Promise+: expandindo a base de dados de requisitos de software Promise_exp

Bruno Silva[1], Rodrigo Nascimento[1], Luis Rivero[1], Geraldo Braz[1], Rodrigo Pereira dos Santos[2], Luiz E. G. Martins[3], Davi Viana[1]
[1]Universidade Federal do Maranhão - UFMA, São Luís, Maranhão, Brasil
[2]Universidade Federal do Estado do Rio de Janeiro - UNIRIO, Rio de Janeiro, Rio de Janeiro, Brasil
[3]Universidade Federal de São Paulo - Unifesp, São José dos Campos, São Paulo, Brasil
{bruno.carvalho1,rodrigo.nascimento}@discente.ufma.br,{luis.rivero,geraldo.braz,davi.viana}@ufma.br
rps@uniriotec.br,legmartins@unifesp.br

ABSTRACT

A classificação de requisitos de software é um dos processos da etapa de análise de requisitos, sendo fundamental para a compreensão do software a ser criado. Realizar essa classificação manualmente é uma tarefa difícil, demorada e sujeita a erros. Nesse sentido, trabalhos na literatura propõem utilizar algoritmos de aprendizado de máquina supervisionado para automatizar essa tarefa. As bases de dado mais comumente usadas para este processo são PROMISE e PROMISE_exp. No entanto, estudos anteriores identificaram questões como o número limitado de requisitos e a falta de diversidade das bases de dados existentes. Essas limitações impactam negativamente o desempenho dos algoritmos de aprendizado de máquina na classificação de requisitos. Este trabalho é uma nova expansão da base de requisitos com classificação feita por especialistas e avaliada no desempenho de seis algoritmos de aprendizado de máquina. Apresentamos a expansão, nomeadamente Promise+, que representa um aumento de quase 280% face ao PROMISE_exp. Para a tarefa de classificação binária, o Promise+ representou uma melhoria na identificação de requisitos funcionais. Quanto às tarefas multiclasse, a maioria dos algoritmos treinados com Promise+ apresentou melhor desempenho em mais classes de requisitos não funcionais. Por fim, o Promise+ estará disponível para toda a comunidade de Engenharia de Software.

KEYWORDS

Engenharia de Requisitos, Repositório, Base de Dados, Aprendizado de Máquina

1 INTRODUÇÃO

A Engenharia de Requisitos é um processo no ciclo de desenvolvimento de software responsável por mediar as necessidades e interesses dos *stakeholders*, bem como os custos e esforços envolvidos no desenvolvimento do software. Esse processo possui uma natureza multidisciplinar, sendo subdividido nas seguintes etapas [25]: elicitação; análise; documentação; verificação e validação.

Para apoiar a análise de requisitos, a ISO 29148:2018 [2] indica que os requisitos contenham atributos que auxiliem na sua identificação. Um dos atributos sugeridos é o tipo, podendo ser requisito funcional, que descreve funções do software ou de seus componentes, e requisitos não-funcionais que descrevem as restrições impostas ao software e ajudam a garantir que o software atenda às necessidades dos usuários. Os requisitos não-funcionais podem ser classificados em: portabilidade, usabilidade, confiabilidade, manutenibilidade e entre outros. É importante destacar que uma correta identificação e classificação de requisitos não-funcionais tem papel fundamental no projeto arquitetural e ajuda na identificação dos *early aspects* [5], isto é, aspectos críticos do sistema que devem ser devidamente tratados no momento inicial do projeto do software.

A classificação manual dos requisitos é uma tarefa difícil, demorada e propensa a erros, especialmente para projetos de software com uma abundância de requisitos [7, 22]. Neste contexto, vários trabalhos se apresentaram na literatura discutindo a tarefa de classificar automaticamente requisitos [33], utilizando algoritmos baseados em regras e aprendizado de máquina (AM).

A tarefa de classificação em algoritmos de aprendizado de máquina se insere no campo da aprendizagem supervisionada, sendo assim, o modelo treina com um conjunto de dados para poder inferir rótulos a dados desconhecidos. O ponto central está, portanto, no conjunto de dados utilizados no treinamento. No contexto de requisitos, Kaur e Kaur [15] indicam que um dos conjuntos de dados mais utilizados é o PROMISE Dataset [24]. Porém, Navarro-Almanza et al. [18] indicam que essa base possui baixa quantidade de dados para aplicações de aprendizado de máquina. Assim, surgem outras iniciativas para expansão, como a PROMISE_exp [16] que incorporou requisitos de diferentes fontes. Apesar dos esforços, ainda há relatos sobre limitações desta expansão, como conjuntos de dados desbalanceados [7, 22], baixa quantidade de dados para aplicações de aprendizado de máquina e conjuntos de dados com amostras pouco diversas em relação ao tipo de software.

A necessidade de avaliar como a ampliação dessas bases de dados impacta o desempenho de algoritmos de aprendizado de máquina em tarefas de classificação de requisitos se torna evidente, pois a introdução de novos dados pode influenciar diretamente na precisão e eficácia desses algoritmos. Outro aspecto que fortalece a proposição de novas expansões é a introdução de algoritmos de redes neurais e aprendizado profundo, algoritmos que necessitam de abundância de dados [27]. Desta forma, o objetivo deste trabalho é realizar uma nova expansão da base de requisitos, tomando como ponto de partida a base PROMISE_exp. Para isso, utilizaram-se documentos de requisitos coletados em repositórios públicos. Em seguida, realizou-se uma classificação manual dos requisitos. Os requisitos também passaram por uma validação de concordância utilizando o teste Kappa. Por fim, conduziu-se uma análise de desempenho comparativa entre as bases PROMISE_exp e a Promise+ com algoritmos de aprendizado de máquina.

A Promise+ trouxe como resultados as seguintes contribuições: expansão em cada classe de requisitos em relação a PROMISE_exp, com mínimo de 89,6% e máximo de 1.293,33%; a Promise+ disponibiliza, como metadado extra, a sua procedência (academia ou indústria); a Promise+ oferece um desempenho melhor na identificação de requisitos funcionais; e, para a maioria dos algoritmos, a Promise+ tem melhor desempenho em mais classes de requisitos não-funcionais. Os resultados indicam que a Promise+ pode ser relevante para a comunidade de Engenharia de Requisitos, especialmente para a tarefa de classificação automática de requisitos.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os algoritmos de aprendizado de máquina utilizados e as métricas de avaliação, além dos trabalhos relacionados. A Seção 3 descreve o método de pesquisa seguido na expansão da base de requisitos. A Seção 4 apresenta o resultado das análises realizadas. A Seção 5 apresenta as discussões relativas aos resultados e ameaças à validade. Por fim, a Seção 6 descreve as considerações finais e perspectivas futuras.

2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Apresenta-se, nessa seção, a base teórica utilizada neste trabalho. Inicialmente tem-se a explanação sobre as etapas de pré-processamento utilizadas, depois os algoritmos de aprendizado de máquina e as métricas consideradas para comparação. Descreve-se, também, uma visão geral da base PROMISE_exp. Por fim, os trabalhos relacionados são apresentados.

2.1 Pré-processamento

O pré-processamento de texto é uma etapa importante nos *pipelines* de Processamento de Linguagem Natural (PLN), especialmente quando associado à aplicação de algoritmos de aprendizado de máquina. Esta etapa envolve a limpeza e preparação dos dados antes de serem utilizados em modelos de AM. Neste trabalho, os requisitos passam por estágios de pré-processamento conforme apresentado na Figura 1.

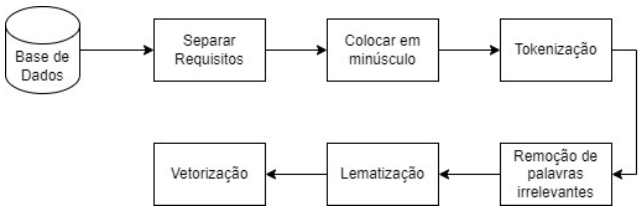

Figura 1: Fases do pré-processamento

A fase de separação de requisitos é responsável por retirar as instâncias da base de dados e separar cada instância em um registro. A próxima fase é o início da normalização, que visa padronizar os requisitos em minúsculo, contribuindo assim para a redução de dimensionalidade dos vetores que posteriormente serão formados.

