Software Engineering Repositories: Expanding the PROMISE Database

Márcia Lima
marcia.lima@icomp.ufam.edu.br
Universidade Federal do Amazonas
Universidade do Estado do Amazonas
Manaus, AM - Brazil

Victor Valle, Estevão Costa, Fylype Lira
Bruno Gadelha
{vfv2,ecc,fwcl,bruno}@icomp.ufam.edu.br
Universidade Federal do Amazonas
Manaus, AM - Brazil

ABSTRACT

Defining and classifying software requirements are critical tasks for determining software functionality and overall software architecture. In this sense, several types of research are being developed aiming to automate the classification of software requirements through the use of machine learning algorithms. However, the feasibility of such studies runs counter to the existence of a public database that is adequate in terms of quantity and quality of sample requirements. A requirement base widely used in this type of task is the PROMISE. However, the number of requirements is considered low for practical applications involving machine learning. This research presents an expansion of the PROMISE corpus. New software requirements were incorporated, and the resulting dataset was evaluated through the use of well-known machine learning algorithms. We observed some improvement in the performance of these algorithms regarding the identification of some types of software requirements.

CCS CONCEPTS

• **Software and its engineering → Software libraries and repositories**.

KEYWORDS

software repositories, requirements classification, machine learning

ACM Reference Format:
Márcia Lima, Victor Valle, Estevão Costa, Fylype Lira, and Bruno Gadelha. 2019. Software Engineering Repositories: Expanding the PROMISE Database. In *XXXIII Brazilian Symposium on Software Engineering (SBES 2019), September 23–27, 2019, Salvador, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3350768.3350776

1 INTRODUÇÃO

Requisitos de software (RSs) constituem artefatos essenciais para a compreensão, planejamento, desenvolvimento e manutenção de um sistema de software [24]. Requisitos especificam funcionalidades (requisitos funcionais - RFs) e restrições (requisitos não funcionais - RNFs) sob as quais o software deve ser desenvolvido. Logo, suas especificações devem ser determinadas o mais cedo possível durante o processo de desenvolvimento do sistema. Contudo, nem sempre os requisitos de um software são bem conhecidos, delimitados ou definidos antes de iniciado o processo de desenvolvimento deste. Cleland *et al.* [9] destacam que RNFs são muitas vezes descobertos de forma *ad hoc* e relativamente tarde no processo de desenvolvimento do software.

A tarefa de classificação de requisitos de software (SRC - *Software Requirement Classification*) consiste em especificar a categoria a qual um determinado RS pertence. As categorias de um requisito correspondem aos tipos que estes podem assumir, como por exemplo: RF e RNF de disponibilidade, segurança, portabilidade, usabilidade, operacional, escalabilidade, performance, tolerância a falha, aparência, disponibilidade e manutenibilidade [6, 23]. Dentre as diversas aplicações, a tarefa de SRC desempenha papel importante para: (1) planejar o projeto arquitetônico do software [9]; (2) identificar *"early aspects"* que permeiam a etapa de definição de requisitos do sistema, ajudando a produzir um sistema de maior qualidade [4]; (3) evitar grandes mudanças futuras, decorrentes de requisitos mal especificados, que podem gerar sobrecusto no projeto [23]; (4) minimizar o problema de ambiguidade de requisitos, evitando soluções que não atendem às necessidades dos *stakeholders* [9]. Entretanto, a classificação manual dos RSs é uma tarefa demorada e propensa a imprecisões, especialmente em projetos com um grande número de requisitos [23].

Nesse contexto, diversas pesquisas foram desenvolvidas com objetivo de realizar a classificação automática dos RSs através de algoritmos de *Machine Learning* (ML) [1, 6, 9, 23]. Porém, a eficácia de tais esforços esbarra na existência de uma base de dados adequada em termos de quantidade e qualidade dos dados nela contido. Algoritmos de ML possuem seus desempenhos associados à quantidade, representatividade e qualidade dos dados manipulados na determinação do modelo usado na tarefa de classificação [14]. Atualmente, a base pública de RSs PROMISE [1] tem sido utilizada nas pesquisas da área. Esta é composta por um conjunto pré-rotulado de 255 RFs e 370 RNFs, estando estes últimos subclassificados em 11 diferentes tipos de RNFs. O objetivo da PROMISE é incentivar a pesquisa de modelos preditivos aplicáveis à Engenharia de Software (ES). No entanto, pesquisadores destacam que a baixa representatividade dos dados da base PROMISE é um ponto negativo da mesma [1, 23].

O objetivo desta pesquisa é conduzir a expansão da base PROMISE, agregando novas instâncias de RFs e RNFs, de modo a garantir a compatibilidade e qualidade dos novos dados inseridos.

[1] https://terapromise.csc.ncsu.edu/!/#repo/view/head/requirements/nfr

SBES 2019, September 23–27, 2019, Salvador, Brazil

ACM ISBN 978-1-4503-7651-8/19/09…$15.00
https://doi.org/10.1145/3350768.3350776

Além disso, pretende-se disponibilizar para a comunidade científica essa expansão, contribuindo para a condução de pesquisas que utilizem algoritmos de ML no contexto de ES. Foram utilizadas métricas estatísticas na validação da expansão realizada, como teste *Kappa* [10], *Precision* [13] e *Recall* [13]. Os resultados obtidos demonstram que houve melhora no desempenho de algoritmos de ML quanto à classificação de RSs em diferentes níveis de granularidade.

O restante deste artigo está organizado da seguinte forma: a Seção 2 aborda conceitos sobre a área de *machine learning*. A Seção 3 descreve o planejamento da execução do estudo. A Seção 4 apresenta os resultados obtidos. A Seção 5 são apresentadas as discussões referentes aos resultados e as limitações do estudo realizado. Na Seção 6, as considerações finais são apresentadas.

2 *MACHINE LEARNING* NA ENGENHARIA DE SOFTWARE

Machine Learning é um campo de estudo no qual programas de computadores são desenvolvidos com o objetivo de aprender a partir de dados [14]. De acordo com Géron [14], a ML possui destaque em cenários onde: (1) a solução do problema exige grande esforço de ajustes manuais ou ainda diversas derivações de regras; (2) as abordagens tradicionais, que não utilizam arcabouços estatísticos e probabilísticos, não propiciam uma solução ótima para o problema; (3) os ambientes são dinâmicos, isto é, os dados sofrem mudanças frequentes (por exemplo, preços de ações); e (4) é necessário a geração de conhecimento a partir de grande quantidade de dados. Devido à tal abrangência, técnicas de ML estão sendo aplicadas em diversos contextos obtendo resultados promissores, como por exemplo: na saúde, possibilitando a previsão do risco de ocorrência de doenças [7]; no comércio, quando aplicáveis à sistemas de recomendação [8]; em visão computacional, com objetivo de gerar descrições textuais a partir de imagens [18]; na área da segurança, promovendo a detecção de diferentes crimes [5, 31].

Na área de engenharia de software, diversos são os estudos realizados com o intuito de aplicar técnicas de ML ao apoio de tarefas que englobam o gerenciamento e o desenvolvimento de software. Hassan e Xie [16] afirmam que os repositórios de dados originários do processo de desenvolvimento de um software (tais como: bases de código, listas de discussão, históricos de manutenções, relatórios de bugs, *feedback* de usuários) compreendem ricas fontes de dados acerca das quais informações podem ser extraídas e compreendidas, através do uso de técnicas de mineração de dados e ML, a fim de serem utilizadas como fonte de conhecimento para futuros projetos. Diferentes pesquisas propõem o uso de técnicas de ML para automatizar e minimizar os desafios relacionados à classificação de requisitos de software [1, 6, 9, 19, 23]. A ML também está sendo usada no suporte a realização de diversas tarefas que compreendem o processo de desenvolvimento de software, como: identificação e extração de *design rationale* relacionados às decisões tomadas durante o processo de desenvolvimento de software [2, 3]; detecção automática de requisitos de software ambíguos [24]; e, identificação de estilos de arquitetura de software [20].

Neste trabalho, algoritmos de ML serão usados com o intuito de avaliar a nova base de RSs gerada após o processo de expansão da PROMISE original.

