Algoritmos de Aprendizagem Supervisionada com Conjuntos de Dados Desbalanceados para Classificação de Requisitos Não-Funcionais

Karolayne Teixeira da Silva
UNICAP ICAM-TECH
Universidade Católica de Pernambuco
Recife, Brasil
karolayne.2019230005@unicap.br

Geovane Miguel da Silva
UNICAP ICAM-TECH
Universidade Católica de Pernambuco
Recife, Brasil
geovane.2019210924@unicap.br

Giulia Falcão de Melo F. Cavalcanti
UNICAP ICAM-TECH
Universidade Católica de Pernambuco
Recife, Brasil
thegiuliacavalcanti@gmail.com

Matheus Marinho
Escola Politécnica de Pernambuco
Universidade de Pernambuco
Recife, Brasil
mblm@ecomp.poli.br

Francisco Madeiro Bernadino Junior
UNICAP ICAM-TECH
Universidade Católica de Pernambuco
Recife, Brasil
francisco.madeiro@unicap.br

***Resumo*—Algoritmos de *Machine Learning* aliados com técnicas de Processamento de Linguagem Natural (PLN) são uma ferramenta importante para classificação textual. A aplicação dessas técnicas na Engenharia de *Software* pode evitar eventuais falhas humanas e garantir maior precisão e agilidade na classificação de requisitos de *software*. Porém, o desempenho das abordagens de Aprendizagem de Máquina é afetado quando existe desbalanceamento entre as classes em conjuntos de dados. Para minimizar os impactos desse problema, técnicas de reamostragem de dados podem ser aplicadas. Para investigar os efeitos da reamostragem em bases desbalanceadas, cinco dos métodos de subamostragem, sobre-amostragem e também a combinação de ambos são aplicados ao conjunto de dados PROMISE_exp, e os desempenhos de sete algoritmos de aprendizagem de máquina supervisionada para a classificação de 11 subcategorias de Requisitos Não-Funcionais são avaliados em um cenário de simulação envolvendo seis experimentos, sendo cinco deles com o corpus modificado por reamostragem e um aplicando apenas as técnicas de PLN para fins comparativos. Como destaque dos resultados obtidos é possível citar o *Extra Trees* que alcançou uma melhoria de em média 7% de F1 e 12% de precisão no ADASYN, técnica de sobre-amostragem, e os algoritmos *Multilayer Perceptron*, *Passive Aggressive* e *Stochastic Gradient Descent*, que obtiveram até 3% de melhora, em termos de precisão, *recall* e F1, quando aplicados aos conjuntos de dados sobre-amostrados.**

***Palavras-chave*—*machine learning*, requisitos não-funcionais, classificação, desbalanceamento.**

I. INTRODUÇÃO

Requisitos de sistema podem ser classificados em dois grupos: Requisitos Funcionais (RFs) e Não-Funcionais (RNFs) [1]. Esses grupos são igualmente importantes nos documentos de especificação de requisitos de *software* [2]. Os RFs descrevem funcionalidades que um *software* deve fornecer, enquanto os RNFs são declarações que apresentam peculiaridades relacionadas às qualidades e restrições de um sistema [3].

Um dos desafios da engenharia de requisitos está ligado à elicitação e definição dos RNFs. A falta de estrutura nos documentos de especificação de requisitos pode dificultar o processo arquitetônico do *software* à medida que esses requisitos muitas vezes se encontram intrínsecos nos RFs [4]. Além disso, requisitos de *software* são escritos em linguagem natural e podem apresentar ambiguidade e inconsistência nas declarações [5]. Assim, a tarefa dos engenheiros de *software* na identificação e classificação desses RNFs pode levar tempo e está sujeita a erros [4].

Desse modo, algoritmos de aprendizagem de máquina, para a classificação textual, vêm sendo amplamente explorados em pesquisas recentes para dar suporte às atividades de Engenharia de Requisitos relacionadas com a identificação e extração de RNFs em documentos de especificação de requisitos de *software* bem como o agrupamento desses artefatos em suas respectivas classes [6], [4], [7].