A terceira fase é a tokenização, um processo responsável por identificar os limites de cada palavra no texto. Como a linguagem é o inglês, ela é delimitada por espaço. Uma vez delimitados os *tokens*, o próximo passo é a remoção de palavras irrelevantes. Palavras irrelevantes são palavras com baixo significado semântico em textos e, portanto, são removidas como uma forma de auxiliar na redução da dimensionalidade. Essas palavras incluem artigos (como "a", "the"), preposições (como "in", "at") e conjunções (como "but", "or").

Outra etapa que contribui para a redução da dimensionalidade é a lematização. Este processo considera o contexto dos *tokens* e os converte para uma base significativa, o lema, que representa o significado central ou fundamental da palavra. A lematização é preferível à redução, pois os lemas são válidos e têm significado linguístico real [12].

Finalmente, antes de servir como entrada para algoritmos de aprendizado de máquina, é necessário criar um vetor de características mapeando os textos para vetores numéricos, também conhecidos como *word embeddings*. Entre os métodos de *word embeddings* mais comumente usados e de melhor desempenho está o método *Term Frequency - Inverse Document Frequency* (TF-IDF). O TF-IDF é um método estatístico para calcular pesos para cada palavra individualmente encontrada em diversos documentos. É basicamente composto por dois scores [20]: (1) frequência do termo que mede com que frequência a palavra aparece em um documento; e (2) frequência inversa que mede a importância de um termo globalmente.

2.2 Algoritmos de Aprendizado de Máquina

Algoritmos de aprendizado de máquina são utilizados para automatizar o processo de classificação de requisitos. Zamani et al. [32] relatam que os algoritmos mais utilizados para a tarefa de classificação de requisitos são KNN, Naive Bayes e MVS. Os algoritmos SGD e PA foram utilizados em Silva e Madeiro [13], enquanto o RL foi utilizado em Canedo e Mendes [7]. Portanto, estes seis algoritmos foram selecionados para avaliar o desempenho da Promise+. Abaixo, apresenta-se uma breve descrição destes algoritmos:

- *K-Nearest Neighbor* (KNN): este algoritmo pertence ao tipo de aprendizado não generalizante, o que significa que não tenta construir um modelo interno geral, mas apenas armazena instâncias de dados de treinamento. A classificação é realizada atribuindo a classe de dados que possui mais representantes entre os vizinhos mais próximos. Um parâmetro sensível neste algoritmo é o número de vizinhos a serem considerados, ou seja, o valor de K. Essa abordagem é adequada para classes multimodais, mas tem baixa eficiência [11];
- *Multinomial Naive Bayes* (MNB): os métodos de Naive Bayes são construídos com base na aplicação do teorema estatístico de Bayes, assumindo independência condicional entre cada par de características (palavras). MNB é uma variação para dados distribuídos de forma multinomial sendo usado na classificação de texto. Este método é eficiente computacionalmente e robusto para vetores com dados esparsos, o que é comum em contextos de texto, porque algumas palavras podem não estar presentes em todos os documentos. No entanto, a suposição de independência condicional entre palavras, frequentemente falha em textos reais e MNB tende a ignorar palavras raras ou reduzir seu peso, o que pode diminuir a eficiência do algoritmo [31];
- Máquina de Vetores de Suporte (MVS): este algoritmo realiza classificação construindo um hiperplano N-dimensional que

separa os dados em categorias [23]. Um ponto-chave nesta abordagem é a escolha da função de *Kernel* para a função de decisão, sendo cada função mais adequada dependendo do problema e do conjunto de dados. MVSs são eficazes em espaços de alta dimensão e em termos de memória computacional. No entanto, as MVSs podem ter dificuldades com conjuntos de dados desbalanceados e são sensíveis a *outliers*;

- Regressão Logística (RL): este algoritmo pertence à classe de algoritmos lineares, assumindo que um valor alvo é uma combinação linear de características. Este modelo usa a função logística para modelar a probabilidade da variável dependente. A regressão logística produz coeficientes interpretáveis que ajudam a entender quais palavras influenciam mais na classificação de texto. Outra característica deste modelo é sua eficiência computacional com grandes volumes de texto. No entanto, este algoritmo é sensível a *outliers*, não é adequado para classes desbalanceadas, sendo limitado para problemas não lineares [29];
- *Stochastic Gradient Descent* (SGD): é uma técnica de otimização numérica simples, mas eficiente, para ajustar classificadores que trabalham com funções de perda convexas. O SGD é frequentemente usado na classificação de texto por possuir bom desempenho em dados de alta dimensão e esparsos. Uma característica desta abordagem é sua diversidade de operação que, dependendo da função de perda, pode assumir características de um modelo linear, modelo quadrático, classificador probabilístico, *perceptron*, entre outros. No entanto, o SGD é sensível à inicialização de hiper parâmetros, e seus resultados podem variar muito dependendo do parâmetro de regularização, número de iterações ou taxa de aprendizado [3, 28];
- *Passive Aggressive* (PA): fazem parte de uma família de algoritmos de aprendizado de máquina chamados algoritmos de aprendizado online, significando que os dados vêm em ordem sequencial e o modelo é atualizado passo a passo. Essa característica é útil ao lidar com uma abundância de dados. Esses algoritmos mantêm o modelo se a previsão estiver correta e alteram o modelo se a previsão estiver incorreta. Adicionalmente, eles tendem a esquecer exemplos de treinamento anteriores quando confrontados com novos exemplos e também têm dificuldades com problemas multi-classe, comumente usados para classificação binária [28].

2.3 Métricas de Desempenho

Quando se trabalha com algoritmos de aprendizado de máquina supervisionado, avaliar o desempenho do modelo é crucial para determinar sua eficácia na resolução de tarefas. Essas métricas de avaliação são essenciais para quantificar seu desempenho em relação aos dados de teste. Geralmente, quatro métricas são utilizadas para avaliar modelos de classificação [15, 16]: precisão, *recall*, acurácia e *F1-score*.

A precisão mede a proporção entre as instâncias classificadas como positivas e as que de fato são positivas. Ou seja, a precisão avalia a qualidade das predições positivas do modelo. O *recall* mede a proporção entre verdadeiros positivos identificadas pelo modelo em relação ao número total de instâncias positivas. Sendo assim, o *recall* avalia a capacidade do modelo em encontrar instâncias positivas. Por sua vez, a acurácia quantifica a proporção de predições corretas em relação ao total de predições feitas pelo modelo, ou seja, representa a capacidade do modelo em classificar corretamente todas as classes.

Por fim, o *F1-score* é uma métrica que quantifica o equilíbrio entre precisão e *recall*, sendo calculada como a média harmônica destas duas métricas. *F1-score* é útil quando existe um desequilíbrio significativo entre as classes. No contexto do presente trabalho, esta métrica será utilizada para a comparação entre o desempenho dos algoritmos em cada *dataset*, por ser uma forma simples e direta de fazer esta comparação [9]. Outro ponto que reforça a escolha desta métrica é que ela consegue quantificar melhor os desempenhos dos algoritmos na situação de desbalanceamento dos *datasets* [17].

2.4 PROMISE_exp: base de dados expandido de requisitos de software

Em Lima et al. [16], realizaram uma expansão da PROMISE, resultando na PROMISE_exp. A PROMISE [24], publicada originalmente em 2005, objetivava fomentar o uso de modelos preditivos na comunidade de Engenharia de Software. A PROMISE_exp já vem sendo utilizada no lugar da versão original [4, 7, 13, 19, 26], por possuir mais amostras rotuladas.

A base de dados PROMISE_exp manteve os requisitos e classes da base original e expandiu em 344 requisitos, retirados de 34 projetos de software encontrados na internet. A expansão resultou em 969 requisitos, sendo 444 requisitos funcionais e 525 requisitos não-funcionais. Cada instância possui três atributos: *ProjectID*, *RequirementText* e *Class*. Cada atributo é separado por vírgula sendo descritos a seguir:

- `ProjectID`: atributo responsável por relacionar ao projeto no qual o requisito foi retirado;
- `RequirementText`: atributo que representa o texto do requisito de software;
- `Class`: atributo que representa o rótulo do RequirementText. Este atributo possui 12 classes possíveis, sendo F para requisitos funcionais e o restante para diferentes classes de requisitos não-funcionais: A (*Availability*), L (*Legal*), LF (*Look and Feel*), MN (*Maintainability*), O (*Operacional*), PE (*Performance*), SC (*Scalability*), US (*Usability*), FT (*Fault Tolerance*) and PO (*Portability*).