2.1 Algoritmos de *Machine Learning*

Os algoritmos de ML podem ser classificados em supervisionados, não supervisionados, semi-supervisionados e algoritmos que realizam a aprendizagem por reforço. Algoritmos supervisionados são aqueles cujos dados de treino devem estar previamente rotulados, em contrapartida, algoritmos não supervisionados são aqueles cujos dados de treino não foram rotulados previamente. Os algoritmos semi-supervisionados são aqueles capazes de lidar com bases de treino parcialmente rotuladas. Algoritmos que implementam a aprendizagem por reforço são aqueles capazes de executar ações de acordo com o *feedback* obtido através de interações com o ambiente em que estão inseridos, objetivando a construção de soluções (modelos) generalizáveis [14].

O problema de classificação é um exemplo típico de tarefa que um algoritmo supervisionado aborda. Esse problema consiste em organizar objetos em categorias pré-definidas [14]. A classificação pode ser distinguida entre: classificação binária e classificação multiclasse. Na classificação binária, objetos analisados são categorizados dentre duas classes possíveis, ao passo que na classificação multiclasse, objetos podem ser classificados dentre um conjunto de mais de duas classes. No contexto deste artigo, tanto a classificação binária quanto a multiclasse serão objetos de estudos para categorizar RSs em diferentes granularidades pré-definidas pela base original PROMISE, conforme seção 2.3.

Os algoritmos de classificação usados neste artigo foram:

- Árvores de decisão (*Decision Tree*)

A Árvore de decisão se enquadra na categoria do aprendizado supervisionado e pode ser usada para resolver problemas de classificação binária e multiclasse. O modelo gerado pelo algoritmo consiste em uma coleção de nós internos e nós folhas, organizados em uma estrutura hierárquica do tipo árvore. Uma árvore de decisão é uma estrutura na qual cada nó representa um recurso (atributo), cada *link* (ramificação) representa uma decisão (regra) e cada folha representa um resultado (classe). A árvore é usada para guiar a tomada de decisão a fim de determinar a classe ou categoria de uma nova instância [21, 25].

- *k*-NN (*k-Nearest Neighbor*)

Trata-se de um classificador que se baseia na distância entre instâncias, considerando valores de proximidade entre dados na realização da classificação. O algoritmo baseia-se no princípio de que dados similares tendem a estar concentrados na mesma região no espaço de dispersão dos dados. A classificação de um dado cuja classe é desconhecida é realizada comparando este exemplar com outros k próximos a ele cujas classes já foram definidas. Dado um novo exemplo a ser classificado, o k-NN calcula a distância entre a nova instância e as outras do conjunto de treino; identifica e seleciona as k instâncias de treino mais similares à nova a ser classificada; e, determinada a categoria da nova instância de acordo com a categoria mais frequente entre as instâncias semelhantes [21, 25].

- *Naïve Bayes*

Trata-se de um classificar estatístico, baseado no Teorema de Bayes [11]. Este algoritmo tomará a decisão sobre a qual classe um novo exemplar de dado deve ser associado por meio de cálculos probabilísticos condicionais, baseado no pressuposto que para um

suposto evento "A" ocorra, o mesmo depende do que ocorra com o evento "B" [25].

- SVM (*Support Vector Machine*)

Trata-se de uma técnica de classificação que possui seus fundamentos na teoria de aprendizagem estatística. No problema de classificação binária, o algoritmo SVM encontra um hiperplano linear de separação ótimo entre as duas classes, ou seja, um hiperplano de margem máxima. A utilização da margem máxima garante que haverá poucas possibilidades de separação dos dados da amostra. Dessa forma, a chance de erros de classificação de novas instâncias é diminuída, mesmo que estas estejam próximas do hiperplano de separação. Quanto menor a margem utilizada, maior a possibilidade de hiperplanos de separação de dados. A margem é a distância entre os pontos de dados, de ambas as classes (vetores de suporte), mais próximos ao hiperplano [21, 25].

2.2 Métricas de avaliação

Métricas de avaliação determinam valores através dos quais desempenho de sistemas podem ser avaliados e comparados. A eficácia da expansão realizada na base de dados original será avaliada de acordo com a eficácia obtida pelos algoritmos de ML ao utilizarem os dados provenientes de ambas as bases na construção da solução (modelo) do problema de classificação de requisitos. Os algoritmos serão avaliados de acordo com as seguintes métricas de medidas quantitativas: precisão, revocação, medida-F e acurácia.

A precisão (*Precision*) é uma medida de qualidade que mede o número total de requisitos classificados corretamente em relação ao número total de requisitos existentes [13]. Ou seja, mensura a proporção de requisitos classificados corretamente pelo algoritmo.

Seja N o conjunto de requisitos de software classificados corretamente por especialistas e R o conjunto de requisitos classificados pelos algoritmos, temos que a precisão avaliará, para cada classe de requisito, a porção de requisitos que foram classificados corretamente. Sua fórmula é:

$$Precision = \frac{|N \cap R|}{|R|} \tag{1}$$

A revocação (*Recall*) é uma medida de completude que mensura, a partir do total de requisitos de uma determinada classe, a fração de requisitos que foram classificados corretamente [13]. Seu valor é obtido pela seguinte fórmula:

$$Recall = \frac{|N \cap R|}{|N|} \tag{2}$$

A faixa de valores que tanto a precisão quanto a revocação podem assumir é o intervalo $[0, 1]$. Uma precisão perfeita, igual a 1, significa que cada elemento categorizado pertence realmente ao critério pesquisado. Em contrapartida, não há informação sobre a completude da classificação, ou seja se todos os elementos da categoria foram identificados. Uma revocação perfeita, de valor 1, significa que todos os elementos que deveriam pertencer a uma determinada categoria foram realmente identificados, em contrapartida não há informação sobre a quantidade de elementos corretamente classificados.

A medida-F (*F-measure*) é a média harmônica obtida através da combinação dos valores de precisão e revocação [28]. Seu resultado também pertence ao intervalo $[0, 1]$.

$$F - measure = \frac{(2 \cdot precision \cdot recall)}{(precision + recall)} \tag{3}$$

A acurácia (*accuracy*) é uma medida quantitativa da eficácia do classificador $\hat{f}$ que fora empregado [12]. Dado um conjunto de dados x com n instâncias, a acurácia equivale a proporção de instâncias classificadas corretamente por $\hat{f}$. Na equação 4, $I(y_i = \hat{f}(x_i))$ vale 1 se uma instância x_i foi classificada corretamente por $\hat{f}$; caso contrário, esse termo vale 0.

$$accuracy(\hat{f}) = \frac{1}{n} \sum_{i=1}^{n} I(y_i = \hat{f}(x_i)) \tag{4}$$

A precisão e a revocação podem ser calculadas para problemas de classificação multiclasse usando uma matriz de confusão. A matriz de confusão ilustra o número de predições corretas e incorretas [12]. Por meio dela têm-se medidas quantitativas das classes em que o algoritmo apresentou mais dificuldades em identificar. As suas linhas representam as classes reais e as colunas representam as classes preditas. Cada elemento da matriz M_c (5) é da forma m_{ij}, tal que i é a i-ésima classe real e j é a j-ésima classe predita. Os elementos pertencentes à diagonal principal da matriz de confusão indicam o número de acertos do classificador em cada classe; os demais elementos, o número de erros.

$$M_c = \begin{bmatrix} m_{11} & m_{12} & m_{13} & \dots & m_{1k} \\ m_{21} & m_{22} & m_{23} & \dots & m_{2k} \\ m_{31} & m_{32} & m_{33} & \dots & m_{3k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{k1} & m_{k2} & m_{k3} & \dots & m_{kk} \end{bmatrix} \tag{5}$$

2.3 PROMISE: base de requisitos de software

O objetivo da base de dados PROMISE é incentivar a criação e melhoria de modelos preditivos de ES através de um catálogo de requisitos de software dos mais variados tipos. Esse repositório tem sido usado em vários trabalhos na ES [1, 6, 9, 23] e constitui-se uma importante base de dados para se abordar o problema de classificação de requisitos de software.