Além das dificuldades causadas pelos aspectos estruturais das declarações de requisitos, a grande variedade de classes, quando agrupadas em um conjunto de dados, pode apresentar um "desequilíbrio". O desbalanceamento, como também é conhecido, ocorre quando existe uma desproporção na distribuição de classes, na qual uma ou mais classes se apresentam de maneira majoritária em relação a outras [8]. É importante salientar que o desbalanceamento influencia no desempenho dos algoritmos de aprendizagem de máquina na classificação [9], [10].

O problema dos conjuntos de dados desbalanceados tem sido objeto de estudo de pesquisadores [9], [11], [12] e uma das técnicas utilizadas nesse cenário é a reamostragem, que tem por objetivo modificar o conjunto de dados de forma a reduzir as grandes diferenças entre as quantidades das amostras de classes [13].

Este artigo apresenta uma avaliação de desempenho das técnicas de reamostragem de dados Tomek Links, *Synthetic Minority Oversampling Technique* (SMOTE), SMOTE-Borderline, *Adaptative Synthetic Sampling* (ADASYN) e SMOTE-TL (combinação de SMOTE e Tomek Links) sobre o conjunto de treinamento, utilizando 11 classes de RNFs do PROMISE_exp [7], com diversos algoritmos de aprendizagem: *Extra Trees* (ET), *Logistic Regression* (LR), *Multilayer Perceptron* (MLP), *Multinomial Naive Bayes* (MNB), *Passive Aggressive* (PA), *Stochastic Gradient Descent* (SGD) e *Support Vector Machine* (SVM) no problema de classificação textual com desbalanceamento de dados. Para avaliação dos classificadores gerados, com e sem reamostragem, foram usadas as métricas de precisão, *recall* e medida F1.

O restante deste artigo está organizado da seguinte forma: a Seção II aborda trabalhos relacionados. A Seção III apresenta os algoritmos de aprendizagem utilizados, aborda o desbalanceamento, as soluções ao problema considerado neste estudo e também o pré-processamento textual que foi utilizado. Na Seção IV é apresentado o planejamento da execução dos experimentos. Os resultados obtidos são discutidos na Seção V. Na Seção VI, a conclusão e os trabalhos futuros são apresentados.

II. TRABALHOS RELACIONADOS

A classificação de requisitos de *software* tem sido objeto de estudos [3], [4], [14], [2]. Um dos conjunto de dados mais frequentemente utilizado neste cenário de pesquisa é o PROMISE [15], um conjunto composto de 625 requisitos, sendo 255 RFs e 370 RNFs, estando estes últimos distribuídos em 11 subcategorias de requisitos. Alguns dos pontos negativos citados por pesquisadores são a baixa representatividade dos dados do PROMISE e o desbalanceamento [16], [17], [3], [7].

Kurtanovic e Maalej [6] utilizaram o algoritmo SVM para classificar requisitos em RFs, RNFs e subcategorias de RNFs. Os autores fizeram uso do conjunto de dados PROMISE e, para equilibrar os dados, empregaram a técnica de subamostragem aleatória. Além disso, avaliaram uma técnica de sobreamostragem com um conjunto de dados derivado de análises de usuários sobre os produtos da Amazon. Fizeram uso de uma validação cruzada de 10-*folds* e obtiveram como melhores resultados os valores de precisão e *recall* de até 90%.

O PROMISE também foi utilizado pelos autores Abab et al. [16], com enfoque em algoritmos de aprendizagem de máquina e pré-processamento de dados. Com o pré-processamento, foram observadas melhorias nos resultados – em termos de precisão, *recall* e acurácia – em relação à classificação realizada sem a aplicação desse pré-processamento. Entre os algoritmos de aprendizado de máquina utilizados, o *Binarized Naive Bayes* teve o melhor desempenho na tarefa de classificar RNFs, com precisão de 91% e *recall* de 90%.

Visando viabilizar futuras pesquisas na área de Engenharia de Requisitos, Lima et al. [7] fizeram buscas *online* por documentos de especificação de requisitos de *software*, e uma classificação manual por especialistas da área foi realizada. Os dados obtidos foram utilizados para expandir o PROMISE, que obteve um aumento de aproximadamente 55% no número de instâncias, formando assim o PROMISE_exp, conjunto de dados utilizado neste trabalho.

