorrência de defeitos, para melhora no direcionamento da equipe de desenvolvimento e controle da qualidade (FACELI et al., 2011).

Um conjunto de dados (do inglês, *dataset*) nada mais é do que um agrupamento

de objetos, também chamado de instâncias, que representa um elemento do mundo real (FACELI et al., 2011; WITTEN et al., 2011). Um objeto pode ser, por exemplo, um computador utilizado na atividade de programação pelo desenvolvedor; pode ser o documento de requisito utilizado para descrever as funcionalidades que necessitam ser desenvolvidas; também pode ser o meio de comunicação utilizado pelos desenvolvedores para tirarem dúvidas ou falarem sobre o projeto; e, ainda, pode representar o próprio desenvolvedor ou o código produzido por ele em sua atividade de programação. Cada um dos exemplos citados caracteriza um objeto distinto dentro de um conjunto de dados. Dessa forma é possível identificar que os objetos podem ser caracterizados de maneiras distintas.

A caracterização de objetos se dá por meio do conjunto de atributos de entrada que representa cada objeto, também chamado de vetor de características (JAIN et al., 1999). Esses atributos representam características do objeto pertencente ao conjunto de dados, por exemplo, para representar o objeto relacionado ao código fonte de um programa: o número de linhas de códigos; a complexidade dos métodos; ou, a linguagem de programação utilizada. Dentre os atributos do conjunto de dados, pode existir um atributo especial, chamado de atributo alvo (também chamado de atributo meta, atributo de saída ou classe). Esse atributo armazena a informação que se pretende analisar para aquele objeto naquele conjunto de dados. Contudo, sua existência no conjunto de dados não é obrigatória.

O atributo alvo pode possuir conteúdo com valores numéricos ou rotulados, sendo que o domínio do conteúdo determina o tipo de problema que se pretende resolver (ALPAYDIN, 2010). Quando os valores são numéricos, trata-se de um problema de regressão, já quando os valores do atributo são rotulados (categorizados ou classificados), trata-se de um problema de classificação. Muito embora o atributo alvo seja importante para resolução do problema existente com o objeto, os valores assumidos pelos outros atributos são também relevantes. Tais atributos podem variar nos aspectos de tipo e escala.

O tipo do atributo pode ser qualitativo ou quantitativo (WITTEN et al., 2011). Um atributo assume valores do tipo qualitativo quando ele exprime qualidade, tendo valores associados a categorias. Exemplos de valores qualitativos são atributos que podem ter valores como: {*alto, baixo*} ou {*baixo, moderado, alto*}. Já um atributo que apresenta valores do tipo quantitativo armazena valores numéricos. Esse tipo de valor, quantitativo ou numérico, ainda pode ser considerado de domínio contínuo ou discreto.

Um atributo de domínio contínuo pode assumir um número numa faixa infinita de valores, por exemplo, um número de ponto flutuante que representa a distância entre duas cidades, ou representa o peso de um dado objeto a ser transportado. Já um atributo do tipo discreto é um atributo que pode assumir valores limitados, como idade.

É importante lembrar que, além do tipo, a escala é outra informação importante a ser considerada na análise dos valores dos atributos e está intimamente relacionada ao tipo de atributo. A escala de valor que o tipo de um atributo pode assumir define as operações que podem ser realizadas com ele (FACELI et al., 2011). Atributos do tipo qualitativo podem assumir escalas classificadas como nominais ou ordinais. A escala nominal trata de valores com nomes diferentes, por exemplo, o tipo de veículo, com as categorias "Popular", "*Premium*" ou "Luxo". As operações mais utilizadas para manipular esses valores são de igualdade ou desigualdade. Já a escala ordinal permite, além das operações mencionadas para a escala nominal, a utilização de operadores como: $>$, $<$, $\geq$, $\leq$. Isso pode ser feito uma vez que as categorias representadas possuem uma relação de ordem de grandeza, como por exemplo, o resultado da avaliação de provas de um aluno: "A","B", "C", "D" ou "E".

De maneira análoga aos atributos do tipo qualitativo, existem as operações aplicáveis aos atributos do tipo quantitativo. Um atributo desse tipo apresenta valores que assumem escalas intervalares ou racionais (FACELI et al., 2011; WITTEN et al., 2011). Os atributos com escalas intervalares são aqueles que apresentam números que variam dentro de um intervalo. Os atributos com escalas racionais são aqueles que têm um significado absoluto, existindo um ponto de comparação absoluto que permite uma checagem precisa sobre o valor apresentado (FACELI et al., 2011). Exemplos de valores que apresentam um parâmetro exato de comparação que, em todos os casos, é nenhuma vez, são: o número de vezes que um aluno reprovou em uma disciplina, ou o número de vezes que o proprietário acionou o seguro de seu veículo.

2.2.2 ALGORITMOS DE APRENDIZADO DE MÁQUINA

Os algoritmos de aprendizado de máquina são utilizados para solucionar vários tipos de problemas (RUSSELL; NORVIG, 2009; ALPAYDIN, 2010; FACELI et al., 2011; GHOTRA et al., 2015). Dentre os problemas, estão aqueles ligados: à detecção de fraude; ao reconhecimento facial; ao reconhecimento de fala; ao reconhecimento de sentimento; aos diagnósticos clínicos; e também, à predição de defeitos em projetos de software. Os algoritmos de aprendizado de máquina são geralmente agrupados de três

formas: conforme o tipo de problema que se pretende resolver; conforme o estilo de aprendizado dos algoritmos; e, conforme a abordagem dos algoritmos para solucionar os problemas.

Com relação ao tipo de problema que se pretende resolver os algoritmos podem ser agrupados como aqueles que resolvem problemas com: classificação; regressão; agrupamento (do inglês, *clustering*); ou, extração de regras (ALPAYDIN, 2010; FACELI et al., 2011). O tipo do problema a ser tratado está diretamente ligado ao valor do atributo alvo pertencente ao conjunto de dados usado pelo algoritmo. Caso o atributo alvo seja nominal, se trata de um problema de classificação (também chamado de categorização ou aprendizado de conceitos), por exemplo: classificar um paciente como portador de uma doença ou livre dela, classificar um código fonte como detentor de defeito ou livre de dele. Nesse caso, o algoritmo de aprendizado de máquina é chamado de "classificador". Contudo, se o valor do atributo alvo é numérico, o problema a ser solucionado é um problema de regressão. A função do algoritmo é tentar prever um valor numérico, por exemplo: cotação da bolsa de valores; variação cambial; números da loteria; valor de venda de um produto. Algoritmos de aprendizado de máquina que trabalham para solucionar esse tipo de problema são chamados de "regressores". Já problemas que exigem o agrupamento de dados conforme algum tipo de similaridade, não necessariamente com a criação de rótulos para os dados, é resolvido com agrupamento, por exemplo: agrupar clientes com potencial de compra de um veículo, agrupar arquivos de código fonte com potencialidade de defeitos. Para outros casos, em que se procura estabelecer uma relação entre elementos, os algoritmos de aprendizado de máquina trabalham com extração de regras, como por exemplo: verificar se clientes que compram certo produto também compra outro determinado produto; ou, determinar se desenvolvedores que cometem um tipo de defeito também geram outro tipo de defeito.

Quanto ao seu estilo de aprendizagem e à forma como são empregados, os algoritmos podem ser divididos em (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 trabalham com um conjunto de dados de treinamento para aprender sobre um determinado problema. Esse conjunto de dados contém obrigatoriamente o atributo alvo. Após "aprender" sobre o problema com os dados do conjunto de dados com o atributo alvo, seu conhecimento é aplicado em outro conjunto de dados, geralmente chamado de conjunto de teste, para que as predições sejam realizadas. É

um estilo de aprendizado bastante utilizado para resolver problemas de classificação ou regressão.

Por sua vez, algoritmos de aprendizado não supervisionado usam conjuntos de dados que não apresentam atributos rotulados. Ao invés disso, os valores presentes no conjunto de dados são utilizados para que deduções sobre eles sejam feitas e uma forma de representação do conhecimento sobre o problema que está sendo investigado seja criada. É um estilo de aprendizado bastante utilizado para solucionar problemas de agrupamento com extração de regras.

Com características dos dois outros estilos de aprendizagem, algoritmos de aprendizado semi-supervisionado trabalham com um conjunto de dados que apresentam atributos parte rotulado e parte não rotulado. Apesar de existir um problema de predição, o algoritmo deve aprender com os dados de entrada para prever corretamente a saída. Assim como os algoritmos de aprendizado supervisionado, é um estilo de aprendizado bastante utilizado para solucionar problemas de classificação e regressão.

Por fim, algoritmos de aprendizado por reforço trabalham com um conjunto de dados que são fornecidos como estímulos e os algoritmos devem responder e reagir a eles. O resultado de seu processamento é apresentado como um conjunto de punições ou recompensas. É um estilo de aprendizado bastante utilizado para solucionar problemas de robótica.

Além dessas duas formas de organização dos algoritmos de aprendizado de máquina, há outra forma baseada na técnica que eles utilizam para solucionar os problemas, sendo as mais comuns: estatística; baseada em árvores de decisão; baseada em regras; redes neurais artificiais; máquina de vetores de suporte; distância entre os dados – vizinhos mais próximos – (do inglês, *nearest neighbor*); agrupamento; e, algoritmos utilizados em conjunto (do inglês, *ensemble*) (ALPAYDIN, 2010; FACELI et al., 2011; GHOTRA et al., 2015).

Como mencionado, uma das técnicas utilizada por algoritmos de aprendizado de máquina para solucionar problemas é o “método estatístico”. Para a criação de um modelo preditivo, as técnicas estatísticas se apoiam em um modelo de probabilidade para encontrar padrões entre os valores dos atributos presentes no conjunto de dados (KOTSIANTIS, 2007). Dentre os métodos probabilísticos mais utilizados está o Bayesiano (JOHN; LANGLEY, 1995). Esse método tem a característica de assumir que todos os atributos do conjunto de dados são independentes entre si e lida bem com dados imprecisos ou incompletos. Uma representação do método Bayesiano, em que $P(A|B)$ é a

probabilidade de uma determinada classe ocorrer baseado na observação dos atributos, é apresentada na Equação 1, no qual:

- $P(B|A)$ é a probabilidade de observar vários atributos com relação à classe;
- $P(A)$ é a probabilidade *a priori*[1] da classe; e,
- $P(B)$ é a probabilidade de ocorrência de cada atributo de forma individual.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

Esse método apresenta algumas vantagens e desvantagens (JOHN; LANGLEY, 1995). Como vantagem, tem-se a sua velocidade de treino e execução, sua não sensibilidade a dados incompletos ou imprecisos e a capacidade de trabalhar com dados reais e discretos. Sua desvantagem está em assumir a independência entre os atributos, em que não se permite estabelecer a correlação entre os atributos e a influência da combinação dos atributos na identificação do atributo alvo.

Outra técnica utilizada por algoritmos de aprendizado de máquina para solucionar problemas são as "árvores de decisão". Assim como as técnicas estatísticas, essa técnica pode ser utilizada para resolver problemas de classificação ou regressão (GHOTRA et al., 2015). Formalmente, uma árvore de decisão é um grafo acíclico, em que cada nó pode ser um nó de divisão (inclusive o nó raiz), com os desvios condicionais, ou um nó folha, representado pelo atributo alvo (FACELI et al., 2011). A estratégia da técnica de árvore de decisão é dividir o problema principal em problemas menores, de forma recursiva, até chegar à solução do problema proposto. Isso se dá com o uso dos valores dos atributos para chegar à classificação de cada objeto do conjunto de dados. Essa classificação ocorre a partir do nó raiz, passa pelos nós de divisão que apresentam testes condicionais baseados nos valores de seus próprios atributos, até chegar ao nó folha. Um exemplo de árvore de decisão pode ser visualizado na Figura 1, em que o conjunto de dados, `weather.symbolic`, disponível na ferramenta Weka[2] é utilizado para representar a tomada de decisão para a prática de uma atividade esportiva qualquer, cujas condições climáticas exercem influência sobre a execução, ou não, da atividade. Essa tomada de decisão tem como referência as condições climáticas que podem interferir no início ou não da partida.

[1] É a probabilidade atribuída à classe antes de qualquer processamento.

[2] `http://www.cs.waikato.ac.nz/ml/weka/` .

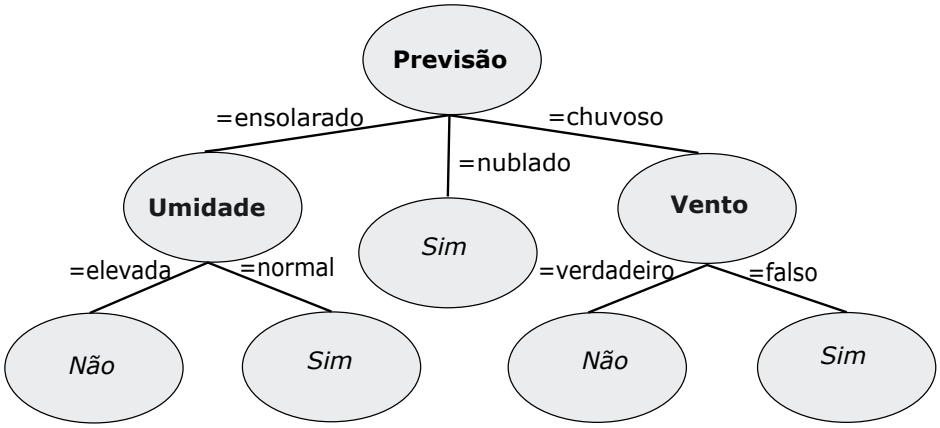

Figura 1: Uma representação de árvore de decisão.
Fonte: Elaborado pelo autor.

Seguindo, mais uma técnica utilizada é a que utiliza um "conjunto de regras" distintas que são aplicadas aos atributos como forma de chegar à resolução do problema (FACELI et al., 2011; GHOTRA et al., 2015). As regras construídas são do tipo *"se" X "então" Y*. Cada análise condicional é construída com base em um atributo do objeto do conjunto de dados nos valores permitidos de seu domínio. Os operadores utilizados nas analises condicionais são: $=$,$\neq$,$<$,$>$,$\leq$,$\geq$, $\in$ e $\notin$. Os conjuntos de regras são construídos em uma estrutura semelhante ao apresentado pelas árvores de decisão, tendo a Figura 1 como base, um exemplo seria: *Previsão=ensolarado & Umidade=normal => Jogar = Sim*. Esse exemplo representa que, em condições de tempo ensolarado e umidade normal, provavelmente a partida seria iniciada.

Outro tipo de técnica utilizada para solucionar problemas com o uso de algoritmos de aprendizado de máquina são aqueles que utilizam os princípios de "redes neurais artificiais" (do inglês, *Artificial Neural Network*) (HAYKIN, 1998). Ela tem como inspiração a estrutura e o funcionamento do sistema nervoso como forma de simular o cérebro humano na aquisição de conhecimento. Como o neurônio é a estrutura principal do sistema nervoso, o que se busca computacionalmente é uma forma de simular seu funcionamento, conforme pode ser observado na Figura 2. O funcionamento de um neurônio pode ser apresentado, de forma simplista, da seguinte forma: os neurônios detectam estímulos que são provenientes de outros neurônios ou do ambiente em que o indivíduo está inserido. Esses estímulos são captados pelos dendritos[3] que são processados pelo corpo celular[4]. O resultado desse processamento faz com que um novo

[3]Prolongamento dos neurônios especializados em recepção de estímulos.
[4]Parte integrante do neurônio responsável pela interpretação dos estímulos elétricos trazidos pelos

estímulo seja gerado e enviado para o axônio[5], para propagação a outros neurônios. A transmissão de informação entre os neurônios recebe o nome de sinapse, que é o meio como as informações, representada por impulsos elétricos, são transmitidas. A Figura 2 é uma exemplo da representação da estrutura de um neurônio conforme apresentado.

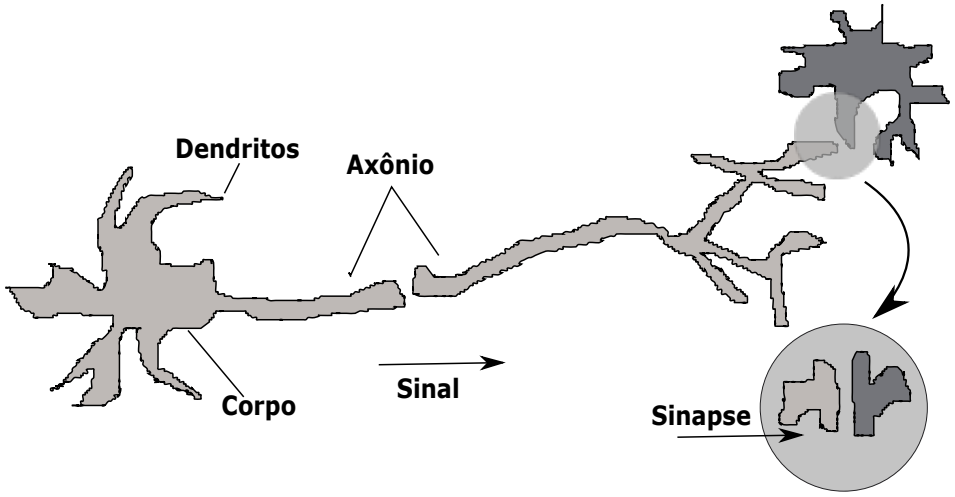

Figura 2: Uma representação de um nerônio biológico.
Fonte: Elaborado pelo autor.

A representação de um neurônio artificial e o seu funcionamento pode ser visto na Figura 3. O neurônio artificial é a base para o funcionamento das redes neurais artificiais, que procura simular o funcionamento das redes neurais biológicas. A simulação do funcionamento depende da arquitetura e do formato de aprendizado utilizado pela técnica de redes neurais artificiais (BRAGA et al., 2000). A arquitetura diz respeito à forma como a solução é estruturada, já o formato de aprendizado está relacionado à maneira como o processo do estímulo, processamento e propagação são tratados. É na arquitetura que é definida a unidade básica de processamento, que desempenha o papel análogo ao do neurônio. Essa unidade básica tem pontos de entrada que simulam o papel dos dendritos. Os valores recebidos pelas unidades de entrada recebem valores ponderados, conforme exibido na Figura 3 por meio dos dados “Peso x”, “Peso y” e “Peso z”, combina-os e realiza o processamento, semelhante ao executado pelo corpo celular e representado na Figura 3 por meio da função “f_a” . A saída do processamento é a resposta dada pela unidade de processamento, que simula a resposta dada pelo neurônio e encaminhada pelo axônio, conforme pode ser visto na Figura 3 como a troca de

dendritos.

[5]Prolongamento do neurônio responsável pela transmissão de impulsos elétricos para outros neurônios.

"Sinal" entre os dois neurônios artificiais representados. A associação dessas unidades de processamento, que simulam o funcionamento de um neurônio, dá origem às redes neurais artificiais.

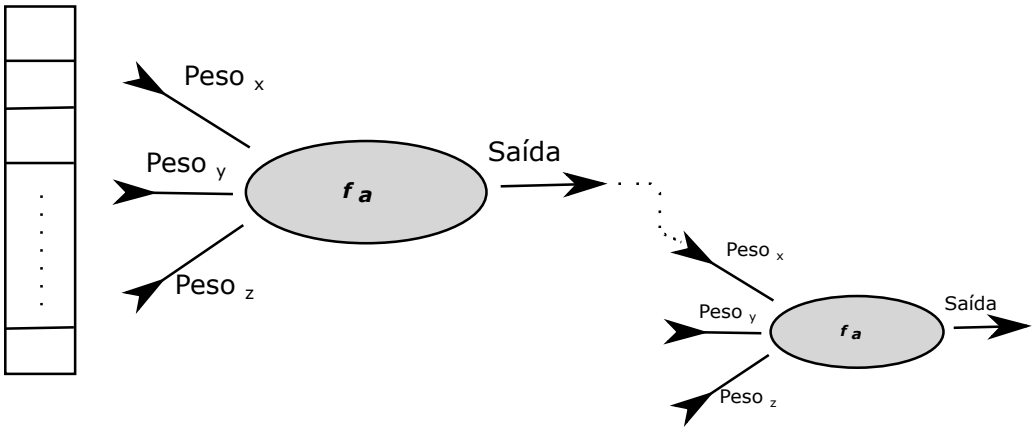

Figura 3: Uma representação de um nerônio artificial.
Fonte: Elaborado pelo autor, baseado em (FACELI et al., 2011).

A utilização de redes neurais artificiais tem vantagens e desvantagens. As dificuldades em sua utilização ainda estão ligadas à interpretação dos resultados que foram gerados, à dificuldade de parametrização e, ainda, à performance (BRAGA et al., 2000). Contudo, a facilidade de generalização, tolerância a falhas e ruídos são características importantes que tem feito com que seu uso prático seja cada vez mais considerado, especialmente em tarefas de percepção e controle, visão computacional e robótica.

Assim como todas as demais técnicas permitem o tratamento de problemas de regressão e classificação, as "máquinas de vetores de suporte" (do inglês, *Support Vector Machine*, SVM) também o fazem. Sua fundamentação está na teoria do aprendizado estatístico (VAPNIK, 1995), que tem o objetivo de inferir regras gerais a partir da observação de exemplos, com a minimização de riscos por meio da aplicação de conjuntos de treino e teste. Essa técnica tem como forma de trabalho a montagem de um hiperplano para as classes representadas pelo atributo alvo com o maior número possível de representações (CRISTIANINI; SHAWE-TAYLOR, 2000).

As vantagens dessa técnica são sua robustez e sua eficiência, especialmente em dados de grandes dimensões. Algumas de suas limitações estão ligadas à complexidade de interpretação de seus resultados, assim como a própria parametrização para uma correta execução e performance de execução (FACELI et al., 2011).

Se as técnicas de redes neurais artificiais e máquina de vetores de suporte são

duas das técnicas mais complexas para solucionar problemas de aprendizado de máquina, a técnica do vizinho mais próximo está entre as mais simples das técnicas. Essa técnica pode ser usada para solucionar problemas de classificação e regressão. A ideia por trás dela é considerar que objetos relacionados ao mesmo conceito são semelhantes entre si. Dessa forma, a abordagem da técnica é memorizar os objetos do conjunto de treino e posicionar os objetos do conjunto de teste pela semelhança entre as amostras.

Dentre as vantagens da utilização dessa abordagem estão à simplicidade do processo de treinamento e a possibilidade de sua aplicação mesmo para problemas complexos. O principal ponto negativo dessa abordagem está relacionado ao processo de teste, que pode ser custoso, pois há necessidade de cálculo da distância do objeto que está sendo testado contra todos os objetos de treino para encontrar a menor distância para agrupamento. Outro ponto negativo é a influencia de atributos irrelevantes ou duplicados (COVER; HART, 1967; AHA et al., 1991; FACELI et al., 2011).

Pode-se perceber que cada forma de solucionar um problema com aprendizado de máquina tem seus pontos positivos e negativos. Também é possível identificar que todas as soluções são aplicáveis para solucionar problemas de classificação ou regressão. Contudo, uma pergunta que surge é se existe uma forma de solucionar problemas que seja melhor e/ou aplicável em todos os tipos de problemas. Essa pergunta é respondida por Wolpert e MacReady (WOLPERT; MACREADY, 1997), que afirmam não existir um algoritmo que seja o melhor para solucionar todos os tipos de problemas. Com base nessa afirmação estudos têm sido direcionado para procurar encontrar formas de combinar diferentes algoritmos de aprendizado de máquina, sendo a “combinação de algoritmos de aprendizado de máquina” (do inglês, *ensemble*), uma abordagem para solucionar esse problema. Algoritmos com essa abordagem apresenta como característica positiva uma possibilidade de melhora da capacidade de predição por permitir o uso em combinação de diferentes algoritmos, contudo, essa combinação também é responsável por suas características negativas, como a dificuldade de interpretação dos resultados alcançados e o custo de execução da técnica, se comparada com as demais.

Por fim, a outra forma de solucionar problemas com o uso de algoritmo de aprendizado de máquina, é com o uso da técnica de agrupamentos. Por uma questão de organização, essa técnica é descrita com mais detalhes na Seção 2.2.6.

2.2.3 TÉCNICAS DE PRÉ-PROCESSAMENTO DE CONJUNTOS DE DADOS

Para minimizar problemas que afetam a qualidade do conjunto de dados, como: ruídos; imperfeições; inconsistências e dados duplicados; e, tornar o conjunto de dados com seus objetos e atributos mais adequados para o uso pelos algoritmos de aprendizado de máquina, existem técnicas de pré-processamento que podem ser aplicadas, como: eliminação manual de atributos; integração de dados; amostragem de dados; tratamento de conjuntos de dados desbalanceados; limpeza de dados; transformação de dados; e, redução da dimensionalidade. Não existe uma sequencia na aplicação e nem a garantia de qualidade nos dados após a utilização dessas técnicas, entretanto, vários problemas podem ser minimizados com seu uso, como apresentado a seguir (FACELI et al., 2011).

A primeira técnica é a “eliminação manual de atributos”. Essa técnica consiste na análise de um especialista em todos os atributos dos objetos do conjunto de dados com o objetivo de encontrar atributos repetidos ou irrelevantes. Esses atributos, uma vez encontrados, são removidos do conjunto de dados. Essa duplicidade de atributos pode ter como origem a formação de um conjunto de dados, com objetos provenientes de várias fontes, como dados de múltiplos sistemas. Para isso a técnica de “integração de dados” é útil para tratar esse tipo de situação. Nessa situação, conjuntos de dados distintos podem ter atributos com tipos de informações ou escalas diferentes. Isso dificulta a análise do conjunto de dados, além de impactar na consistência dos dados. É recomendada a unificação dos valores e mantém o mesmo tipo e escala. Muito embora o uso dessa técnica consiga reduzir o número de objetos ou atributos para análise, o problema do tempo de processamento e recursos computacionais necessários para análise de uma amostra grande de dados não é solucionado. Para atacar esse problema, a técnica de “amostragem de dados” pode ser utilizada. Seu objetivo é encontrar uma amostra que possa ser utilizada pelos algoritmos de aprendizado de máquina sem a necessidade de utilização de todo conjunto de dados. O uso de todo conjunto de dados pode comprometer recursos computacionais e tornar custoso o processo de obtenção dos resultados esperados. Para isso, uma amostra dos dados é selecionada mantendo as mesmas características que a base de dados original, contudo, com um conjunto de dados menor em comparação com a base de dados como um todo.