A base de dados PROMISE possui 625 requisitos de software que são chamados instâncias. Cada instância é composta por 3 (três) atributos: *ProjectID*, *RequirementText* e *class*. Cada atributo é descrito a seguir:

- *ProjectID*. Este atributo consiste em um número que identifica unicamente um projeto de onde um requisito foi obtido. Originalmente, a base PROMISE compreende requisitos de 15 projetos de software.
- *RequirementText*. Este atributo consiste no texto descrevendo o requisito de software relativo ao projeto identificado pelo *ProjectID*.
- *Class*. Este atributo consiste no rótulo do *RequirementText*. Este rótulo tem por objetivo classificar o requisito de software que está sendo apresentado. Através dessa classificação, os algoritmos de ML podem aprender o tipo de cada atributo. A PROMISE classifica os requisitos em Requisito Funcional (RF) ou um dos seguintes Requisitos Não Funcionais (RNFs): Usabilidade (US), Tolerância à Falha (FT), Escalabilidade (SC),

Performance (PE), Portabilidade (PO), Operacional (O), Segurança (SE), Legal (L), Aparência (LF, do inglês, *Look and Feel*), Disponibilidade (A, do ingês, *Availability*) e Manutenibilidade (MN).

A Figura 1 apresenta a distribuição dos RNFs da base de dados original PROMISE, aqui referenciada por PROMISE_orig.

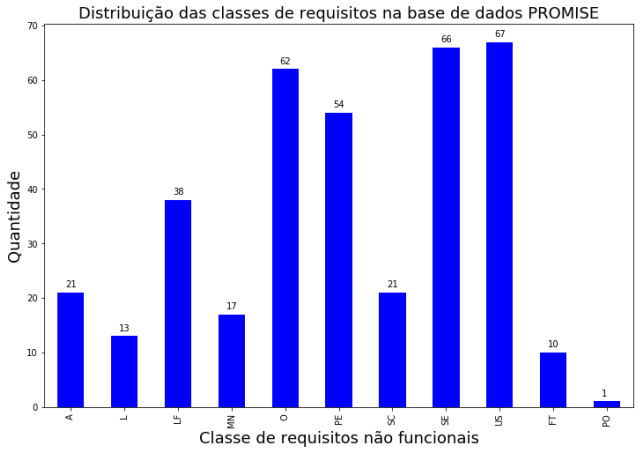

Figura 1: Distribuição das classes de requisitos na base de dados PROMISE

2.4 Trabalhos relacionados

Práticas recomendadas pela IEEE para especificações de requisitos de software [17], destacam a importância da completude e da corretude dos artefatos gerados na fase de elicitação de requisitos de software, pois são eles que definem as funções e as restrições do produto a ser desenvolvido. Os requisitos de software são considerados os alicerces do produto [23], portanto desafios relacionados à elicitação destes são alvos de constantes pesquisas. A tarefa de classificação de requisitos de software em diferentes granularidades constitui alvo de interesse deste artigo, pois algoritmos classificadores oriundos da ML serão utilizados no processo de validação da base de dados aqui proposta.

Existem tentativas de se desenvolver modelos classificadores de Requisitos de Software que exploram diversos recursos da ML. Navarro *et al.* [23] propôs um modelo baseado em uma Rede Neural Convolutiva para classificar RSs. Neste trabalho, a base de dados PROMISE_orig foi usada no treino da geração do modelo. Os autores afirmam que os resultados obtidos foram promissores, porém destacaram a dificuldade em encontrar conjuntos de dados públicos que pudessem ser explorados em pesquisas que aplicam aprendizagem máquina ao domínio da ES.

Abab *et al.* [1] também utilizam a base original PROMISE na construção de modelos classificadores de RSs gerados a partir de algoritmos de aprendizagem de máquina e de técnicas de processamento de linguagem natural (PLN). Os autores propuseram um pré-processamento na base de dados utilizando técnicas de PLN como, por exemplo, rotulação *Part Of Speech* e generalização de expressões temporais. Com isso, observaram uma melhoria nos resultados - em termos de precisão, revocação e acurácia - em relação à classificação que foi feita sem utilizar esse pré-processamento. Os autores citaram o desbalanceamento, a baixa representatividade do RNF de portabilidade e a precisão dos rótulos atribuídos às instâncias da base PROMISE_orig como pontos que ameaçam a validade do modelo proposto.

Cleland-Huang *et al.* [9] propuseram uma ferramenta de classificação de RNFs baseada em palavras-chave que utiliza técnicas de Recuperação de Informação (RI): a NFR-Classifier. Os autores afirmam que o classificador é capaz de identificar e classificar NFRs espalhados em documentos estruturados e não estruturados. Afirmam também que o classificador de RNF apresenta melhor desempenho para os tipos de RNF que possuem um número maior de instâncias no conjunto de treinamento.

A base de dados PROMISE_orig, também foi usada por Casamayor *et al.* [6], para treinar e propor um modelo de classificador de RNFs baseado em aprendizagem semi-supervisionada. Os autores também destacam que bons níveis de precisão no processo de classificação de RNFs estão diretamente relacionados à quantidade de requisitos pré-categorizados, ou seja, está relacionada com a representatividade e balanceamento da base de dados usada para treino. Além disso, os autores observam que o uso de vocabulários distintos, terminologia de domínio e estilos de escrita em diferentes projetos, influenciam na eficácia dos modelos de classificação de requisitos de software escritos em linguagem natural.

O trabalho aqui proposto, tem por objetivo expandir a base de dados PROMISE_orig. Objetivo justificado pelos recorrentes relatos de pesquisas já realizadas, onde a base de dados PROMISE_orig é sempre citada com uma limitação dos estudos [1, 6, 23]. Verificamos não haver iniciativas de expansão da base PROMISE_orig, porém os trabalhos que a utilizam destacam a falta de representatividade e diversidade da base como ponto negativo na pesquisa. A expansão da base será feita manualmente, sua validação será endossada por testes estatísticos e a mesma será avaliada através de métricas que comprovam a eficácia de algoritmos de ML na construção de modelos de classificação de requisitos de software. Como parte da contribuição deste trabalho, a base de dados expandida (PROMISE_exp) será disponibilizada para uso da comunidade científica.

3 METODOLOGIA

A condução da expansão da base de dados pública PROMISE tem por objetivo a geração e disponibilização de uma nova base de requisitos de software rotulada por especialistas. O aumento quantitativo do número de instâncias da base passará por um processo de validação e avaliação, no qual métricas adequadas serão usadas para assegurar a qualidade da base gerada. Na etapa de validação será mensurado o consenso de classificação dos especialistas quanto ao tipo de requisitos de software por eles determinado. Na etapa de avaliação, a expansão executada será avaliada quanto à sua eficácia na geração de modelos de classificadores. Destaca-se, ainda, a importância da manutenção de compatibilidade estrutural com a base original, proporcionando a utilização desta última por todas as pesquisas que já fizeram uso da base original.

Conforme esquema da Figura 2, a construção da PROMISE_exp seguiu as seguintes etapas:

(1) Busca focada na *web* por documentos de Especificação de Requisitos de Software (ERs);
(2) Análise manual dos documentos de ERs retornados;
(3) Extração e classificação manual dos RSs descritos nos documentos;
(4) Validação dos novos RSs identificados;
(5) Geração da nova base de dados expandida, PROMISE_exp;
(6) Avaliação da PROMISE_exp.