Em 2020, Canedo e Mendes [3] utilizaram os algoritmos SVM, MNB, *K-Nearest Neighbor* (KNN) e LR em conjunto com as técnicas de vetorização de texto: *Bag of Words*, *term frequency–inverse document frequency* (TF-IDF) e *Chi-square* para classificar RFs e RNFs no conjunto de dados PROMISE_exp. Para a escolha de hiperparâmetros utilizaram a classe GridSearchCV[1] e para classificação fizeram uso de uma validação cruzada de 10-*folds*. Seu melhor resultado foi utilizando a técnica de LR em conjunto com a abordagem TF-IDF, atingindo um F1 de 74%.

III. FUNDAMENTAÇÃO TEÓRICA

Nesta seção, os principais conceitos relacionados às técnicas utilizadas são abordados.

A. Algoritmos de aprendizagem de máquina supervisionada

Na Aprendizagem de Máquina Supervisionada, os elementos de entrada de um determinado conjunto de dados são fornecidos com rótulos conhecidos que correspondem às saídas desejadas [18]. Assim, o classificador pode construir um modelo com base em generalizações feitas sobre os dados de entrada em relação aos rótulos fornecidos e então esse modelo pode ser utilizado para prever novas instâncias não apresentadas no seu treinamento [19].

Sete algoritmos foram selecionados para a análise comparativa de desempenho na aprendizagem com conjunto de dados desbalanceado.

- *Extra Trees* (ET): é uma técnica de *ensemble* que agrega os resultados de várias árvores de decisão não correlacionadas coletadas em uma "floresta" para produzir seu resultado de classificação.
- *Logistic Regression* (LR): apesar do nome, é um método linear que usa uma função logística, descrita graficamente por uma curva sigmoide, para representar as probabilidades que descrevem os possíveis resultados da classificação de uma amostra de dados [20].
- *Multilayer Perceptron* (MLP): é uma rede neural que tem várias camadas computacionais com um número predeterminado de neurônios. O algoritmo comumente utilizado para o seu treinamento é o *backpropagation* [21].
- *Multinomial Naive Bayes* (MNB): Trata-se de um modelo probabilístico baseado no teorema de Bayes para dados distribuídos multinomialmente. O MNB calcula a probabilidade de um dado pertencer a uma classe, partindo do pressuposto de que para que um evento *a* ocorra, o mesmo depende do que ocorra com o evento *b* [22], [7].
- *Passive Aggressive* (PA): é definido como um algoritmo de aprendizagem *online*, em que os dados de entrada vêm em ordem sequencial e o modelo de aprendizado de máquina é atualizado passo a passo, ao contrário do

[1] https://scikit-learn.org/stable/modules/grid_search.html

aprendizado em lote, em que todo o conjunto de dados de treinamento é usado de uma vez [23].

- *Support Vector Machine* (SVM): o algoritmo cria um hiperplano com a maior margem possível que melhor separa duas classes. Os pontos de dados das classes mais próximas ao hiperplano são chamados de vetores de suporte [3], [7].
- *Stochastic Gradient Descent* (SGD): é uma abordagem para ajustar classificadores lineares e regressores sob funções de perda convexa, como, por exemplo, o LR e o SVM. O SGD é considerado apenas como uma técnica de otimização de gradiente, sendo assim, não correspondente a nenhuma família específica de algoritmo de aprendizado de máquina, em suma, apenas uma forma de treinar um modelo [24].

B. Aprendizagem em conjuntos de dados desbalanceados

Uma das principais soluções para lidar com conjuntos de dados desbalanceados são os métodos de reamostragem, que têm por objetivo modificar a massa de dados em uma etapa de pré-processamento [9].

Os métodos de reamostragem podem ser subdivididos em duas vertentes: uma que remove as instâncias das classes majoritárias, denominada subamostragem, e outra que cria instâncias das classes minoritárias, denominada sobre-amostragem [9]. Apesar de as técnicas de subamostragem e sobre-amostragem resolverem, até certo ponto, o problema do desbalanceamento, ainda existem desafios: na subamostragem, a perda de amostras da classe majoritária; e na sobre-amostragem, a sobreposição de classes na geração de exemplos sintéticos e a inclusão de objetos artificiais que podem produzir sobre-ajuste [25], [12]. Dessa forma, para reduzir os problemas presentes em ambas as categorias de técnicas, foram propostos na literatura algoritmos que combinam subamostragem com sobre-amostragem [12].