Contudo, independentemente da análise estar sendo feita na base de dados toda ou em uma amostra, um problema que ainda pode persistir é o desbalanceamento dos dados, para isso a técnica “tratamento de conjuntos de dados desbalanceados” pode

ser utilizada. Dizer que um conjunto de dados é desbalanceado significa dizer que o conjunto de objetos varia significativamente para as diferentes classes. Nesse caso, as classes apresentam frequências diferentes e, no caso de um atributo alvo com duas classes distintas, a classe com maior número de ocorrências é nomeada classe majoritária, já a classe com a menor frequência é nomeada de classe minoritária. Um conjunto de dados desbalanceado compromete o desempenho dos algoritmos de classificação, de maneira que, a classificação de objetos da classe majoritária seja predominantemente privilegiada. Para minimizar esse problema, algumas técnicas para tratar o desbalanceamento podem ser aplicadas ao conjunto de dados e dentre as mais utilizadas na literatura estão (FACELI et al., 2011; WITTEN et al., 2011):

Redefinição do tamanho do conjunto de dados. Essa técnica tem o objetivo de reduzir o número de objetos da classe majoritária ou acrescentar objetos a classe minoritária. Ambas as opções acabam por trazer riscos ao conjunto de dados. Na primeira situação, a remoção de informações relevantes pode impactar nos resultados corretos da execução do modelo, o que pode provocar *underfitting*[6] no modelo de predição. No segundo caso informações que não são relevantes, que nunca ocorrerão, podem ser adicionadas além da possibilidade de ocasionar *overfitting*[7] no modelo.

Atribuição de diferentes custos de classificação para as diferentes classes. Essa técnica estabelece custos diferentes de classificação para as classes majoritárias e minoritárias do conjunto de dados. Significa que, se a classe majoritária tem duas vezes mais ocorrências que a classe minoritária, poder-se-ia atribuir um custo de duas vezes o valor para a classe com menor número de ocorrências. Ocorre que o estabelecimento desta proporção não é algo tão trivial assim, além de que a atribuição desses custos aos algoritmos de classificação não é algo amplamente aceito.

Uso de um modelo para cada classe. Essa abordagem propõe o uso de apenas uma das classes de cada vez, ou a minoritária ou a majoritária, para treino do modelo.

Muito embora o problema de desbalanceamento seja um elemento que compromete a qualidade do conjunto de dados, ele não é o único ponto de atenção. Os

[6] *Underfitting* é muitas vezes resultado de um modelo muito simples, em que o modelo de predição não se encaixa com os algoritmos de aprendizado.

[7] *Overfitting* é quando o modelo de predição está super ajustado ao conjunto de dados, de forma que suas execuções em outros conjuntos de dados, de teste, não apresentarão o mesmo desempenho anterior. Geralmente ocorre em modelos muito complexos.

objetos de um conjunto de dados podem apresentar dados incompletos, inconsistentes, redundantes ou com ruído, que comprometem o desempenho dos algoritmos de classificação. Tais problemas podem ser tratados pela técnica de "limpeza de dados". Os valores dos atributos dos objetos de um conjunto de dados podem ser considerados: incompletos, quando existem ocorrências de um atributo em um objeto e não em outro; inconsistentes, quando valores de atributos relacionados apresentam valores conflitantes; redundantes, quando objetos, ou atributos, apresentam os mesmos valores muito próximos; e, com ruído, quando objetos são agrupados erroneamente em um conjunto de dados.

Outra técnica que pode ser utilizada com objetivo de melhorar a qualidade dos dados do conjunto de dados é a técnica de "transformação de dados". Ela tem o objetivo de tornar os valores dos atributos adequados para o uso pelos algoritmos de aprendizado de máquina. Por exemplo, existem algoritmos que trabalham apenas com dados numéricos, como as redes neurais artificiais e máquina de vetores de suporte. Para algoritmos com essa necessidade os atributos precisam ser transformados para esse tipo de representação, caso contrário, o funcionamento do algoritmo de aprendizado de máquina pode ser comprometido ou simplesmente não funcionar. Essa transformação dos atributos em valores numéricos, valores simbólicos ou transformações na escala de valores, são exemplos de aplicação da técnica.

Além dos dados serem limpos ou transformados, existe outro fator que exerce impacto na qualidade e custo do processo de predição: o número de atributos que compõem os objetos do conjunto de dados. Quanto mais atributos disponíveis, mais custoso é o processo a ser executado. A técnica de "redução da dimensionalidade" tem o objetivo de solucionar esse problema. De maneira geral, existem duas formas propostas para isso, uma delas é a agregação de atributos, e a outra, a seleção de atributos. A primeira procura encontrar redundâncias entre atributos com intenção de agrupá-los. A segunda, apresentada de maneira detalhada na próxima seção, procura encontrar os atributos que são mais relevantes para caracterização do objeto, com o descarte dos demais.

2.2.4 ALGORITMOS DE SELEÇÃO DE ATRIBUTOS

A técnica de seleção de atributos é importante, pois permite identificar os atributos mais relevantes para o processo de predição (GUYON; ELISSEEFF, 2003; KHOSHGOFTAAR; GAO, 2009; MUTHUKUMARAN et al., 2015). Essa técnica pode ser inter-

pretada como a combinação de algoritmos que procuram encontrar os subconjuntos de atributos mais relevantes para uso dos classificadores. Os algoritmos utilizados pela técnica de seleção de atributos são comumente categorizados conforme seu tipo de utilização. Quanto ao tipo de utilização, eles variam em: filtro (do inglês, *filter*) ou empacotado (do inglês, *wrapprer*) (GAO; KHOSHGOFTAAR, 2015). O entendimento correto sobre o funcionamento de cada categoria de algoritmo pode levar a escolhas mais acertadas para uso conjunto da técnica com o processo de predição de defeitos.

De forma geral, o uso 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, com a redução dos custos computacionais e a multidimencionalidade dos atributos. Por exemplo, o projeto fictício "Projeto Alpha", pode apresentar um conjunto de dados formado por 25.000 objetos e cada um deles apresentar 120 atributos. A utilização da técnica de seleção de atributos pode identificar que desses 120 atributos, apenas 10 são relevantes, ou seja, que de fato exercem influência direta na identificação do atributo alvo. Dessa forma, os outros 110 atributos seriam descartados dos objetos do conjunto de dados para execução do modelo de predição. Portanto, é possível notar que o processo de predição se torna mais ágil com seu uso, uma vez que menos atributos precisam ser avaliados. Também é objetivo do uso da técnica de seleção de atributos que os resultados de predição sejam melhorados, haja vista que os atributos resultantes são os que melhor estabelecem uma relação com o atributo alvo (KHOSHGOFTAAR et al., 2012).

Entretanto, apesar do benefício do uso dos algoritmos de seleção de atributos, existem alguns pontos de atenção que precisam ser observados para seu uso (HALL; SMITH, 1998; ALTIDOR et al., 2009). Cada tipo de utilização da técnica, se como filtro ou empacotado, exige cuidados que precisam ser observados. O uso da técnica de seleção de atributos como filtro funciona como o próprio nome diz, ou seja, como um filtro aplicado aos atributos do conjunto de dados para uso pelos algoritmos de aprendizado de máquina utilizados no processo de predição. Os algoritmos de seleção de atributos utilizados que funcionam como filtro têm um comportamento de considerar que os atributos são independentes para identificação do atributo alvo, não se estabelece correlação direta entre eles. Um fator positivo para aplicação como filtro é que não existe dependência dos algoritmos de aprendizado de máquina para execução da técnica de seleção de atributos. A utilização de algoritmos de seleção de atributos como empacotados tem um comportamento diferente da técnica apresentada anteriormente. Com o uso da técnica de empacotamento, há o empacotamento do algoritmo de apren-

dizado de máquina com algoritmos de seleção de atributos em diferentes combinações de subconjuntos de atributos. Cada subconjunto é utilizado para treinar o modelo por meio do uso do algoritmo de aprendizado de máquina, e a taxa de erro da classificação de cada subconjunto é analisada para estabelecer o subconjunto que melhor representa todo conjunto de atributos. Um dos grandes benefícios de seu uso em relação ao método de filtro é a possibilidade de estabelecer correlação entre os atributos. Contudo, suas desvantagens são: dependência gerada entre o algoritmo de seleção de atributos e o algoritmo de aprendizado de máquina; maior custo de execução, se comparado com o tipo de utilização como filtro.

As técnicas de seleção de atributos ainda podem ser categorizadas de outras duas formas, como ranqueadoras (do inglês, *ranking*) ou selecionadoras de subconjunto (do inglês, *subset selection*) (GAO; KHOSHGOFTAAR, 2015). As ranqueadoras criam um "ranking" de influência dos atributos com relação ao atributo alvo. Já as selecionadoras de subconjunto procuram encontrar um subconjunto de atributos, relacionados, que exerçam influência sobre o atributo alvo. Em ambos os casos, o objetivo é encontrar os atributos que darão o melhor desempenho ao modelo de predição e diminuição do custo de criação do modelo.

Independente da forma como a técnica de seleção de atributos é categorizada, se como ranqueadora ou selecionadora de subconjunto, se como filtro ou empacotado, seu funcionamento depende de dois tipos de algoritmos utilizados em conjunto (CHANDRASHEKAR; SAHIN, 2014; GAO; KHOSHGOFTAAR, 2015). O primeiro tipo agrupa algoritmos que estabelecem formas por meio dos quais subconjuntos de atributos são pesquisados e encontrados, chamados de métodos de pesquisa (do inglês, *search methods*). O segundo tipo são os algoritmos que procuram estabelecer uma relação entre os atributos que foram encontrados, com a geração de uma análise entre eles, chamados de algoritmos de avaliação de atributos (do inglês, *attribute evaluators*).

Segundo a literatura, os principais algoritmos utilizados como método de pesquisa pela técnica de seleção de atributos são apresentados a seguir (HALL; SMITH, 1998; HALL; HOLMES, 2003; PITT; NAYAK, 2007; WITTEN et al., 2011).

***Ranker*.** Esse algoritmo procura encontrar os atributos analisando-os individualmente e com a criação de um *Ranking* para utilização pelos algoritmos de seleção de atributos. Na verdade, os algoritmos de avaliação de atributo utilizam os algoritmos de pesquisa de atributos para criar uma lista com os atributos e sua influência para com o atributo alvo. Essa lista é utilizada para identificar quais atributos devem

ser mantidos ou descartados com a utilização dos algoritmos de aprendizado de máquina utilizados no processo de predição.

***Best First*.** Implementa uma pesquisa por conjuntos de atributos tendo como ponto de partida um subconjunto vazio de atributos. Os atributos vão sendo testados e adicionados ao subconjunto de atributos à medida que melhoram o processo preditivo. Nesse processo o algoritmo pode mudar o sentido da busca e voltar atrás em decisões tomadas, sem perder as informações já encontradas até aquele momento.

***Genetic Search*.** Utiliza os princípios de seleção natural da biologia e, para isso, utiliza conceitos de mutação e cruzamento de atributos. O conceito de mutação trabalha com a adição ou remoção de atributos em subconjuntos formados para análise, já o conceito de cruzamento procura agrupar atributos diferentes em pares, também para checagem. A ideia é encontrar conjuntos de atributos que permitam a evolução dos atributos ao longo do tempo, com o objetivo de gerar melhores predições.

***Greedy Stepwise*.** Percorre subconjuntos de atributos e busca características para melhorar a predição, parando apenas quando não consegue melhorar os resultados encontrados. A forma de busca dos atributos ocorre percorrendo o conjunto de atributos, com a adição ou remoção deles noutro subconjunto, sendo que isso ocorre, geralmente, de duas formas. Quando os atributos vão sendo analisados um a um e vão sendo adicionados em um subconjunto com melhor capacidade preditiva, dá-se o nome de pesquisa de atributos para frente (do inglês, *forward*), já quando um subconjunto maior é formado, por todos os atributos e eles vão sendo removidos para teste da melhora preditiva, tem-se o nome de pesquisa de atributos para trás (do inglês, *backward*).

Segundo a literatura, os principais algoritmos utilizados para avaliação de atributos pela técnica de seleção de atributos são descritos a seguir (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 da quantidade de informação que cada atributo oferece para identificação do atributo alvo, de forma que as

informações dos atributos reduzam a entropia[8] sobre o atributo alvo. Esse algoritmo tem uma tendência em considerar que atributos com valores altos, mesmo que eles não influenciem o atributo alvo, sejam selecionados (HALL; SMITH, 1998).

***Gain-Ratio*.** Possui uma abordagem semelhante ao *Information Gain*, contudo, sua fórmula de cálculo procura reduzir a influência de valores altos sobre os atributos, o que permite aos atributos com valores mais baixos terem sua influência considerada, contudo, passa a ter uma tendência de considerá-los com maior influência sobre o atributo alvo (HALL; SMITH, 1998).

***Symmetrical Uncertainty*.** Assim como *Gain-Ratio* e *Information Gain*, é uma medida baseada no conceito de entropia, com origem na teoria da informação (WITTEN et al., 2011). De forma semelhante ao *Gain-Ratio*, tem uma tendência para considerar atributos com valores baixos como tendo maior influência sobre o atributo alvo. Seu cálculo procura estabelecer uma relação de simetria entre as entropias do atributo que está sendo analisado e o atributo alvo (HALL; SMITH, 1998).

***Relief*.** Avalia o valor de um atributo nos objetos múltiplas vezes e procura estabelecer uma relação de proximidade com o atributo alvo. Esse algoritmo atribui um peso para cada atributo tendo como base a capacidade de predição para com o atributo alvo (ROBNIK-SIKONJA; KONONENKO, 2003).

One-R. Esse algoritmo trabalha com a criação de regras simples para cada atributo e procura estabelecer a relação entre as classes por meio da aplicação das regras criadas. Essas regras são treinadas e aquelas com menor número de erros são utilizadas (HOLTE, 1993).

***Chi-Square*.** Sua abordagem é por meio do cálculo estatístico do *Chi-Square* entre o atributo em análise com relação ao atributo alvo. Por meio do seu cálculo, tem-se o objetivo de observar se há dependência entre o atributo analisado e o atributo alvo. As hipóteses analisadas são: não há dependência entre o atributo analisado 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*). É um algoritmo que considera os atributos de forma independente, mas que analisa a capacidade preditiva de

[8]Entropia é uma medida de incerteza sobre uma determinada informação.

cada um e estima o grau de redundância e intercorrelação entre eles para identificação do atributo alvo (WITTEN et al., 2011).

Os métodos de pesquisa e de avaliação de atributos não podem ser indiscriminadamente usados uns com os outros. Por exemplo, o algoritmo de método de pesquisa *Ranker* tem uma abordagem estatística baseada em *Ranking*, o que o qualifica para trabalhar com todos os algoritmos de avaliação de atributos apresentados, exceto CFS. Esse algoritmo, por sua vez, pela característica de procurar chegar ao atributo alvo por meio da correlação entre atributos, utiliza todos os demais algoritmos de método de pesquisa exibidos, exceto *Ranker*.

2.2.5 VALIDAÇÃO CRUZADA

A validação cruzada é o nome dado à técnica que procura utilizar o próprio conjunto de dados do projeto como opção para treino e teste pelo modelo de predição (REFAEILZADEH et al., 2009). Serve de opção para situações em que um projeto não possui dados suficientes que possam servir de base para treinar o modelo de predição. Sua utilização também permite melhorar a performance dos modelos preditivos. Isso se dá pela possibilidade de aplicar diferentes algoritmos de classificação em fragmentos do conjunto de dados, o que permite a comparação dos resultados obtidos e identificação daqueles que apresentam melhor desempenho para predição naquele conjunto de dados.

A validação cruzada consiste em fragmentar o conjunto de dados utilizado pelo modelo de predição em K partições (REFAEILZADEH et al., 2009). O ideal é que cada uma das K partições tenham o mesmo número de objetos e que a distribuição do atributo alvo esteja também equiparada. Após isso, ocorrem K interações nas quais, em cada interação, são utilizados $K-1$ partições para treinar o modelo e uma partição é separado para teste. Dessa forma, todas as partições são utilizadas para testar o modelo, e os resultados obtidos podem ser analisados.

A análise se dá por meio de medidas de desempenho que permitem comparar a performance de diferentes algoritmos de aprendizado de máquina em cada partição do conjunto de dados (REFAEILZADEH et al., 2009). Isso ocorre uma vez que o conjunto de dados fragmentado em K partições pode ser aplicado igualmente em diferentes algoritmos. Isso permite que os resultados encontrados – tais como “sensibilidade”, “precisão” e “acurácia” – sirvam para comparar o desempenho entre os algoritmos aplicados

igualmente em cada partição.

A comparação de desempenho pode servir de parâmetro para escolha de um algoritmo vencedor (KOHAVI, 1995). Os resultados também podem servir para que os algoritmos sejam otimizados quanto a própria performance, por exemplo, para a modificação de parâmetros de configuração dos algoritmos de classificação com o objetivo de melhora do desempenho. Ou seja, a validação cruzada pode ser utilizada para que algoritmos tenham sua própria performance testada com intuito de obter os melhores resultados preditivos.

Um fator importante a observar é o número de partições utilizado na fragmentação do conjunto de dados. Uma medida bastante utilizada na predição de defeito com algoritmos de aprendizado de máquina é a estratificação em 10 partes, chamada *10-fold cross-validation* (KOHAVI, 1995). Essa forma de estratificação considera que 9 partições são separadas para treino do modelo e outra para testá-lo, com a geração de um total de 10 interações. A Figura 4 é um exemplo desse processo, contudo, nela é possível observar a estratificação do conjunto de dados em 3 partições e o uso dessas partes no processo de treino e teste pelo modelo de predição.

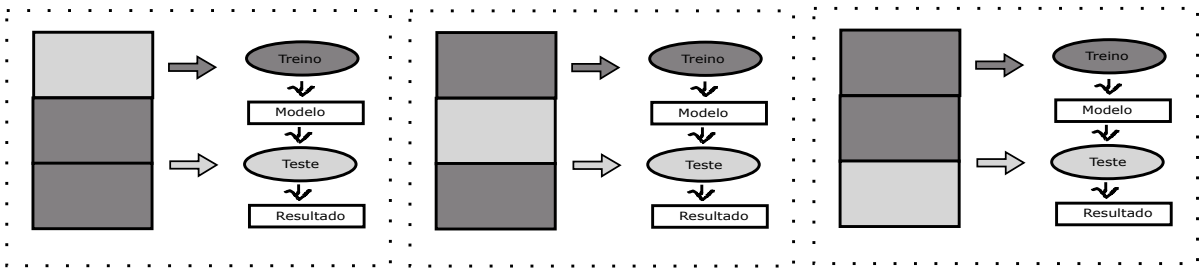

Figura 4: Uma representação de validação cruzada.
Fonte: Elaborado pelo autor, baseado em (REFAEILZADEH et al., 2009).

A validação cruzada é uma possibilidade para treinar modelos de predição, especialmente para projetos com dados históricos. Contudo, para projetos que estão em fase inicial de desenvolvimento, com poucos dados disponíveis, o desempenho dos modelos de predição são comprometidos mesmo com o uso de validação cruzada (REFAEILZADEH et al., 2009).

2.2.6 ALGORITMOS DE AGRUPAMENTO POR SIMILARIDADE

A técnica de agrupamento tem o objetivo de agrupar objetos que apresentem algum tipo de similaridade entre si (JAIN; DUBES, 1988). Para que esse agrupamento

aconteça existem algumas atividades, como: preparação dos dados; identificação de uma medida de similaridade; criação do agrupamento; validação dos agrupamentos criados; e, interpretação dos agrupamentos formados. Com a execução dessas atividades é esperada uma melhor formação dos agrupamentos e entendimento das informações geradas. Essas atividades são representadas na Figura 5 e são detalhadas a seguir.

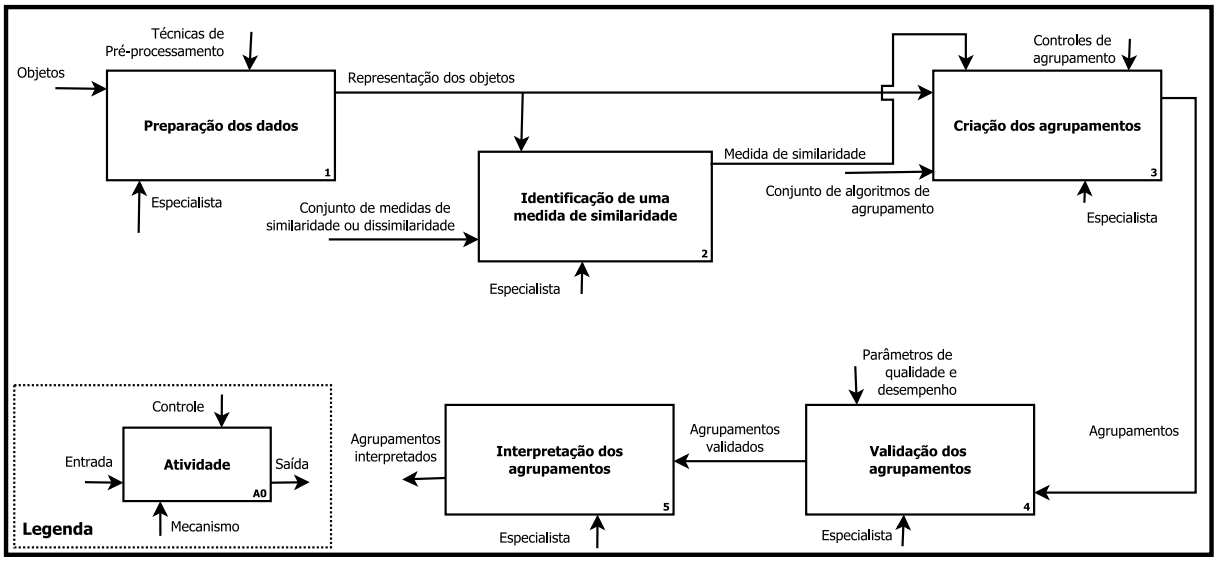

Figura 5: Atividades de criação dos agrupamentos pelos algoritmos de clusterização. Fonte: Elaborado pelo autor, baseado em (JAIN; DUBES, 1988).

A primeira atividade é a “1 Preparação dos dados”. Nessa etapa há a preocupação em ter uma forma apropriada de representação dos objetos para posterior utilização por um algoritmo de agrupamento. As principais formas de representação dos objetos são matrizes de objetos, matrizes e grafos de similaridade/dissimilaridade. Todas elas fornecem meios para que as atividades “2 Identificação de uma medida de similaridade” e “3 Criação dos agrupamentos” sejam executadas. Mas para que isso ocorra, atividades relacionadas ao pré-processamento dos dados, conforme relatadas na Seção 2.2.1, são utilizadas para tratamento dos dados do conjunto de objetos.

A próxima atividade se refere a “2 Identificação de uma medida de similaridade” para ser utilizada com a representação estabelecida dos objetos. Essa medida de similaridade, ou dissimilaridade, tem intenção de identificar quão semelhante, ou diferente, são os objetos que serão agrupados (JAIN; DUBES, 1988). Ela é utilizada para que os agrupamentos sejam criados, sendo que as medidas mais utilizadas para isso são a distância euclidiana e a correlação (FACELI et al., 2011).

A distância euclidiana calcula a distância entre dois objetos, sendo uma repre-

sentação da soma da distância entre os objetos que compõem o agrupamento. Seu cálculo é dado pela Equação 2, no qual:

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

- p_i e q_i são os objetos que terão as distâncias calculadas;
- i é o índice da dimensão para comparação entre os objetos;
- n é o número total de dimensões comparadas dos objetos; e,
- d é a somatória das distâncias calculadas entre os objetos em todas as suas dimensões.

Já o cálculo da medida de correlação é uma medida estatística que mede o grau de correlação entre os objetos que se pretende agrupar. Essa medida apresenta valores entre -1 e 1. O módulo dos valores $\geq$ 0,70 indica forte correlação entre objetos, o módulo dos valores $\geq$ 0,30 e $\leq$ 0,70 indicam correlação moderada, o módulo dos valores entre $\geq$ 0,00 e $\leq$ 0,30 indicam fraca correlação (FACELI et al., 2011). A fórmula de cálculo da correlação é dada pela Equação 3, no qual:

$$P_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{3}$$

- "X" e "Y" são os objetos que terão a medida de correlação calculada;
- *cov* é a covariância, o grau de interindependência entre os objetos;
- σ é o produto do desvio padrão dos objeto; e,
- P é o valor da correlação entre os objetos.

A medida para cálculo da distância que vai ser selecionada é o resultado gerado pela atividade "2 Identificação de uma medida de similaridade". Após isso, a próxima atividade a ser executada é a de "3 Criação dos agrupamentos". É nessa etapa que os objetos são apresentados para um ou mais algoritmos de agrupamento para a formação de possíveis grupos (do inglês, *clusters*). Cada algoritmo de agrupamento é baseado em um critério de agrupamento de dados e eles podem ser: agrupamento por compactação, agrupamento por encadeamento ou agrupamento por separação espacial (HANDL et al.,

2005). O agrupamento por compactação apresenta uma variação intragrupo pequena que produz agrupamentos compactos e tem no algoritmo K-means (ALPAYDIN, 2010) um exemplo; agrupamento por encadeamento (ou ligação) trabalha com a ideia que os objetos vizinhos devem compartilhar do mesmo grupo; agrupamento por separação espacial são aqueles que trabalham com a separação dos objetos baseado na distância entre os grupos.

Outra forma de pensar nos algoritmos de agrupamento está relacionada com o tipo de estrutura que pode ser gerado por eles, sendo duas as principais opções (JAIN; DUBES, 1988). A primeira é chamada de "agrupamento exclusivo x não exclusivo", no qual um objeto pode estar em apenas um agrupamento formado, quando o tipo de agrupamento é exclusivo, ou ele pode participar da formação de mais que um agrupamento, quando o tipo de agrupamento é não exclusivo. Em estruturas não exclusivas cada objeto apresenta um grau de pertinência a cada um dos agrupamentos existentes. Algoritmos probabilísticos ou *fuzzy* são exemplos de algoritmos que trabalham com essa abordagem. A segunda forma de agrupar os algoritmos é chamada de "agrupamento sequencial x hierárquico". Essa forma de agrupar é uma subdivisão dos agrupamentos exclusivos apresentados anteriormente. Os agrupamentos sequenciais geram uma única partição de dados, enquanto os hierárquicos produzem uma hierarquia aninhada de partições (FACELI et al., 2011).

Dentre os algoritmos sequenciais mais utilizados está o BSAS (do inglês, *Basic Sequencial Algorithmic Scheme*) (KAINULAINEN; KAINULAINEN, 2002). Ele é um algoritmo que recebe o conjunto de dados para processamento, analisa-o e processa-o de forma sequencial sem exigir que o conjunto de dados seja apresentado ao algoritmo mais que uma vez. Cada objeto do conjunto de dados é processado e analisado, sendo direcionado para um grupo já existente ou então um novo grupo é criado para recebê-lo. Para isso, o algoritmo BSAS analisa os padrões passados de cada objeto do conjunto de dados, analisa a medida de similaridade que foi estabelecida e analisa se o número máximo de *clusters* permitidos ainda não foi atingido (THEODORIDIS; KOUTROUMBAS, 2008). Uma representação de alto nível do algoritmo BSAS é exibida no Algoritmo 1. Esse algoritmo foi o utilizado neste trabalho e a medida de similaridade utilizada foi a distância euclidiana.

Algoritmo 1 Uma representação de alto nível do algoritmo BSAS.