2.5 Trabalhos relacionados

A literatura apresenta trabalhos que analisam o desempenho de algoritmos de aprendizado de máquina, com diferentes tipos de tratamentos para resolver, em diferentes graus de granularidade, a problemática da classificação automática de requisitos de software. Canedo e Mendes em [7] analisaram diferentes técnicas de vetorização de texto (Bag-of-Words, TF-IDF e Chi-Squared) e diferentes algoritmos de aprendizado de máquina (MVS, KNN, MNB e RL) para três tarefas diferentes: classificação binária entre requisitos funcionais e não-funcionais; classificação entre requisitos não-funcionais e classificação com todas as classes do conjunto de dados. O conjunto de dados utilizado foi o PROMISE_exp, e os melhores resultados foram obtidos com TF-IDF e RL usando validação cruzada com k definido como 10. Os autores destacaram

que expandir a base de dados poderia levar a melhores resultados e ao uso de outras técnicas de extração de características, como word2vec e Bag-of-Words (BoW) hierárquico.

Silva e Madeiro [13] se concentraram na análise de desempenho de algoritmos de aprendizado de máquina para todas as tarefas de classificação de requisitos. Foram utilizadas as técnicas KNN e MVS. As técnicas de vetorização de texto utilizadas foi BoW. O conjunto de dados utilizado foi o PROMISE_exp. Os autores concluíram que MVS performa melhor que o KNN para todos os casos que o conjunto de dados utilizado possui um baixo número de amostras.

Além disso, Karolayne Teixeira da Silva and Madeiro [13] investigaram o desempenho de algoritmos de aprendizado de máquina para a tarefa de classificar requisitos não-funcionais. Os algoritmos avaliados foram: *Extra Trees*, RL, *Multi Layer Perceptron*, MNB, PA, MVS e SGD. Os autores enfatizaram o desequilíbrio de classe no conjunto de dados PROMISE_exp e conduziram comparações entre métodos que abordam esse problema. Os autores identificaram que o algoritmo de RL tem o melhor desempenho sem reamostragem. Quando se considera a reamostragem o algoritmo *Extra Tree* apresentou o melhor desempenho. Os autores destacaram a necessidade de realizar novos experimentos com conjuntos de dados maiores.

No trabalho de Patel et al. [19], foi investigado o desempenho de algoritmos de aprendizado de máquina para três tarefas de classificação. Os algoritmos avaliados foram: KNN, MNB, MVS e DT. Adicionalmente, BoW e TF-IDF foram avaliados como algoritmos de vetorização. O conjunto de dados utilizado foi o PROMISE_exp. O melhor desempenho foi alcançado com a combinação de TF-IDF e MNB.

A pesquisa apresentada neste trabalho realizou uma nova expansão do *dataset* PROMISE_exp [16] que foi denominado PROMISE+. Adicionalmente, realizaram-se avaliações comparativas entre o PROMISE_exp e o PROMISE+ utilizando os seguintes algoritmos de aprendizado de máquina: KNN, MNB, MVS, RL, SGD e PA.

3 MÉTODO DE PESQUISA

A condução desta nova expansão da base de requisitos PROMISE teve como objetivo aumentar o número de requisitos, mantendo a qualidade e seguindo os princípios orientadores da primeira expansão (PROMISE_exp) [16]. O aumento quantitativo do número de instâncias do conjunto de dados também passou por um processo de validação e avaliação. Na etapa de validação, o objetivo é verificar o consenso de classificação entre os especialistas em relação à classificação do requisito de software. Já na etapa de avaliação, a expansão será analisada quanto à sua eficácia na geração de modelos classificadores.

Esta pesquisa reproduziu a primeira expansão [16], mantendo a estrutura do conjunto de dados já definida, bem como as classes de requisitos possíveis. Esta abordagem possibilita que todas as pesquisas realizadas com os dois conjuntos de dados anteriores utilizem a Promise+. O esquema adaptado possui as seguintes etapas:

- Busca focada na web por documentos de especificação de software (do inglês, *Software Requirements Specification* ou SRS);
- Análise dos SRS encontrados;
- Extração de dados dos SRS;
- Classificação dos requisitos que não possuem uma classe válida;
- Validação dos requisitos classificados;
- Geração da Promise+;
- Avaliação da Promise+.

O motor de busca da Google[1] foi usado para recuperar documentos de novas especificações de requisitos. As buscas foram realizadas utilizando a sequência "*Software Requirements Specification*". Inicialmente, a única restrição foi quanto ao uso da língua inglesa no documento, por ser a linguagem encontrada tanto na PROMISE quanto na PROMISE_exp. Qualquer documento que violasse essa restrição não seria considerado para análise. Além disso, era restrita também a duplicação de documentos cujos requisitos já estivessem na base.

As etapas de análise de documentos e extração de dados foram realizadas simultaneamente. Na análise de documentos, o objetivo era verificar tanto a estrutura quanto a forma de documentação dos requisitos. Apesar da existência de normas e recomendações para a criação de documentos de requisitos [1, 2], é comum encontrar documentos com formatos confusos ou improvisados. Esta etapa visou eliminar documentos com estruturas incompreensíveis. Outro aspecto analisado, foi como os requisitos são documentados, uma vez que existem diferentes métodos de documentação (protótipos, diagramas de caso de uso, histórias de usuário, entre outros). O único método válido em nosso contexto era considerar documentos nos quais os requisitos fossem documentados na forma de frases.

Na extração de dados, dois pesquisadores preencheram um formulário com os seguintes dados:

- Ano: ano de publicação do documento;
- Descrição do Software: os pesquisadores selecionavam os campos de resumo e introdução dos documentos. Este item foi posteriormente utilizado para realizar a análise quanto ao tipo de software;
- Requisitos classificados: os pesquisadores selecionavam os requisitos que estivessem descritos em um tipo válido (Funcional ou alguma das onze classes de não-funcionais contidas na PROMISE_exp);
- Requisitos não classificados: os pesquisadores selecionavam os requisitos descritos com um tipo diferente das 12 categorias definidas na PROMISE_exp.

Assim como na primeira expansão, neste trabalho também optou-se por preservar as classificações trazidas no documento, caso fossem válidas, ou seja, requisitos com alguma das 12 classes da PROMISE_exp (Seção 2.4). Portanto, os requisitos já classificados foram inseridos diretamente na Promise+. No entanto, os requisitos não classificados precisavam ser classificados por especialistas.

Nesta etapa, entraram cinco especialistas, cada um classificou aproximadamente 20% dos requisitos extraídos. Os especialistas receberam uma lista com os requisitos, uma descrição destes requisitos e eram instruídos a classificar os requisitos conforme as classes pré-definidas (Seção 2.4). Os avaliadores também eram encorajados a realizar um comentário caso encontrassem algum problema de escrita ou detectassem que aquele requisito era de uma classificação diferente às classes pré-determinadas.

[1] www.google.com.br

Para validar a classificação realizada pelos cinco especialistas, um sexto especialista classificou todos os requisitos. Para analisar o consenso na classificação, utilizou-se o teste Kappa [6]. Os resultados da aplicação do teste Kappa são detalhados na Seção de Resultados. A experiência de cada especialista em Engenharia de Requisitos é listada na Tabela 1. Destaca-se que os especialistas têm vasta experiência trabalhando com Engenharia de Requisitos em projetos de desenvolvimento e/ou orientações e com produção científica.

Tabela 1: Experiência dos especialistas

Esp	Escolaridade	Experiência (em anos)	Artigos em ER
A	Dr.	13	12
B	Me.	11	3
C	Dr.	14	3
D	Dr.	24	36
E	Me.	6	2
F	Dr.	14	14

Para a avaliação da Promise+, realizaram-se as três tarefas de classificação de requisitos de software [16]: (1) classificação binária entre requisitos funcionais e não-funcionais; (2) classificação multiclasse entre os requisitos não-funcionais e, por fim, (3) classificação multiclasse com todas as classes disponíveis da base. Os resultados comparados são entre a PROMISE_exp e a Promise+. Com os requisitos passando pelo pré-processamento descrito na Figura 1. Foram utilizados os algoritmos de AM (KNN, MVS, RL, SGD, MNB, PA), instanciados sob as mesmas condições de hiper parâmetros, com divisão treinamento / teste numa proporção 90% / 10%. Utilizou-se também o processo de validação cruzada Kfold no treinamento, com k=10. Os resultados apresentados neste trabalho são da aplicação dos algoritmos treinados com o melhor *fold* e aplicados nos dados de teste.