Neste trabalho, foram comparados os desempenhos de cinco abordagens para lidar com desbalanceamento de classes:

- *Tomek Links*: este algoritmo de subamostragem busca eliminar o desequilíbrio entre as classes fazendo com que os Tomek Links de classes majoritárias sejam removidos até que todos os pares de vizinhos mais próximos pertençam à mesma classe [26]. Um Tomek Link pode ser definido por duas instâncias a e b se: (1) b é o vizinho mais próximo da instância a, (2) a é o vizinho mais próximo da instância b e (3) a e b são de classes diferentes.
- *Synthetic Minority Oversampling Technique* (SMOTE): é uma técnica de sobre-amostragem que gera novas amostras sintéticas de casos minoritários existentes no conjunto de dados fornecido como entrada. O SMOTE seleciona exemplos que estão próximos no espaço, criando uma "linha" entre eles e gerando uma nova amostra em um ponto ao longo dessa "linha" [8].
- SMOTE-Borderline 1 (BD-SMOTE): é um método que apresenta uma versão modificada do SMOTE original. Ao invés de criar exemplos sintéticos de forma aleatória, como o SMOTE, ele utiliza todos os exemplos de minorias e apenas sobre-dimensiona as minorias que são consideradas "Borderline". O método classifica cada exemplo minoritário em três categorias: ruído, perigo e seguro, atendendo ao número de vizinhos mais próximos na classe majoritária. Finalmente, na fase SMOTE, apenas os exemplos em perigo serão usados para gerar novas amostras, assim apenas instâncias de difícil classificação são sobre-amostradas [27].
- *Adaptative Synthetic Sampling* (ADASYN): baseado no SMOTE, utiliza uma distribuição ponderada para diferentes exemplos de classes minoritárias como critério para determinar automaticamente a quantidade de dados sintéticos a serem gerados para essas amostras [28].
- SMOTE-TL: é a combinação dos algoritmos SMOTE e Tomek Links, sendo o algoritmo Tomek Links utilizado como método de limpeza de dados. Desta forma, ao invés de remover apenas exemplos de classes majoritárias que formam Tomek Links, exemplos de classes minoritárias também são removidos [10].

Para implementação dos algoritmos de reamostragem apresentados, foi utilizada a biblioteca Imbalanced-Learn[2], que fornece uma série de técnicas de reamostragem para conjuntos de dados desequilibrados.

C. Pré-processamento de dados

De acordo com Binkhonain e Zhao [4], o pré-processamento consiste em aplicar diferentes técnicas de PLN em um documento textual. Algumas dessas técnicas em conjunto compõem a normalização de texto, que é essencial para padronização, limpeza e organização dos dados, de modo que possam ser utilizados como entrada por outras técnicas de PLN, sistemas analíticos e aplicativos [3].

O processo descrito anteriormente foi aplicado ao conjunto de dados, em que as palavras nos documentos pertencentes ao corpus foram convertidas para minúsculo, as palavras numéricas foram convertidas para numerais, as palavras irrelevantes e numerais foram removidos e em todas as palavras restantes foi aplicada a técnica de lematização. Todo esse pré-processamento foi feito utilizando recursos das bibliotecas Spacy[3] e Word2number[4].

Em razão dos algoritmos de aprendizagem de máquina não operarem com dados de entrada textuais, para classificar esses dados, é necessário transformá-los em vetores numéricos. Para isto, foi utilizado o TF-IDF, que é muito aplicado em pesquisas que envolvem análise textual [3], [4], [9], [29]. O modelo TF-IDF tem o objetivo de indicar a importância de um termo em relação à coleção de documentos ao qual ele pertence [4].