1: $m = 1$; {Inicialização do contador que controla o número de agrupamentos criados.}
2: $Cm = \{x1\}$;
3: **for** $x = 2 \rightarrow N$ **do**
4: *Encontrar o agrupamento Ck que tenha a menor* $d(x,Ck)$ {Utiliza uma medida de distância para o cálculo.}
5: **if** $d(x,Ck) > \theta$ *AND* $(m < q)$ **then**
6: $m = m + 1$; {Incrementa o contator que controla o número de agrupamentos criados.}
7: $Cm = \{x\}$; {Novo agrumapemento criado.}
8: **else**
9: $Ck = Ck + \{x\}$; {Amostra adicionada ao agrupamento mais próximo.}
10: *Atualizar representante do agrupamento, quando precisar.*
11: **end if**
12: **end for**

Assim que os agrupamentos são criados, torna-se necessária a atividade de "4 Validação dos agrupamentos criados". Essa atividade busca analisar cada agrupamento de forma a identificar a significatividade e adequação frente ao conjunto de dados (JAIN et al., 1999). Essa análise tem como base alguns parâmetros para checar a influência na qualidade e desempenho dos agrupamentos criados, 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 *outliers*[9]. Essa atividade é pré-requisito pra que a próxima atividade, "5 Interpretação dos dados formados", seja executada. Tal atividade procura oferecer ao especialista meios pelos quais ele possa analisar cada agrupamento com a análise de sua relevância; significado; qualidade; e, correlação entre os componentes do agrupamento. Como resultado final dessa atividade, tem-se a interpretação dos agrupamentos gerados.

2.3 PREDIÇÃO DE DEFEITOS DE SOFTWARE

2.3.1 PROCESSO DE PREDIÇÃO DE DEFEITOS

Segundo o dicionário Michaelis (MICHAELIS, 2009), predição é um substantivo feminino que assume o significado de "ato ou efeito de predizer; prognóstico; afirmar o que vai acontecer no futuro". Quando o termo predição é aplicado em engenharia de

[9]Outliers são objetos com valores atípicos, que destoam dos demais integrantes do agrupamento criado.

software, mais precisamente com intuito de predizer defeitos, seu significado é semelhante ao apontado pelo dicionário; antecipar a ocorrência de defeitos que acontecerão em versões futuras de um software. O uso da predição de defeitos em engenharia de software está intimamente relacionado à qualidade do software que está sendo produzido. Para que a predição ocorra, um primeiro passo é selecionar um conjunto de dados com os atributos (métricas) que se pretende analisar. Isso pode ser feito com a aplicação de técnicas de mineração de dados. Após os dados estarem minerados e preparados para uso, um modelo de predição é criado, treinado, testado e indicadores de performance são estabelecidos para aferir a qualidade das predições realizadas. Assim, é possível observar que todas essas atividades tornam a predição de defeitos uma tarefa complexa; e, infelizmente não há um padrão que possa ser seguido para que a predição de defeitos seja criada, treinada e testada em um projeto de software o que aumenta os desafios para sua utilização (HALL et al., 2012).

Apesar das dificuldades apresentadas existem muitos benefícios que justificam e motivam o uso da predição de defeitos em projetos de software (MOSER et al., 2008; HALL et al., 2012). Dentre eles está a maior produtividade da equipe envolvida no desenvolvimento do projeto, uma vez que gastarão menos tempo com a procura de defeitos, e com isso, terão mais tempo para se dedicarem na criação de novas implementações. Estudos mostram que até 60% do tempo gasto em inspeções pela equipe de qualidade pode ser reduzido com o uso de predição de defeito (PETERS et al., 2013). Isso sem falar da percepção de qualidade dos usuários finais, que pode potencializar novas receitas e novos clientes (MOSER et al., 2008).

Para que a predição de defeitos seja viável são necessárias informações sobre o projeto que se deseja analisar. Essas informações têm origem, geralmente, em *releases*[10] do software em desenvolvimento. A ideia é que cada *release* já desenvolvida do software forneça dados para a criação de um conjunto de dados distinto para uso no processo de predição que será criado. Quanto mais conjuntos de dados formados, melhor é para o modelo de predição. Dessa forma, é possível identificar que projetos que estão em fase inicial de desenvolvimento apresentam poucos dados, talvez não possuam nem uma *release* liberada para que os dados sejam coletados. Assim, existe uma dificuldade natural em criar modelos de predição para tais projetos.

Os modelos de predição precisam de dois conjuntos de dados, um conjunto de treino e outro conjunto de teste. O conjunto de treino, frequentemente, é formado por

[10]Release é uma distribuição pública ou privada de uma nova versão de um software ou sua atualização.

todos os conjuntos de dados de *releases*, exceto da mais recente, que é utilizada para formar o conjunto de teste. Por exemplo, um projeto fictício “Projeto Alfa”, tem sua décima *release* liberada para uso de seus usuários. Esse projeto pode fazer uso das informações das 9 *releases* anteriores para criação de um conjunto de treino para o modelo de predição e utilizar os dados disponíveis da décima *release* para testar o modelo e predizer ocorrências de defeito que podem acontecer na *release* de número 11 que ainda estará sendo desenvolvida. Assim, é possível observar que os modelos de predição utilizam dados do passado para aprender como predizer defeitos em *releases* futuras (WEYUKER et al., 2006; BACCHELLI et al., 2010). Na Figura 6 é possível visualizar o processo descrito anteriormente. Nela é possível visualizar a formação dos conjuntos de treino e teste, para uso pelo modelo para geração da predição. Os conjuntos de treino e teste têm, geralmente, suas origens nas *releases* de software armazenado em um repositório, comumente, controlado por ferramentas de controle de versão como CVS[11], SVN[12] ou GIT[13].

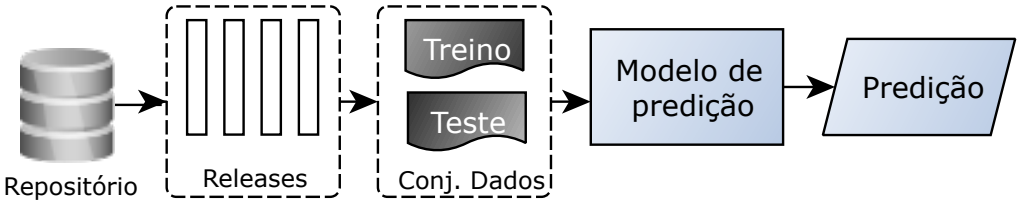

Figura 6: Uma representação da formação do conjunto de dados de treino e teste. Fonte: Elaborado pelo autor.

É importante notar que o conjunto de dados que forma o conjunto de treino e teste, é composto por um grupo de objetos, que por sua vez, pode representar vários elementos que fizeram parte de cada *release* desenvolvida. O elemento mais comum representado é o próprio código fonte, assim sendo, cada código fonte ou pacote do projeto pode ser representado como um objeto dentro do conjunto de dados. Por exemplo, uma *release* do projeto fictício “Projeto Alfa”, pode ser formada pelos pacotes: gráfico, que contêm 5 arquivos com códigos fonte com as interfaces gráficas de interação com o usuário; negócio, que contêm um código fonte que faz o controle da regra de negócio do sistema; e, controle, composto por outro código fonte que cuida de todas as requisições

[11] http://savannah.nongnu.org/projects/cvs/.
[12] http://subversion.apache.org/.
[13] certo http://git-scm.com/.

feitas pelos usuários do sistema, feitas com a utilização das interfaces gráficas. Com a análise desse exemplo, se o conjunto de dados for formado pelos pacotes que essa *release* do sistema possui, ele será formado por três objetos, sendo eles: gráfico, negócio e controle. Já se o conjunto de dados for formado pelos códigos fonte da *release* desse projeto teremos um conjunto de dados formado por 7 objetos, que é o número total de arquivos com códigos fonte que o projeto contém.

Cada objeto adicionado ao conjunto de dados precisa fornecer dados relevantes para apoiar o processo de predição (ALPAYDIN, 2010; WITTEN et al., 2011). Esses dados são os valores obtidos por meio das métricas. É importante notar que, conforme apresentado na Subseção 2.3.2, existe uma vasta relação de métricas disponíveis na literatura e também que elas podem ser de vários tipos, como: código, processo, sociais ou de contexto. Com isso, aquelas métricas mais relevantes podem ser mapeadas em atributos e utilizadas no processo de predição em que assumem o papel de preditores.

Após a formação do conjunto de dados, nele são aplicadas técnicas de pré-processamento com o objetivo de tratar os atributos e os objetos que o compõem de forma que o conjunto de dados esteja minimamente tratado para criação do modelo de predição. A criação de um modelo de predição se dá por meio da seleção de um algoritmo de aprendizado de máquina, da escolha de uma forma de tratar os atributos dos objetos do conjunto de dados, para reduzir sua dimensionalidade e a identificação de indicadores que permitam a análise dos resultados, após a execução do modelo de predição (RUSSELL; NORVIG, 2009; FACELI et al., 2011).

Os algoritmos são aplicados aos conjuntos de dados de treino e teste. Como informado anteriormente, o conjunto de treino é utilizado para que os algoritmos aprendam a reconhecer padrões, estabelecer correlações, e aprender qual a relação entre os atributos dos objetos com o atributo alvo. Depois que o conjunto de treino é aplicado, o conjunto de teste é apresentado ao modelo de predição que usa as informações aprendidas na execução do passo anterior para realizar a predição de eventos futuros (THEODORIDIS; KOUTROUMBAS, 2008). A averiguação da qualidade das predições geradas é a etapa final, geralmente com análise de um especialista, que busca interpretar as informações de desempenho obtidas.

2.3.2 MÉTRICAS DE SOFTWARE COMO PREDITORES

Uma métrica de software é um dado coletado durante o desenvolvimento de um software que pode ser utilizado para representar algum tipo de conteúdo (HALL et

al., 2012). As métricas de software podem apresentar as mais variadas informações, por exemplo: o número de atributos declarados em um sistema; o número de métodos; o número de linhas de código de cada método; o grau de acoplamento e coesão entre os módulos; o número de vezes que um desenvolvedor contribuiu com o desenvolvimento de um determinado módulo de um projeto; ou, o número de projetos impactados com uma mudança do software. Enfim, existe na literatura um vasto conjunto de métricas de software (RADJENOVIC et al., 2013).

Esse vasto número de métricas tem um lado bom e outro que merece cuidado. Por um lado, ter um grande conjunto de métricas que podem contribuir para que o processo de desenvolvimento do software ocorra com qualidade é bom, pois disponibiliza uma série de dados para análise. Por outro lado, se o uso delas não for feito de forma criteriosa, há risco de incorrer no problema da multidimensionalidade, que pode comprometer a performance dos algoritmos de aprendizado de máquina que utilizarem tais métricas. Assim, entender como elas estão agrupadas e quais são as opções disponíveis pode funcionar como critério de filtro para escolha das métricas mais adequadas (DÁMBROS et al., 2010).

Segundo alguns pesquisadores da área, as métricas de software podem ser agrupadas em três principais grupos (CATAL; DIRI, 2009; RADJENOVIC et al., 2013):

Grupo de elementos técnicos. As métricas de software que têm dados referentes a critérios técnicos, como: complexidade do projeto; número de linhas de código; número de linhas comentadas no código; número de linhas em branco no código; número de classes ao qual uma classe é acoplada; número de métodos que podem ser executados para responder uma mensagem; número de métodos de uma classe; complexidade existente entre interfaces de componentes; e, número de linhas de código geradas automaticamente (MCCABE, 1976; HALSTEAD, 1977; PINZGER et al., 2008; HALL et al., 2012).

Grupo de elementos de processo. As métricas de software que têm dados referentes a elementos do processo de desenvolvimento, como: falta de completude de um requisito; a quantidade de códigos fontes que serão afetados por um requisito; nível de experiência dos programadores; e, número de funcionalidades que foram adicionados e que afetaram um novo módulo (KASZYCKI, 1999; JIANG et al., 2007).

Grupo de elementos sociais. As métricas de software que têm dados referentes a in-

formações sociais relacionados ao processo de desenvolvimento, como: grau de conhecimento dos desenvolvedores; características das empresas envolvidas no processo; característica do cliente; critérios utilizados na comunicação entre os desenvolvedores; foco em um único módulo de desenvolvimento ou foco em múltiplos módulos; e, modelo de gestão utilizado (NAGAPPAN et al., 2006; PINZGER et al., 2008; DÁMBROS et al., 2010; WIESE et al., 2014).

Além desses agrupamentos, alguns autores destacam outro conjunto de métricas que exercem influência no processo de predição, as chamadas métricas de contexto (HALL et al., 2012; ZHANG et al., 2014). Alguns exemplos de métricas de contexto são: domínio da aplicação; grau de maturidade dos sistemas; uso de ferramentas de controle de atividades e linguagem de programação. Essas métricas são, segundo alguns estudos, consideradas de grande importância para o processo de predição de defeitos (HALL et al., 2012; ZHANG et al., 2014).

2.3.3 MEDIDAS DE AVALIAÇÃO DE DESEMPENHO DE MODELOS DE PREDIÇÃO

Os resultados obtidos pelos modelos de predição precisam ser avaliados. O meio pelo qual essa avaliação é feita é com o uso de uma medida de desempenho. Na literatura existe um amplo conjunto de medidas que podem ser utilizadas de forma independente ou em conjunto e a seguir as mais utilizadas conforme a literatura é apresentada (OSTRAND; WEYUKER, 2007; BOWES et al., 2012):

Matriz de confusão. Sua representação é em forma de tabela que permite a visualização de resultados, sendo muito utilizada para avaliar modelos que utilizam algoritmos de classificação. Ela é representada por quatro conjuntos de dados conforme exibido na Tabela 1, e que serão descritos a seguir. Importante observar que os demais indicadores descritos na sequencia têm como base as informações provenientes da matriz de confusão:

Tabela 1: Exemplo de matriz de confusão.

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

Fonte: Elaborado pelo autor.

Na Tabela 1, TP retrata o valor de verdadeiro positivo (do inglês, *True Positive*). Esse valor informa quantas ocorrências do atributo alvo foram previstas pelo modelo como defeituosa e, após os testes, ela foi comprovada com defeito. O valor de FP retrata o valor de falso positivo (do inglês, *False Positive*). Esse valor informa quantas ocorrências do atributo alvo foram previstas pelo modelo como defeituosa, mas, após os testes, ela não apresentou defeito. O valor de TN retrata o valor de verdadeiro negativo (do inglês, *True Negative*). Esse valor informa quantas ocorrências do atributo alvo foram previstas pelo modelo como livre de defeito e, após os testes, ela realmente não apresentou defeito. O valor de FN retrata o valor de falso negativo (do inglês, *False Negative*). Esse valor informa quantas ocorrências do atributo alvo foram previstas pelo modelo como livre de defeito, mas, após os testes, foram constatadas ocorrências de defeito.

Acurácia (do inglês, *Accuracy*). Uma medida de exatidão de como as classes estão sendo classificadas corretamente, como verdadeiro positivo ou verdadeiro negativo. Valores mais próximos de 1 identificam a acurácia do modelo. Sua fórmula de cálculo é: $Acuracia = \frac{(TP+TN)}{(TP+TN+FP+FN)}$;

Precisão (do inglês, *Precision*). Uma medida de precisão que representa o quanto se consegue classificar corretamente arquivos com defeito, no qual valores próximos de 1 identificam a precisão do modelo. Sua fórmula de cálculo é: $Precisão = \frac{(TP)}{(TP+FP)}$;

Sensibilidade (do inglês, *Recall*). Uma medida da proporção de arquivos classificados erradamente como livre de defeito, no qual valores próximos de 1 indicam menor ocorrência de falso negativo. Sua fórmula de cálculo é: $Sensibilidade = \frac{(TP)}{(TP+FN)}$;

F-Measure (Medida da média harmônica) Trata-se da média harmônica entre os valores de sensibilidade e precisão. Quanto mais próximo de 1 for o valor encontrado, melhor o desempenho do modelo de predição. Sua fórmula de cálculo é: $F-Measure = \frac{(2 \cdot Recall \cdot Precision)}{(Recall+Precision)}$;

ROC (do inglês, *Receiver Operation Characteristic*). A Curva ROC é uma técnica de comparação dos resultados da classificação do modelo de predição, que cruza em um plano cartesiano os resultados obtidos de FP, no eixo-x, e TP, no eixo-y. Ela pode ser interpretada como a probabilidade que, ao selecionar uma amostra classificada como positivo (TP) e outra como negativo (FP), o sistema atribua uma pontuação mais elevada para o positivo. Quanto maior a probabilidade, melhor

é o desempenho do modelo, pois representa a precisão de identificação da classe com defeito. Uma representação da Curva ROC é exibida na Figura 7, nela é possível verificar a representação gráfica de um classificador aleatório, também é exibida a plotagem de outros dois classificadores, "A" e "B". A curva ROC pode ser utilizada como medida para identificar qual classificador tem melhor desempenho. No caso representado na Figura 7, o classificador de melhor desempenho é o classificador "B", por ter suas medidas superiores ao classificador "A"; e,

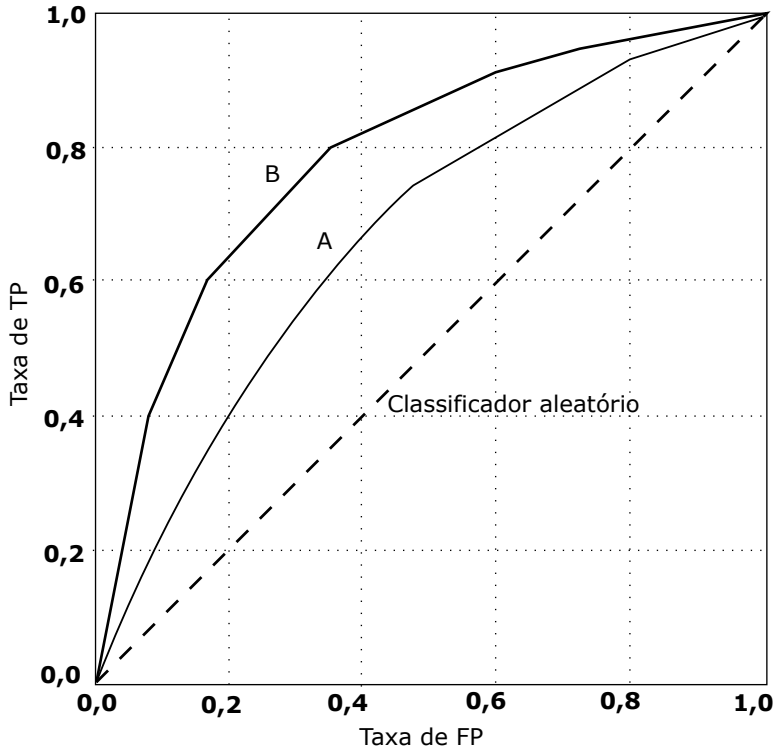

Figura 7: Uma representação da curva ROC.
Fonte: Elaborado pelo autor.

AUC (do inglês, *Area under an ROC Curve*). O valor de AUC é uma medida utilizada para comparação do desempenho de algoritmos com base nas informações extraídas a partir da curva ROC. Os valores de AUC são calculados a partir da curva ROC e variam entre 0 e 1. Valores que sejam mais próximos de 1 são considerados melhores, dessa forma, na comparação dos valores de AUC de dois algoritmos de classificação, aquele que apresentar o valor de AUC mais alto é o que tem melhor desempenho. Essa comparação visual do desempenho entre classificadores fica difícil de ser feita quando mais de um classificador está sendo analisado por meio da curva ROC. Essa dificuldade existe quando em algum momento na curva ocorre interseção entre as áreas de plotagem dos classificadores. Com isso, é inviável a

identificação visual de qual dos algoritmos foi o que apresentou melhor desempenho, pois com a observação da área de plotagem, em algum momento cada um deles pode ter tido melhor desempenho. Para sanar essa dúvida é que o cálculo do valor de AUC deve ser utilizado. A Figura 8 exibe tal situação com a exibição dos classificadores “A” e “B”.

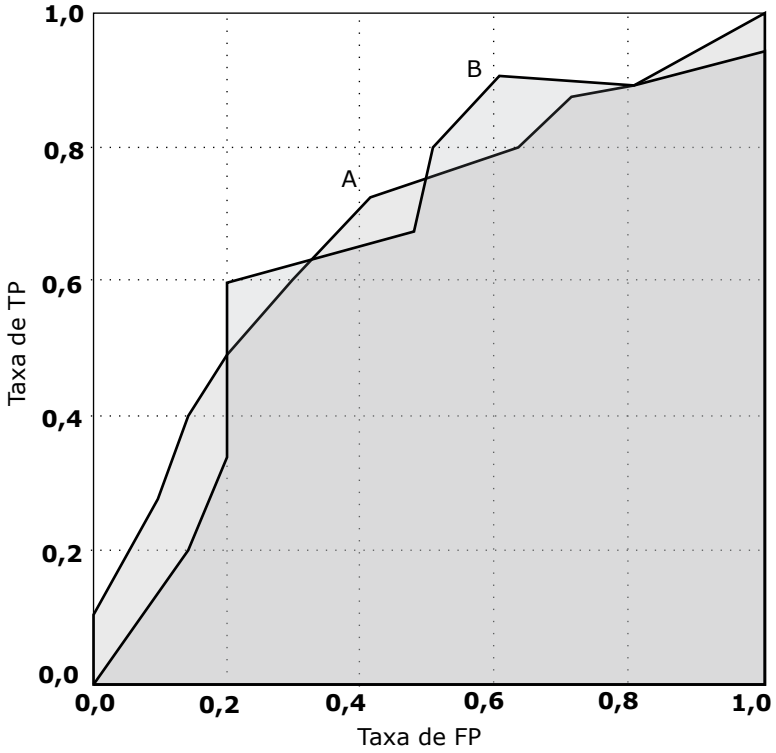

Figura 8: Uma representação da Área sob a curva ROC - AUC.
Fonte: Elaborado pelo autor.

2.3.4 PREDIÇÃO CRUZADA ENTRE PROJETOS DE SOFTWARE

A predição cruzada entre projetos é uma alternativa interessante para aqueles projetos que estão em fase inicial (TURHAN et al., 2009; ZIMMERMANN et al., 2009; JURECZKO; MADEYSKI, 2010; MENZIES et al., 2011; ZHANG et al., 2014). Como esses projetos possuem poucos dados para que um modelo de predição seja criado e testado, a utilização de conjunto de dados de outros projetos como parte da criação do modelo de predição, pode ser uma opção a ser considera. Para que isso ocorra existem desafios que precisam ser vencidos, como: estabelecer uma medida de semelhança entre projetos; encontrar projetos que sejam semelhantes ao que se pretende analisar (projeto

alvo); tratar os dados de forma que os conjuntos de treino e teste sejam compatíveis e aplicados ao modelo de predição tendo seu desempenho averiguado.

Utilizar dados de um projeto para predizer defeito em outro é possível segundo alguns autores em algumas situações (MENZIES et al., 2007; ZIMMERMANN et al., 2009; HALL et al., 2012; ZHANG et al., 2014). Segundo pesquisas, projetos que tenham características em comum têm maiores chances de compartilhar dados para que um modelo de predição cruzada entre projetos seja criado. Dessa forma, o primeiro desafio é estabelecer medidas por meio das quais se pretende encontrar projetos que tenham características semelhantes e assim, compartilhar do conjunto de dados. Algumas características parecem ser melhor preditoras do que outras, como o caso de métricas de contexto (HALL et al., 2012; ZHANG et al., 2014), contudo, quais exatamente deve ser alvo para comparação entre projetos, ainda é um assunto em aberto (OSTRAND et al., 2005; NAGAPPAN et al., 2006). A forma mais comumente utilizada para procurar estabelecer comparação entre projetos é pelo conjunto de métricas disponíveis nos projetos. Com isso, se o projeto alvo permite que essas métricas sejam coletadas, então há uma chance de encontrar outros projetos com essas mesmas características.

Com as medidas que estabelecem semelhança entre projetos determinada, o próximo passo é encontrar os projetos que tenham semelhança com o projeto alvo por meio dessas medidas (JURECZKO; MADEYSKI, 2010). Tanto empresas públicas ou privadas, quanto a comunidade podem usar informações de outros projetos que já tenham sido produzidos. A ideia é que, por exemplo, uma empresa possa encontrar dentre os projetos que já tenha desenvolvido aqueles que apresentam as mesmas características que o projeto alvo apresenta. Da mesma forma, a comunidade de software livre pode pesquisar nos repositórios de software livre projetos com características semelhantes ao projeto alvo, para replicação. Com relação a projetos de software livre, os repositórios mais utilizados por trabalhos sobre predição de defeito, são: NASA MDP[14], PROMISE[15] e Eclipse[16] (MALHOTRA, 2015).

Um ponto importante a observar é a forma de comparar as medidas de similaridade. Por exemplo, voltando para o projeto fictício “Projeto Alfa”, se as medidas de similaridade selecionadas forem as métricas: número de linha de código; linguagem de programação; número de desenvolvedores e nível de conhecimento deles é importante observar que cada uma dessas métricas tem seu tipo e escala próprio; e, consequente-

[14] http://nasa-softwaredefectdatasets.wikispaces.com
[15] http://openscience.us/repo/
[16] http://www.st.cs.uni-sb.de/softevo/bug-data/eclipse

mente uma forma de comparar seus valores de referência. A maneira como essa comparação é construída varia e precisa ser observada para que os conjuntos de dados sejam compatíveis e possíveis de aplicação ao modelo (ZIMMERMANN et al., 2009; ZHANG et al., 2014).

Com as medidas de similaridade estabelecidas e com os projetos encontrados, o modelo de predição pode ser construído. Quanto mais projetos forem encontrados com semelhança, melhor. A forma de usar o conjunto de dados dos projetos semelhantes pode variar. Podem-se formar pares entre o projeto similar e o projeto alvo (ZIMMERMANN et al., 2009), ou então, agrupamento do conjunto de dados dos projetos (SATIN et al., 2015). O fato é que o conjunto de dados dos projetos encontrados será utilizado para treinar o modelo e o conjunto de dados do projeto alvo será utilizado para testar o modelo de predição. A partir dai o processo de análise segue como apresentado na Subseção 2.3.3, em que a análise das medidas de desempenho informará a qualidade da predição realizada.

Os resultados encontrados no processo de predição cruzada entre projetos são variados. Existem estudos que analisam que a transferência dos modelos de predição não são viáveis (BELL et al., 2006) contudo, outros reforçam que há a possibilidade, inclusive, com apresentação de resultados satisfatórios, semelhantes às predições realizadas com os dados de dentro dos próprios projetos (NAGAPPAN et al., 2006; ZHANG et al., 2014). Fato é que ainda se trata de uma área que necessita de maiores estudos, sendo alguns dos principais desafios o estabelecimento de medidas de similaridade, a forma de agrupamento de projetos por algum tipo de semelhança e a real influência dos algoritmos de aprendizado de máquina no resultado dos processos de predição (LESSMANN et al., 2008; GHOTRA et al., 2015).