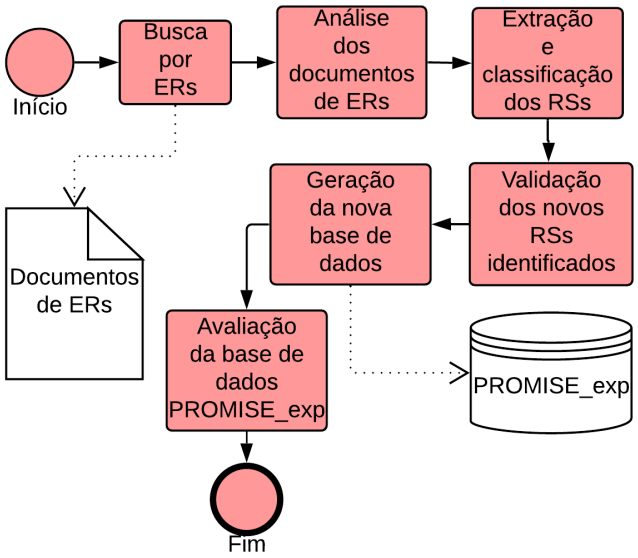

Figura 2: Diagrama das etapas da expansão da base de dados PROMISE

A busca focada por documentos de ERs foi feita através da máquina de busca do Google [2]. Os termos da consulta que compuseram a busca foram *"Software Requirement Specification"*. O objetivo desta etapa era a obtenção de documentos de ERs públicos no idioma Inglês que descrevessem o sistema documentado e contivesse uma lista explicitando os requisitos do software. A lista de documentos de ERs retornados pelo motor da máquina de busca foi explorada a fim de identificar novos requisitos de software. A restrição do idioma se justifica pela necessidade de manutenção de compatibilidade com o idioma da base PROMISE original (PROMISE_orig). Documentos de ERs que não satisfizessem os objetivos desta etapa não foram analisados.

Embora existam recomendações oficiais que norteiam a elaboração de bons documentos de ERs [17], a estrutura, a distribuição do conteúdo e o formalismo utilizado na criação desses documentos podem variar de acordo com o tipo de sistema em desenvolvimento e o processo de desenvolvimento usado [26]. Logo, uma leitura sequencial dos documentos de ERs recuperados pela máquina de busca foi conduzida por dois dos autores deste trabalho que, concomitantemente, realizaram a identificação, extração e classificação dos RSs contidos nos documentos de ERs. Para manter a compatibilidade das classes de requisitos pré-determinadas na PROMISE_orig, a classificação dos novos requisitos extraídos foi baseada na granularidade de RSs já existente, conforme seção 2.3.

O processo de classificação de requisitos extraídos ocorreu de duas formas: (1) baseado em uma classificação já existente nos documentos de ERs analisados, e (2) baseado na classificação do julgamento dos dois pesquisadores que conduziram essa etapa da expansão. As duas formas são mutuamente excludentes. Existindo o registro do tipo de requisito de software no documento de ER, os autores faziam uso desse dado para adicionar rótulos de classificação aos requisitos. A não especificação do tipo do requisito no documento ER exigia que autores a fizessem manualmente.

A classificação dos novos requisitos foi validada através da quantificação de consenso com um terceiro pesquisador. O teste estatístico *Kappa* [10] foi usado para mensurar a concordância de classificação dos requisitos existente entre os pesquisadores. O resultado da etapa de validação é relatado na seção 4.

Em seguida, os RSs identificados foram adicionados à base PROMISE, dando origem à nova base de RSs: PROMISE_exp, disponível em: https://tinyurl.com/PROMISE-exp. A base contém descrição textual de vários RFs e RNFs de software. A execução do processo de expansão foi feita de forma a preservar a compatibilidade dos novos dados com os dados definidos na base original. Sendo assim, o sistema de codificação original, a granularidade na classificação dos requisitos, o idioma e o formalismo na escrita dos requisitos foram mantidos.

Por fim, a PROMISE_exp foi avaliada quanto à sua capacidade de influenciar a criação de modelos de classificadores mais precisos. A PROMISE_exp foi usada nos treinamentos e nos testes de eficácia dos algoritmos de *machine learning* usados para classificar requisitos de software em diferentes granularidades: (1) requisitos funcionais e não funcionais, (2) 11 tipos distintos de requisitos não funcionais, e (3) diferentes tipos de requisitos, onde requisitos funcionais podem estar misturados a diferentes tipos de requisitos não funcionais.

Os desempenhos de quatro algoritmos de ML foram mensurados quanto às suas respectivas capacidades de classificação ao serem treinados com a base de dados expandida. A ferramenta WEKA [3] foi utilizada no apoio da realização dos experimentos conduzidos. WEKA disponibiliza um conjunto de software de *machine learning*, assim como uma coleção de ferramentas de visualização que dão suporte a várias tarefas de ML e mineração de dados, como: pré-processamento de dados, agrupamento de dados (*clustering*), classificação, regressão e seleção de atributos [15].

Como contribuição, uma base de dados numericamente maior contendo novas instâncias de RSs foi gerada. Além disso, houve um incremento de diversidade de contexto dos RSs, pois dados oriundos de diferentes cenários de software foram considerados. Acredita-se que o aumento na quantidade e na diversidade de dados ajudem a comunidade científica a promover estudos mais ricos na área de ML e engenharia de software, pois a base de dados constitui fator primordial no sucesso dos algoritmos de ML [14].

4 RESULTADOS

O gráfico apresentado na Figura 3 mostra a distribuição da expansão quantitativa realizada em cada uma das 11 classes de requisitos

[2] https://www.google.com

[3] https://www.cs.waikato.ac.nz/ ml/weka/

Tabela 1: Comparação da expansão quantitativa.

Tipo de RSs	PROMISE orig	PROMISE exp	Total expandido	Taxa Aumento (%)
RF	255	444	189	74,12
RNF-A	21	31	10	47,62
RNF-L	13	15	2	25,38
RNF-LF	38	49	11	28,95
RNF-MN	17	24	7	41,18
RNF-O	62	77	15	24,19
RNF-PE	54	67	13	24,07
RNF-SC	21	22	1	4,76
RNF-SE	66	125	59	89,39
RNF-US	67	85	18	26,87
RNF-FT	10	18	8	80,00
RNF-PO	1	12	11	1.100,00
Total	625	969	344	55,04

não funcionais manipuladas. Os dados da expansão gerados foram obtidos através de uma busca focada por documentos contendo RSs feita na *web*, conforme descrito na seção 3. A base original teve seu número de instâncias aumentado em aproximadamente 55%, passando de 625 para 969 requisitos. O conjunto de RFs sofreu um acréscimo de 189 instâncias, já o conjunto de RNFs foi expandido em 155 instâncias, representando um aumento de 74% e 42%, respectivamente, conforme consta na Tabela 1.

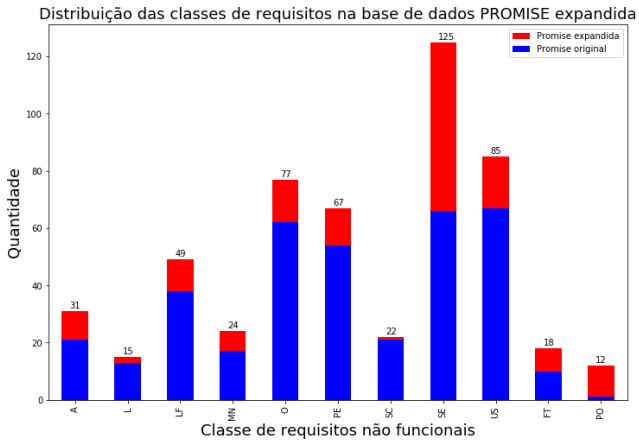

Figura 3: Distribuição das classes de requisitos na base de dados PROMISE expandida

O aumento quantitativo proporcional mais expressivo ocorrido foi no número de instâncias de RNF de portabilidade (RNF-PO), conforme Tabela 1, passando de 1 para 12 instâncias. Além deste, destaca-se um aumento proporcional significativo na amostra nos requisitos não funcionais de segurança (RNF-SE), nos requisitos funcionais (RF) e nos requisitos não funcionais de tolerância à falhas (RNF-FT), aumento de 89%, 74% e 80% respectivamente. Em números reais, a classe de RFs foi a mais contemplada com novas instâncias.

A validação da expansão manualmente feita pelos pesquisadores foi estatisticamente endossada pelo uso do teste estatístico *Kappa* [10]. Dentre as novas instâncias adicionadas à base, requisitos foram selecionados, aleatoriamente, com objetivo de compor uma amostra de validação. Em seguida, um terceiro pesquisador deste trabalho classificou de forma independente os requisitos pertencentes à amostra. Por fim, o valor de *Kappa* foi calculado com objetivo de quantificar a concordância do terceiro pesquisador com a classificação realizada na etapa de classificação manual dos requisitos de software extraídos do documentos de especificação de requisitos. O valor de *Kappa* obtido (*kappa = 0,684*) retrata que houve uma concordância substancial entre os avaliadores[22].