IV. METODOLOGIA

A. Conjunto de dados

O conjunto de dados utilizado foi o PROMISE_exp [7], uma versão expandida do PROMISE, composta por 969 re-

[2]https://imbalanced-learn.org/stable/

[3]https://spacy.io/

[4]https://pypi.org/project/word2number/

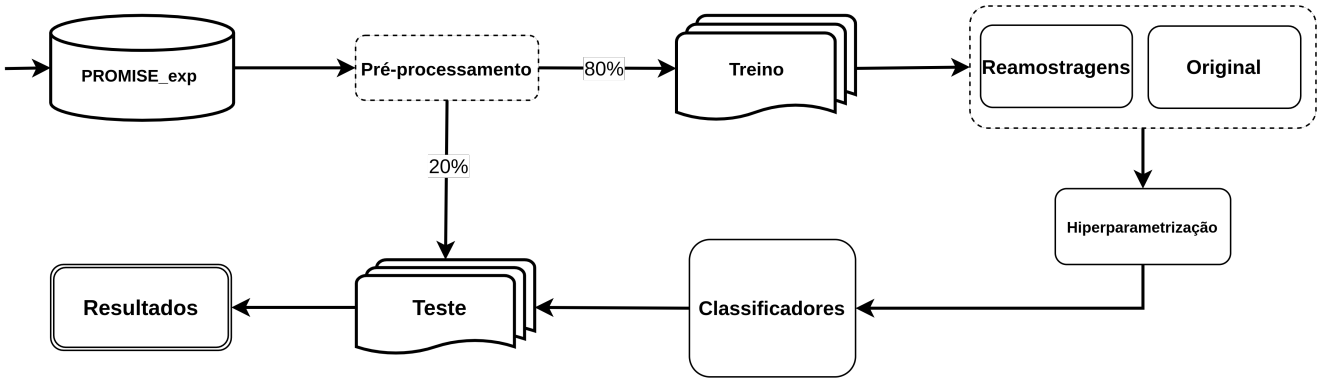


Figura 1: Diagrama de blocos dos experimentos.

quisitos, sendo 444 requisitos funcionais e 525 requisitos não-funcionais. Este conjunto de dados compreende 11 classes de requisitos não-funcionais. A distribuição desses requisitos por classe pode ser vista na Tabela I.

Tabela I: Quantidade de requisitos por classe.

Classes	Quantidade
Disponibilidade (A)	31
Tolerância a falhas (FT)	18
Legal (L)	15
Aparência e comportamento (LF)	49
Manutenibilidade (MN)	24
Operacional (O)	77
Performance (PE)	67
Portabilidade (PO)	12
Escalabilidade (SC)	22
Segurança (SE)	125
Usabilidade (US)	85
Total	525

B. Métricas de avaliação

Para avaliação de modelos de aprendizado de máquina, métricas como acurácia, precisão, *recall* ou sensibilidade e a medida F1 (*F-measure* ou *F1-score*) têm sido utilizadas [6], [3], [14]. No entanto, para problemas envolvendo desequilíbrio de dados, a acurácia pode não fornecer informações adequadas sobre a capacidade discriminatória de um classificador em relação ao conjunto de dados. Por esse motivo, os algoritmos foram avaliados de acordo com as seguintes métricas [8]:

Precisão (P): é a medida utilizada para observar a razão entre a quantidade de dados de um determinado tipo *a* classificados corretamente (verdadeiros positivos) e o total de dados classificados como sendo do tipo *a*, embora alguns desses dados não pertençam a este tipo (falsos positivos).

Recall (R): tem como objetivo apresentar o percentual entre o total de dados de um tipo *a* classificados corretamente e a quantidade real de dados do tipo *a*, ou seja, a sensibilidade indica a porcentagem de dados classificados verdadeiramente como pertencentes à sua classe original.

Medida F1 (F1): é possível ser interpretada como média ponderada entre precisão e *recall*.

C. Metodologia aplicada

Para a condução da classificação de requisitos não-funcionais com e sem reamostragem de dados no PROMISE_exp, seis etapas foram seguidas, conforme ilustra a Figura 1:

1) Pré-processamento: nesta etapa ocorre a normalização, onde os dados são limpos e em seguida o corpus de requisitos de *software* é convertido em vetores numéricos que representam as informações contidas nesses requisitos. Para isso foi utilizado o TF-IDF;