2.4 TRABALHOS RELACIONADOS À PREDIÇÃO CRUZADA DE DEFEITOS ENTRE PROJETOS DE SOFTWARE

Um ponto importante é que, de maneira geral, é difícil realizar uma comparação direta de resultados entre trabalhos com modelos de predição de defeitos. Essa dificuldade tem diferentes causas, como por exemplo, conjuntos de dados de origens diferentes (dados livres ou dados de projetos proprietários, de certa forma inacessível pela maioria da comunidade científica), diferentes níveis de qualidade dos conjuntos de dados, diferentes organizações dos conjuntos de dados, uso de diferentes classificadores, uso de diferentes maneiras de medir o desempenho do modelo, uso de agrupamento

dos projetos ou não, dentre outros.

Nesta seção são discutidas pesquisas relacionadas com o tema predição cruzadas de defeitos entre projetos. Muito embora estudos mostrem uma variedade de trabalhos sobre esse assunto, são apresentados nesta seção aqueles considerados mais importantes pelos autores desse trabalho. O fato de não haver, até o momento, uma revisão sistemática da literatura sobre o tema predição cruzada de defeito entre projetos que possa ser utilizada como referência para identificação de trabalhos, o presente estudo selecionou um conjunto de pesquisas e teve como base alguns autores, que conforme a literatura especializada apresentam trabalhos relevantes sobre predição cruzada de defeito entre projetos. Dessa forma foram selecionados os trabalhos de Kitchenham *et al.* (KITCHENHAM et al., 2007), Watanabe *et al.* (WATANABE et al., 2008), Turhan *et al.* (TURHAN et al., 2009), Rahman *et al.*(RAHMAN et al., 2012), Jureczko *et al.* (JURECZKO; MADEYSKI, 2010), Zimmermann *et al.* (ZIMMERMANN et al., 2009), Menzies *et al.*(MENZIES et al., 2011) e Zhang *et al.* (ZHANG et al., 2014), brevemente descritos a seguir.

O trabalho proposto por Kitchenham *et al.* (KITCHENHAM et al., 2007) foi uma revisão sistemática da literatura, sobre o tema predição cruzada. Apesar do foco do trabalho de Kitchenham *et al.* estar mais direcionado para estimativas de custo, as considerações também são pertinentes para defeitos. Até sua publicação, em 2007, Kitchenham *et al.* não catalogou trabalhos que mostrassem melhores resultados que aqueles que utilizavam dados internos de seus projetos. Ainda existem ressalvas apontadas pelo trabalho que o próprio tempo é um fator determinante para que predições sejam executadas levando em conta dados históricos, por exemplo, devidas mudanças tecnológicas, processos e conhecimento que impactariam nas predições. O resultado de sua revisão sistemática é classificado como inconclusivo, apesar de ter encontrado evidencias de sua possibilidade de uso, não conseguiu chegar a respostas claras de quando é viável a utilização de dados de um projeto para predizer ocorrências noutro.

O trabalho proposto por Watanabe *et al.* (WATANABE et al., 2008) procurou responder a pergunta se é possível realizar a predição cruzada entre projetos mesmo sendo eles escritos em linguagens diferentes (métrica de contexto). Para responder essa pergunta ele trabalhou com um conjunto de dados composto por uma *release* dos projetos `sakura-editor` [17] e `jedit` [18]. Ambos os programas são editores de código fonte, contudo, o primeiro desenvolvido em linguagem C++, enquanto o segundo em

[17] http://sourceforge.net/projects/sakura-editor/
[18] http://sourceforge.net/projects/jedit/

Java. Como o conjunto de dados foi formado por apenas uma *release* de cada projeto o autor utilizou predição cruzada com o uso de *10-fold cross validation*. Os resultados encontrados pelo trabalho sinalizam que a linguagem em que os programas são escritos não é um impedimento para que o conjunto de dados de um projeto seja utilizado como treino para predizer defeitos para outro projeto, muito embora o autor mencione a importância que os projetos são do mesmo contexto (editores). Um aspecto que chama atenção é o fato de ter sido usada apenas uma *release* de cada projeto e também o baixo número de projetos, o que pode comprometer uma generalização. Outro ponto foram os baixos valores de precisão e, especialmente, de sensibilidade que a predição cruzada entre os projetos apresentaram. Quando o modelo de predição foi treinado com os dados do projeto `sakura-editor` e o conjunto de dados do projeto `jedit` foram aplicados para testar o modelo, os valores de precisão e sensibilidade encontrados foram 0,872 e 0,402 respectivamente. Quando o inverso foi feito, em que o projeto jedit teve seu conjunto de dados utilizado como treino para o modelo de predição e o conjunto de dados do projeto `sakura-editor` foi apresentado como teste para o modelo, os valores de precisão e sensibilidade encontrados foram 0,622 e 0,596 respectivamente.

O trabalho de Turhan *et al.* (TURHAN et al., 2009) analisou 12 projetos disponíveis no repositório da NASA. Um ponto de atenção sobre esses projetos é que foram desenvolvidos com base nos rigorosos princípios da ISO-9001 (INCE, 1994), estabelecido pela NASA. Segundo os autores, o fato de seguirem essa norma e o contexto de sua aplicação os diferencia de outros projetos de software livre, o que os torna difíceis preditores para outros projetos. Contudo, mesmo com essa peculiaridade, foi possível evidenciar o benefício do uso de predição cruzada entre esses projetos para encontrar defeitos, mesmo com resultados inferiores que as predições realizadas dentro dos próprios projetos. Esse estudo também mostrou um fator de risco interessante que é o aumento da taxa de falso positivo quando a predição cruzada é utilizada. Em seu trabalho foi encontrada uma média de 52,00% de taxa de falso positivo (tendo atingido um limite superior de 73,00%), o que caracteriza um risco para transferência de modelos. A forma utilizada pelo autor para classificar os projetos foi com o uso de algoritmos de aprendizado de máquina baseado no vizinho mais próximo. Os projetos tiveram as métricas de código como base para identificação da similaridade que foi calculada pela distância euclidiana.

O trabalho proposto por Rahman *et al.* (RAHMAN et al., 2012) utilizou um conjunto de dados formado por 38 *releases* de 9 projetos de software livre. O conjunto de dados das *releases* dos projetos analisados apresentam, preferencialmente, métricas de

código. Sua pesquisa procurou verificar se predições realizadas em projetos agrupados por similaridade têm desempenho semelhante às predições realizadas com o conjunto de dados dos próprios projetos. Seus resultados encontrados apontaram que as predições realizadas internamente nos projetos apresentam desempenho superior. Os parâmetros de desempenho avaliados foram precisão e AUC. Contudo, apesar da técnica proposta não ter encontrado evidencias de um desempenho superior para predições realizadas entre projetos, com sua aplicação foi possível constatar um menor custo da predição em comparação a qualquer modelo aleatório.

O trabalho de Jureczko *et al.* (JURECZKO; MADEYSKI, 2010) é mencionado por sua abordagem de criação de agrupamentos com conjunto de dados de projetos semelhantes. Seu objetivo com a criação desses agrupamentos é a melhora da predição cruzada entre projetos. Para isso, ele coletou 92 *releases* de 38 projetos de software livre ou acadêmicos com a criação de agrupamento entre projetos com a utilização de algoritmos hierárquicos de clusterização e K-means. Os resultados encontrados mostraram um desempenho melhor nas predições realizadas internamente nos agrupamentos criados que aquelas realizadas sem qualquer tipo de agrupamento.

O trabalho realizado por Zimmermann *et al.* (ZIMMERMANN et al., 2009) teve como escopo 12 projetos de grande porte. Esses projetos foram agrupados com a formação de 622 pares para execução de predição cruzada entre eles. De todos esses pares formados, apenas 3,40% conseguiram predizer defeitos entre si. A forma como Zimmermann *et al.* considera um projeto capacitado para predição de defeito em outros projetos é se seus valores de precisão, sensibilidade e acurácia, obtidos na fase de classificação, são todos acima de 0,75. O trabalho de Zimmermann *et al.* apontou que projetos terem o mesmo contexto ou usarem os mesmos processos não é o suficiente para torná-los preditores eficientes. Outra contribuição importante do trabalho de Zimmermann *et al.* foi a produção de uma árvore de decisão para os valores de precisão, sensibilidade e acurácia de forma que interessados na utilização da predição cruzada entre projetos possam usá-las como parâmetro para avaliar se com os resultados obtidos de seus projetos para precisão, sensibilidade e acurácia seria possível, ou não, a aplicação de predição cruzada para eles. Uma característica da árvore de decisão proposta por Zimmermann *et al.* e utilizada neste trabalho diz respeito à escolha do projeto alvo utilizado para testar os modelos de predição cruzada. Segundo Zimmermann *et al.* projetos com maior número de instâncias fornecem mais informações para predição em outros projetos.

O estudo conduzido por Menzies *et al.* (MENZIES et al., 2011) sugere que o trabalho de predição de defeitos para projetos agrupados por algum tipo de similaridade apresenta melhores resultados do que quando analisados sem qualquer agrupamento. Para comprovar essa afirmação, o autor utilizou um conjunto de projetos disponíveis no repositório PROMISE: `Luciane`; `Xalan`; `JEdit`; `Synapse`; `Tomcat`; `Velocity` e `Xerces`. A forma de agrupamento foi com uso do algoritmo `WHERE`, criado pelo autor, que procura agrupar projetos por semelhança entre seus atributos.

A proposta de Zhang *et al.* (ZHANG et al., 2014) é a criação de um modelo universal para predição de defeitos entre projetos. Sua proposta passa pela criação de uma técnica que tem como objetivo padronizar os valores dos atributos do conjunto de dados dos projetos. Sua proposta é que, com essa padronização, o processo de comparação para encontrar projetos similares se tornaria mais fácil. A forma como Zhang *et al.* construiu a padronização de valores para comparação da similaridade entre projetos têm alguns passos. Em primeiro lugar, busca-se para cada atributo, de cada par de projetos que se pretende comparar, identificar a inexistência de diferença estatística significativa entre os valores ou, caso haja, que a diferença não seja grande. Os projetos que não passam por essa qualificação são descartados e não passam pela segunda atividade, que justamente realiza a padronização dos valores de seus atributos. Essa atividade acontece com a divisão dos valores dos atributos em decis, divisão dos valores em 10 partes iguais e conversão desses valores para outra escala, de forma que os valores estejam sempre entre 1 e 10. Portanto, se um projeto deseja utilizar informações de outro projeto para o processo de predição de defeito, bastaria transformar os valores dos seus atributos conforme apresentado por Zhang *et al.*, de forma que todos eles estejam com as mesmas faixas de valores.

Os resultados encontrados por Zhang *et al.* evidenciam a possibilidade que resultados de predição de defeitos entre projetos possa ser uma opção às predições realizadas internamente no próprio projeto, com o conjunto de dados formato por suas *releases*. As evidências encontradas por Zhang *et al.* são de valores de AUC, obtidos nas predições entre projetos, terem sido superiores às predições realizadas, internamente, em cada projeto. Outro ponto interessante é que Zhang *et al.* utilizou as medidas estabelecidas por Zimmermann *et al.* (ZIMMERMANN et al., 2009) contendo valores de precisão, sensibilidade e acurácia acima de 0,75 para comparar os resultados encontrados em seu trabalho. Ele também utilizou os parâmetros estabelecidos por He:2012 *et al.* (HE et al., 2012) com valores de precisão acima de 0,50 e sensibilidade acima de 0,70 para analisar como foi o desempenho de sua proposta de modelo de predição cruzada

entre projetos. Com base nesses parâmetros Zhang *et al.* apresentaram resultado similar ao apresentado por Zimmermann *et al.*, com 3% de taxa de sucesso na realização de predição cruzada entre projetos. Contudo, chegaram em 14% de taxa de sucesso com a comparação pelos indicadores de He *et al.*.

2.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou uma revisão sobre os principais conceitos relacionados ao tema aprendizado de máquina, que são necessários para entendimento de como os modelos de predição de defeito são construídos. Também foram apresentadas as informações teóricas necessárias para entendimento de como os modelos de predição, e predição cruzada entre projetos, são criados, testados e tem seu desempenho avaliado. Além disso, uma relação de trabalhos que apresentam relação com o tema proposto pelo presente trabalho, e que contribuíram para a presente pesquisa, foram discutidos. No próximo capítulo é apresentada a criação, execução e avaliação inicial dos resultados obtidos com a criação da técnica de predição cruzada de defeito entre projetos criado no presente trabalho.

3 MODELOS DE PREDIÇÃO CRUZADA DE DEFEITOS APOIADO POR SIMILARIDADE DE PROJETOS

3.1 CONSIDERAÇÕES INICIAIS

O objetivo do presente capítulo é apresentar os passos dados na criação da técnica de predição cruzada de defeitos utilizada neste trabalho. Na Seção 3.2 é apresentada uma introdução que caracteriza a criação dos modelos de predição cruzada de defeitos, em que são revisados conceitos chaves sobre predição de defeitos, assim como são apresentados os principais passos a serem dados para criação de modelos de predição. Da Seção 3.3 até a Seção 3.7 são descritos detalhadamente os passos para criação dos modelos. Vale destacar que em algumas dessas seções os resultados dos estudos sobre algoritmos de classificação, de seleção de atributos e de agrupamento são mostrados minuciosamente.

3.2 CRIAÇÃO DO MODELO DE PREDIÇÃO CRUZADA DE DEFEITOS

A predição de defeitos pode ser obtida por meio da aplicação de modelos de predição em projetos nos quais se pretende avaliar a presença ou ausência de defeitos. Quando esses modelos são aplicados no conjunto de dados gerados pelos próprios projetos, se tem o que é chamado de predição local (do inglês, *within project*), em que informações históricas dos projetos são utilizadas para prever a ocorrência ou ausência de defeitos em versões futuras. Esse tipo de aplicação de predição de defeitos pode fazer uso do conjunto de dados de forma que ele seja fragmentado, tendo parte utilizada para treinar o modelo e outra parte utilizada para testar o modelo, a discutida validação cruzada. Contudo, como já mencionado, muitos projetos que estão em fase inicial apresentam dificuldade natural em antecipar a ocorrências de defeitos pela falta de dados históricos, como abordagem para minimizar esse problema a predição cruzada entre projetos pode ser utilizada. Independente da abordagem seguida, se predição local ou predição cruzada, um modelo de predição de defeitos é construído conforme crité-

rios de uma metodologia que permita seu desenvolvimento, treino, teste e avaliação de desempenho (HALL et al., 2012). Procurou-se seguir esses passos neste trabalho, no entanto, como a presente proposta visa propor uma técnica de predição cruzada entre projetos, com a utilização de similaridade entre eles, tais passos sofreram modificações, conforme ilustrado na Figura 9.

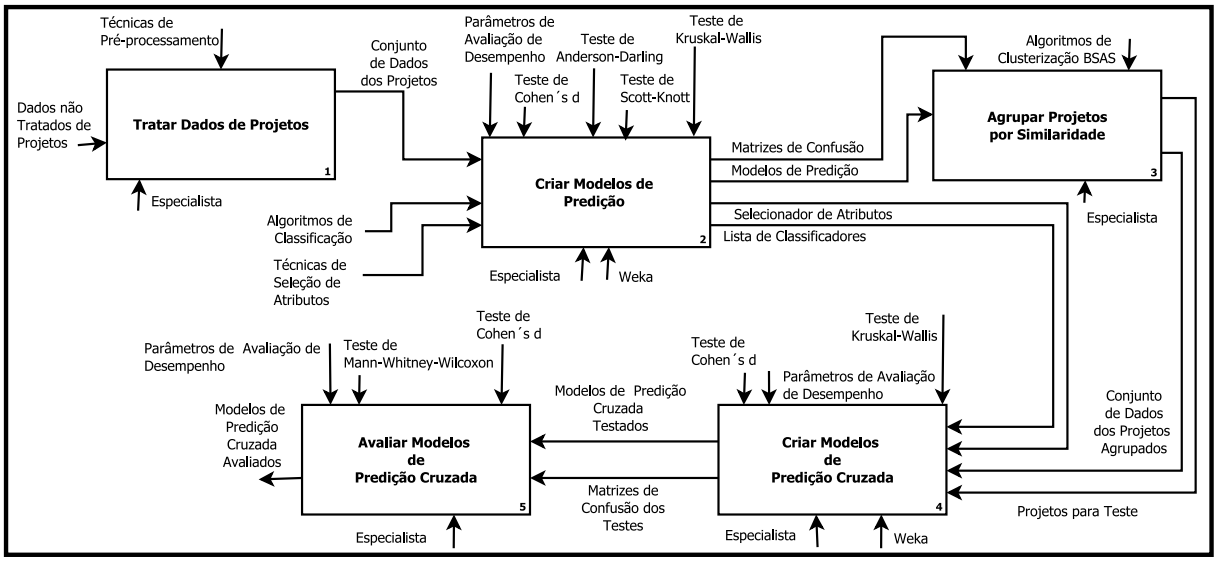

Figura 9: Atividades de criação dos modelos de predição cruzada de defeitos com o uso de similaridade de projetos.

Fonte: Elaborado pelo autor.

3.3 TRATAR DADOS DE PROJETOS

A atividade “1 Tratar Dados de Projetos” é responsável pelo tratamento dos dados que serão utilizados pela técnica proposta para construção o modelo de predição cruzada de defeitos entre projetos. Para cumprir seu objetivo, a atividade processa um conjunto de dados de projetos de software livre, minerados a partir de repositórios de software livre. Esse conjunto de dados é analisado por um especialista que tem como base para controle as técnicas de pré-processamento de dados e gera, como resultado de seu processamento, um conjunto de dados consistente e minimamente aceitável de projetos para uso da atividade “2 Criar Modelos de Predição”.

O conjunto de dados processado por essa atividade tem como origem o proposto por Zhang *et al.* (ZHANG et al., 2014). Esse conjunto de dados é composto por

1385 projetos de software livre minerado a partir do *Source Forge*[1] ou *Google Code*[2]. Todos esses projetos apresentam os mesmos 72 atributos, predominantemente de métricas de código. Um desses atributos é a variável dependente do modelo de predição. Essa variável é dicotômica e pode assumir o valor *Buggy* ou *Clean*. Essas duas classes, *Buggy* ou *Clean*, indicam respectivamente a presença ou ausência de defeitos em um dado arquivo – ou instância – dos projetos. Outro atributo relevante do conjunto de dados é o que representa a linguagem de programação que foi utilizada no desenvolvimento do software. O conjunto de dados deste trabalho é formado por projetos desenvolvidos em: C, C++, C#, Java e Pascal. Esse atributo é relevante por representar uma métrica de contexto. Tal métrica já foi estudada em outro trabalho (WATANABE et al., 2008), em que sua influência na transferência dos modelos de predição foi analisada. Apesar de estudos que mostram que atributos de contexto auxiliam na melhora da capacidade preditiva (ZHANG et al., 2014), outros afirmam que por si só, tais métricas não são suficientes para tornar os modelos de predição aptos para replicação para outros projetos (ZIMMERMANN et al., 2009). Dessa forma, esse ainda é um tema em aberto que necessita de maiores estudos.

Os atributos que compõem o conjunto de dados foram tratados conforme técnicas de pré-processamento. As técnicas aplicadas ao conjunto de dados foram: eliminação manual de atributos; limpeza de dados e redução manual de dimensionalidade (FACELI et al., 2011). Com o tratamento feito, os 72 atributos originais foram reduzidas para 67, e os projetos foram reduzidos de 1385 para 1283. Projetos com menos de 10 instâncias foram descartados, porque a técnica de treino de validação cruzada usada na atividade “2 Criar Modelos de Predição”, exige um mínimo de 10 instâncias. Ademais, atributos que não colaboram para o processo de predição foram removidos. Um exemplo de atributo descartado é o “origem dos projetos”, um atributo que tem valor nominal que armazena de onde o projeto foi minerado, por exemplo, *Source Forge* ou *Google Code*.

Muito embora o conjunto de dados tenha que passar pelos tratamentos adequados, não se pode garantir sua qualidade. A qualidade dos dados depende basicamente de atividades que antecedem o próprio processo de mineração e da construção do conjunto de dados, fases que não foram atacadas neste trabalho. Esse problema da qualidade dos dados pode influenciar no desempenho dos classificadores conforme discutido por alguns estudos (LESSMANN et al., 2008; GHOTRA et al., 2015). Outro ponto que

[1] http://sourceforge.net/.
[2] https://code.google.com/.

vale ressaltar é que existem outros conjuntos de dados disponíveis na literatura especializada, como os dados disponíveis no repositório PROMISE, NASA MDP e Eclipse. No entanto, mesmo com essa disponibilidade, existem trabalhos que mostram que não se pode garantir a qualidade desses conjuntos de dados também (ZIMMERMANN et al., 2009; DÁMBROS et al., 2010; GHOTRA et al., 2015). Além disso, de maneira geral, os conjuntos de dados disponíveis possuem poucos projetos, por exemplo, o conjunto de dados do PROMISE, tem 76 projetos em sua maior versão. O conjunto de dados usado neste trabalho foi escolhido justamente por possuir um número muito maior de projetos e também por não ser apontado na literatura especializada como sendo um conjunto de dados com baixa qualidade. Muito embora o conjunto de dados seja formado por apenas uma *release* de cada projeto, o que torna o processo de predição cruzada formado por esse conjunto de dados um desafio. Isso porque um maior número de informações sobre os projetos formados por mais *releases*, geralmente, colaboram para que o modelo de predição tenha melhor desempenho (HALL et al., 2012), mesmo levando em conta que o tempo pode atuar como um limitador para que dados históricos sejam utilizados irrestritamente por modelos de predição, devido a evoluções de elementos como: tecnologias; processos; e, conhecimento (KITCHENHAM et al., 2007).

3.4 CRIAR MODELOS DE PREDIÇÃO

O objetivo da atividade “2 Criar Modelos de Predição” é produzir: uma lista de classificadores que se desejou avaliar, conseguido a partir de algoritmos de classificação; um selecionador de atributos que trabalhe em conjunto com os classificadores selecionados, escolhido a partir de técnicas de seleção de atributos; e a matriz de confusão, produzida a partir do treino e teste dos modelos de predição de defeitos de cada um dos projetos que compõem o conjunto de dados. A execução dessa atividade tem como entrada o conjunto de dados dos projetos tratados pela atividade “1 Tratar Dados de Projetos”; um conjunto de algoritmos de classificação e outro de seleção de atributos encontrados na literatura especializada. Para geração dos resultados esperados um especialista utilizou a ferramenta Weka; como mecanismo de controle foi aplicado: parâmetros de avaliação de desempenho nos modelos de predição; e, testes estatísticos por meio das técnicas de Anderson-Darling (ANDERSON; DARLING, 1952), Kruskal-Wallis (KRUSKAL; WALLIS, 1952), Scott-Knott (JELIHOVSCHI et al., 2014) e Cohen´s *d* (COHEN, 1977).

Para um melhor detalhamento de todos os passos necessários para execução da

atividade “2 Criar Modelos de Predição”, ela foi dividida em subatividades conforme exibido na Figura 10. De forma geral, as atividades que detalham a criação dos modelos de predição seguem a proposta de Hall *et al.* (HALL et al., 2012): desenvolvimento, treino, teste e avaliação de desempenho. As subatividades “2.1 Escolher Classificadores”, “2.2 Escolher Selecionador de Atributos”, “2.3 Construir Modelos de Predição” e “2.4 Determinar Selecionador de Atributos” são detalhadas a seguir.

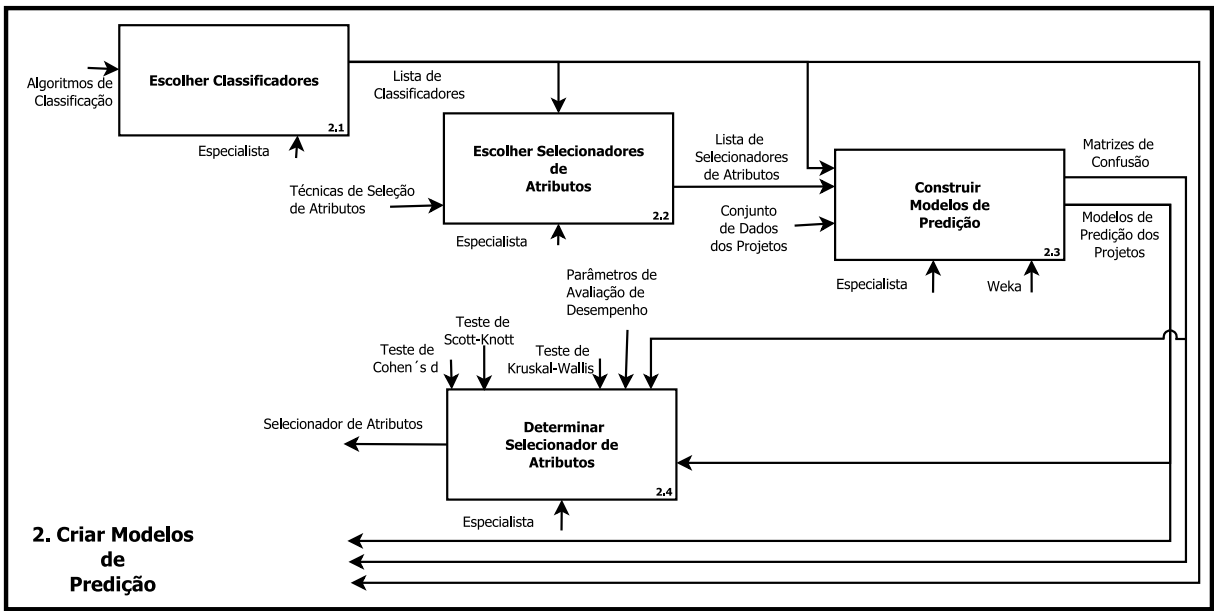

Figura 10: Atividades de criação dos modelos de predição individual dos projetos.
Fonte: Elaborado pelo autor.

O objetivo da primeira subatividade, “2.1 Escolher Classificadores”, é selecionar um conjunto de algoritmos de aprendizado de máquina para utilização pelo modelo de predição de defeitos. Como o atributo alvo das instâncias do conjunto de dados é nominal o problema que se pretende resolver é de classificação, dessa forma, foram selecionados algoritmos de classificação para compor a lista de algoritmos utilizados para criação do modelo de predição.

A escolha de classificadores adequados ainda é um assunto em aberto quando se trata de construção de modelos de predição de defeitos. Estudos são controversos sobre a influência dos algoritmos de classificação no desempenho obtido pelo processo de predição (LESSMANN et al., 2008; GHOTRA et al., 2015). Lessman *et al.* (LESSMANN et al., 2008) aponta em seu trabalho que não existem diferenças estatísticas relevantes nos resultados gerados pelo classificador utilizado no processo de predição. No entanto, em um estudo mais recente, essa afirmação é refutada por Ghotra *et al.* (GHOTRA et

al., 2015), que apontou problemas na qualidade do conjunto de dados dos projetos, e consequentemente das métricas, usados no experimento conduzido por Lessman *et al.*, o que pode ter influenciado nos resultados obtidos.