Três outras avaliações foram conduzidas com o objetivo de determinar a qualidade da base de dados expandida: (1) eficácia na classificação binária dos requisitos; (2) eficácia na classificação multiclasse dos requisitos não funcionais; e (3) eficácia na classificação multiclasse de requisitos, incluindo RNFs e RFs. No contexto desses três experimentos foram utilizados os algoritmos de ML descritos na seção 2.1. Como *baseline* utilizou-se a base de dados PROMISE original, ou seja, os valores de eficácia dos algoritmos utilizados foram contrastados quando as bases original (PROMISE_orig) e expandida (PROMISE_exp) foram à eles submetidas. As métricas utilizadas na avaliação dos algoritmos foram: acurácia (eq. 4), *Precision* (eq. 1), *Recall* (eq. 2) e *F-measure* (eq. 3).

A avaliação de desempenho dos algoritmos de ML na classificação binária dos requisitos de software foi conduzida com objetivo de determinar a qualidade da expansão na tarefa de distinção entre RFs e RNFs. O resultado de um algoritmo de classificação binária é um modelo (classificador), que pode ser usado para prever, dentre quais duas classes, como as novas instâncias sem rótulo serão classificadas. Com objetivo de maximizar a eficácia de execução dos algoritmos de ML usados, os dados das bases utilizadas foram submetidos a uma etapa de pré-processamento: eliminação de *stopwords* e aplicação do algoritmo de *Porter* que determina o radical das palavras [29]. O pré-processamento da base de dados é comumente adotado em pesquisas de ML que manipulam dados textuais. Objetivando a exploração exaustiva das bases de dados, optou-se por utilizar a técnica de validação cruzada [14] nos experimentos realizados: as bases foram divididas em dez subconjuntos (10-*folds*), dos quais 9 são usados para treino dos algoritmos – o que corresponde a 90% da base – e 1 usado para teste – correspondente a 10% da base de dados. Para avaliar o modelo proposto, utilizou-se a média das métricas de precisão, revocação e medida-F obtida em cada *fold*.

A Tabela 2, demonstra os resultados de eficácia dos algoritmos de ML ao realizarem a classificação binária dos requisitos existentes nas bases de dados. Este experimento foi conduzido com o objetivo avaliar o uso da base expandida na tarefa de diferenciar RFs de RNFs. Com esta finalidade, ambas as bases foram versionadas para que representassem apenas requisitos funcionais e requisitos não funcionais e suas instâncias foram submetidas ao treinamento e teste dos algoritmos de aprendizagem usados (Árvore de decisão, *k*-NN, Naïve Bayes e SVM).

Os valores de precisão demonstram superioridade discreta na classificação dos requisitos funcionais (RF) no algoritmo Naïve Bayes e SVM. O mesmo valor de precisão foi obtido na execução do algoritmo *k*-NN, significando que, em 2 dos 4 algoritmos analisados, houve um aumento no percentual de RFs classificados corretamente após a expansão da base de dados original. Já na avaliação de precisão da base expandida em relação a classificação de requisitos não funcionais (RNF), o algoritmo SVM demonstrou superioridade.

Tabela 2: Eficácia da classificação binária de requisitos de software.

Classificação Binária						
	PROMISE original			PROMISE expandida		
	Precision	*Recall*	*F-measure*	*Precision*	*Recall*	*F-measure*
Árvore de Decisão						
F	0,78	0,77	0,77	0,76	0,76	0,76
NF	0,84	0,85	0,84	0,80	0,80	0,80
***k*-NN**						
F	*0,75*	0,85	*0,80*	*0,75*	**0,86**	*0,80*
NF	0,89	0,81	0,84	0,86	0,76	0,81
Naïve Bayes						
F	0,76	0,80	0,78	**0,79**	0,79	**0,79**
NF	0,86	0,81	0,84	0,82	**0,82**	0,82
SVM						
F	0,85	0,80	0,83	0,80	**0,86**	**0,84**
NF	0,87	0,90	0,89	**0,90**	0,84	0,86

O que nos permite concluir que, embora haja uma superioridade discreta, a base de dados expandida pode ser usada na tarefa de classificação binária dos requisitos de software. Por possuir mais exemplos de instância, a PROMISE_exp contribui para o aumento de representatividade e diversidade dos dados. A métrica *F-measure* demonstra que houve ganho de eficácia pelos algoritmos Naïve Bayes e SVM ao usarem da base de dados expandida para treino e geração do modelo de classificação binária de requisitos de software.

A Tabela 3 demonstra os resultados da avaliação dos algoritmos de ML usados na tarefa de classificação multiclasse dos RNFs. Neste experimento, é avaliada a eficácia da classificação dos RNFs em 11 classes distintas, conforme rótulos originalmente definidos na PROMISE_orig. O objetivo é identificar o impacto da expansão da base no desempenho da classificação de cada tipo de RNF e avaliar a viabilidade de incorporação das novas instâncias à base de dados original.

Foram verificados que existem diferenças de ganho de acordo com o algoritmo usado e o tipo de RNF avaliado. Analisando valores da métrica *F-measure*, percebe-se que nos 4 algoritmos analisados a classificação dos requisitos RNF-PO e RNF-FT obteve melhora significativa quando utilizada a base de dados expandida para treino dos classificadores. O melhor valor de *F-measure* alcançado foi de 0, 38 para o RNF-PO e 0, 48 para o RNF-FT. Consequentemente, os valores de *precision* e *recall* de ambos os RNFs obtiveram melhoras se comparados as valores de *precision* e *recall* obtidos pelo uso da base original. Fato que está diretamente associado à quantidade de novos requisitos destas classes específicas que foram adicionados à base. As duas classes de requisitos citadas foram as que mais aumentaram em quantidade de instâncias.

Analisando os valores de *F-measure* da classe RNF-MN, observa-se que dos 4 algoritmos, utilizados para teste de validação, 3 demonstraram valores de melhora na classificação enquanto o algoritmo *k*-NN manteve o mesmo comportamento (*F-measure = 0*) quando treinado a partir de ambas as bases, não sendo este último capaz de classificar corretamente nenhuma requisito do tipo RNF-MN. O percentual de expansão do requisito RNF-MN corresponde a 41,18%.

O classificador Árvore de Decisão obteve resultados superiores na classificação de requisitos não funcionais do tipo L, SC e SE. Tal fato possibilita observar que as novas instâncias de exemplo desses requisitos podem ser utilizados por outros algoritmos que possuem características semelhantes à Árvore de Decisão.

As novas instâncias de RNF-SE, também contribuíram para a melhora da determinação deste tipo de requisito. Analisando os valores de *precision*, *recall* e *F-measure*, observa-se que os valores de *recall* da classe fez com que a media harmônica *F-measure* melhorasse. Isso significa que a fração de requisitos do tipo RNF-SE que foram classificados corretamente aumentou e os algoritmos foram capaz de melhor distinguir instâncias da classe RNF-SE.

Através das medidas de avaliação do requisito do tipo operacional (RNF-O), observa-se que a inserção de novas instâncias não trouxe melhora de desempenho a nenhum dos algoritmos utilizados. Acredita-se que o número de instâncias adicionadas não tenha sido representativa a ponto de proporcionar melhorias na construção dos modelos de classificação avaliados.

A Tabela 4 sumariza os valores de desempenho dos algoritmos de ML quando as bases PROMISE_orig e PROMISE_exp são usadas para treino do modelo de classificação. Neste experimento, os algoritmos são avaliados quanto às suas respectivas capacidade de classificação de RSs em 12 diferentes classes, conforme especificação original da PROMISE. A análise dos valores de *precision*, *recall* e *F-measure* nos permite afirmar que, neste contexto, (1) todos os modelos gerados pelos classificadores utilizando a base PROMISE_exp para treino, foram capazes de diferenciar RF das outras categorias de requisitos existentes; (2) houve ganho no desempenho em 3 dos 4 classificadores usados para identificar RNF-L; (3) as taxas de acerto de classificação dos requisitos não funcionais do tipo PO foram melhoradas em todos os algoritmos; (4) o algoritmo *k*-NN demonstrou desempenho superior na classificação de 7 dos 12 tipos de RSs utilizados ao utilizar a base PROMISE_exp como base de treino.