Para a análise quanto ao tipo de software, foram utilizadas as seguintes classificações de software: sistema, aplicativo, científico/de engenharia, embarcado, linha de produtos, web e inteligência artificial [21].

4 RESULTADOS

Após a busca no Google, 85 documentos foram encontrados. No entanto, após excluir seis documentos com requisitos já presentes na PROMISE_exp, apenas os documentos identificados por Ferrari et al. [8], no *dataset* PURE, permaneceram na análise deste trabalho. O *dataset* PURE consiste em 79 documentos de especificação de requisitos públicos coletados na web e escritos em linguagem natural. Os autores [8] tinham como objetivo fornecer um conjunto de dados para tarefas típicas de PLN em RE, como identificação de requisitos e avaliação da estrutura do documento. Eles também previram que os requisitos poderiam ser manualmente anotados para servir como *benchmarks* para tarefas como classificação de requisitos.

Na Figura 2, apresentam-se a condução e os resultados das seis iniciais etapas da metodologia (Seção 3). Após análise dos documentos, 37 foram rejeitados e 42 foram aceitos. Dos 42 aceitos, 57% (24) são documentos de software da Indústria e 43% (18) são de documentos de software de projetos universitários. O dataset PURE possui requisitos espalhados em seus documentos. Para a correta extração dos requisitos, dois pesquisadores analisaram todos os documentos internos conforme a classificação das seções nos documentos (e.g., *"Requirements", "Non-Functional", "Functional requirements", "Quality requirements"*). Assim, foram extraídos os requisitos de cada documento com a sua respectiva classificação. Portanto, dos documentos aceitos, foram extraídos 2703 requisitos, sendo que 53,6% (1448) continham uma classe válida e foram diretamente para a Promise+. Os 46,4% (1255) restantes eram requisitos sem classificação, sendo enviados para os especialistas.

Na Figura 2, tem-se também o resultado da validação do sexto especialista com os cinco especialistas. Para essa validação, utilizou-se o resultado do índice Kappa que foi, respectivamente: 0,6462, 0,7213, 0,7112, 0,4126 e 0,4094. Conforme Cohen [6], houve, portanto, acordo substancial para os três primeiros e acordo moderado para os dois últimos. Para os casos em que a concordância foi moderada (Kappa <0,6), optou-se por inserir na base a classificação informada pelo especialista de maior experiência.

No momento da classificação manual, os especialistas encontraram requisitos que, em seus entendimentos, pertenceriam a uma classe diferente das pré-determinadas (Seção 2.4). As classes de requisitos não-funcionais citadas foram: hardware, linguagem de programação, integração entre sistemas e interoperabilidade. Por vezes, alguns requisitos foram também considerados regras de negócio. Os especialistas entraram em consenso que 25 requisitos são de classe inválida. Estes requisitos estão adicionados à base em separado. A Promise+ está disponível para uso e consulta[2].

Na Figura 3 é apresentada a distribuição anual dos documentos. Percebe-se que boa parte dos documentos tem data após 2005 (72%). Dos documentos encontrados, somente em três não se teve nenhuma identificação direta do ano em que foi feito.

Quanto aos tipos de software identificados na base, observou-se que somente 4 dos 7 tipos de software foram encontrados, conforme Figura 4. Não foram encontrados os tipos de software científico/de engenharia, software de linha de produtos e software de inteligência artificial. A ausência dos dois primeiros tipos se deve, provavelmente, ao fato de que esse tipo de software geralmente resulta em patente ou propriedade intelectual. Já a ausência do último tipo pode estar relacionada ao período, conforme visto anteriormente, os documentos encontrados datam até 2010, momento em que esse tipo de software (inteligência artificial) não era tão difundido.

A Tabela 2 exibe a comparação quantitativa entre a PROMISE_exp e a Promise+. Na PROMISE_exp as classes minoritárias eram L (15 amostras), PO (12 amostras) e FT (18 amostras), as duas primeiras receberam um aumento significativo em suas amostras, respectivamente, 1286,67% e 516,67%. Cabe destacar, também, o aumento expressivo da classe MN que foi de 587,5%. A classe de maior aumento absoluto em amostras foi a classe de requisitos funcionais, no valor de 1814. Verifica-se, também, que o aumento de classes que possuíam, na base anterior, acima de 150 amostras saiu de 1 para 7, e que a classe com menos amostras, na base anterior, saiu de 12 para 35 na Promise+. A Figura 5 apresenta uma visualização do acréscimo realizado na nova base de requisitos para os requisitos não-funcionais.

[2]https://zenodo.org/records/12805484

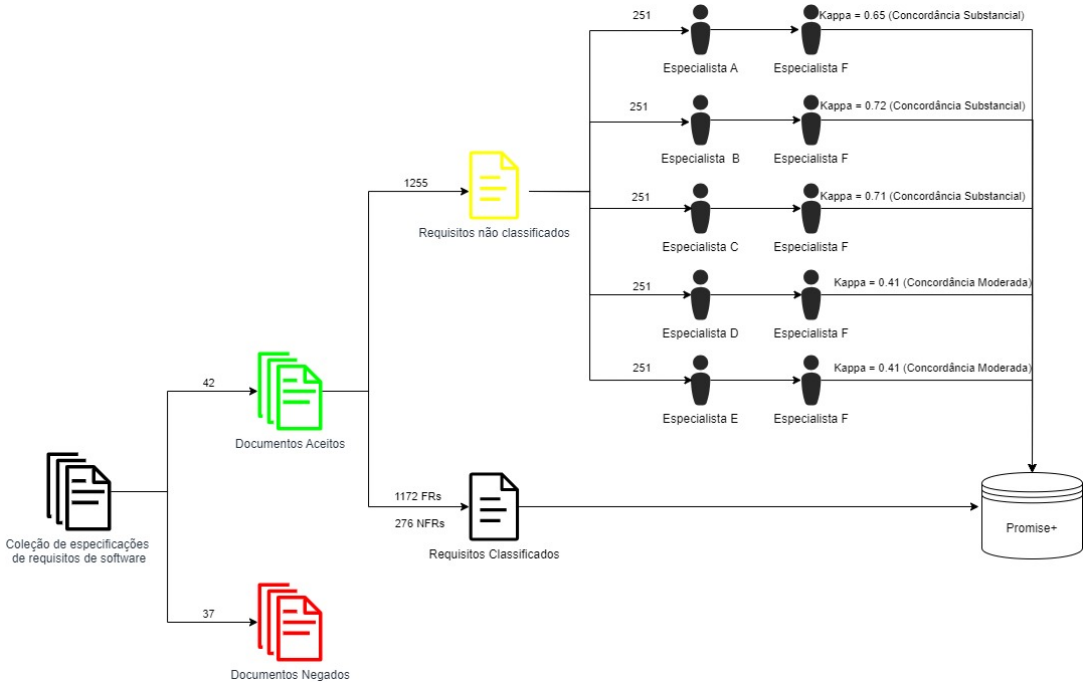

Figura 2: Resultados passo a passo da metodologia.

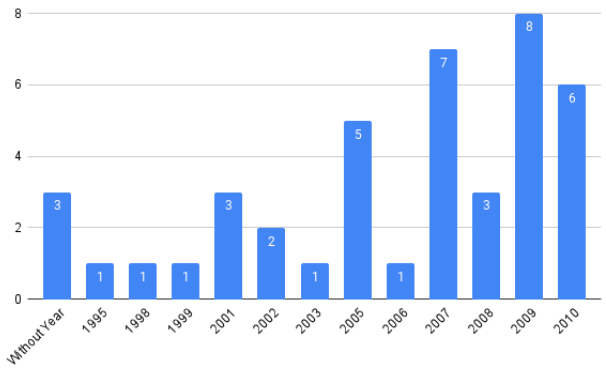

Figura 3: Lista anual de documentos.