Nessa atividade, a análise de um especialista buscou dentre os classificadores mais utilizados em trabalhos similares (ZIMMERMANN et al., 2009; MENZIES et al., 2011; HERBOLD, 2013), e considerou as pesquisas de Ghotra *et al.* (GHOTRA et al., 2015) e Lessman *et al.* (LESSMANN et al., 2008), selecionar algoritmos com bom desempenho, mas com abordagens distintas. Isso foi feito com o intuito de avaliar a influência dos classificadores nos resultados do processo de predição, uma vez que esse assunto é controverso. Os algoritmos utilizados na construção dos modelos de predição foram: com atuação estatística, *Naive Bayes* e *Simple Logistic*; baseados em árvores de decisão, J48 e *Random Forest*; baseados em regras de decisão, *Decision Table* e OneR; baseado em redes neurais artificiais, *Multilayer Perceptron*; e, baseado em máquinas de vetores de suporte, SMO. Todos esses algoritmos tiveram como entrada os atributos dos projetos contidos no conjunto de dados, também chamados de métricas, além de terem sido executados conforme parametrização padrão de suas implementações disponibilizadas pela ferramenta Weka. É importante notar que essas métricas, na construção dos modelos de predição, conforme a literatura, assumem o papel de preditores para os modelos (GRAVES et al., 2000; OSTRAND et al., 2005).

A subatividade "2.2 Escolher Selecionador de Atributos" tem o objetivo de escolher técnicas para seleção de atributos, e assim permitir a diminuição da dimensionalidade dos dados para melhora do desempenho dos modelos de predição e viabilizar sua transferência para outros projetos (HALL et al., 2012). Para isso, essa atividade utiliza a relação de classificadores gerada pela atividade anterior, pois a ideia é que eles funcionem em conjunto com os algoritmos de seleção de atributos. Essa opção fez com que este trabalho optasse pela utilização de técnicas de seleção de atributo. Uma técnica de seleção de atributos procura, por meio do uso de algoritmos especializados, encontrar dentre todos os atributos do conjunto de dados, aqueles que conseguem dar as melhores informações para os algoritmos de classificação realizar seu trabalho. Uma alternativa, mas descartada por este trabalho, seria a construção de um algoritmo próprio para realizar a seleção de atributos, conforme proposto por (MENZIES et al., 2007; LIU et al., 2014; ZHANG et al., 2014). O motivo do descarte foi justamente para avaliar o trabalho conjunto da técnica de avaliação de atributos com os classificadores.

Também nesta atividade, a análise de um especialista buscou dentre os tipos de

aplicação da técnica de seleção de atributos as mais utilizadas em trabalhos similares (MALHOTRA, 2015). A opção, conforme a literatura, foi pelo uso conjunto dos algoritmos de seleção de atributos com os classificadores, no formato de empacotamento (GAO; KHOSHGOFTAAR, 2015). Com isso, duas abordagens foram escolhidas: a utilização dos classificadores com algoritmos de seleção de atributo baseada em seleção de subconjunto de atributos e a baseada em *ranking*. A primeira opção procura estabelecer correlações entre atributos, com objetivo de explicar a ocorrência do atributo alvo. Na segunda opção, os atributos são considerados de forma independente e aqueles que melhor explicam a ocorrência do atributo alvo são selecionados para aplicação nos classificadores. Entretanto, para um melhor funcionamento dos classificadores com algoritmos de seleção de atributos, quando a técnica baseada em *ranking* foi utilizada, foi fixado um número máximo de atributos que devem ser considerados na execução pelos classificadores. Essa fixação, geralmente, ocorre mediante análise de um especialista ou por requisitos de projeto, com intuito de otimização dos recursos computacionais. Contudo, para este trabalho, a forma utilizada não foi nenhuma dessas duas e sim com a aplicação da fórmula de $log_2(n)$ para a identificação do total de atributos do conjunto de dados que devem ser utilizados, conforme apresentado por (GAO; KHOSHGOFTAAR, 2015), no qual n é o número de atributos do conjunto de dados. Dessa forma, com $n = 67$, foram escolhidos no máximo 6 atributos que, conforme a técnica, melhor descrevem a relação com o atributo alvo.

Para a opção de subconjunto de atributos os algoritmos selecionados foram: como avaliador de atributos, CFS, com a execução em conjunto com os algoritmos de método de pesquisa *Best First*, *Genetic Search* e *Greedy Stepwise*. Para a opção de *ranking* os algoritmos de avaliação de atributos selecionados, para uso com o algoritmo de método de pesquisa *Ranker*, foram: *Information Gain*, *Gain-Ratio* e *Chi-Square*.

O objetivo da subatividade "2.3 Construir Modelos de Predição" é desenvolver, treinar e testar os modelos de predição. Portanto essa atividade consome o conjunto de dados dos projetos, a lista de classificadores e selecionadores de atributos, gerados pelas atividades anteriores. O processamento desses dados acontece por meio do uso da ferramenta Weka com a análise de um especialista. As informações geradas por essa atividade são a matriz de confusão e os modelos de predição para cada projeto

Nessa atividade foram criados 48 cenários de desenvolvimento de modelos de predição, conforme pode ser visto na Tabela 2. Cada um dos 48 cenários foi aplicado individualmente aos 1283 projetos, com a composição dos modelos de predição. Foram

Tabela 2: Cenários para criação de modelos de predição.

Classificador	Técnica de Seleção de Atributos					
	CFS			Ranker		
	Best First	***Genetic Search***	***Greedy Stepwise***	***Info Gain***	***Gain Ratio***	***Chi Square***
Naive Bayes	Cenário 01	Cenário 09	Cenário 17	Cenário 25	Cenário 33	Cenário 41
Random Forest	Cenário 02	Cenário 10	Cenário 18	Cenário 26	Cenário 34	Cenário 42
Decision Table	Cenário 03	Cenário 11	Cenário 19	Cenário 27	Cenário 35	Cenário 43
Simple Logistic	Cenário 04	Cenário 12	Cenário 20	Cenário 28	Cenário 36	Cenário 44
J48	Cenário 05	Cenário 13	Cenário 21	Cenário 29	Cenário 37	Cenário 45
OneR	Cenário 06	Cenário 14	Cenário 22	Cenário 30	Cenário 38	Cenário 46
Multilayer Perceptron	Cenário 07	Cenário 15	Cenário 23	Cenário 31	Cenário 39	Cenário 47
SMO	Cenário 08	Cenário 16	Cenário 24	Cenário 32	Cenário 40	Cenário 48

Fonte: Elaborado pelo autor.

desenvolvidos, então, 61584 modelos de predição de defeitos. Na coluna "Classificador", da Tabela 2, são exibidos os algoritmos de classificação utilizados. A coluna "Técnica de Seleção de Atributos" mostra os algoritmos de métodos de pesquisa e seus respectivos algoritmos de avaliação de atributos. A ideia dessa atividade foi permitir a avaliação dos classificadores e das técnicas de seleção de atributos nos diversos modelos de predição.

Os 61 584 modelos foram treinados e testados com o uso da técnica de validação cruzada com a opção *10-fold cross-validation* (KOHAVI, 1995). Os valores obtidos para verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos foram coletados, com a formação da matriz de confusão de cada modelo. A matriz de confusão foi base para chegar ao valor de AUC a partir da curva ROC (FAWCETT, 2006).

As amostras presentes na matriz de confusão foram analisadas, conforme teste de normalidade por meio da técnica Anderson-Darling. Isso foi feito com o objetivo de identificar se os valores das amostras apresentam uma distribuição normal ou não. Uma distribuição normal é uma distribuição simétrica à volta da média[3], que é uma medida de localização, e o desvio padrão, que é uma medida de dispersão (SHESKIN, 2007). A importância na identificação da normalidade das amostras é para um correto direcionamento dos testes estatísticos que podem ser aplicados para a análise dos resultados obtidos, com a aplicação da técnica de predição cruzada de defeito proposta por este trabalho.

Os testes que devem ser aplicados para dados que apresentem uma distribuição

[3]Uma distribuição simétrica à volta da média implica que a média, a mediana e a moda são coincidentes.

Tabela 3: Pares de algoritmos para seleção de atributo.

Par	Avaliador de atributos	Métodos de pesquisa
Par 01	CFS	*Best First*
Par 02	CFS	*Genetic Search*
Par 03	CFS	*Greedy Stepwise*
Par 04	*Chi-Square*	*Ranker*
Par 05	*Gain-Ratio*	*Ranker*
Par 06	*Information Gain*	*Ranker*

Fonte: Elaborado pelo autor.

normal são os chamados testes paramétricos, também chamados de teste "t". Para amostras que não apresentem uma distribuição normal, os testes não paramétricos são recomendados. A aplicação do teste Anderson-Darling gerou um valor para p de $2{,}2 \times 10^{-16}$, menor que $0{,}05$, que indica que as amostras não compõem uma distribuição normal. Dessa forma, o teste não paramétrico para amostras independentes Kruskal-Wallis e os testes Scott-Knott e Cohen´s *d*, adequados para dados paramétricos e não paramétricos, foram selecionados para avaliação nos resultados gerados pela técnica proposta por esse trabalho.

Esses testes estatísticos são aplicados na próxima atividade, em que a criação dos modelos de predição passa por uma última etapa representada pela subatividade "2.4 Determinar Selecionador de Atributos". Essa subatividade tem o objetivo de determinar o método de pesquisa e avaliação de atributos, que trabalhem em conjunto, para uso pela técnica de seleção de atributos para reduzir a dimensionalidade do conjunto de dados. Ela recebe como dado de entrada os modelos de predição criados e um especialista utiliza como mecanismo de controle a matriz de confusão gerada na atividade anterior juntamente com testes estatísticos com a utilização das técnicas de Kruskal-Wallis, Scott-Knott e Cohen´s *d*.

O objetivo é identificar qual par de algoritmos deve ser utilizado pela técnica de seleção de atributos. Os pares analisados são exibidos na Tabela 3. Nela, a coluna "Avaliador de atributos", apresenta os algoritmos de avaliação de atributos que foram selecionados. A coluna "Métodos de pesquisa" apresenta os algoritmos de método de pesquisa que foram selecionados e que, pelas características, trabalham em conjunto com o algoritmo de avaliação de atributos.

Afim de averiguar se existe diferença de desempenho dentre os 6 pares formados, o teste estatístico de Kruskal-Wallis foi aplicado aos valores de AUC obtidos a partir do teste dos projetos nos diferentes cenários de criação de modelos de predição. O valor

obtido para p foi de $7{,}837 \times 10^{-6}$, menor que 0,05, que indica haver diferença estatística relevante entre as 6 amostras. Para verificar qual das 6 amostras obteve melhor desempenho, o teste estatístico de Scott-Knott foi aplicado. O teste de Scott-Knott faz uma análise hierárquica de agrupamentos para estatisticamente classificar (do inglês, *ranking*), dentro dos agrupamentos, as técnicas de seleção de atributos. Assim, essa identificação é possível pelo destaque do par vencedor dos demais pela formação de um agrupamento separado dos demais pares.

A Figura 11 exibe os dois agrupamentos formados com a aplicação da técnica. Entretanto, é possível observar que nenhum dos agrupamentos formados contém um único elemento, o que evidenciaria qual seria o par de algoritmos com melhor desempenho. Os agrupamentos formados foram o 1 e o 2. O agrupamento 1 é formado pelos pares: 2, 4, 6 e 5, já o agrupamento 2 é formado pelos pares: 1 e 3. Com base na coluna "Valor AUC Médio" da Tabela 4, que representa o valor médio de AUC obtidos a partir do teste dos projetos nos diferentes cenários de criação de modelos de predição, com cada par de algoritmos de seleção de atributos, é possível identificar que os pares que compõem o agrupamento 1 são os que apresentam o melhor desempenho. Entretanto, por estarem todos no mesmo agrupamento significa que há um empate entre esses quatro pares de algoritmo, haja vista a diferença entre eles ser muito pequena. Contudo, há uma diferença, que mesmo sendo pequena, será utilizada como parâmetro para identificar o par de algoritmos com melhor desempenho. Assim sendo, a comparação dos resultados apresentados na Figura 11 com os valores exibidos na Tabela 4, destaca o par 2, formado pela combinação dos algoritmos CFS+*Genetic Search*, como o que apresentou desempenho superior, tendo valores médios de AUC de 0,5744411, levemente superior aos demais integrantes do mesmo agrupamento.

As demais informações exibidas na Tabela 4 são: referencia ao par de algoritmos de seleção de atributos, representado pela coluna "Par"; algoritmo de avaliação de atributo e algoritmo de método de pesquisa, representados respectivamente nas colunas "Avaliador de Atributos" e "Método de Pesquisa"; e, a informação de em qual agrupamento os pares de algoritmos foi encapsulada, conforme aplicação da técnica Scott-Knott, e que pode ser comparado com a informação exibida na Figura 11. Uma informação importante que pode ser obtida com os dados presentes nas colunas "Valor AUC Min." e "Valor AUC Max." da Tabela 4, é que todos os pares de algoritmos apresentaram valores de AUC mínimo e máximo semelhantes, com valores respectivos de 0 e 1. Isso significa que são valores de AUC significantes e não são *outliers* presentes na amostra.

Tabela 4: Resultados do teste estatístico de Scott-Knott para os 6 pares de algoritmos de seleção de atributos.

Par	Avaliador de Atributos	Método de Pesquisa	Valor AUC Médio	Valor AUC Mín.	Valor AUC Máx.	Agrup.
Par 02	CFS	*Genetic Search*	0,5744411	0,00	1,00	1
Par 04	*Chi-Square*	*Ranker*	0,5713352	0,00	1,00	1
Par 06	*Information Gain*	*Ranker*	0,5708108	0,00	1,00	1
Par 05	*Gain-Ratio*	*Ranker*	0,5694079	0,00	1,00	1
Par 01	CFS	*Best First*	0,5614537	0,00	1,00	2
Par 03	CFS	*Greedy Stepwise*	0,5611120	0,00	1,00	2

Fonte: Elaborado pelo autor.

O teste de Scott-Knott conseguiu dividir os pares de algoritmos em dois grupos, com 4 e 2 elementos conforme exibida na Figura 11, entretanto, a diferença entre eles é bastante pequena, mesmo tendo apontado para uma pequena superioridade do par formado pelos algoritmos CFS e *Genetic Search*, conforme exibido na Tabela 4. Afim de confirmar o par de algoritmos de seleção de atributos que apresentou melhor desempenho, o teste de *effect size* de Cohen´s *d* para amostras independentes foi aplicado e tem seu resultado exibido na Tabela 5. O *effect size* é uma medida quantitativa da força, tamanho, ou magnitude de um fenômeno analisado, importante porque muitas vezes é usada para mediar à importância científica do dado fenômeno. Neste trabalho, o *effect size* foi usado para determinar a magnitude da diferença entre as diferentes técnicas de seleção de atributos. Na Tabela 5, as colunas e as linhas apresentam os resultados dos testes aplicados nos pares de algoritmos. O teste Cohen´s *d* trabalha sempre com duas amostras, uma amostra de tratamento e outra amostra de controle. A amostra de controle é aquela com o que se deseja comparar e a amostra de tratamento é a que está sendo analisada no momento (COHEN, 1977). Os resultados da comparação podem assumir três escalas de valores: $< 0; > 0$; ou, 0. Quando os valores encontrados são 0, significa que não existe diferença entre as amostras; quando os valores são > 0 significa que a amostra de tratamento apresentou valores superiores que a amostra de controle; e, quando os valores são < 0 representa que a amostra de tratamento apresentou valores inferiores que a amostra de controle. Dessa forma, pode-se ver que quanto maior for a diferença, maior é o *effect size* entre as amostras comparadas.

Para representar essa comparação, na Tabela 5, valores negativos estão destacados em negrito. É possível observar que de todos os pares, aquele que apresentou melhor desempenho foi o par formado pelos algoritmos CFS e *Genetic Search*. É possí-

vel visualizar que em todas as comparações tendo como amostra de controle os valores dos outros pares de algoritmos, ele foi o vencedor. Um detalhe a observar é que a linha diagonal composta pela comparação das amostras de controle e tratamento com os mesmos algoritmos apresentam valores zerados, devida comparação das amostras apresentarem valores iguais e, por isso, devem ser descartados. A ordem de desempenho dos pares de algoritmos, segundo o teste estatístico Cohen´s *d* exibido na Tabela 5 é: o par vencedor foi o formado pelos algoritmos CFS e *Genetic Search*, seguido por *Chi-Square* e *Ranker*; *Information Gain* e *Ranker*; *Gain-Ratio* e *Ranker*; CFS e *Best First*; e por último, CFS e *Greedy Stepwise*.

A escala de valores para análise dos resultados segundo o teste estatístico Cohen´s *d* é dividida em: insignificante; pequena; média; e, grande (COHEN, 1977). O resultado ideal, do teste aplicado às amostras, é apresentar diferenças com escala de grandeza média ou grande. A escala de grandeza é dividida em: pequena, quando a grandeza dos valores é $\geq 0,20$ e $< 0,40$; média, quando a grandeza dos valores é $\geq 0,40$ e $< 0,80$; grande, quando a gradeza de valores são $\geq 0,80$; e, insignificantes, quando a grandeza dos valores é $< 0,20$.

Os resultados encontrados com a aplicação do teste Cohen´s *d*, conforme pode ser observado na Tabela 5, mostra que existe uma diferença insignificante entre os valores das amostras. Contudo, tal diferença existe e este trabalho usou essa diferença, mesmo sendo pequena, como parâmetro para identificação do par de algoritmos vencedor, assim como *rankear* os demais pares de algoritmos utilizados. A comparação dos resultados encontrados com a aplicação do teste Cohen´s *d*, com os resultados obtidos com o teste Scott-Knott, mostra consistência no par de algoritmos selecionado como o de melhor desempenho, assim como é possível identificar que a ordem de desempenho dos demais pares foi a mesma, segundo apresentado pelas duas técnicas e que pode ser observado na Tabela 5, na Figura 11 e na Tabela 4. Dessa forma, ambos os testes estatísticos apontam que o par vencedor foi o formado pelos algoritmos CFS e *Genetic Search*, seguido por *Chi-Square* e *Ranker*; *Information Gain* e *Ranker*; *Gain-Ratio* e *Ranker*; CFS e *Best First*; e por último, o par formado pelos algoritmos CFS e *Greedy Stepwise*.

3.5 AGRUPAR PROJETOS POR SIMILARIDADE

O objetivo da atividade “3 Agrupar Projetos por Similaridade” foi agrupar os projetos que são similares segundo o critério proposto neste trabalho. Para isso, os modelos de predição desenvolvidos na atividade anterior são utilizados como dado de

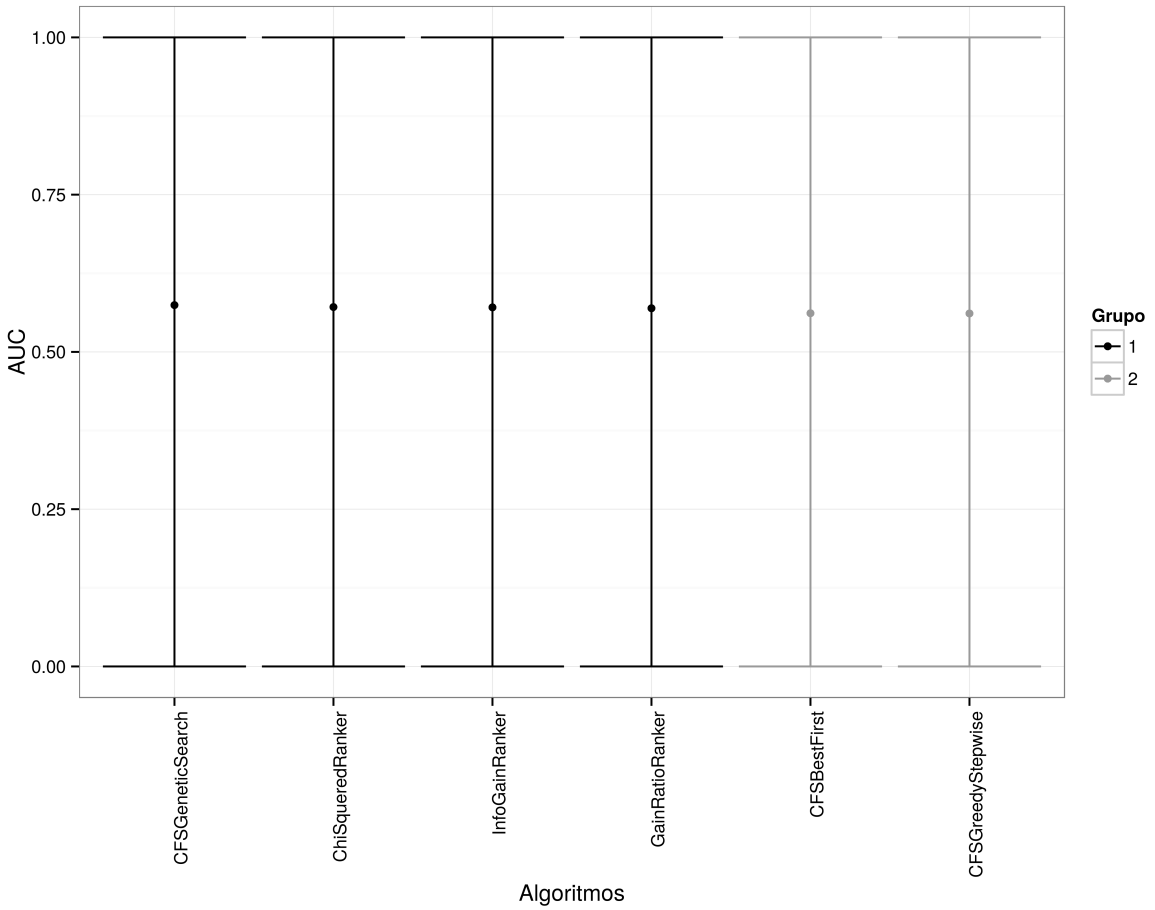

Figura 11: Desempenho dos pares de algoritmos utilizados pela técnica de seleção de atributos.

Fonte: Elaborado pelo autor.

entrada. Um especialista utiliza como mecanismo de controle a matriz de confusão, também obtida na atividade anterior e um algoritmo de clusterização, o BSAS. Os resultados gerados por essa atividade são o conjunto de dados de projetos agrupados e a relação de projetos selecionados para testar cada um dos modelos de predição cruzada na atividade seguinte. Cada agrupamento de projetos criado deu origem a um modelo de predição cruzada de defeitos[4].

Uma das principais medidas utilizadas para identificação da similaridade entre projetos são as métricas. No entanto, este trabalho não procurou estabelecer a relação

[4]O termo “agrupamento” é usado quando se deseja referenciar os projetos e suas instâncias, contidos no conjunto de dados usado, agrupados segundo a aplicação de um método de agrupamento por similaridade. Já o termo “modelo de predição cruzada” é usado quando se deseja referenciar os projetos e suas instâncias já agrupados, juntamente com um algoritmo de classificação e um algoritmo selecionador de atributos. O termo “modelo de predição cruzada” é usado indistintamente para modelos não testados, testados, e testados e avaliados.

Tabela 5: Resultado do teste estatístico de *effect size* de Cohen´s *d* aplicado aos pares de algoritmos de seleção de atributos.

Par de Algoritmos	*CFS + Best First*	*CFS + Genetic Search*	*CFS + Greedy Stepwise*	*Chi-Square + Ranker*	*Gain-Ratio + Ranker*	*Information Gain + Ranker*
CFS + Best First		**-0,067153**	0,001720	**-0,051220**	**-0,041359**	**-0,048511**
CFS + Genetic Search	0,067153		0,068946	0,016574	0,026946	0,019376
CFS + Greedy Stepwise	**-0,001720**	**-0,068946**		**-0,053011**	**-0,043152**	**-0,050301**
Chi-Square + Ranker	0,051220	**-0,016574**	0,053011		0,010345	0,002806
Gain-Ratio + Ranker	0,041359	**-0,026946**	0,043152	**-0,010345**		**-0,007532**
Information Gain + Ranker	0,048511	**-0,019376**	0,050301	**-0,002806**	0,007532	

Fonte: Elaborado pelo autor.

de similaridade entre os projetos por meio das métricas e sim pelos valores de AUC calculados a partir das matrizes de confusão obtidas na atividade anterior. O valor de AUC de cada um dos modelos não foi usado para avaliar individualmente os modelos, como usualmente é feito, ao invés disso, ele foi usado para estabelecer a relação de similaridade entre os projetos para o processo de predição cruzada. Isso foi feito porque se espera obter bons resultados dos modelos de predição cruzada com agrupamento de projetos, mesmo aqueles projetos com valores de AUC considerados ruins. Dessa maneira, a transferência de modelos de predição entre projetos seria beneficiada.

Para que o agrupamento propriamente dito fosse realizado, foi necessário selecionar um algoritmo de clusterização, e o elevado número de modelos de predição foi um fator determinante na escolha. Isso porque, alguns algoritmos de clusterização, como por exemplo o K-means (ALPAYDIN, 2010), precisam de um elemento centróide inicial como forma de inicializar o processo de agrupamento. Diferentes escolhas desse centróide inicial resultam em diferentes agrupamentos o que, consequentemente, pode levar a resultados não satisfatórios. Assim, não é uma tarefa trivial escolher um modelo de predição ótimo, que represente o centróide inicial, tendo em consideração o grande número de modelos de predição obtidos. Portanto, o elevado número de modelos de predição juntamente com a dificuldade de identificação de uma forma de inicializar os agrupamentos levou a escolha de um algoritmo de avaliação sequencial, que pudesse avaliar individualmente cada modelo (THEODORIDIS; KOUTROUMBAS, 2008). Den-

tre esses algoritmos, um dos mais comuns é o BSAS (KAINULAINEN; KAINULAINEN, 2002).

No entanto, o BSAS possui como característica negativa a tendência em gerar um grande número de agrupamentos, inclusive alguns com poucos elementos. Essa tendência pode ser, de certa forma, controlada por dois parâmetros: o primeiro é chamado de medida de similaridade e o segundo de limite de agrupamento. Essa medida de similaridade é responsável por estabelecer se um modelo de predição será adicionado em um agrupamento já existente, ou se será criado um novo para acomodá-lo. A medida de similaridade é calculada levando em conta os parâmetros estabelecidos para definir a similaridade entre os projetos. O cálculo sempre mantém a referência de quem é o modelo de predição centróide, ou seja, aquele cujas informações seriam a mais central dentre os integrantes daquele agrupamento (KAINULAINEN; KAINULAINEN, 2002). Já o limite de agrupamento é um parâmetro de parada para a execução do algoritmo, ele serve como um mecanismo de segurança que estabelece um limite de agrupamentos que podem ser formados, independente do número de elementos que compuseram cada agrupamento. Esse controle sobre o número de agrupamentos que pode ser formado, estabelecidos pela medida de similaridade e o limite de agrupamentos que podem ser formados, é importante tanto para garantir a existência de recursos computacionais, como para facilitar análise de informações dos agrupamentos. Essa última opção foi a mais relevante para este trabalho e fez com que o número máximo de agrupamento fosse limitado em 20. Com a determinação do limite superior de número de agrupamentos, a medida de similaridade foi calculada a partir do valor de AUC, e fixada em $0,034$.