Com objetivo de avaliar o total geral de acertos dos algoritmos treinados com a base de dados PROMISE_exp, foi calculado o valor de acurácia de cada modelo gerado conforme Seção 2.2. Dentre os 4 modelos avaliados o classificador SVM obteve maior valor de acurácia: 0,73, significando que, dos modelos avaliados, foi o que mais acertou no total de instâncias classificadas. A árvore de decisão obteve valor de acurácia 0,63, o *k*-NN obteve 0,64 e o Naïve Bayes obteve 0,70. A tabela apresentada na Figura 4 corresponde à matriz de confusão gerada pelo algoritmo SVM ao utilizar a PROMISE_exp como base de treino. A matriz de confusão é útil para que se perceba quais classes de requisitos o modelo gerado melhor distingue e quais classes o modelo encontra dificuldades quando o mesmo tenta identificá-la. A matriz de confusão da Figura 4 demonstra que apesar do modelo diferenciar requisitos funcionais dos demais, este ainda é bastante confundido com os RNF-SE. Já os diferentes tipos de RNFs são confundidos, pelo modelo, com os RF. O desbalanceamento da base pode ser considerado como fator decisivo na confusão gerada pelo modelo. Bases desbalanceadas são caracterizadas pela desproporção de representatividades das instâncias de cada classe.

Os resultados aqui apresentados demonstram que os ganhos obtidos com a utilização da base PROMISE_exp, na tarefa de SRC, foram identificados em classes independentes de requisitos de software, destacando a necessidade de progressivas expansões da base PROMISE.

Tabela 3: Eficácia da classificação multiclasse de requisitos não funcionais de software.

Classificação RNFs em 11 granularidades												
	Árvore de Decisão						**Naïve Bayes**					
	PROMISE original			PROMISE expandida			PROMISE original			PROMISE expandida		
	Precision	*Recall*	*F-measure*	*Precision*	*Recall*	*F-measure*	*Precision*	*Recall*	*F-measure*	*Precision*	*Recall*	*F-measure*
A	0,63	0,71	0,67	0,58	0,71	0,64	0,69	0,86	0,77	0,68	0,74	0,71
L	0,47	0,54	0,50	**0,56**	**0,60**	**0,58**	0,88	0,54	0,67	0,82	**0,60**	**0,69**
LF	0,50	0,34	0,41	0,45	0,29	0,35	0,69	0,82	0,75	0,67	0,73	0,70
MN	0,29	0,29	0,29	**0,47**	*0,29*	**0,36**	0,75	0,53	0,62	0,71	**0,63**	**0,67**
O	0,63	0,65	0,63	0,50	0,43	0,46	0,62	0,69	0,66	0,55	0,56	0,55
PE	0,84	0,69	0,76	0,68	0,61	0,65	0,73	0,76	0,75	0,72	0,76	0,74
SC	0,36	0,38	0,37	**0,44**	**0,50**	**0,47**	0,59	0,62	0,60	0,57	0,55	0,56
SE	0,58	0,53	0,56	0,47	**0,78**	**0,59**	0,78	0,74	0,76	0,76	**0,76**	0,76
US	0,46	0,66	0,54	**0,62**	0,34	0,44	0,70	0,70	0,70	0,61	0,65	0,63
FT	0,00	0,00	0,00	**0,50**	**0,33**	**0,40**	0,50	0,10	0,17	**0,70**	0,39	**0,40**
PO	0,00	0,00	0,00	**0,50**	**0,25**	**0,19**	0,00	0,00	0,00	**0,30**	**0,25**	**0,19**
	k-**NN**						**SVM**					
A	0,27	0,81	0,40	0,24	**0,90**	0,38	0,81	0,81	0,81	0,71	*0,81*	0,76
L	0,37	0,54	0,44	0,35	0,47	0,40	0,82	0,69	0,75	0,69	0,60	0,64
LF	0,47	0,61	0,53	**0,50**	0,49	0,49	0,72	0,74	0,73	0,67	0,69	0,68
MN	0,00	0,00	0,00	0,00	0,00	0,00	0,75	0,35	0,48	*0,75*	**0,50**	**0,60**
O	0,61	0,69	0,65	0,57	0,52	0,54	0,61	0,82	0,70	0,57	0,62	0,60
PE	0,76	0,65	0,70	**0,83**	0,52	0,64	0,77	0,76	0,77	0,76	0,75	0,75
SC	0,41	0,57	0,48	0,38	0,55	0,44	0,60	0,57	0,59	0,48	0,45	0,47
SE	0,81	0,58	0,67	0,73	**0,64**	**0,68**	0,77	0,82	0,79	0,73	0,79	0,76
US	0,83	0,49	0,62	0,64	**0,52**	0,57	0,79	0,73	0,76	0,65	0,68	0,67
FT	1,00	0,20	0,33	0,80	**0,22**	**0,40**	1,00	0,30	0,46	0,86	**0,33**	**0,48**
PO	0,00	0,00	0,00	**0,18**	**0,17**	**0,19**	0,00	0,00	0,00	**0,43**	**0,25**	**0,38**

Tabela 4: Eficácia da classificação multiclasse de requisitos de software.

Classificação RSs em 12 granularidades												
	Árvore de Decisão						**Naive Bayes**					
	PROMISE original			PROMISE expandida			PROMISE original			PROMISE expandida		
	Precision	*Recall*	*F-measure*	*Precision*	*Recall*	*F-measure*	*Precision*	*Recall*	*F-measure*	*Precision*	*Recall*	*F-measure*
F	0,69	0,89	0,78	**0,70**	**0,90**	**0,79**	0,76	0,80	0,78	**0,80**	**0,81**	**0,81**
A	0,47	0,38	0,42	0,35	0,23	0,27	0,88	0,71	0,79	0,70	0,61	0,66
L	0,36	0,31	0,33	**0,38**	**0,40**	**0,39**	0,70	0,54	0,61	0,69	**0,60**	**0,64**
LF	0,57	0,34	0,43	0,46	0,24	0,32	0,68	0,68	0,68	0,61	0,63	0,62
MN	0,27	0,18	0,21	0,21	0,13	0,16	0,75	0,53	0,62	0,74	**0,58**	**0,65**
O	0,47	0,48	0,48	0,42	0,43	0,43	0,62	0,66	0,64	0,54	0,57	0,55
PE	0,83	0,63	0,72	0,76	**0,67**	0,71	0,60	0,72	0,66	**0,61**	**0,76**	**0,68**
SC	0,33	0,29	0,31	**0,44**	**0,32**	**0,37**	0,60	0,57	0,59	0,52	0,55	0,53
SE	0,69	0,50	0,58	0,60	*0,50*	0,54	0,65	0,64	0,64	0,63	0,63	0,63
US	0,69	0,60	0,64	**0,70**	0,45	0,55	0,66	0,61	0,64	0,59	0,55	0,57
FT	0,40	0,20	0,27	0,00	0,00	0,00	0,67	0,20	0,31	0,60	**0,33**	**0,43**
PO	0,00	0,00	0,00	**0,17**	**0,17**	**0,17**	0,00	0,00	0,00	**0,38**	**0,25**	**0,30**
	k-**NN**						**SVM**					
F	0,72	0,87	0,79	**0,73**	**0,89**	**0,80**	0,74	0,92	0,82	**0,75**	*0,92*	**0,83**
A	0,27	0,76	0,40	*0,27*	**0,84**	**0,41**	0,84	0,76	0,80	0,71	**0,77**	0,74
L	0,35	0,62	0,44	**0,44**	0,53	**0,48**	0,73	0,62	0,67	**0,75**	0,60	*0,67*
LF	0,34	0,32	0,33	**0,42**	**0,33**	**0,37**	0,77	0,53	0,63	0,69	0,45	0,54
MN	*0,00*	*0,00*	*0,00*	*0,00*	*0,00*	*0,00*	0,56	0,29	0,38	**0,73**	**0,33**	**0,46**
O	0,63	0,52	0,57	**0,65**	0,51	*0,57*	0,72	0,71	0,72	0,65	0,56	0,60
PE	0,81	0,54	0,64	**0,84**	0,46	0,60	0,91	0,74	0,82	0,88	0,73	0,80
SC	0,42	0,48	0,44	**0,45**	0,45	**0,45**	0,58	0,52	0,55	0,56	0,41	0,47
SE	0,74	0,44	0,55	0,63	0,43	0,51	0,72	0,67	0,69	0,69	0,63	0,66
US	0,74	0,46	0,57	0,58	0,41	0,48	0,80	0,66	0,72	0,70	0,59	0,64
FT	0,67	0,20	0,31	**1,00**	**0,22**	**0,36**	1,00	0,30	0,46	0,88	**0,39**	**0,54**
PO	0,00	0,00	0,00	**0,18**	**0,17**	**0,17**	0,00	0,00	0,00	**0,40**	**0,17**	**0,24**