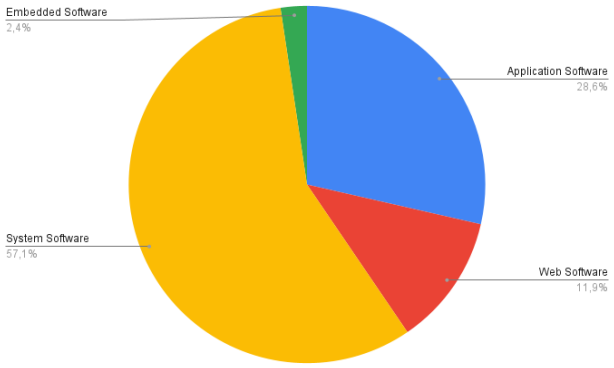

Figura 4: Distribuição de tipos de software.

Na Tabela 3, tem-se o resultado da classificação binária. Para o algoritmo KNN, tem-se um desempenho melhor na PROMISE_exp para requisitos não-funcionais, enquanto a Promise+ melhora para os funcionais. Para o MVS, tem-se um incremento de desempenho para os requisitos funcionais e valores próximos para os não-funcionais na Promise+. No RL, MNB e PA o desempenho fica bem nivelado, com melhora nos índices de requisito funcional. Já no SGD, tem-se melhora com a utilização da Promise+ na identificação de requisitos funcionais. De maneira geral, a Promise+ oferta uma melhora nos índices relativos à classe funcional.

Os resultados da classificação de requisitos não-funcionais em 11 classes estão na Tabela 4. Para a classe A tem-se, de maneira geral, melhores índices na utilização da Promise_exp, exceto na utilização de RL. Para a classe FT no KNN e PA a execução é melhor com a PROMISE_exp, enquanto no MVS e SGD fica melhor com a Promise+. Para a classe L nos algoritmos KNN, MVS e SGD ao usar a PROMISE_exp tem-se desempenho de 100%, enquanto no RL e MNB tem-se desempenho de 0%. Em relação a Promise+, o desempenho é mais uniforme em todos os algoritmos.

Para a classe LF, a Promise+ melhora em todos os algoritmos, com exceção do KNN e PA, cujo melhor desempenho fica com a PROMISE_exp. Na classe MN, a aplicação da Promise+ resulta em um desempenho superior em todos os algoritmos. Para as classes O e SE, o desempenho é melhor com a aplicação da PROMISE_exp, com exceção dos algoritmos PA e SGD. Já para a classe PE, o desempenho com a Promise+ melhora em todos os algoritmos, com exceção dos

Tabela 2: Comparação quantitativa da expansão

Classes	PROMISE_exp	Promise+	Taxa de Aumento
F	444	2258	408,56%
A	31	71	129%
L	15	209	1293,33%
LF	42	102	142,85%
MN	24	75	212,5%
O	77	157	103,9%
PE	72	163	126,38%
SC	22	95	331,81%
SE	125	237	89,6%
US	85	224	163,53%
FT	18	35	94,44%
PO	12	76	533,33%
Total	969	3677	279,46%

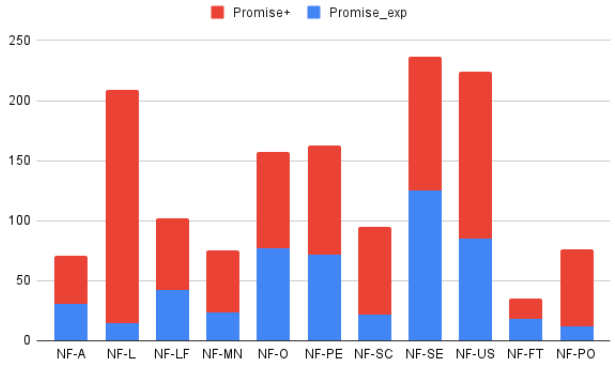

Figura 5: Comparação entre as bases de requisitos para requisitos não-funcionais.

algoritmos RL e PA. Nas classes PO e US, o desempenho melhora com a utilização da Promise+. Para a classe o SC, o desempenho é melhor na PROMISE_exp, com exceção aos algoritmos SGD e PA.

O resultado da classificação com todas as 12 classes (incluindo requisitos funcionais) é apresentado na Tabela 5. Para as classes F e A, Promise+ é melhor nos algoritmos MVS, RL e PA. Para a classe FT e PO, em todos os algoritmos o desempenho é de 0%, com exceção da classe PO nos algoritmos PA e MVS. Na Promise+ estes dois algoritmos possuem desempenho bem superior. Para a classe L, em todos os algoritmos, o desempenho é melhor na Promise+. Nas classes LF e O, a PROMISE_exp desempenha melhor nos algoritmos KNN e SGD, enquanto a Promise+ desempenha melhor nos algoritmos MVS e RL.

Na classe MN, a aplicação da Promise+ resulta em um desempenho superior em todos os algoritmos, exceto no PA. Para a classe PE, o desempenho é melhor na PROMISE_exp em todos os algoritmos. Para as classes SC e SE, o desempenho é melhor na PROMISE_exp, com exceção ao algoritmo RL. Por fim, na classe US, o desempenho da Promise+ é melhor à aplicação dos algoritmos RL, SGD e MVS.

Tabela 3: Classificação Binária

Classificação Binária						
Classe	PROMISE_exp			Promise+		
	Precisão	**Recall**	**F-1**	**Precisão**	**Recall**	**F-1**
KNN						
F	0,766	0,81	0,79	**0,834**	**0,867**	**0,85**
NF	**0,84**	**0,79**	**0,82**	0,77	0,72	0,75
MVS						
F	0,81	0,7727	0,79	**0,88**	**0,86**	**0,87**
NF	0,82	0,85	0,83	0,78	0,83	0,80
RL						
F	0,86	0,7272	0,79	0,84	**0,89**	**0,86**
NF	0,8	**0,91**	**0,85**	0,80	0,74	0,77
SGD						
F	0,777	0,8	0,786	**0,887**	**0,84**	**0,86**
NF	**0,827**	0,81	**0,82**	0,77	**0,831**	0,79
MNB						
F	**0,829**	0,77	0,8	0,78	**0,92**	**0,84**
NF	0,82	**0,868**	**0,84**	**0,83**	0,577	0,68
PA						
F	0,8	0,82	0,81	**0,88**	**0,85**	**0,86**
NF	**0,85**	0,83	**0,84**	0,77	0,824	0,796

5 DISCUSSÃO

Inicialmente, dos documentos de requisitos de software aceitos neste trabalho, mais da metade é de projetos da indústria, o que por sua vez aproxima os resultados obtidos ao ambiente real de produção. Salienta-se que mesmo os projetos acadêmicos também representam projetos de software. Em ambos os casos, são documentos estruturados para garantir o desenvolvimento de um software. Além disso, ao considerar estes projetos, aumenta-se o quantitativo de dados disponíveis, uma vez que há a carência deste tipo de dado publicamente na indústria. A relação de ids de projetos que são da indústria ou academia está disponível junto à Promise+.

Cabe destacar também que a maior concentração de documentos está entre os anos de 2005 a 2010. Ressalta-se que, apesar do tempo, tais documentos continuam relevantes, pois os requisitos de software não possuem prazo de validade. Os requisitos não estão vinculados a uma tecnologia específica, mas sim às necessidades dos clientes/usuários do sistema, mantendo sua finalidade de especificação, principalmente os requisitos funcionais. Outro fator que justifica o uso desses documentos é a escassez de documentos de requisitos públicos.