O algoritmo de agrupamento BSAS foi executado especificamente para cada um dos classificadores escolhidos anteriormente. Na Tabela 6 e na Tabela 7, na coluna “Agrup.” são listados os agrupamentos obtidos segundo a lista de classificadores escolhidos para serem avaliados neste trabalho. É possível observar que o número de agrupamentos obtidos varia de 12 a 19. Dessa maneira, o número de agrupamentos e os projetos que compõem cada agrupamento variam de acordo com o classificador, conforme apresentado na Tabela 6. Por exemplo, pode-se observar que com o uso do *Naive Bayes* e o BSAS, foram obtidos 19 agrupamentos. Portanto, a partir desses 19 agrupamentos, foi possível construir 19 modelos de predição cruzada de defeitos entre projetos. Na Tabela 6 e na Tabela 7 pode-se observar que o agrupamento “01”, formado por 6 projetos e 191 instâncias com a utilização de *Naive Bayes*, é diferente do agrupamento “01”, formado por 22 projetos e 991 instâncias com a utilização do *Random*

Forest; esses valores são destacados em negrito nas respectivas tabelas.

Tabela 6: Número de projetos de cada agrupamento formado segundo os diferentes classificadores.

Agrup.	Naive Bayes	Random Forest	Decision Table	Simple Logistic	OneR	Multilayer Perceptron	J48	SMO
01	**6**	**22**	**4**	9	3	18	5	4
02	120	118	215	225	545	139	162	557
03	86	185	109	172	178	199	296	239
04	**314**	68	186	202	51	113	**367**	**150**
05	100	4	26	219	173	129	51	175
06	5	82	45	89	21	31	73	95
07	199	234	218	42	24	86	25	20
08	153	129	87	159	74	249	35	7
09	41	100	34	54	135	60	20	15
10	50	48	80	13	36	24	187	2
11	7	24	38	19	15	23	26	2
12	23	50	109	36	10	25	15	4
13	75	16	51	8	–	103	8	–
14	8	103	16	19	–	30	–	–
15	16	23	6	2	–	39	–	–
16	9	54	43	–	–	–	–	–
17	17	3	–	–	–	–	–	–
18	15	–	–	–	–	–	–	–
19	20	–	–	–	–	–	–	–
Total	1264	1263	1267	1268	1265	1268	1270	1270

Fonte: Elaborado pelo autor.

O segundo produto desta atividade foi a lista de projetos para teste, específicos de cada modelo de predição cruzada. Com a intenção de obter uniformidade nos projetos para teste, foi escolhido para cada modelo de predição cruzada o projeto com o maior número de instâncias. A opção pelo projeto com maior número de instâncias foi baseada na análise da árvore de decisão proposta por Zimmermann *et al.* (ZIMMERMANN et al., 2009), em que aqueles projetos com o maior número de instâncias poderiam ser melhores opções para o processo de predição, por fornecerem mais dados para aprendizado do modelo. Dessa forma, o agrupamento "01", por exemplo, tinha originalmente 7 projetos com o total de 260 instâncias. No entanto, o projeto com o maior número de instâncias chamado `Uclmda`, com 69 instâncias, foi escolhido para ser o projeto de teste desse agrupamento. Assim, resultaram no agrupamento "01", 6 projetos com o total de 191 instâncias.

Ainda sobre a lista de projetos de teste, é possível observar na Tabela 8 que os projetos utilizados para testar cada agrupamento também variaram. Por exemplo, o projeto chamado `Uclmda`, utilizado para testar o agrupamento "01" com o classificador *Naive Bayes* não foi o mesmo utilizado para testar o agrupamento "01" com o uso do

Tabela 7: Número de instâncias dos projetos de cada agrupamento formado segundo os diferentes classificadores.

Agrup.	Naive Bayes	Random Forest	Decision Table	Simple Logistic	OneR	Multilayer Perceptron	J48	SMO
01	**191**	**991**	**54**	230	179	790	84	97
02	14.998	17.398	28.226	32.943	52.470	18.076	15.294	66.949
03	5.653	17.377	6.321	9.531	22.551	21.203	26.670	27.582
04	42.909	2.657	18.031	28.234	4.430	14.337	44.287	16.114
05	5.725	64	626	24.354	17.470	10.420	2.081	6.793
06	80	10.095	5.639	5.892	485	1.193	3.340	8.510
07	29.144	27.247	22.694	5.676	954	3.931	887	460
08	10.957	17.263	4.485	10.240	5.756	34.536	2.073	168
09	1.458	6.942	1.460	1.819	17.753	2.026	501	450
10	2.265	3.003	12.705	308	2.631	688	23.834	24
11	146	1.926	1.345	530	450	812	817	57
12	2.227	2.248	13.005	2.733	496	698	1.066	54
13	3.986	461	1.818	192	–	6.526	518	–
14	152	11.430	461	845	–	1.862	–	–
15	467	563	539	74	–	4.970	–	–
16	219	1.784	1707	–	–	–	–	–
17	495	124	–	–	–	–	–	–
18	805	–	–	–	–	–	–	–
19	547	–	–	–	–	–	–	–
Total	122.424	121.573	119.116	123.601	125.625	122.068	121.452	127.258

Fonte: Elaborado pelo autor.

classificador *Random Forest*, sendo que o projeto utilizado, para esse caso, foi o chamado `Moast`. Ao todo, 60 projetos distintos foram selecionados, conforme cada agrupamento. Isso representa 4,68% de todos os projetos que fazem parte do conjunto de dados. Essa mudança no projeto selecionado para testar se deu devida variação na formação de cada agrupamento, conforme já explicado.

3.6 CRIAR MODELOS DE PREDIÇÃO CRUZADA

O objetivo da atividade “4 Criar Modelos de Predição Cruzada” foi desenvolver treinar e testar os modelos de predição cruzada. Para isso, essa atividade recebe como dados para processamento a relação de projetos de teste e o conjunto de dados dos projetos agrupados, conforme atividade anterior. A execução da atividade foi conduzida por um especialista com o uso da ferramenta Weka. O especialista fez uso de parâmetros de avaliação de desempenho e testes estatísticos, por meio das técnicas de Kruskal-Wallis e Cohen´s *d*, como ferramenta de controle para análise dos resultados gerados. Os resultados são representados pela matriz de confusão com as informações do teste realizado e o teste dos modelos de predição cruzada.

Tabela 8: Projeto de teste utilizado em cada agrupamento segundo cada algoritmo de classificação.

Agrup.	Naive Bayes	Random Forest	Decision Table	Simple Logistic	OneR	Multilayer Perceptron	J48	SMO
01	**uclmda**	**moast**	**ivef-sdk**	**ivef-sdk**	**moast**	**moast**	**ivef-sdk**	**ivef-sdk**
02	lportal	officefloor	brlcad	lportal	lportal	lportal	lportal	lportal
03	massiv	unladen-swallow	log4net	nh3d	wow-qrsk	yale	unladen-swallow	applet2app
04	qtwin	log4net	unicase	qtwin	baadengine	unladen -swallow	jfreereport	wow-qrsk
05	irrlicht	kicq	wxlua	officefloor	yale	mplayer-ce	gotjava	thout
06	jbookshelf	vdsf	gexpert	mb-unit	libmesh	gotjava	baadengine	uwom
07	unladen-swallow	applet2app	moast	wow-qrsk	arm-webradio	impala	qtwin	libopenmetaverse
08	mplayer-ce	lportal	qtwin	flexpay	emeraldemu	applet2app	brlcad	uface
09	redshell	massiv	applet2app	xpontus	applet2app	jvcl	yale	gexpert
10	impala	ivataopenportal	jnode	jphototagger	magicwars	log4net	jgossipforum	zoie
11	fm-classic	smartgwt	unladen-swallow	gexpert	gexpert	gexpert	ossim	moast
12	jasperreports	vxl	wpdev	jbpm	uclmda	redshell	gexpert	gpu
13	gotjava	gexpert	wow-qrsk	tcvp	–	nh3d	moast	–
14	airhead-research	enlightenment	lportal	log4net	–	jasperreports	–	–
15	gexpert	nsis	massiv	gotjava	–	wow-qrsk	–	–
16	jchassis	redshell	gotjava	–	–	–	–	–
17	reaper3d	google-collections	–	–	–	–	–	–
18	moast	–	–	–	–	–	–	–
19	gnome-jabber	–	–	–	–	–	–	–

Fonte: Elaborado pelo autor.

Os modelos de predição foram desenvolvidos a partir dos agrupamentos formados. Muito embora cada um dos projetos tenha sido treinado individualmente na atividade anterior, o treino foi novamente efetuado para cada agrupamento com cada um dos classificadores selecionados: 19 modelos treinados com o uso de *Naive Bayes*; 17 modelos treinados com o uso de *Random Forest*; 16 modelos treinados com o uso de *Decision Table*; 15 modelos treinados com o uso de *Simple Logistic*; 12 modelos treinados com o uso de OneR; 15 modelos treinados com o uso de *Multilayer Perceptron*; 13 modelos treinados com o uso de J48; e 12 modelos treinados com o uso de SMO. O teste foi feito com auxílio da lista de projetos para teste obtido na atividade anterior e exibida na Tabela 8. Os resultados desta atividade foram os modelos de predição cruzada testados, e as matrizes de confusão dos testes.

Por meio do teste foi possível encontrar o classificador que teve o melhor desempenho, com a observação do valor de AUC obtido para cada modelo de predição cruzada testado com um projeto específico pertencente ao agrupamento que deu origem aquele modelo de predição. Na Tabela 9 são apresentados os valores de AUC dos centróides de cada agrupamento juntamente com o teste do projeto selecionado previamente. Os resultados são separados por algoritmo de classificação.

Tabela 9: Valores de AUC dos centróides e do teste de projeto previamente selecionado em cada agrupamento.

Agrup.	Naive Bayes		Random Forest		Decision Table		Simple Logistic		OneR		Multilayer perceptron		J48		SMO	
	AUC Centróide	AUC Teste	AUC Centróide	AUC Teste	AUC Centróide	AUC Teste	AUC Centróide	AUC Teste	AUC Centróide	AUC Teste	AUC Centróide	AUC Teste	AUC Centróide	AUC Teste	AUC Centróide	AUC Teste
01	**0,998**	**0,876**	**0,986**	0,364	**1,000**	0,441	**0,989**	0,085	**0,990**	0,298	**0,989**	**0,955**	**0,997**	0,463	**1,000**	0,413
02	**0,824**	**0,729**	**0,752**	0,583	0,667	0,500	**0,784**	**0,822**	0,483	0,500	**0,803**	0,671	**0,796**	0,475	0,503	0,500
03	0,534	**0,741**	0,582	0,620	0,417	0,500	0,448	*0,570*	0,651	0,500	0,615	0,534	0,504	0,500	0,612	0,588
04	0,675	**0,875**	0,436	0,640	0,570	0,673	**0,707**	**0,928**	**0,747**	0,500	0,666	0,501	0,611	0,576	0,698	**0,733**
05	0,490	0,563	0,140	0,560	0,102	**0,975**	0,632	**0,718**	0,539	0,500	0,556	0,504	0,052	*0,509*	0,464	0,657
06	0,049	***0,933***	0,869	0,441	**0,865**	**0,855**	0,502	0,564	0,363	0,500	0,071	**0,947**	0,395	*0,528*	**0,791**	**0,718**
07	**0,754**	**0,762**	0,695	0,592	0,479	0,500	**0,865**	**0,879**	**0,843**	0,500	0,435	0,518	0,331	0,579	0,400	*0,502*
08	0,587	0,551	**0,801**	0,646	0,243	0,500	0,548	0,615	**0,705**	0,500	**0,724**	0,616	0,261	0,500	**0,917**	0,495
09	0,313	*0,561*	0,489	0,548	0,045	0,500	0,332	0,658	0,595	0,500	0,353	*0,523*	0,000	***1,000***	0,000	***1,000***
10	0,447	*0,579*	0,530	0,527	**0,798**	0,500	0,116	0,500	**0,784**	0,500	0,163	**0,776**	0,692	0,482	0,280	*0,692*
11	0,156	0,618	**0,924**	**0,808**	0,153	0,500	0,000	***1,000***	0,000	***1,000***	0,002	***1,000***	**0,923**	0,456	**0,957**	0,500
12	**0,903**	**0,881**	0,385	0,558	**0,737**	0,500	**0,922**	0,623	**0,914**	**0,744**	0,249	*0,603*	0,197	0,619	0,343	0,650
13	0,390	***0,965***	0,000	***1,000***	0,364	0,500	0,185	0,500	–	–	0,500	0,495	0,143	*0,607*	–	–
14	0,080	0,605	0,635	0,473	0,000	***1,000***	0,253	0,527	–	–	**0,925**	0,643	–	–	–	–
15	0,000	***1,000***	0,213	*0,678*	**0,936**	0,052	0,043	0,636	–	–	**0,878**	**0,921**	–	–	–	–
16	0,116	***0,762***	0,316	0,546	0,308	0,500	–	–	–	–	–	–	–	–	–	–
17	0,259	0,615	**0,948**	**0,806**	–	–	–	–	–	–	–	–	–	–	–	–
18	**0,954**	0,022	–	–	–	–	–	–	–	–	–	–	–	–	–	–
19	0,214	0,639	–	–	–	–	–	–	–	–	–	–	–	–	–	–
MÉDIA	0,460	0,699	0,543	0,611	0,480	0,562	0,488	0,642	0,557	0,545	0,529	0,680	0,454	0,561	0,580	0,621

Fonte: Elaborado pelo autor.

Assim, na Tabela 9, valores de AUC $>= 0,7$ são destacados em negrito, e valores de AUC das colunas "AUC Teste" que são maiores do que seus respectivos valores das colunas "AUC Centróide" são apresentados emoldurados. O valor que foi adotado de AUC $>= 0,7$ teve como base pesquisas anteriores que precisaram estabelecer um parâmetro de comparação para os resultados obtidos com os classificadores (MENZIES et al., 2007; LESSMANN et al., 2008; MALHOTRA, 2015). Esse valor é tido como aceitável para considerar que o resultado obtido com um algoritmo de classificação seja de boa qualidade. Em verdade, Malhotra *et al.* (MALHOTRA, 2015) menciona que os valores de AUC nos trabalhos de predição de defeitos ficam entre $0,7$ e $0,83$ na maior parte dos casos.

A análise dos valores de AUC também permite interpretar quais são os resultados de um classificador tidos como de desempenho ruim. Se por um lado valores de AUC $>= 0,7$ representam resultados obtidos com um algoritmo de classificação, como de boa qualidade, aqueles algoritmos que representam os piores resultados são os que se aproximam de um modelo randômico de classificação (FAWCETT, 2006; MALHOTRA, 2015). Um algoritmo de classificação apresenta resultados tidos como randômicos quando apresentam valores de AUC próximos de $0,5$, mais precisamente entre $0,4$ e $0,6$. Resultados como esse inviabilizam uma interpretação segura das informações, se livre de defeito ou com a presença dele. Esse tipo de resultado se assemelha à escolha aleatória de um código fonte para a realização de testes, com intuito de identificar a existência de defeitos. Situações como essa são as mais custosas em um ambiente de produção, pois provoca a perda da referência sobre o que testar ou deixar de testar.

Ter essa referência é uma das razões de utilização dos modelos de predição de defeito. Contudo, essa referência pode estar nos resultados gerados pelos algoritmos de classificação, mas não apenas para valores de AUC $>= 0,7$. A interpretação das taxas de TP e FP, gerados a partir da Curva ROC, que é a base para que o valor de AUC seja obtido, pode ser feita de uma forma alternativa. Essa forma alternativa é com a análise dos valores de AUC nos agrupamentos formados que apresentam valores consistentes ruins. Para este trabalho, a interpretação de resultados consistentemente ruins vem da avaliação dos resultados dos agrupamentos e os respectivos testes, representado pelas colunas "AUC Centróide" e "AUC Teste" da Tabela 9. Quando os valores de AUC do centróide e do teste apresentam resultados randômicos ou inferiores ao randômico, significa que tanto o agrupamento formado pelos projetos, conforme atividade "2 Criar Modelos de Predição", quanto o processo de predição cruzada realizada, conforme atividade "4 Criar Modelos de Predição Cruzada", tem baixa capacidade preditiva.

Esse comportamento de resultados ruins, do agrupamento e do teste, permite classificar aquele agrupamento como um preditor consistentemente ruim. Dessa forma, esse padrão permite que um classificador tenha suas taxas de FP ou TP consideradas invertidas no processo de predição cruzada, ou seja, se a resposta dada pelo classificador é que um código está livre de defeito, na verdade a interpretação que deve ser feita é justamente o inverso, tal artefato deve ser alvo de inspeção, pois deve apresentar a incidência de defeitos em futuras *releases*. A mesma análise vale para situações em que o classificador apresentar que um determinado código fonte estará com a presença de defeito, sendo assim, não há a necessidade de testá-lo em versões futuras. Com isso, uma interpretação sobre o valor de AUC gerado pelo classificador pode ser modificada, como $1 - AUC$, o que provê a inversão do valor de AUC gerado inicialmente pelo classificador. Isso só é possível de ser feito, pois a priori, sabemos que os resultados que foram gerados, inicialmente, são ruins.

Portanto, este trabalho propõe como alternativa na interpretação dos valores de AUC, obtidos por meio de algoritmos de classificação, que apresentem resultados ruins de forma consistente, a aplicação da inversão da análise dos valores de AUC gerados pelo processo de predição cruzada mediante a seguinte regra (FAWCETT, 2006).

Regra de inversão. Quando os valores de AUC, obtidos com a aplicação da predição cruzada forem consistentemente ruins, ou seja: valor de AUC do centróide do agrupamento for inferior a $0,5$; e, o valor de AUC do teste também for inferior a $0,5$, o valor de AUC do teste deve ser submetido a seguinte regra de inversão: $1 - AUC$.

Com isso, quando os valores de AUC do teste e do centróide, representado pelas colunas "AUC Teste" e "AUC Centróide" da Tabela 9 se enquadraram na regra, o valor do AUC do teste, da coluna "AUC Teste", foi submetido à regra de inversão. Dessa forma, os resultados invertidos são exibidos em itálico na coluna "AUC Teste" da Tabela 9.

Afim de averiguar se existe diferença de desempenho entre os classificadores, o teste estatístico de Kruskal-Wallis foi aplicado aos valores de AUC obtidos a partir do teste dos projetos de cada um dos modelos de predição cruzada, apresentados nas colunas "AUC Teste" da Tabela 9. O valor obtido para p foi de $0,1877 \times 10^{-2}$, menor que $0,05$, que indica que houve diferença estatística relevante entre as amostras.

Tabela 10: Resultado do teste estatístico de *effect size* de Cohen´s *d* entre os classificadores.

Classificadores	Naive Bayes	Random Forest	Decision Table	Simple Logistic	OneR	Multilayer Perceptron	J48	SMO
Naive Bayes	–	0,4333364	0,6098599	0,2586222	0,8286042	0,0882994	0,7112647	0,3911519
Random Forest	**-0,4333364**	–	0,2875031	**-0,1351703**	0,5243947	**-0,3799637**	0,3808100	**-0,0258607**
Decision Table	**-0,6098599**	**-0,2875031**	–	**-0,3546101**	0,1260326	**-0,5646174**	0,0060413	**-0,2906607**
Simple Logistic	**-0,2586222**	0,1351703	0,3546101	–	0,5474923	**-0,1891406**	0,4272618	0,1073219
OneR	**-0,8286042**	**-0,5243947**	**-0,1260326**	**-0,5474923**	–	**-0,8308501**	**-0,1575789**	**-0,5330906**
Multilayer Perceptron	**-0,0882994**	0,3799637	0,5646174	0,1891406	0,8308501	–	0,7068297	0,3398657
J48	**-0,7112647**	**-0,3808100**	**-0,0060413**	**-0,4272618**	0,1575789	**-0,7068297**	–	**-0,3960448**
SMO	**-0,3911519**	0,0258607	0,2906607	**-0,1073219**	0,5330906	**-0,3398657**	0,3960448	–

Fonte: Elaborado pelo autor.

Com o resultado do teste de Kruskal-Wallis, a investigação seguiu para encontrar qual dos classificadores apresentou melhor desempenho. Devido ao fato do teste estatístico Scott-Knott exigir que as amostras tenham a mesma quantidade de elementos, não foi possível sua aplicação para investigar o classificador campeão. Com isso, a solução empregada foi a aplicação do teste de *effect size* de Cohen´s *d* aos valores de AUC exibidos da coluna "AUC Teste" da Tabela 9. Neste trabalho, o *effect size* foi usado para determinar a magnitude da diferença entre os diferentes algoritmos de classificação. O resultado do teste aplicado é exibido na Tabela 10. Para facilitar a comparação, valores negativos são mostrados em negrito. Tendo o algoritmo *Naive Bayes* como amostra de tratamento, na Tabela 10 é possível verificar que ele foi o que apresentou melhor desempenho frente a todos demais classificadores. A comparação de *Naive Bayes* com os demais classificadores, representados em colunas na Tabela 10, permite identificar que ele foi o que tem maior *effect size* comparado com os demais classificadores. A comparação da magnitude, par a par, entre o algoritmo *Naive Bayes* e demais classificadores revela escala de grandeza: muito pequena, insignificante, em comparação com o classificador *Multilayer Perceptron*; pequena, em comparação com os classificadores: *Simple Logistic*; e, SMO; média, em comparação com os classificadores: *Random Forest*; *Decision Table*; e, J48; grande, em comparação com o classificador OneR. Essas informações qualificam *Naive Bayes* como o escolhido para determinar quais modelos de predição cruzada fazem parte da proposta final deste trabalho.

3.7 AVALIAR MODELOS DE PREDIÇÃO CRUZADA

A predição realizada por um modelo necessita ser avaliada e ter seus resultados comparados para ter seu desempenho identificado (OSTRAND; WEYUKER, 2007; BOWES et al., 2012). Assim, o objetivo da atividade "5 Avaliar Modelos de Predição Cruzada" foi avaliar os modelos de predição cruzada testados por meio das medidas de avaliação de desempenho calculadas a partir das matrizes de confusão. Para isso, foram utilizados os dados provenientes de testes dos modelos de predição cruzada testado e da matriz de confusão. Um especialista analisou as informações geradas com o uso de parâmetros de avaliação de desempenho e testes estatísticos com o uso da técnica de Mann-Whitney-Wilcoxon (SHESKIN, 2007) e Cohen´s *d*. Os produtos gerados por essa atividade são os modelos de avaliação cruzada avaliada.

A avaliação dos modelos de predição precisa ser feita por indicadores de desempenho que possam aferir o quão bom foi à predição realizada. Segundo Ostrand e

Weyuker (OSTRAND; WEYUKER, 2007) e Bowees *et al.* (BOWES et al., 2012), as medidas mais utilizadas para este fim são: precisão, sensibilidade, F-Measure e acurácia. Todos esses indicadores podem ser calculados a partir da matriz de confusão, que por si só também pode ser usada para avaliar o desempenho de modelos de predição. O valor de AUC é outra medida de desempenho muito utilizada em trabalhos similares, na verdade, segundo Zhang *et al.* (ZHANG et al., 2014), são poucos os trabalhos que não utilizam o valor obtido de AUC como indicador de desempenho.

Tabela 11: Resultados da execução dos modelos aplicados localmente nos agrupamentos e globalmente.

Agrup.	Instâncias do Cluster	Instâncias do Projeto	AUC Centróide	AUC Predição Local	AUC Predição Global
01	191	69	**0,998**	**0,876**	**0,743**
02	14998	3583	**0,824**	**0,729**	0,510
03	5653	562	0,534	**0,741**	**0,750**
04	42909	2489	0,675	**0,875**	**0,880**
05	5725	224	0,490	0,563	0,572
06	80	32	0,049	**0,933**	0,500
07	29144	2959	**0,754**	**0,762**	0,531
08	10957	628	0,587	0,551	0,552
09	1458	194	0,313	0,561	0,429
10	2265	208	0,447	0,579	0,428
11	146	71	0,156	0,618	0,642
12	2227	503	**0,903**	**0,881**	**0,887**
13	3986	491	0,390	**0,965**	**0,847**
14	152	49	0,080	0,605	0,491
15	467	147	0,000	**1,000**	0,000
16	219	68	0,116	**0,762**	0,227
17	495	108	0,259	0,615	0,666
18	805	258	**0,954**	0,022	0,687
19	547	97	0,214	0,639	0,587

Fonte: Elaborado pelo autor.

Neste trabalho, a avaliação do desempenho dos modelos de predição cruzada foi feita com a confrontação do valor de AUC obtido no teste dos projetos selecionados especificamente para os agrupamentos, o que foi chamado de predição local, com o AUC do teste desses mesmos projetos, mas com um modelo de predição cruzada treinado com todos os outros projetos do conjunto de dados, o que foi chamado de predição global. O resultado das execuções dos modelos de predição com o uso do algoritmo de classificação *Naive Bayes*, é exibido na Tabela 11. Nela é possível observar os valores de AUC de três diferentes execuções de modelos: "AUC Centróide", é o valor de AUC

com as execuções dos modelos de predição nos projetos que deram origem ao agrupamento, conforme já apresentado na Tabela 9; "AUC Predição Local", é o valor de AUC com a execução dos modelos de predição cruzada obtido com o teste dos projetos, previamente selecionados, dentre os respectivos agrupamentos, a predição local, também apresentado na coluna "AUC Teste" da Tabela 9; e, "AUC Predição Global", é o valor de AUC com a execução dos modelos de predição cruzada com o teste dos projetos previamente selecionados dentro os respectivos agrupamentos contra todos os outros projetos do conjunto de dados, a predição global. Na Tabela 11, valores de AUC $>= 0,7$ são destacados em negrito. Valores de AUC das colunas "AUC Predição Local" que são maiores do que seus respectivos valores das colunas "AUC Centróide" são mostrados em molduras. Seguindo a mesma ideia, valores de AUC das colunas "AUC Predição Global" que são maiores do que seus respectivos valores das colunas "AUC Predição Local" são mostrados em molduras também.

Para verificar os resultados dos modelos de predição cruzada, o teste de Mann-Whitney-Wilcoxon foi aplicado nos valores de AUC mostrados na coluna "AUC Predição Local" e "AUC Predição Global" da Tabela 11. O resultado do teste estatístico encontrou para p o valor de $0{,}04395$, menor que $0{,}05$, que mostrou que há diferença estatística relevante entre os resultados. Uma vez encontrada diferença estatística relevante entre as amostras, o teste de *effect size* de Cohen´s d para amostras independentes foi aplicado para determinar a magnitude das diferenças encontradas. O valor obtido foi $-0{,}5616158$, o que representa uma escala de grandeza média, considerado um bom valor pela literatura. Dessa forma, é possível afirmar que além de haver diferença estatística significativa entre as amostras, conclui-se que essa diferença é realmente relevante.