5 DISCUSSÃO

O Teorema *No Free Lunch* define que não é possível apontar um método de aprendizagem de máquina como sendo superior à outro para todos os problemas existentes [30]. A superioridade de determinados métodos em detrimento a outros pode estar relacionado à ajustes na otimização de parâmetros e/ou pesos, através da base de treinamento. Por este motivo, o *baseline* usado neste estudo foi a base original da PROMISE, ou seja, os resultados do modelo treinado com a nova base proposta foram confrontados com os resultados obtidos através do treinamento com a base original. Desta forma, é garantido que diferentes ajustes de otimização não influenciaram a comparação dos classificadores analisados, já que as configurações de parâmetros de otimização usadas foram iguais para o treino em ambas as bases.

Dentre os grandes desafios associados à área de ML estão a quantidade insuficiente de dados para treino dos algoritmos, a não representatividade das bases de treino e a baixa qualidade dos dados da base de treino [14]. Nesta pesquisa, a expansão da base de dados foi motivada pelo desafios relatados em relação ao tamanho da base PROMISE [1, 9, 23] . O objetivo deste estudo é prover uma base de dados maior (em relação à quantidade de instâncias) e de qualidade (em relação à qualidade de rotulagem manual dos requisitos)

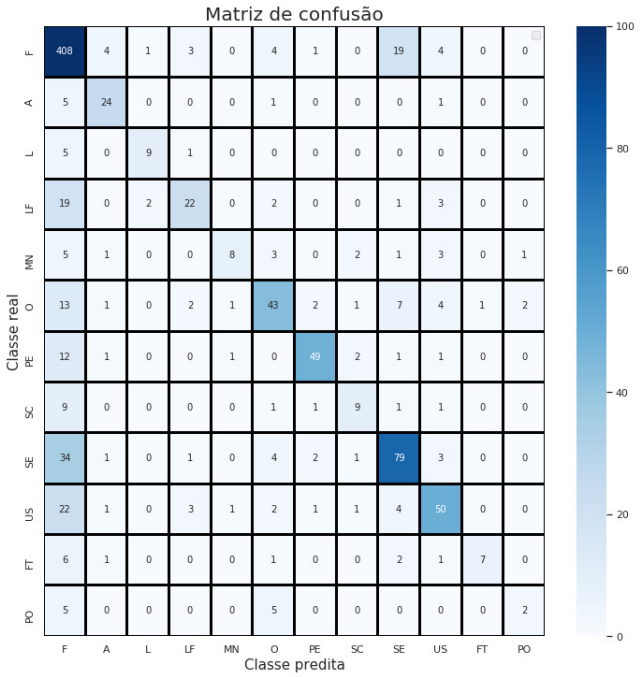

Figura 4: Matriz de confusão classificador SVM treinado pelo PROMISE_exp.

para promover a inserção da Inteligência Artificial (IA), através de *machine learning*, no suporte às tarefas da área de engenharia de software.

Apesar de persistir o problema da não representatividade da base de dados, comum em muitos contextos do mundo real [27], foram acrescidos à classe de RNF-PO, 11 novos requisitos de portabilidade contribuindo para um aumento na representatividade deste tipo de RNF. Fato este positivamente comprovado pelo aumento na capacidade dos quatro classificadores analisados de identificar RNF-PO. A base PROMISE_orig registra apenas uma instância do RNF-PO: *"The product is expected to run on Windows CE and Palm operating systems."*. Com a inserção de novas instâncias do tipo RNF-PO o uso da base PROMISE_exp permitiu que algoritmos de classificação identificassem com uma taxa de precisão de 43% os requisitos desta categoria. Analisando as respostas dos algoritmos de ML não é possível identificar com precisão quais das novas instâncias de RNF-PO foram classificadas erroneamente, porém é possível perceber que todos os algoritmos de ML treinados com a base PROMISE_orig não foram capazes de classificar corretamente instâncias do tipo PNF-PO.

Na tarefa de diferenciação de RFs de RNFs, os resultados demonstram que no contexto da classificação binária, a base expandida obteve superioridade na classificação dos RSs em requisitos funcionais. Na base expandida, o aumento no número de requisitos funcionais é superior ao aumento no número de RNFs, tal diferença constitui fator para que o desempenho dos classificadores binários fosse superior na identificação de RFs comparados à determinação de RNFs, conforme discutido por Géron [14] quando afirma que algoritmos de ML necessitam de quantidades de dados de treino para funcionarem "apropriadamente".

Apesar de discreta a melhora obtida nas tarefas de classificação dos requisitos de software, a expansão da base introduziu diversidade aos diferentes tipos de requisitos registrados na PROMISE_orig. Neste caso, a diversidade é obtida quando são agregadas informações de diferentes contextos de software, contribuindo para mitigar as limitações descritas Casamayor *et al.* [6].

Como contribuição principal deste estudo, foi gerada uma nova base de requisitos de software rotulada através da expansão de uma base já existente e utilizada pela comunidade científica. A PROMISE_exp tem por objetivo viabilizar a realização de novos trabalhos na aplicação de ML à engenharia de software, estando publicamente disponível em: https://tinyurl.com/PROMISE-exp.

5.1 Ameaças à validade

Alguns pontos da pesquisa podem influenciar em sua validade, dentre os quais destacam-se:

(1) Expansão feita manualmente, neste caso o julgamento humano pode influenciar os resultados obtidos. Para mitigar esta ameaça o processo de expansão foi validado e avaliado através de métricas estatísticas comprobatórias.
(2) Uso de documentos de definição de requisitos de software disponíveis publicamente na *web*. A obtenção de documentos públicos impossibilita a garantia de que os dados sejam oriundos de um ambiente real de desenvolvimento de software e que seu conteúdo esteja correto. Para mitigar esta ameaça dois autores realizaram a leitura dos documentos de ERs retornados e filtraram os documentos não associados ao contexto desta pesquisa.
(3) Extração e classificação manual dos RSs descritos nesses documentos. A fim de minimizar o impacto desta ameaça, o teste estatístico *Kappa* foi usado para mensurar o nível de concordância dos pesquisadores em relação à classe de requisito associada a cada nova instância da expansão.
(4) A quantidade de requisitos identificados. Devido a natureza manual da expansão a quantidade de requisitos acrescentados à PROMISE_exp não foi suficiente para permitir o balanceamento da base original, ou seja, o número de RFs continua superior ao número de RFN-PO, por exemplo.

6 CONCLUSÃO

Este trabalho apresenta a proposta de expansão de um repositório de dados aplicável à tarefas de classificação automática de requisitos de software. Com esta finalidade foi conduzida uma busca focada na *web* por documentos contendo registros de requisitos de software. A partir de uma análise manual feita em tais documentos, foram identificados e extraídos requisitos de software usados na expansão. Todo o processo priorizou a compatibilidade e a qualidade da expansão. Tal expansão foi avaliada através do uso de algoritmos de ML e seus resultados foram comparados aos resultados da base original quando submetida aos mesmos algoritmos.