Em termos quantitativos, a expansão realizada no presente trabalho foi alta em comparação com a PROMISE_exp. Neste trabalho, o aumento absoluto foi de 2255 novas instâncias, o que representa 6,5 vezes mais do que foi realizado na primeira expansão. Todas as classes receberam aumento relativo de, no mínimo, 86%, chegando até 1286,67%, como o ocorrido com a classe L. A expansão contribui na promoção ao acesso de mais informações sobre requisitos e, portanto, diversidade na base de dados. A diversidade promove convergência em algoritmos de aprendizado de máquina ao disponibilizar um número maior de exemplos com maior diversidade. Esses fatores contribuem para a redução de *overfiting* e capacidade de

Tabela 4: Classificando RNFs em 11 classes

Classificando RNFs em 11 classes												
	KNN						MVS					
	PROMISE_exp			Promise+			PROMISE_exp			Promise+		
Classe	**Precisão**	**Recall**	**F-1**	**Precisão**	**Recall**	**F-1**	**Precisão**	**Recall**	**F-1**	**Precisão**	**Recall**	**F-1**
A	**0,42**	**1**	**0,6**	0,385	0,71	0,5	**1**	**0,666**	**0,8**	0,75	0,428	0,54
FT	**0,5**	**0,5**	**0,5**	0,166	0,33	0,22	0	0	0	0	0	0
L	**1**	**1**	**1**	0,87	0,95	0,91	**1**	**1**	**1**	0,772	0,81	0,79
LF	**0,6**	**0,6**	**0,6**	0,55	0,55	0,55	0,5	0,4	0,44	**1**	**0,555**	**0,714**
MN	0	0	0	**0,375**	**0,43**	**0,4**	0	0	0	**1**	**0,148**	**0,25**
O	1	**0,75**	**0,85**	0,733	0,6875	0,71	0,533	**1**	0,695	**0,619**	0,8125	0,70
PE	0,5	0,57	0,533	**0,846**	**0,6875**	**0,75**	0,714	0,714	0,714	**0,737**	**0,875**	**0,8**
PO	0	0	0	**0,6**	**0,375**	**0,461**	0	0	0	**0,75**	**0,375**	**0,5**
SC	0,5	0,5	0,5	**0,72**	**0,8**	**0,76**	0,5	0,5	0,5	**0,88**	**0,8**	**0,84**
SE	**0,92**	**0,923**	**0,923**	0,82	0,75	0,762	**1**	0,92	**0,96**	0,76	0,92	0,83
US	**0,857**	**0,66**	**0,75**	0,76	0,619	0,68	**0,7**	0,778	0,737	0,642	**0,857**	0,73
	RL						SGD					
	PROMISE_exp			Promise+			PROMISE_exp			Promise+		
A	0	0	0	**1**	**0,43**	**0,6**	0,75	**1**	**0,86**	**0,8**	0,57	0,66
FT	0	0	0	0	0	0	0	0	0	**1**	**0,5**	**0,667**
L	0	0	0	**0,73**	**0,76**	**0,74**	**1**	**1**	**1**	0,84	0,76	0,8
LF	1	0,2	0,33	1	**0,33**	**0,5**	0,66	0,8	0,72	**1**	**0,667**	**0,8**
MN	0	0	0	0	0	0	0	0	0	**0,5**	**0,14**	**0,222**
O	**0,778**	**0,875**	**0,823**	0,62	0,81	0,69	0,66	0,5	0,57	**0,69**	**0,8125**	**0,74**
PE	0,625	**0,7143**	0,667	**0,769**	0,625	**0,69**	**1**	0,43	0,6	0,75	**0.94**	**0.8334**
PO	0	0	0	**1**	**0,375**	**0,545**	0	0	0	**0,555**	**0,625**	**0,59**
SC	0	0	0	**1**	**0,6**	**0,75**	0.5	0.5	0.5	**0,88**	**0,8**	**0,84**
SE	0,565	**1**	0,7222	**0,62**	0,958	**0,754**	0,65	**1**	0,78	**0,74**	0,8	**0,842**
US	0,5833	0,7778	0,667	0,5882	**0,95**	**0,73**	**0,875**	**0,777**	**0,823**	0,76	0,74	0,75
	MNB						PA					
	PROMISE_exp			Promise+			PROMISE_exp			Promise+		
A	0	0	0	**1**	**0,14**	**0,25**	0,75	**1**	0,86	**1**	1	**1**
FT	0	0	0	0	0	0	1	**0,5**	**0,667**	1	0,333	0,5
L	0	0	0	**0,70**	**0,90**	**0,79**	**1**	**1**	**1**	0,54	0,90	0,67
LF	0	0	0	**1**	**0,1**	**0,18**	0,8	**0,8**	**0,8**	**1**	0,5	0,667
MN	0	0	0	0	0	0	0,33	0,5	0,4	**1**	**0,765**	**0,87**
O	**0,71**	**0,625**	**0,666**	0,5	0,44	0,4667	0,6	0,5	0,54	0,6	**0,5625**	**0,58**
PE	0,4	0,286	0,333	**0,75**	**0,4**	**0,52**	**1**	**0,71**	**0,833**	0,64	0,6	0,62
PO	0	0	0	0	0	0	0	0	0	**0,5**	**0,57**	**0,533**
SC	0	0	0	0	0	0	0.33	0.5	0.4	**0,6**	**0,76**	**0,667**
SE	0,45	**1**	0,62	**0,41**	0,956	**0,57**	**1**	0,23	0,374	0,8	**0,87**	**0,8334**
US	0,583	0,7778	**0,667**	0,45	**0,82**	0,58	**1**	**0,888**	**0,94**	0,667	0,73	0,695

generalização dos algoritmos [10].Contudo, como o maior aumento absoluto foi na classe F (1339 instâncias) o desbalanceamento, que já era presente, se intensificou. Portanto, trabalhos futuros que utilizem a Promise+ devem observar tal desbalanceamento.

Na etapa de classificação manual, o presente trabalho evidencia alguns problemas como escrita dos requisitos e classes diferentes das classes originais da PROMISE. Para o problema da escrita, isso pode impactar tanto na rotulação manual, quanto na utilização dos algoritmos de classificação. Para o problema de classes diferentes, isso pode indicar que uma nova avaliação pode ser feita futuramente, para que todos os requisitos aqui encontrados. Um dos objetivos desta nova avaliação pode ser rotular novas classes e, assim, ter um *dataset* mais granular e mais diverso em relação ao tipo de requisitos não-funcionais. Ao final da classificação manual, a validação conduzida com o índice Kappa demonstrou-se bem aceitável, com índices de concordância majoritariamente substanciais.

O Teorema *No Free Lunch* estabelece não haver um método de aprendizado de máquina que possa ser considerado superior a todos os outros para resolver todos os problemas existentes [30]. Sendo assim, um determinado método pode ser superior a outro dependendo do ajuste que se faz no treinamento. Portanto, a *baseline*

Tabela 5: Classificando todas as classes

Classificando todas as classes												
	KNN						MVS					
	PROMISE_exp			Promise+			PROMISE_exp			Promise+		
Classe	**Precisão**	**Recall**	**F-1**	**Precisão**	**Recall**	**F-1**	**Precisão**	**Recall**	**F-1**	**Precisão**	**Recall**	**F-1**
F	0,74	0,9	0,81	**0,78**	0,9	**0,83**	0,61	0,98	0,754	**0,76**	0,98	**0,86**
A	**0,75**	**1**	**0,85**	0,625	0,714	0,666	1	0,666	0,8	1	0,667	0,8
FT	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	**0,70**	**0,57**	**0,63**	0	0	0	**0,875**	**0,666**	**0,756**
LF	**0,75**	**0,6**	**0,666**	**1**	0,555	**0,71**	1	0,2	0,333	1	**0,33**	**0,5**
MN	0	0	0	**0,2**	**0,14**	**0,167**	0	0	0	**0,7778**	**0,4117**	**0,538**
O	0,571	**0,5**	**0,533**	0,5	0,3125	0,384	**0,75**	0,375	**0,5**	0,667	0,375	0,48
PE	0,857	**0,857**	**0,857**	0,7	0,4375	0,5384	**1**	**0,571**	**0,7272**	0,9	0,5625	0,692
PO	0	0	0	**1**	**0,5**	**0,667**	0	0	0	**1**	**0,25**	**0,4**
SC	1	**0,5**	**0,6667**	1	0,4	0,57	1	**0,5**	**0,6667**	1	0,4	0,57
SE	0,846	**0,846**	**0,846**	0,61	0,667	0,64	0,833	**0,77**	**0,8**	0,8125	0,5417	0,65
US	**1**	**0,444**	**0,6153**	0,6429	0,429	0,5143	1	0,333	0,5	0,785	**0,523**	**0,629**
	RL						SGD					
	PROMISE_exp			Promise+			PROMISE_exp			Promise+		
F	0,55	1	0,701	**0,6219**	**0,983**	**0,762**	0,76	**0,932**	0,837	0,7843	0,90	0,835
A	0	0	0	**1**	**0,714**	**0,8334**	0,5	1	0,667	**1**	1	**1**
FT	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	**0,875**	**0,333**	**0,483**	0	0	0	**0,786**	**0,5238**	**0,628**
LF	0	0	0	**0,2**	**0,1**	**0,1333**	**0,8**	**0,8**	**0,8**	0,556	0,5	0,526
MN	0	0	0	0	0	0	**1**	**1**	**1**	0,5625	0,53	0,545
O	**0,6667**	0,25	0,363	0,625	**0,3125**	**0,4167**	0,5	0,125	0,2	**0,545**	**0,75**	**0,6315**
PE	1	**0,43**	**0,6**	1	0,2	0,333	0,75	**0,857**	**0,8**	**0,875**	0,4667	0,61
PO	0	0	0	0	0	0	0	0	0	0	0	0
SC	0	0	0	**1**	**0,25**	**0,4**	1	**0,5**	0,667	1	0,25	0,4
SE	**1**	0.615	0,762	0,875	**0,82**	**0,8456**	**0,785**	0,846	**0,8148**	0,619	**0,91**	0,7365
US	**1**	0,333	0,5	0,8	**0,82**	**0,81**	0,333	0,555	0,416	**0,526**	**0,4545**	**0,4878**
	MNB						PA					
	PROMISE_exp			Promise+			PROMISE_exp			Promise+		
F	0,463	1	0,633	**0,61**	1	**0,76**	0,78	0,91	0,842	**0,88**	**0,934**	**0,91**
A	0	0	0	0	0	0	**1**	0,667	**0,8**	0,71	**0,71**	0,71
FT	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	**0,78**	**0,86**	**0,82**
LF	0	0	0	0	0	0	**0,75**	**0,6**	**0,667**	0,71	0,555	0,625
MN	0	0	0	0	0	0	0,667	**1**	0.8	**1**	0,857	**0,923**
O	0	0	0	0	0	0	**0,667**	**0,5**	**0,57**	0,41	0,43	0,42
PE	0	0	0	0	0	0	**0,75**	**0,857**	**0,8**	0,667	0,625	0,4645
PO	0	0	0	0	0	0	0	0	0	**0,667**	**0,5**	**0,57**
SC	0	0	0	0	0	0	0,5	0,5	0,5	**0,857**	**0,6**	**0,70**
SE	**1**	**0,15**	**0,266**	0	0	0	**0,785**	**0,846**	**0,82**	0,77	0,71	0,74
US	0	0	0	0	0	0	**0,857**	0,667	0,75	0,74	**0,81**	**0,772**