Como complemento a avaliação de desempenho obtida por meio do valor de AUC, este trabalho avaliou o desempenho dos modelos de predição cruzada por meio dos valores obtidos de precisão, sensibilidade e acurácia. A escolha dessas medidas de desempenho foi feita para permitir uma comparação, ainda que simplista com resultados de trabalhos semelhantes, como por exemplo, os trabalhos propostos por Zimmermann *et al.* (ZIMMERMANN et al., 2009) e He *et al.* (HE et al., 2012). Os resultados foram obtidos por meio do teste dos projetos selecionados especificamente para os agrupamentos, o que foi chamado de predição local, e são comparados com um modelo de predição cruzada treinado com todos os outros projetos do conjunto de dados, o que foi chamado de predição global. Os valores encontrados são exibidos na Tabela 12 e Tabela 13. As colunas "Prec. Predição Local", "Sens. Predição Local" e "Acur. Predição Local" da Tabela 12, apresentam respectivamente os valores de precisão, sensibilidade e

acurácia obtidos na predição local com o uso do algoritmo de classificação *Naive Bayes*. Nessa mesma tabela, as colunas "Prec. Predição Global", "Sens. Predição Global" e "Acur. Predição Global" também exibem, respectivamente, os valores de precisão, sensibilidade e acurácia, obtidos na predição global com o uso do algoritmo de classificação *Naive Bayes*. Na Tabela 12, os valores de precisão, sensibilidade e acurácia $> 0,75$ são apresentados em negrito. Quando as três medidas apresentam valor $> 0,75$, seja para predição local ou global, seus valores são também exibidos em moldura. Isso porque segundo Zimmermann *et al.*, essa medida pode indicar a possibilidade de transferência de um modelo de predição para outros projetos.

Assim sendo, a predição realizada localmente nos agrupamentos apresentou 6 conjuntos de valores para precisão, sensibilidade e acurácia $> 0,75$, nos agrupamentos: 1, 4, 11, 13, 18 e 19, que totaliza 31,58% dos agrupamentos formados. Já a predição realizada globalmente teve 5 conjuntos de valores para precisão, sensibilidade e acurácia $> 0,75$, nos agrupamentos: 1, 2, 4, 7 e 16, que totaliza 26,32% dos agrupamentos formados.

Outra indicação de um modelo de predição cruzada poder ser compartilhado com outros projetos, segundo He *et al.* é quando os valores de precisão são $> 0,50$ e sensibilidade $> 0,70$. Afim de validar essa informação a Tabela 13 exibe os valores de precisão e sensibilidade, obtidos da mesma forma como explicado anteriormente. Nela as colunas "Prec. Predição Local" e "Sens. Predição Local", apresentam respectivamente os valores de precisão e sensibilidade obtidos na predição local com o uso do algoritmo de classificação *Naive Bayes*. As colunas "Prec. Predição Global" e "Sens. Predição Global" exibem os valores de precisão e sensibilidade, obtidos na predição global com o uso do algoritmo de classificação *Naive Bayes*. Na Tabela 13, os valores de precisão $> 0,50$ e sensibilidade $> 0,70$ são apresentados em negrito. Quando as duas medidas apresentam esses valores, seja para predição local ou global, eles são também exibidos em moldura.

A predição realizada localmente nos agrupamentos apresentou 8 conjuntos de valores para precisão e sensibilidade conforme os valores de referência estabelecidos por He *et al.*, nos agrupamentos: 1, 4, 7, 11, 12, 13, 18 e 19, que totaliza 42,11% dos agrupamentos formados. Já a predição realizada globalmente teve 6 conjuntos de valores para precisão e sensibilidade conforme os valores de referência estabelecidos por He *et al.*, nos agrupamentos: 1, 2, 4, 7, 12 e 16, que totaliza 31,58% dos agrupamentos formados.

Tabela 12: Resultados da execução dos modelos aplicados localmente nos agrupamentos e globalmente, para os valores de precisão, sensibilidade e acurácia, conforme sugerido por Zimmermann *et al.*.

Agrup.	Prec. Predição Local	Sens. Predição Local	Acur. Predição Local	Prec. Predição Global	Sens. Predição Global	Acur. Predição Global
01	**0,981**	**0,982**	**0,981**	**0,961**	**0,923**	**0,923**
02	**0,941**	0,691	**0,784**	**0,919**	**0,959**	**0,958**
03	**0,993**	0,114	0,193	**0,989**	0,504	0,504
04	**0,998**	**0,781**	**0,874**	**0,995**	**0,921**	**0,921**
05	0,684	0,379	0,384	0,690	0,531	0,531
06	**0,752**	0,156	0,219	**0,863**	0,375	0,375
07	**0,859**	0,737	**0,779**	**0,782**	**0,884**	**0,884**
08	0,205	0,452	0,282	0,532	0,492	0,492
09	**0,905**	0,356	0,491	**0,913**	0,510	0,510
10	**0,753**	0,462	0,552	**0,774**	0,457	0,457
11	**0,917**	**0,958**	**0,937**	**0,929**	0,451	0,451
12	**0,954**	0,738	**0,811**	**0,954**	0,732	0,732
13	**0,996**	**0,998**	**0,997**	**0,998**	0,591	0,591
14	**0,860**	0,673	0,737	**0,826**	0,633	0,633
15	0,000	0,000	0,000	**1,000**	0,381	0,381
16	**0,930**	0,676	**0,783**	**0,935**	**0,779**	**0,779**
17	**0,873**	0,574	0,674	**0,915**	0,491	0,491
18	**0,784**	**0,849**	**0,815**	**0,845**	0,271	0,271
19	**0,900**	**0,948**	**0,923**	**0,910**	0,485	0,485
Média	0,804	0,607	0,643	0,881	0,598	0,598

Fonte: Elaborado pelo autor.

3.8 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os passos dados para a criação, execução e teste pela técnica de predição cruzada de defeito entre projetos, proposto por este trabalho. Nesse capítulo foram apresentadas: as técnicas de pré-processamento aplicadas ao conjunto de dados; as atividades executadas para criação do modelo de predição; as avaliações sobre desempenho dos pares de algoritmos de seleção de atributos utilizados neste trabalho; a avaliação do desempenho dos classificadores junto ao modelo de predição cruzada de defeito entre projetos; a criação dos agrupamentos de projetos pela medida de similaridade, valor de AUC; os testes realizados pelos modelos de predição nos agrupamentos formados; o comparativo de desempenho das predições realizadas nos agrupamentos formados (predição local) frente às predições realizadas sem qualquer tipo de agrupamento (predição global). No próximo capítulo os resultados detalhados

Tabela 13: Resultados da execução dos modelos aplicados localmente nos agrupamentos e globalmente, para os valores de precisão, sensibilidade, conforme sugerido por He *et al.*.

Agrup.	Prec. Predição Local	Sens. Predição Local	Prec. Predição Global	Sens. Predição Global
01	**0,981**	**0,982**	**0,961**	**0,923**
02	**0,941**	0,691	**0,919**	**0,959**
03	**0,993**	0,114	**0,989**	0,504
04	**0,998**	**0,781**	**0,995**	**0,921**
05	**0,684**	0,379	**0,690**	0,531
06	**0,752**	0,156	**0,863**	0,375
07	**0,859**	**0,737**	**0,782**	**0,884**
08	0,205	0,452	**0,532**	0,492
09	**0,905**	0,356	**0,913**	0,510
10	**0,753**	0,462	**0,774**	0,457
11	**0,917**	**0,958**	**0,929**	0,451
12	**0,954**	**0,738**	**0,954**	**0,732**
13	**0,996**	**0,998**	**0,998**	0,591
14	**0,860**	0,673	**0,826**	0,633
15	0,000	0,000	**1,000**	0,381
16	**0,930**	0,676	**0,935**	**0,779**
17	**0,873**	0,574	**0,915**	0,491
18	**0,784**	**0,849**	**0,845**	0,271
19	**0,900**	**0,948**	**0,910**	0,485
Média	0,804	0,607	0,881	0,598

Fonte: Elaborado pelo autor.

serão discutidos e comparados com alguns resultados encontrados por projetos semelhantes.

4 DISCUSSÃO DOS RESULTADOS

4.1 CONSIDERAÇÕES INICIAIS

O presente capítulo tem o objetivo de apresentar os resultados que foram obtidos com a execução da técnica de predição cruzada de defeitos utilizada neste trabalho. Para isso, o capítulo está organizado em duas partes: na Seção 4.2, os principais resultados encontrados com a aplicação da técnica de predição cruzada de defeitos são apresentados; e, na Seção 4.3 é apresentado uma comparação entre os resultados obtidos com a execução deste trabalho e os resultados encontrados por trabalhos semelhantes.

4.2 RESULTADOS ENCONTRADOS

Existem evidências que mostram que é mais vantajoso o processo de predição cruzada quando ela é feita em conjuntos de dados de diferentes projetos agrupados por algum tipo de similaridade (JURECZKO; MADEYSKI, 2010; MENZIES et al., 2011; ZHANG et al., 2014). Em um dos estudos, Menzies *et al.* (MENZIES et al., 2011) relata que os resultados com a predição cruzada entre projetos agrupados por similaridade superam os resultados de predições ocorridos em um ambiente sem qualquer tipo de agrupamento. Dessa maneira, esperava-se que a construção dos modelos de predição cruzada com o uso da similaridade dos valores de AUC, ao invés de por métricas dos projetos, trouxessem alguns benefícios esperados: melhoria da qualidade dos dados do conjunto de dados; eliminação de problemas relacionados ao desbalanceamento de classes da variável dependente (do inglês, *class imbalance*); aumento do poder preditivo dos algoritmos de classificação; e, diminuição do custo de treino dos modelos de predição. A discussão dos resultados é norteada segundo esses benefícios e tem a análise dos resultados encontrados descrita em quatro partes: a Subseção 4.2.1, em que observações sobre os agrupamentos formados são delineadas; a Subseção 4.2.2, em que observações sobre desempenho dos diferentes algoritmos selecionadores de atributos são mostradas; a Subseção 4.2.3, em que observações sobre o desempenho dos dife-

rentes algoritmos de classificação são feitas; e, a Subseção 4.2.4 em que observações sobre o desempenho dos modelos de predição cruzada de defeitos que foram criados com o uso do algoritmo de agrupamento BSAS apoiado pela similaridade da medida de desempenho são descritas.

4.2.1 OBSERVAÇÕES SOBRE OS AGRUPAMENTOS FORMADOS

A respeito dos resultados dos agrupamentos, vale destacar dois pontos importantes que podem ser vistos pela análise dos resultados da Tabela 9: a dispersão dos valores de AUC dos centróides dos agrupamentos; e, o desempenho dos modelos de predição com o uso de *cross validation* nos projetos dentro dos agrupamentos. Independentemente do classificador usado, os valores de AUC variam muito: *Naive Bayes* varia de $0,022$ a $1,000$; *Random Forest* varia de $0,364$ a $1,000$; *Decision Table* varia de $0,052$ a $1,000$; *Simple Logistic* varia de $0,085$ a $1,000$; OneR varia de $0,298$ a $1,000$; *Multilayer Perceptron* varia de $0,495$ a $1,000$; J48 varia de $0,456$ a $1,000$; e, SMO varia de $0,413$ a $1,000$. O desempenho dos modelos de predição dentro dos agrupamentos em termos de valores de AUC apresentou variação de frequência de valores com base no classificador usado, com a apresentação de valores bons e outros não.

A Tabela 14 exibe que: *Naive Bayes* teve 10 valores de AUC $>= 0,7$ nos 19 agrupamentos (52,63%); *Simple Logistic* teve 5 valores de AUC $>= 0,7$ nos 15 agrupamentos (33,33%); *Multilayer Perceptron* teve 5 valores de AUC $>= 0,7$ nos 15 agrupamentos (33,33%); SMO teve 3 valores de AUC $>= 0,7$ nos 12 agrupamentos (25,00%); *Decision Table* teve 3 valores de AUC $>= 0,7$ nos 16 agrupamentos (18,75%); *Random Forest* teve 3 valores de AUC $>= 0,7$ nos 17 agrupamentos (17,65%); OneR teve 5 valores de AUC $>= 0,7$ nos 12 agrupamentos (16,67%); J48 teve 1 valor de AUC $>= 0,7$ nos 13 agrupamentos (7,69%). É possível observar que os classificadores que apresentaram melhor desempenho, foram: *Naive Bayes*; *Simple Logistic* e *Multilayer Perceptron*. Os demais valores representados na Tabela 14 significam: $<= 0,3$, valores obtidos pelos modelos de predição tidos como de muito baixa qualidade preditiva; entre $> 0,3$ e $<= 0,4$, valores também com baixa qualidade de predição que começam a se aproximar de um modelo randômico; entre $> 0,4$ e $< 0,5$ e entre $>= 0,5$ e $< 0,6$, são valores que se aproximam muito e passam por um modelo randômico; entre $>= 0,6$ e $< 0,7$ são valores que se aproximam de um valor considerado de boa qualidade.

A média dos valores de AUC dos testes apresentados na Tabela 9 exibe para todos os classificadores valores acima de um classificador randômico: *Naive Bayes* obteve

Tabela 14: Resultados de valores de AUC dos modelos de predição cruzada.

Classificador	<=0,3	>0,3 e <=0,4	>0,4 e <0,5	>=0,5 e <0,6	>=0,6 e <0,7	>=0,7	% >=0,7	%>0,4 e <0,6
Naive Bayes	1	0	0	4	4	10	52,63%	21,05%
Simple Logistic	1	0	0	5	4	5	33,33%	33,33%
Multilayer Perceptron	0	0	1	5	4	5	33,33%	40,00%
SMO	0	0	2	4	3	3	25,00%	50,00%
Decision Table	1	0	1	10	1	3	18,75%	68,75%
Random Forest	0	1	2	7	4	3	17,65%	52,94%
OneR	1	0	0	9	0	2	16,67%	75,00%
J48	0	0	4	6	2	1	7,69%	76,92%

Fonte: Elaborado pelo autor.

$0,699$; *Multilayer Perceptron* obteve $0,680$; *Simple Logistic* obteve $0,642$; SMO obteve $0,621$; *Random Forest* obteve $0,611$; *Decision Table* obteve $0,562$; J48 obteve $0,561$; e, OneR obteve $0,545$. Contudo, a dispersão dos valores de AUC resultou em agrupamentos com número de projetos e instâncias bastante variados, com amplitude também considerável, conforme pode ser visto nos valores em negrito na Tabela 6 e na Tabela 7.

Ainda levando em consideração os agrupamentos, ao analisar especificamente o classificador *Naive Bayes*, pode-se perceber que os 10 agrupamentos que obtiveram valores de AUC $>= 0,7$ concentram 64,68% dos projetos do conjunto de dados e 81,58% de todas as instâncias que foram distribuídas em todos os 19 agrupamentos formados com esse classificador. Portanto, pode-se concluir que, com o uso dos dados dos próprios projetos para treino e teste, o desempenho dos modelos de predição foi satisfatório.

Voltando a questão da distribuição dos projetos aos modelos de predição cruzada formada, essa amplitude exibida na Tabela 6 e na Tabela 7 fornece outra informação relevante a respeito da questão do custo de treino e teste dos modelos de predição cruzada entre projetos. Com relação a esse assunto, por exemplo, para utilizar os modelos de predição cruzada para predizer um projeto X qualquer, basta apenas treinar o projeto X, conforme feito na atividade "2.3 Construir Modelos de Predição", encontrar o agrupamento adequado em que o projeto X se encaixa e testá-lo com o modelo de predição cruzada referente ao agrupamento já treinado. Por exemplo: se o treino do projeto X posicioná-lo no agrupamento 1, conforme a Tabela 11, é possível perceber, baseado na coluna "Instâncias do cluster", que o projeto X será testado em um conjunto de dados formado por 191 instâncias. Essa abordagem faz com que o custo de treino e teste dos modelos seja reduzidos comparado com o mesmo procedimento, mas com um modelo

de predição cruzada treinado com todos os outros projetos do conjunto de dados, o que é chamado de predição global. Essa constatação é condizente com o reportado por Rahman *et al.* (RAHMAN et al., 2012), que em seu trabalho chegou a conclusão que o custo de treino e teste dos modelos de predição cruzada entre projetos é reduzido com aplicação de técnicas que agrupam projetos por algum tipo de similaridade.

4.2.2 OBSERVAÇÕES SOBRE OS ALGORITMOS SELECIONADORES DE ATRIBUTOS

Este trabalho avaliou o desempenho dos 6 pares de algoritmos que foram selecionados para uso pela técnica de seleção de atributos. Essa avaliação se deu com a aplicação dos testes estatísticos Kruskal-Wallis, Scott-Knott e Cohen´s *d*. Inicialmente o teste estatístico Kruskal-Wallis foi aplicado nas amostras formadas pelos pares de algoritmos, conforme exibido na Tabela 3, com o objetivo de identificar se existe diferença significativa entre elas. O valor obtido para p foi de $7{,}837 \times 10^{-6}$, menor que $0{,}05$, o que indica haver diferença estatística significante entre as amostras. Para identificar qual dos 6 pares é o que apresenta melhor desempenho, os testes estatísticos Scott-Knott e Cohen´s *d* foram aplicados. Apesar das diferenças entre os pares terem sido pequenas, conforme exibido na Figura 11, na Tabela 4 e na Tabela 5, ambos os testes são consistentes e apontaram para o par formado pela combinação dos algoritmos CFS+*Genetic Search*, como o que apresentou melhor desempenho para utilização com a técnica proposta, seguido por: *Chi-Square* e *Ranker*; *Information Gain* e *Ranker*; *Gain-Ratio* e *Ranker*; CFS e *Best First*; e por último, CFS e *Greedy Stepwise*

4.2.3 OBSERVAÇÕES SOBRE OS ALGORITMOS DE CLASSIFICAÇÃO

Este trabalho avaliou o desempenho dos 8 algoritmos de classificação utilizados pela técnica proposta por este trabalho, para predição de defeito entre projetos. Essa avaliação toca no assunto da influência dos classificadores nos resultados dos modelos de predição. A avaliação de desempenho dos modelos feita neste trabalho têm resultados semelhantes aos obtidos por Ghotra *et al.* (GHOTRA et al., 2015), em contraposição aos resultados apresentados por Lessmann *et al.* (LESSMANN et al., 2008), no qual apresentou evidências de que os classificadores não exercem influência significativa nos resultados dos modelos de predição de defeito. O presente trabalho fez uso de alguns dos algoritmos de classificação utilizados por Lessmann *et al.* e Ghotra *et al.*, entretanto, o conjunto de dados utilizado aqui é diferente do utilizado por esses trabalhos. A avaliação feita por este trabalho se deu com a aplicação dos testes

estatísticos Kruskal-Wallis e Cohen´s *d*. A constatação sobre a influência dos classificadores nos resultados dos modelos de predição de defeito, de forma significativa, se deu com a aplicação do teste estatístico de Kruskal-Wallis aos valores de AUC obtidos a partir do teste dos projetos de cada um dos modelos de predição cruzada, apresentados nas colunas "AUC Teste" da Tabela 9. O valor obtido para p foi de $0{,}001\,877$, menor que $0{,}05$, que indica que houve diferença estatística relevante entre as amostras dos diferentes algoritmos de classificação. A investigação seguiu para encontrar qual dos 8 classificadores apresentou melhor desempenho. A solução empregada foi então o teste estatístico *effect size* de Cohen´s *d* junto aos valores de AUC exibidos da coluna "AUC Teste" da Tabela 9. O resultado do teste aplicado é apresentado na Tabela 10. Nela é possível verificar que o classificador *Naive Bayes*, além de apresentar diferença estatística significante entre os resultados gerados por ele frente aos demais classificadores, apresentou melhor desempenho comparado com os demais classificadores, sendo eleito o classificador vencedor.

Também vale ressaltar que os resultados dos diferentes classificadores influenciaram a formação dos agrupamentos, tanto na quantidade de projetos como na distribuição desses projetos nos diferentes grupos, conforme exibido na Tabela 6 e na Tabela 7. Nelas é possível observar que os agrupamentos formados foram diferentes. Por exemplo: o agrupamento "01" variou de 4 a 22 projetos, conforme apresentado na Tabela 6. Nela é possível identificar que o número de instâncias também sofreu variação, de 54 a 991, conforme exibido na Tabela 7. O projeto que foi selecionado para testar o agrupamento formado também sofreu variação, tendo representações distintas, conforme exibido na Tabela 8.

Alguns algoritmos de classificação mostraram certa tendência a produzir predições randômicas, valores de AUC $> 0,4$ e $< 0,6$, conforme pode ser observada na coluna "% >0,4 e <0,6" da Tabela 14. Nela é possível observar que: *Naive Bayes* teve 4 valores de AUC (21,05%); *Simple Logistic* teve 5 valores de AUC (33,33%); *Multilayer Perceptron* teve 6 valores de AUC (40,00%); SMO teve 6 valores de AUC (50,00%); *Random Forest* teve 9 valores de AUC (52,94%); *Decision Table* teve 11 valores de AUC (68,75%); OneR teve 9 valores de AUC (75,00%); e, J48 teve 10 valores de AUC (76,92%). Vale destacar, ainda, que alguns algoritmos apresentaram valores iguais a $0,5$, conforme exibido na Tabela 9, sendo: *Decision Table* com 10 valores de AUC (62,50%) e OneR com 9 valores de AUC (75,00%), ou seja, completamente randômicos.

4.2.4 OBSERVAÇÕES SOBRE O DESEMPENHO DOS MODELOS DE PREDIÇÃO CRUZADA DE DEFEITOS CRIADOS

Este trabalho avaliou o desempenho preditivo dos agrupamentos que foram formados com o uso do algoritmo de agrupamento BSAS, tendo o valor de AUC como parâmetro para similaridade entre projetos. O principal resultado que se pretende obter com essa análise é identificar se existe diferença nos resultados da predição cruzada de defeitos, quando a técnica proposta neste trabalho é utilizada nesses agrupamentos, o que, para este trabalho, é chamado de predição local, em comparação com as predições realizadas fora deles, o que, para este trabalho, é chamado de predição global. A resposta que foi encontrada é positiva e, para detalhar essa conclusão, em que os modelos de predição cruzada propostas obtiveram resultados melhores que o modelo global, alguns pontos mostrados na Tabela 11 e na Tabela 14 são discutidos aqui. Primeiramente são discutidos os resultados obtidos pela predição local em comparação com a predição feita com dados dos próprios projetos, representada pelos valores de AUC dos centróides dos agrupamentos. Em um segundo momento os resultados da predição local e da predição global são confrontados.

Pode-se observar que a maioria dos valores de AUC da predição local (14 valores, 73,68%) foram maiores do que o valor de AUC do centróide. A maioria das diferenças é sensivelmente maior, como nos agrupamentos 6, 11, 13, 14, 15, 16, 17 e 19. Assim, os números indicam que o agrupamento de projetos por AUC melhorou o desempenho da predição local com relação ao centróide do agrupamento. Mesmo nos casos em que AUC do centróide foi maior do que o AUC da predição local, pode-se perceber que a predição local obteve sempre resultados $>= 0,7$. Tal fato é uma evidencia de que boas predições dentro do próprio projeto geram agrupamentos capazes de realizar boas predições locais também.

Os casos com resultados menos satisfatórios da predição local contra o centróide podem ser vistos no agrupamento “15” e no “18”. Esses agrupamentos foram fortemente influenciados pelo desbalanceamento de classes. O desbalanceamento de classes traz sérias consequências sobre o poder preditivo dos algoritmos de classificação, e algumas técnicas para enfrentar o problema são propostas na literatura (KHOSHGOFTAAR; GAO, 2009; KHOSHGOFTAAR et al., 2010). Como pode ser visto os valores de AUC das predições relacionadas ao agrupamento “15” são em todos os casos $0,000$, porque todos os projetos agrupados ali são completamente desbalanceados, por possuírem apenas uma classe de variável dependente (apenas a classe *Buggy*). O agrupamento

"18" também sofre os efeitos dos desbalanceamentos de classes, mas com menor intensidade. O valor do AUC do centróide desse agrupamento teve valor $0,954$ porque a técnica de seleção de atributos (CFS+*Genetic Search*) permitiu ao classificador *Naive Bayes* obter bons resultados durante o processo de *cross-validation*.

Dessa forma, pode-se perceber que o agrupamento pelo valor de AUC também colaborou para evitar o problema de desbalanceamento de classes, uma vez que ocorreram apenas dois agrupamentos que contêm somente 31 projetos compostos de 1272 instâncias (2,416% dos projetos e 1,04% das instâncias do conjunto de dados), muito embora não tenha evitado esse problema por completo.

A comparação dos resultados das predições locais e globais apresentam resultados promissores. A avaliação feita por este trabalho se deu com a aplicação dos testes estatísticos Mann-Whitney-Wilcoxon e Cohen´s *d*. Conforme teste estatístico pode-se comprovar que existe diferença de desempenho do poder preditivo dos modelos de predição. O resultado do teste estatístico Mann-Whitney-Wilcoxon, aplicado nos valores de AUC mostrados na coluna "AUC Predição Local" e "AUC Predição Global" da Tabela 11, foi para p igual $0{,}043\,95$, menor que $0{,}05$, o que mostra haver diferença estatística relevante entre os resultados obtidos com a predição local frente a global. Com intuito de identificar a magnitude das diferenças estatísticas encontradas, o teste de *effect size* de Cohen´s *d* para amostras independentes foi aplicado. O valor obtido foi de $-0{,}561\,615\,8$, o que representa uma escala de grandeza média. Dessa forma, é possível identificar que além de haver diferença significativa entre as amostras, conclui-se que essa diferença é de significância média, o que permite afirmar que sua aplicação supera o modelo de predição global em poder de predição.

4.3 COMPARAÇÃO DOS RESULTADOS OBTIDOS E OS TRABALHOS RELACIONADOS SELECIONADOS

De maneira geral, é difícil de realizar a comparação direta entre os resultados encontrados por diferentes modelos de predição cruzada entre projetos. Essas dificuldades provem da: diferença das técnicas de modelagem que são empregadas; diferença nos parâmetros de desempenho utilizados para avaliação dos modelos; diferença no conjunto de dados. Um dos fatores que exercem maior influência na dificuldade de replicação dos modelos de predição está no conjunto de dados. Por exemplo, segundo alguns autores, características de contexto podem fazer com que um conjunto de dados de um projeto não seja aplicável para treinar outros modelos de predição (KITCHE-

NHAM et al., 2007; HALL et al., 2012). Com tantas diferenças, fica difícil comparar resultados, contudo, desde que respeitadas tais diferenças existentes entre os modelos de predição, que haja entendimento que essas diferenças podem influenciar nos resultados da comparação, que critérios comuns sejam estabelecidos para análise, os resultados obtidos em trabalhos semelhantes podem ter seus resultados comparados.