Baseando-se nos resultados obtidos, conclui-se que a nova base de dados PROMISE_exp pode ser utilizada em pesquisas que utilizem algoritmos de ML no apoio a tarefa de classificação automática de requisitos de software. Assim como, em qualquer pesquisa que

necessite de uma base de requisitos de software previamente rotulada. Dentre os benefícios obtidos com o uso algoritmos que automatizem a tarefa de classificação de requisito, pode-se citar: evitar as suposições humanas, diminuir tempo de execução e aumentar a precisão da realização da tarefa. A nova base também pode ser usada como artefato para apoiar o estudo da engenharia de software, fornecendo uma base de amostra de exemplos de diferentes tipos de requisitos de software que pode ser usada no apoio ao ensino de ES através de realização de exercícios de inspeção, definições de cenários operacionais com caso de uso e criação de *stories* de BDD (*Behavior Driven Development*).

A realização de experimentos para avaliar o impacto da inserção dos novos registros à base de dados original revelaram que, apesar da discreta melhora obtida nas tarefas de classificação, binária e multiclasse, de requisitos de software, a expansão da base introduziu diversidade aos diferentes tipos de requisitos registrados na PROMISE_orig. Fator que também influencia no desempenho dos algoritmos [14].

Como pontos positivos oriundos da expansão executada, cita-se: o aumento de aproximadamente 55% da base original PROMISE; a inclusão de novas instâncias do tipo de requisito não funcional PO, possibilitando a sua determinação; o aumento aproximado de 89% de instâncias dos requisitos do tipo RNF-SE; superioridade na identificação de requisitos funcionais; e, melhora de desempenho na determinação de classes de requisitos do tipo RNF-FT e RNF-PO. Contudo, devido a natureza manual da expansão realizada, não foi possível atingir o balanceamento da base de dados original, ou seja, persistem a existência de classes com baixa representatividade de instâncias. Além disso, a expansão de instâncias de requisitos não funcionais do tipo operacional (RNF-O) não trouxe melhora de desempenho a nenhum dos algoritmos de ML avaliados.

Por fim, a nova base gerada está disponível para comunidade científica no endereço: https://tinyurl.com/PROMISE-exp.

AGRADECIMENTOS

Os autores agradecem o apoio financeiro da CAPES, através do processo: PROCAD 175956/2013 e Financiamento 001, da FAPEAM através do processo: PPP 04/2017 e da UFAM através da resolução N.012/2018 PAIC-AM 2018-UFAM. Agradecemos também aos pesquisadores do USES-UFAM pelas contribuições na execução deste trabalho.

REFERÊNCIAS

[1] Zahra Shakeri Hossein Abad, Oliver Karras, Parisa Ghazi, Martin Glinz, Günther Ruhe, and Kurt Schneider. 2017. What Works Better? A Study of Classifying Requirements. *2017 IEEE 25th International Requirements Engineering Conference (RE)* (2017), 496–501.

[2] Rana Alkadhi, Teodora Lata, Emitza Guzmany, and Bernd Bruegge. 2017. *Rationale in development chat messages: an exploratory study.* IEEE.

[3] Rana Alkadhi, Manuel Nonnenmacher, Emitza Guzman, and Bernd Bruegge. 2018. How do developers discuss rationale?. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER).* IEEE, IEEE, Campobasso, Italy, 357–369.

[4] Elisa Baniassad, Paul C Clements, Joao Araujo, Ana Moreira, Awais Rashid, and Bedir Tekinerdogan. 2006. Discovering early aspects. *IEEE software* 23, 1 (2006), 61–70.

[5] Anna L Buczak and Erhan Guven. 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials* 18, 2 (2016), 1153–1176.

[6] Agustín Casamayor, Daniela Godoy, and Marcelo Campo. 2009. Semi-Supervised Classification of Non-Functional Requirements: An Empirical Analysis. *Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial, ISSN 1137-3601, Vol. 13, Nº. 44, 2009, pags. 35-44* (05 2009). https://doi.org/10.4114/ia.v13i44.1044

[7] Min Chen, Yixue Hao, Kai Hwang, Lu Wang, and Lin Wang. 2017. Disease prediction by machine learning over big data from healthcare communities. *Ieee Access* 5 (2017), 8869–8879.

[8] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems.* ACM, ACM, Boston, MA, USA, 7–10.

[9] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc. 2006. The Detection and Classification of Non-Functional Requirements with Application to Early Aspects. In *14th IEEE International Requirements Engineering Conference (RE'06).* IEEE, Minneapolis/St. Paul, MN, USA, 39–48. https://doi.org/10.1109/RE.2006.65

[10] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20, 1 (1960), 37–46.

[11] Bradley Efron. 2013. Bayes' theorem in the 21st century. *Science* 340, 6137 (2013), 1177–1178.

[12] Katti Faceli et al. 2011. *Inteligência Artificial: Uma Abordagem de Aprendizagem de Máquina.* LTC.

[13] Yeongsu Kim et. al. 2018. Improving Classifiers for Semantic Annotation of Software Requirements with Elaborate Syntatic Structure. *International Journal of Advanced Science and Technology, ISSN 2005-4238 IJAST, Vol. 112, Nº. 44, 2009, pags. 123-136* (2018), 14. https://doi.org/10.14257/ijast.2018.112.12

[14] Aurélien Géron. 2017. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* "O'Reilly Media, Inc.", USA.

[15] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter* 11, 1 (2009), 10–18.

[16] Ahmed E Hassan and Tao Xie. 2010. *Mining software engineering data.* IEEE.

[17] IEEE. 1998. IEEE Recommended Practice for Software Requirements Specifications. (1998), 37. https://doi.org/10.1109/IEEESTD.1998.88286

[18] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. 2016. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, Las Vegas, NV, USA, 4565–4574.

[19] Reyes Ju, Guillermo Licea, et al. 2017. Towards supporting software engineering using deep learning: A case of software requirements classification. In *2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT).* IEEE, IEEE, Mérida, Mexico, 116–120.

[20] Qadeem Khan, Usman Akram, Wasi Haider Butt, and Saad Rehman. 2016. Implementation and evaluation of optimized algorithm for software architectures analysis through unsupervised learning (clustering). In *2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA).* IEEE, IEEE, Sousse, Tunisian, 266–276.

[21] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. 2007. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering* 160 (2007), 3–24.

[22] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.

[23] R. Navarro-Almanza, R. Juárez-Ramírez, and G. Licea. 2017. Towards Supporting Software Engineering Using Deep Learning: A Case of Software Requirements Classification. In *2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT).* IEEE, Mérida, Mexico, 116–120. https://doi.org/10.1109/CONISOFT.2017.00021

[24] Mohd Hafeez Osman and Mohd Firdaus Zaharin. 2018. Ambiguous software requirement specification detection: an automated approach. In *2018 IEEE/ACM 5th International Workshop on Requirements Engineering and Testing (RET).* IEEE, IEEE, Gothenburg, Sweden, Sweden, 33–40.

[25] Fabrizio Sebastiani. 2002. Machine Learning in Automated Text Categorization. *ACM Comput. Surv.* 34, 1 (March 2002), 1–47. https://doi.org/10.1145/505282.505283

[26] I. Sommerville. 2011. *Engenharia de software.* PEARSON BRASIL.

[27] Jason Van Hulse, Taghi M. Khoshgoftaar, and Amri Napolitano. 2007. Experimental Perspectives on Learning from Imbalanced Data. In *Proceedings of the 24th International Conference on Machine Learning (ICML '07).* ACM, New York, NY, USA, 935–942. https://doi.org/10.1145/1273496.1273614

[28] C. J. van Rijsbergen. 1979. Information Retrieval. http://www.dcs.gla.ac.uk/Keith/Preface.html. Acessado em 8 de maio de 2019.

[29] Peter Willett. 2006. The Porter stemming algorithm: then and now. *Program* 40, 3 (2006), 219–223.

[30] David H Wolpert, William G Macready, et al. 1997. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation* 1, 1 (1997), 67–82.

[31] Ong Shu Yee, Saravanan Sagadevan, and Nurul Hashimah Ahamed Hassain Malim. 2018. Credit card fraud detection using machine learning as data mining technique. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 10, 1-4 (2018), 23–27.