utilizado no nosso trabalho foi a base PROMISE_exp. Optou-se também por instanciar os classificadores sob as mesmas condições de hiper parâmetros para tentar diminuir fatores extras na análise e comparar somente as bases. Contudo, dado que o TF-IDF foi utilizado como método de vetorização, a presença de mais dados resulta em vetores de maior dimensão. Na Promise+, os vetores possuem mais dimensões e isso pode impactar negativamente nos resultados dos classificadores.

Para a classificação binária, em todos os algoritmos, a Promise+ fornece melhora no desempenho dos índices para a classe F. Este resultado deve-se a mudança de classe majoritária, que na PROMISE_exp era a NF, e na Promise+ passou a ser a classe F. Vale

destacar que o algoritmo MVS é o que melhor desempenha para ambas as classes (NF e F).

Para a classificação dos não-funcionais em 11 classes, a Promise+ traz melhores resultados para a maioria das classes. Vale destacar que nas classes MN e PO, a Promise+ representou sair do rendimento 0% para ter rendimento razoável. Vale destacar que os algoritmos SGD e MVS são os que possuem melhor desempenho para as classes MN e PO.

Por fim, na classificação em 12 classes (com todas as classes da base), a PROMISE_exp desempenha melhor em três dos seis algoritmos, enquanto a Promise+ lidera em dois algoritmos. Acredita-se que tal rendimento da PROMISE_exp se deva por ser um *dataset* menos desbalanceado do que a Promise+. Vale destacar que esta classificação é a mais afetada pelo desbalanceamento, mas a diferença não é substancial, conforme relatado. Entretanto, em trabalhos futuros, deve-se utilizar métodos que compensem tal desbalanceamento para uma melhor qualidade dos resultados.

Como implicações para a pesquisa, tem-se que a Promise+ pode incentivar a utilização de algoritmos de aprendizado de máquina para classificação de requisitos. Pode também ser utilizada para validar e comparar métodos já existentes de classificação de requisitos, amenizando a ameaça à validade do problema da quantidade de amostras [7, 13, 22]. Por fim, fora do escopo de aprendizagem de máquina, pode auxiliar os pesquisadores na análise padrões e tendências sobre como os requisitos são documentados.

Como implicações para a prática de classificação e análise de requisitos, tem-se que a Promise+ pode auxiliar na melhoria da precisão de modelos de classificação, o que pode resultar na redução do tempo e custo do desenvolvimento, pois os modelos classificadores podem classificar os requisitos de forma rápida e eficiente.

5.1 Ameaças à Validade

Alguns fatores podem influenciar na validade dos resultados apresentados na presente pesquisa, cabendo destacar:

- Uso de documentos de especificação de requisitos de software disponíveis publicamente na web. Para diminuir esta ameaça, os autores optaram por utilizar os documentos levantados no *dataset* PURE que já havia sido publicado anteriormente. Ademais, dois pesquisadores ainda conduziram uma análise prévia dos documentos, para utilizar somente aqueles que mais se adequaram ao escopo desta pesquisa;
- A classificação manual dos requisitos de software descritos nos documentos pode ser impactada pelo julgamento humano. Para diminuir os impactos desta ameaça, três iniciativas foram realizadas (1) os especialistas selecionados para participar possuem experiência destacada na Tabela 1 ;(2) junto aos requisitos, os especialistas receberam um manual de instruções que continha uma descrição para cada classe, de forma a normalizar o conceito para a avaliação; e (3) o teste Kappa foi utilizado para mensurar a concordância entre a avaliação dos cinco especialistas com sexto especialista;
- Texto de requisitos estavam confusos. Por vezes, os especialistas relataram a presença de requisitos confusos ou inconsistentes. Para esta ameaça, não foi possível realizar nenhum tratamento, pois poderia desconfigurar o texto original e inserir viés. A única ressalva feita foi para o caso de ter mais de um requisito na mesma estrutura textual, onde os especialistas foram instruídos a rotularem com a classe que fosse majoritária no texto.
- Desbalanceamento da base. A quantidade de requisitos adicionada não foi suficiente para permitir o balanceamento dos dados da PROMISE+. A classe F, por exemplo, continua com mais amostras do que a SC. Tal desbalanceamento pode impactar nos resultados dos classificadores.

6 CONCLUSÃO

Este artigo apresentou a Promise+, uma expansão do *dataset* PROMISE_exp, que visou fornecer novas instâncias de requisitos. Estes requisitos foram obtidos de documentos de especificação disponíveis na web, que passaram por uma análise inicial e tiveram seus requisitos extraídos e rotulados manualmente por cinco especialistas. Tal classificação foi validada com o índice Kappa, mediante a avaliação de um sexto especialista.

A Promise+ apresenta análises extras com relação ao tipo de projeto, sendo os documentos utilizados majoritariamente de projetos da indústria. Por fim, a Promise+ foi avaliada em comparação à PROMISE_exp na tarefa de classificação de requisitos utilizando algoritmos de aprendizado de máquina.

A Promise+ proporciona os seguintes pontos positivos: aumento do número de requisitos totais (aproximadamente 233%) em relação a PROMISE_exp; o aumento de requisitos em 537% na classe MN, que possibilitou uma melhoria nos algoritmos de classificação; aumento de requisitos (1287,6%) para a classe L, possibilitando a identificação deste tipo de requisito de maneira mais uniforme, com resultados promissores. No entanto, devido à abordagem manual na expansão realizada, não foi alcançado o balanceamento da PROMISE_exp. Significando que ainda existem classes com baixa representatividade de instâncias e que devem ser investigadas futuramente.

Como trabalhos futuros, podem ser feitas novas análises dos textos dos requisitos, de modo a identificar e tratar os requisitos que possuam estrutura e/ou sentido confuso. Outra análise que pode ser realizada é sobre as classes que a PROMISE possuia. Os especialistas destacaram que alguns requisitos classificados por eles podem pertencer a uma classe diferente das 12 classes utilizadas nesta pesquisa, na PROMISE_exp e na PROMISE. Para a tarefa de classificação, a Promise+ pode ser avaliada utilizando-se métodos de *word embedding* mais atuais, como os baseados em *transformers* [14]. Devido à presença de mais instâncias, a Promise+ possibilita que outros estudos com novos algoritmos baseados em aprendizado profundo possam ser executados.