Dessa forma, o presente trabalho procurou estabelecer alguns parâmetros de comparação com trabalhos semelhantes. O critério estabelecido de comparação foram as medidas de desempenho. Dentre as medidas de desempenho listadas na literatura, as escolhidas foram: valor de AUC; precisão; sensibilidade; acurácia; e, taxa de falso positivo. O motivo de escolha dessas medidas de desempenho foi a disponibilidade dessas informações em trabalhos semelhantes. Com isso, alguns dos trabalhos presentes na Seção 2.4, "Trabalhos relacionados à predição cruzada de defeitos entre projetos de software", tiveram seus resultados comparados com os resultados obtidos com o presente trabalho, sendo apresentados a seguir.

Dentre os trabalhos relacionados exibidos na Seção 2.4, os que apresentam resultados passiveis de comparação, conforme os critérios estabelecidos anteriormente, são: Watanabe *et al.* (WATANABE et al., 2008), Turhan *et al.* (TURHAN et al., 2009), Zhang *et al.* (ZHANG et al., 2014) e Zimmermann *et al.* (ZIMMERMANN et al., 2009). Para que as comparações sejam possíveis, alguns pontos exibidos na Tabela 12 e na Tabela 13 são discutidos aqui.

Os resultados encontrados por Watanabe *et al.* (WATANABE et al., 2008) mostram valores médios de precisão e sensibilidade, encontrados no processo de predição cruzada por um par de projetos, que permite comparação. Apesar do conjunto de dados ser diferente e o número de projetos também ser distinto, pois Watanabe *et al.* comparou a predição cruzada realizada entre 2 projetos, enquanto que o presente trabalho desenvolveu 19 modelos de predição cruzada entre projetos, guardadas essas diferenças, os valores absolutos são possíveis de comparação. Os valores médios de precisão e sensibilidade encontrados por Watanabe *et al.* foram inferiores aos encontrados por este trabalho com a aplicação da predição cruzada, conforme exibido na Tabela 13. Por meio das colunas "Prec. Predição Local" e "Sens. Predição Local", é possível observar que os valores médios encontrados foram de: $0,804$; e, $0,607$, bem acima dos encontrados por Watanabe *et al.* para precisão e sensibilidade que foram: $0,747$; e, $0,499$, respectivamente.

Semelhante ao proposto por Watanabe *et al.*, o conjunto de dados utilizado por

este trabalho é composto por projetos escritos em múltiplas linguagens, sendo elas: C, C++, C#, Java e Pascal. Diferente do trabalho de Watanabe *et al.*, esse trabalho não cria modelos de predição cruzada com apenas 2 projetos – pares –, mas sim com 1283 projetos que foram executados em uma combinação com 48 cenários de modelos de predição, com a geração de 61584 modelos de predição cruzada de defeitos.

Outro trabalho que apresenta indicadores de desempenho, com valores médios, que permite comparação é o proposto por Turhan *et al.* (TURHAN et al., 2009). Da mesma forma que o anterior, apesar do conjunto de dados ser diferente, a taxa de falso positivo, encontrada pela presente proposta, é inferior se comparada com Turhan *et al.*. Este trabalho obteve uma média 48,40% de taxa de falsos positivos, contra uma média de 52,00% de Turhan *et al.*, para os modelos de predição cruzada desenvolvido, o que representa uma taxa de falso positivo 7,49% inferior. Existem diferenças entre este trabalho e o proposto por Turhan *et al.*. O número de projetos e seu contexto são diferentes, os algoritmos de classificação e agrupamento também.

Um dos trabalhos relacionados de maior relevância para a presente pesquisa foi o desenvolvido por Zimmermann *et al.* (ZIMMERMANN et al., 2009). Nesse trabalho há a informação que resultados de precisão, sensibilidade e acurácia $> 0,75$ classificam o modelo como bom e o credencia para replicação a outros projetos. Esses indicadores foram utilizados neste trabalho, para avaliar os 19 modelos de predição cruzada entre projetos que foram criados. As informações obtidas localmente de precisão, sensibilidade e acurácia são representadas, respectivamente, na Tabela 12 pelas colunas: “Prec. Precisão Local”; “Sens. Precisão Local” e “Acur. Precisão Local”. Nelas, os valores $> 0,75$ estão representados em negrito e a combinação dos três valores $> 0,75$ é representada emoldurada. É possível identificar que 31,58% das amostras apresentam valores $> 0,75$ para as três medidas de desempenho. Os agrupamentos 1, 4, 11, 13, 18 e 19 são os agrupamentos que, segundo Zimmermann *et al.*, teriam as melhores chances de ter seus modelos de predição compartilhados. Essa performance de 31,58%, nos 19 modelos de predição cruzada criados, é bem superior aos 3,40% de taxa de sucesso encontrados por Zimmermann *et al.* em seu trabalho.

Uma abordagem semelhante ao proposto por Zimmermann *et al.* no que diz respeito ao estabelecimento de indicadores para identificar a qualidade alcançada pelos modelos de predição cruzada, é apresentada por He *et al.* (HE et al., 2012), contudo, para os valores de precisão e sensibilidade. Modelos de predição com valores de precisão $> 0,50$ e sensibilidade $> 0,70$ estariam mais aptos por terem seus modelos de

predição replicados para outros projetos. Esses valores foram obtidos e são exibidos na Tabela 13. As informações obtidas localmente de precisão e sensibilidade são representadas, respectivamente, pelas colunas: “Prec. Precisão Local” e “Sens. Precisão Local”. Na Tabela 13 os valores precisão $> 0,50$ ou sensibilidade $> 0,70$ estão representados em negrito e a combinação dos dois valores é representada emoldurada. É possível identificar que 42,11% das amostras apresentam os valores sinalizados para as duas medidas de desempenho. Os agrupamentos 1, 4, 7, 11, 12, 13, 18 e 19 são os agrupamentos que, segundo He *et al.*, seriam os com melhores chances de ter seus modelos de predição compartilhados. Essa performance de 42,11% nos 19 modelos de predição cruzada criados, é bem superior aos 0,32% de taxa de sucesso encontrados por He *et al.* (HE et al., 2012).

Dentre os trabalhos relacionados, aquele que apresenta maiores semelhanças com o presente trabalho é o proposto por Zhang *et al.* (ZHANG et al., 2014). Ambos têm em comum o conjunto de dados, se bem que para o presente trabalho, o conjunto de dados sofreu algumas modificações, conforme apresentado na Seção 3.3. Os mesmos indicadores de performance utilizados por este trabalho foi também utilizado por Zhang *et al.*, tendo os indicadores propostos pelos trabalhos de He *et al.* e Zimmermann *et al.* como parâmetro de comparação. Dessa forma, como apresentado anteriormente, esse trabalho apresentou, segundo os critérios estabelecidos por Zimmermann *et al.*, 31,58% de taxa de sucesso nas predições cruzadas realizadas; e, segundo os critérios estabelecidos por He *et al.*, uma taxa de sucesso nas predições cruzadas de 42,11 %, índices superiores os encontrados por Zhang *et al.* para os mesmos critérios, 3% e 14%, respectivamente.

4.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os principais resultados encontrados no presente trabalho. Os agrupamentos formados pela semelhança entre projetos, pela similaridade da medida de desempenho, tiveram seus resultados preditivos comparados com as predições realizadas pela técnica proposta entre projetos sem qualquer tipo de agrupamento por similaridade, o que foi chamado neste trabalho de predição global. Alguns dos resultados encontrados, como medidas de sensibilidade, precisão, acurácia e taxa de falso positivo, foram comparados com os resultados disponibilizados por trabalhos relacionados. No próximo capítulo serão apresentadas as considerações finais do presente trabalho, limitações e sugestões para trabalhos futuros.

5 CONCLUSÃO

5.1 CONSIDERAÇÕES FINAIS

A utilização da predição cruzada de defeitos entre projetos é uma alternativa importante a ser considerada durante o desenvolvimento e manutenção para aumentar a percepção de qualidade do software, especialmente para projetos que estão em fase inicial de desenvolvimento, o que permite reduzir a ocorrência de defeitos em seus produtos. Um dos princípios para sua utilização é a identificação de semelhanças entre projetos. Algoritmos de agrupamento permitem encontrar padrões de similaridade, sendo uma técnica utilizada noutras pesquisas, que utilizam métricas de software para encontrar projetos semelhantes (ZIMMERMANN et al., 2009; JURECZKO; MADEYSKI, 2010; ZHANG et al., 2014).

Os algoritmos de agrupamento são, corriqueiramente, um dos elementos que fazem parte dos modelos de predição de defeito, que também é composto por algoritmos para seleção de métricas, e geralmente, algoritmos de classificação. Como existem estudos com resultados controversos sobre o impacto nos modelos de predição dos algoritmos de seleção de métricas (KHOSHGOFTAAR et al., 2012; GAO; KHOSHGOFTAAR, 2015; MALHOTRA, 2015); algoritmos de classificação (LESSMANN et al., 2008; GHOTRA et al., 2015); e, sobre o sucesso das predições realizadas com a utilização de dados proveniente de outros projetos (KITCHENHAM et al., 2007; TURHAN et al., 2009; ZIMMERMANN et al., 2009; HE et al., 2012), este trabalho propôs um estudo exploratório sobre esses componentes de forma que pudessem ter seus desempenhos avaliados. Para isso, desenvolveu um modelo para predição cruzada de defeito entre projetos com o uso do algoritmo de agrupamento BSAS e uma medida de desempenho para estabelecer e agrupar projetos por similaridade.

O presente trabalho avaliou o desempenho de 6 diferentes pares de algoritmos de seleção de atributos, 8 de classificação e um de agrupamento em dados de 1283 projetos com a construção de 61 584 diferentes modelos de predição cruzada. O de-

sempenho dos algoritmos de seleção de atributos e classificação foram avaliados por diferentes testes estatísticos, que mostraram haver diferenças estatísticas significantes, tanto para os resultados gerados pelos algoritmos de seleção de atributos, quanto para resultados gerados pelos algoritmos de classificação. Dentre os classificadores, *Naive Bayes* foi o que apresentou melhor desempenho, em comparação com os outros 7 algoritmos, já entre os pares de algoritmos para seleção de atributos, o que apresentou melhor desempenho foi o par formado pelo avaliador de atributos CFS e o método de busca *Genetic Search*, em comparação com outros 6 pares.

Considerando os agrupamentos formados pelo algoritmo BSAS, com o uso do valor de AUC como medida de similaridade, os resultados obtidos pelas predições realizadas nesses agrupamentos mostram evidências de terem sido melhores que as predições realizadas sem qualquer agrupamento por similaridade, além de mostrar a diminuição do custo de treino e teste durante o processo de predição. Dessa forma, a presente proposta parece ser promissora e em linhas gerais apresenta três principais contribuições:

Selecionadores de métricas. Um estudo exploratório sobre o uso de diferentes algoritmos selecionadores de métricas e seu uso conjunto com algoritmos de classificação;

Classificadores. Um estudo exploratório sobre o uso de diferentes classificadores e sua influência nos resultados gerados pelos modelos de predição cruzada de defeito entre projetos; e,

Modelo de predição cruzada de defeitos entre projetos. Um estudo sobre a criação de modelos de predição cruzada de defeito entre projetos, com uso de uma medida de desempenho para agrupamento de projetos por similaridade, por meio do algoritmo de agrupamento BSAS. Um modelo de predição cruzada que permite que outros projetos possam se beneficiar para: criação, desenvolvimento e teste da predição cruzada de defeito entre projetos, por meio da identificação do agrupamento de projetos com melhores condições de fornecer um conjunto de dados para treino do projeto alvo.

5.2 LIMITAÇÕES DO TRABALHO

O presente trabalho apresenta limitações discutidas a seguir. Um primeiro ponto a ser avaliado é a qualidade dos dados do conjunto de dados. Conforme apresen-

tado, a qualidade dos dados pode influenciar os resultados obtidos pelos classificadores. Neste trabalho, foi assumido que os dados foram minerados e tratados corretamente pelo trabalho de Zhang *et al.* (ZHANG et al., 2014). Além disso, os resultados aqui mostrados são válidos apenas para o conjunto de dados estudado, novos conjuntos de dados devem ser estudados para melhorar a generalização dos modelos de predição cruzada.

Foram escolhidos 8 classificadores para a construção dos modelos de predição aqui propostos. No entanto, faz-se necessário um estudo mais aprofundado, com mais classificadores, conforme feito por Ghotra *et al.* (GHOTRA et al., 2015) e Lessman *et al.* (LESSMANN et al., 2008), que usaram 22 classificadores diferentes.

De maneira semelhante à escolha dos classificadores, neste trabalho escolheu-se um grupo pequeno de métodos selecionadores de atributos, e apenas um método de agrupamento por similaridade, o BSAS. Faz-se, portanto, necessários mais estudos para entender o impacto dessas técnicas no resultado dos classificadores, no resultado dos agrupamentos e no resultado final dos modelos de predição cruzada.

5.3 TRABALHOS FUTUROS

Atacar o problema de desbalanceamento do conjunto de dados é um trabalho que traria contribuições significativas para a construção dos modelos de predição. Pode-se encontrar na literatura estudos com o objetivo de melhorar o desempenho dos modelos de predição com desbalanceamento no conjunto de dados (FLACH; WU, 2005; KHOSHGOFTAAR et al., 2010; ROKACH, 2010; FANDOS et al., 2013; MAJNIK; BOSNIć, 2013; WANG et al., 2014). No entanto, estudos específicos em modelos de predição cruzada de defeitos não foram encontrados na literatura.

Três outros pontos que se relacionam também podem ser explorados para evoluir este trabalho: algoritmos de classificação; algoritmos selecionadores de atributos; e, algoritmos de agrupamento por similaridade. Conforme citado, estudos são controversos quando se trata do desempenho dos algoritmos de classificação em modelos de predição. A condução de mais estudos, inclusive com experimentos controlados, seria de grande valia. O mesmo pode ser dito a respeito de algoritmos selecionadores de atributos. Estudos controlados sobre o uso dos algoritmos selecionadores de atributos e seu impacto no desempenho dos modelos de predição devem ser conduzidos para aumentar o conhecimento sobre a influência da seleção de atributos nos modelos de

predição. Além disso, tais estudos podem ser úteis para melhorar as técnicas de agrupamento por similaridade, uma vez que encontrar projetos similares é uma tarefa bastante importante em modelos de predição cruzada. Estudos complementares, com diferentes algoritmos de agrupamento por similaridade, devem ser levados à frente com o objetivo de avaliar seu impacto nos resultados dos modelos de predição.

O presente trabalho utilizou uma medida de desempenho para realizar o agrupamento entre projetos, para predição cruzada de defeitos. Os resultados encontrados foram promissores e apontam para o seu uso como opção às métricas, para o processo de predição cruzada entre projetos. Novas medidas de desempenho e, o uso conjunto delas, deve ser explorado para identificar seus impactos nos modelos de predição.

Outro fator que precisa ser mais explorado é a influência das métricas de contexto no processo de predição. Conforme Hall *et al.* (HALL et al., 2012), os elementos de contexto são fatores que podem potencializar o processo de transferência dos modelos de predição entre projetos. Essa verificação também foi feita por Watanabe *et al.* (WATANABE et al., 2008), que buscou verificar se a diferença entre linguagens de programação seriam um limitante para que os modelos de predição fossem compartilhados. Zhang *et al.* (ZHANG et al., 2014) também constatou a influência dos elementos de contexto no processo de predição. Para Zhang *et al.*, quanto mais elementos de contexto são compartilhados, melhores são os resultados do processo de predição, entretanto, Zimmermann *et al.* (ZIMMERMANN et al., 2009) afirma que esses elementos de contexto, por si só, não são suficientes para que haja o compartilhamento dos modelos de predição. Dessa forma, estudos mais abrangentes sobre esse assunto necessitam ser realizados.

REFERÊNCIAS

AHA, D.; KIBLER, D.; ALBERT, M. Instance-based learning algorithms. **Machine Learning**, Kluwer Academic Publishers, v. 6, n. 1, p. 37–66, 1991. ISSN 0885-6125.

ALPAYDIN, E. **Introduction to Machine Learning**. 2nd. ed. [S.l.]: The MIT Press, 2010. ISBN 026201243X, 9780262012430.

ALTIDOR, W.; KHOSHGOFTAAR, T. M.; NAPOLITANO, A. Wrapper-based feature ranking for software engineering metrics. In: **Proceedings of the 2009 International Conference on Machine Learning and Applications**. Washington, DC, USA: IEEE Computer Society, 2009. (ICMLA '09), p. 241–246. ISBN 978-0-7695-3926-3.

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

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

BELL, R. M.; OSTRAND, T. J.; WEYUKER, E. J. Looking for bugs in all the right places. In: **Proceedings of the 2006 International Symposium on Software Testing and Analysis**. New York, NY, USA: ACM, 2006. (ISSTA '06), p. 61–72. ISBN 1-59593-263-1.

BOWES, D.; HALL, T.; GRAY, D. Comparing the performance of fault prediction models which report multiple performance measures: recomputing the confusion matrix. In: WAGNER, S. (Ed.). **PROMISE**. [S.l.]: ACM, 2012. p. 109–118. ISBN 978-1-4503-1241-7.

BRAGA, A. d. P.; LUDERMIR, T.; CARVALHO, A. **Redes Neurais: Teoria e Aplicações**. [S.l.]: LTC-Livros Técnicos e Científicos Editora SA, Rio, 2000.

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

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

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

COVER, T.; HART, P. Nearest neighbor pattern classification. **Information Theory, IEEE Transactions on**, v. 13, n. 1, p. 21–27, January 1967. ISSN 0018-9448.

CRISTIANINI, N.; SHAWE-TAYLOR, J. **An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods**. New York, NY, USA: Cambridge University Press, 2000. ISBN 0-521-78019-5.

DÁMBROS, M.; LANZA, M.; ROBBES, R. An Extensive Comparison of Bug Prediction Approaches. **Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on**, p. 31–41, 2010.

FACELI, K. et al. Inteligência artificial: Uma abordagem de aprendizado de máquina. **Livros Técnicos e Científicos**, p. 381, 2011.

FANDOS, R.; DEBES, C.; ZOUBIR, A. M. Resampling methods for quality assessment of classifier performance and optimal number of features. **Signal Process.**, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 93, n. 11, p. 2956–2968, nov. 2013. ISSN 0165-1684.

FAWCETT, T. An introduction to roc analysis. **Pattern Recogn. Lett.**, Elsevier Science Inc., New York, NY, USA, v. 27, n. 8, p. 861–874, jun. 2006. ISSN 0167-8655.

FLACH, P. A.; WU, S. Repairing concavities in roc curves. In: **Proceedings of the 19th International Joint Conference on Artificial Intelligence**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. (IJCAI'05), p. 702–707.

GAO, K.; KHOSHGOFTAAR, T. 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, B.; MCINTOSH, S.; HASSAN, A. E. Revisiting the impact of classification techniques on the performance of defect prediction models. **37th International Conference on Software Engineering (ICSE 2015)**, 2015.

GRAVES, T. L. et al. Software Change History. **IEEE TRANSACTIONS ON SOFTWARE ENGINEERING**, v. 26, n. 7, p. 653–661, 2000.

GUYON, I.; ELISSEEFF, A. 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.; 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, M. A.; SMITH, L. A. **Practical Feature Subset Selection for Machine Learning**. 1998.

HALL, T. et al. 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.

HALSTEAD, M. H. **Elements of Software Science (Operating and Programming Systems Series)**. New York, NY, USA: Elsevier Science Inc., 1977. ISBN 0444002057.

HANDL, J.; KNOWLES, J.; KELL, D. B. Computational cluster validation in post-genomic data analysis. **Bioinformatics**, Oxford University Press, Oxford, UK, v. 21, n. 15, p. 3201–3212, ago. 2005. ISSN 1367-4803.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. 2nd. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998. ISBN 0132733501.

HE, Z. et al. 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, S. Training data selection for cross-project defect prediction. In: **Proceedings of the 9th International Conference on Predictive Models in Software Engineering**. New York, NY, USA: ACM, 2013. (PROMISE '13), p. 6:1–6:10. ISBN 978-1-4503-2016-0.

HOLTE, R. 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.

INCE, D. **ISO 9001 and software quality assurance**. [S.l.]: McGraw-Hill, Inc., 1994.

JAIN, A. K.; DUBES, R. 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, E. G.; FARIA, J. C.; ALLAMAN, I. B. 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.

JIANG, Y.; CUKIC, B.; MENZIES, T. Fault prediction using early lifecycle data. p. 237–246, Nov 2007. ISSN 1071-9458.

JOHN, G. H.; LANGLEY, P. Estimating continuous distributions in bayesian classifiers. In: **Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995. (UAI'95), p. 338–345. ISBN 1-55860-385-9.

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

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

KASZYCKI, G. Using process metrics to enhance software fault prediction models. **In Tenth international symposium on software reliability engineering**, 1999.

KHOSHGOFTAAR, T. M.; GAO, K. Feature selection with imbalanced data for software defect prediction. In: **Proceedings of the 2009 International Conference on Machine Learning and Applications**. Washington, DC, USA: IEEE Computer Society, 2009. (ICMLA '09), p. 235–240. ISBN 978-0-7695-3926-3.

KHOSHGOFTAAR, T. M.; GAO, K.; NAPOLITANO, A. 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.

KHOSHGOFTAAR, T. M.; SELIYA, N.; DROWN, D. J. Evolutionary data analysis for the class imbalance problem. **Intell. Data Anal.**, IOS Press, Amsterdam, The Netherlands, The Netherlands, v. 14, n. 1, p. 69–88, jan. 2010. ISSN 1088-467X.

KITCHENHAM, B. et al. Cross versus within-company cost estimation studies: A systematic review. **Software Engineering, IEEE Transactions on**, IEEE, v. 33, n. 5, p. 316–329, 2007.

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

KOTSIANTIS, S. B. Supervised machine learning: A review of classification techniques. In: **Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies**. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2007. p. 3–24. ISBN 978-1-58603-780-2.

KRUSKAL, W. H.; WALLIS, W. A. 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. et al. 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**. [S.l.: s.n.], 1995. p. 388–391. ISSN 1082-3409.

LIU, S. et al. FECAR: A feature selection framework for software defect prediction. In: **IEEE 38th Annual Computer Software and Applications Conference, COMPSAC 2014, Vasteras, Sweden, July 21-25, 2014**. [S.l.: s.n.], 2014. p. 426–435.

MAJNIK, M.; BOSNIć, Z. Roc analysis of classifiers in machine learning: A survey. **Intell. Data Anal.**, IOS Press, Amsterdam, The Netherlands, The Netherlands, v. 17, n. 3, p. 531–558, maio 2013. ISSN 1088-467X.

MALHOTRA, R. 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.

MCCABE, T. J. A complexity measure. **IEEE Trans. Softw. Eng.**, IEEE Press, Piscataway, NJ, USA, v. 2, n. 4, p. 308–320, jul. 1976. ISSN 0098-5589.

MENZIES, T. et al. Local vs. global models for effort estimation and defect prediction. In: **Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering**. Washington, DC, USA: IEEE Computer Society, 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.

MICHAELIS. **Moderno Dicionário da Língua Portuguesa**. 2009. Disponível em: <http://michaelis.uol.com.br/moderno/>. Acesso em: 17 de julho de 2015.

MOSER, R.; PEDRYCZ, W.; SUCCI, G. 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**. New York, NY, USA: ACM, 2008. (ICSE '08), p. 181–190. ISBN 978-1-60558-079-1.

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

NAGAPPAN, N.; BALL, T.; ZELLER, A. Mining metrics to predict component failures. ACM, New York, NY, USA, p. 452–461, 2006.

NOVAKOVIC, J.; STRBAC, P.; BULATOVIC, D. 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, T. J.; WEYUKER, E. 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**. New York, NY, USA: ACM, 2007. (SOQUA '07), p. 25–30. ISBN 978-1-59593-724-7.

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

PETERS, F.; MENZIES, T.; MARCUS, A. Better cross company defect prediction. In: **Proceedings of the 10th Working Conference on Mining Software Repositories**. Piscataway, NJ, USA: IEEE Press, 2013. (MSR '13), p. 409–418. ISBN 978-1-4673-2936-1.

PINZGER, M.; NAGAPPAN, N.; MURPHY, B. Can developer-module networks predict failures? ACM, New York, NY, USA, p. 2–12, 2008.

PITT, E.; NAYAK, R. 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**. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2007. (AIDM '07), p. 83–93.

RADJENOVIC, D. et al. Software fault prediction metrics: A systematic literature review. **Information & Software Technology**, v. 55, n. 8, p. 1397–1418, 2013.

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

REFAEILZADEH, P.; TANG, L.; LIU, H. Cross-validation. Springer US, p. 532–538, 2009.

ROBNIK-SIKONJA, M.; KONONENKO, I. Theoretical and empirical analysis of relieff and rrelieff. **Machine Learning**, Kluwer Academic Publishers, v. 53, n. 1-2, p. 23–69, 2003. ISSN 0885-6125.

ROKACH, L. Ensemble-based classifiers. **Artif. Intell. Rev.**, Kluwer Academic Publishers, Norwell, MA, USA, v. 33, n. 1-2, p. 1–39, fev. 2010. ISSN 0269-2821.

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

SATIN, R. F. de P.; WIESE, I. S.; RÉ, R. 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**. [S.l.]: IEEE Press, 2015. (CLEI 2015). To appear.

SHESKIN, D. J. **Handbook of Parametric and Nonparametric Statistical Procedures**. 4. ed. [S.l.]: Chapman & Hall/CRC, 2007. ISBN 1584888148, 9781584888147.

SHIVAJI, S. et al. 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, S.; KOUTROUMBAS, K. **Pattern Recognition, Fourth Edition**. 4th. ed. [S.l.]: Academic Press, 2008. ISBN 1597492728, 9781597492720.

TURHAN, B. et al. 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.

VAPNIK, V. N. (Ed.). **The Nature of Statistical Learning Theory**. [S.l.]: Springer, 1995.

WANG, P. et al. Multiobjective genetic programming for maximizing roc performance. **Neurocomput.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 125, p. 102–118, fev. 2014. ISSN 0925-2312.

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

WEYUKER, E. J.; OSTRAND, T. J.; BELL, R. M. Adapting a fault prediction model to allow widespread usage. In: **Proceedings of the Second International Promise Workshop**. [S.l.: s.n.], 2006.

WIESE, I. S. et al. Social metrics included in prediction models on software engineering: a mapping study. In: **The 10th International Conference on Predictive Models in Software Engineering, PROMISE '14, Torino, Italy, September 17, 2014**. [S.l.: s.n.], 2014. p. 72–81.

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

WOLPERT, D.; MACREADY, W. No free lunch theorems for optimization. **Evolutionary Computation, IEEE Transactions on**, v. 1, n. 1, p. 67–82, Apr 1997. ISSN 1089-778X.

ZHANG, F. et al. Towards building a universal defect prediction model. ACM, New York, NY, USA, p. 182–191, 2014.

ZIMMERMANN, T. et al. Cross-project defect prediction: A large scale experiment on data vs. domain vs. process. ACM, New York, NY, USA, p. 91–100, 2009.