AGRADECIMENTOS

Agradecemos a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Código de Financiamento 001. Além de agradecermos ao PROCAD-Amazônia da CAPES (88887.200532/2018-00). Agradecemos a FAPEMA (BEPP-03906/23). Agradece-se ainda ao CNPq (Proc. 316510/2023-8) e à UNIRIO (PPQ 2023). Por fim, agradecemos ao INCT de Internet do Futuro para Cidades Inteligentes (CNPq proc. 465446/2014-0, FAPESP proc. 14/50937-1, and FAPESP proc. 15/24485-9). Aproveita-se para agradecer aos pesquisadores que atuaram como especialistas.

REFERÊNCIAS

[1] 1998. IEEE Recommended Practice for Software Requirements Specifications. *IEEE Std 830-1998* (1998), 1–40. https://doi.org/10.1109/IEEESTD.1998.88286

[2] 2018. ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering. *ISO/IEC/IEEE 29148:2018(E)* (2018), 1–104. https://doi.org/10.1109/IEEESTD.2018.8559686

[3] Fawaz S. Al-Anzi. 2022. An Effective Hybrid Stochastic Gradient Descent for Classification of Short Text Communication in E- Learning Environments. In *2022 8th International Conference on Control, Decision and Information Technologies (CoDIT)*, Vol. 1. 1096–1101. https://doi.org/10.1109/CoDIT55151.2022.9804138

[4] Manal Binkhonain and Liping Zhao. 2023. A machine learning approach for hierarchical classification of software requirements. *Machine Learning with Applications* 12 (02 2023), 100457. https://doi.org/10.1016/j.mlwa.2023.100457

[5] Jane Cleland-Huang, Raffaella Settimi, Xuchang Zou, and Peter Solc. 2006. The Detection and Classification of Non-Functional Requirements with Application to Early Aspects. In *14th IEEE International Requirements Engineering Conference (RE'06)*. 39–48. https://doi.org/10.1109/RE.2006.65

[6] Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20 (1960), 37 – 46. https://api.semanticscholar.org/CorpusID:15926286

[7] Edna Dias Canedo and Bruno Cordeiro Mendes. 2020. Software Requirements Classification Using Machine Learning Algorithms. *Entropy* 22, 9 (2020). https://doi.org/10.3390/e22091057

[8] Alessio Ferrari, Giorgio Oronzo Spagnolo, and Stefania Gnesi. 2017. PURE: A Dataset of Public Requirements Documents. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*. 502–505. https://doi.org/10.1109/RE.2017.29

[9] Aurelien Geron. 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed.). O'Reilly Media, Inc.

[10] Zhiqiang Gong, Ping Zhong, and Weidong Hu. 2019. Diversity in Machine Learning. *IEEE Access* 7 (2019), 64323–64350. https://doi.org/10.1109/ACCESS.2019.2917620

[11] Jianglin Huang, Jacky Wai Keung, Federica Sarro, Yan-Fu Li, Yuen-Tak Yu, WK Chan, and Hongyi Sun. 2017. Cross-validation based K nearest neighbor imputation for software quality datasets: an empirical study. *Journal of Systems and Software* 132 (2017), 226–252.

[12] Nitin Indurkhya and Fred J. Damerau. 2010. *Handbook of Natural Language Processing* (2nd ed.). Chapman & Hall/CRC.

[13] Giulia Falcão de Melo F. Cavalcanti Matheus Barreto Lins Marinho Karolayne Teixeira da Silva, Geovane Miguel da Silva and Francisco Madeiro. 2021. Algoritmos de Aprendizagem Supervisionada com Conjuntos de Dados Desbalanceados para Classificação de Requisitos Não-Funcionais. In *Anais do 15 Congresso Brasileiro de Inteligência Computacional*. SBIC, Joinville, SC, 1–7.

[14] Kamaljit Kaur and Parminder Kaur. 2023. Improving BERT model for requirements classification by bidirectional LSTM-CNN deep model. *Computers and Electrical Engineering* 108 (2023), 108699.

[15] Kamaljit Kaur and Parminder Kaur. 2024. The application of AI techniques in requirements classification: a systematic mapping. *Artificial Intelligence Review* 57 (02 2024). https://doi.org/10.1007/s10462-023-10667-1

[16] Marcia Lima, Victor Valle, Estevão Costa, Fylype Lira, and Bruno Gadelha. 2019. Software Engineering Repositories: Expanding the PROMISE Database. *SBES 2019: Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, 427–436. https://doi.org/10.1145/3350768.3350776

[17] Zachary C. Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. 2014. Optimal Thresholding of Classifiers to Maximize F1 Measure. In *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, Berlin, Heidelberg, 225–239.

[18] Raul Navarro-Almanza, Reyes Juarez-Ramirez, and Guillermo 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)*. 116–120. https://doi.org/10.1109/CONISOFT.2017.00021

[19] Vrutik Patel, Priya Mehta, and Kruti Lavingia. 2023. Software Requirement Classification Using Machine Learning Algorithms. 1–6. https://doi.org/10.1109/ICAIA57370.2023.10169588

[20] Rajvardhan Patil, Sorio Boit, Venkat Gudivada, and Jagadeesh Nandigam. 2023. A Survey of Text Representation and Embedding Techniques in NLP. *IEEE Access* 11 (2023), 36120–36146. https://doi.org/10.1109/ACCESS.2023.3266377

[21] R. Pressman and B. Maxim. 2016. *Engenharia de Software - 8ª Edição.* https://books.google.com.br/books?id=wexzCwAAQBAJ

[22] Gaith Y Quba, Hadeel Al Qaisi, Ahmad Althunibat, and Shadi AlZu'bi. 2021. Software Requirements Classification using Machine Learning algorithm's. In *2021 International Conference on Information Technology (ICIT)*. 685–690. https://doi.org/10.1109/ICIT52682.2021.9491688

[23] Duksan Ryu, Okjoo Choi, and Jongmoon Baik. 2016. Value-cognitive boosting with a support vector machine for cross-project defect prediction. *Empirical Software Engineering* 21 (2016), 43–71.

[24] J. Sayyad Shirabad and T.J. Menzies. 2005. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada. http://promise.site.uottawa.ca/SERepository

[25] Ian Sommerville. 2015. *Software Engineering* (10th ed.). Pearson.

[26] Ahmad Subahi. 2023. BERT-Based Approach for Greening Software Requirements Engineering Through Non-Functional Requirements. *IEEE Access* PP (01 2023), 1–1. https://doi.org/10.1109/ACCESS.2023.3317798

[27] Mohammad Mustafa Taye. 2023. Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions. *Computers* 12, 5 (2023). https://doi.org/10.3390/computers12050091

[28] Saurabh Tiwari, Santosh Singh Rathore, Shreya Sagar, and Yash Mirani. 2020. Identifying use case elements from textual specification: A preliminary study. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 410–411.

[29] Wentao Wang, Nesrin Hussein, Arushi Gupta, and Yinglin Wang. 2017. A regression model based approach for identifying security requirements in open source software development. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*. IEEE, 443–446.

[30] D.H. Wolpert and W.G. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1, 1 (1997), 67–82. https://doi.org/10.1109/4235.585893

[31] Shuo Xu, Yan Li, and Zheng Wang. 2017. Bayesian Multinomial Naïve Bayes Classifier to Text Classification. In *Advanced Multimedia and Ubiquitous Engineering*. Springer Singapore, Singapore, 347–352.

[32] Kareshna Zamani, Didar Zowghi, and Chetan Arora. 2021. Machine Learning in Requirements Engineering: A Mapping Study. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*. 116–125. https://doi.org/10.1109/REW53955.2021.00023

[33] Liping Zhao, Waad Alhoshan, Alessio Ferrari, Keletso J. Letsholo, Muideen A. Ajagbe, Erol-Valeriu Chioasca, and Riza T. Batista-Navarro. 2021. Natural Language Processing for Requirements Engineering: A Systematic Mapping Study. *ACM Comput. Surv.* 54, 3, Article 55 (apr 2021), 41 pages. https://doi.org/10.1145/3444689