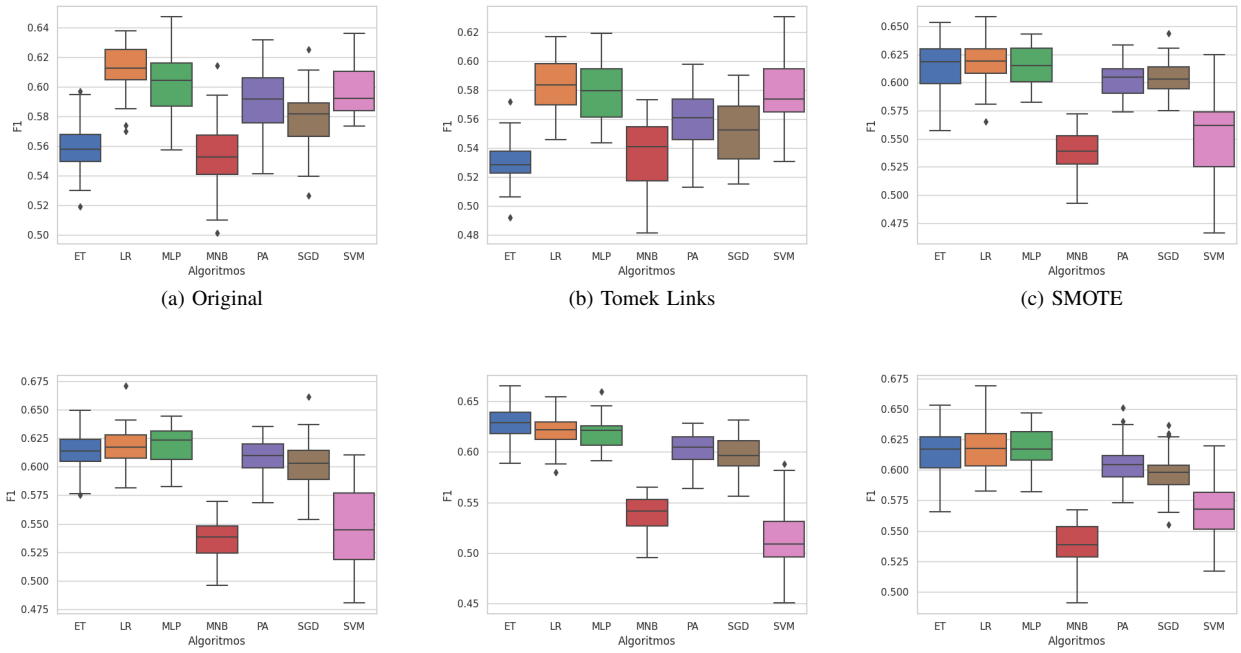
2) Divisão do conjunto de dados: o conjunto de dados é dividido em 80% (520 requisitos) para treino e 20% (105 requisitos) para teste;

3) Reamostragem de dados e conjunto original: são aplicadas, separadamente, todas as técnicas de reamostragem de dados, descritas na seção anterior, e também é mantido o conjunto sem aplicação dessas técnicas;

4) Hiperparametrização: para encontrar a combinação de parâmetros que levaram aos resultados apresentados, a classe BayesianSearchCV[5] foi aplicada aos algoritmos. Nesta classe, um número de repetições é estipulado e, para cada repetição, é realizada uma validação cruzada para um dado conjunto de parâmetros inicialmente selecionados aleatoriamente e, em seguida, com base no desempenho do classificador com esse conjunto de parâmetros, o algoritmo bayesiano escolhe os próximos hiperparâmetros [30]. Para a definição dos hiperparâmetros foram realizadas 150 interações com 6-*folds* para cada validação cruzada e a medida F1 foi a eleita para a avaliação;

5) Avaliação: a metodologia descrita na Figura 1 representa o processo geral de avaliação que foi repetido 30 vezes. Para cada repetição foi realizada uma validação cruzada estratificada de 6-*folds*. Nesse tipo de validação as dobras preservam

[5]https://scikit-optimize.github.io/stable/modules/bayessearchcv.html



(a) Original (b) Tomek Links (c) SMOTE

(d) BD-SMOTE (e) ADASYN (f) SMOTE-TL

Figura 2: *Box-plots* dos resultados com conjunto de dados sem reamostragem e com diferentes técnicas de reamostragem na classificação de RNFs.

o percentual de amostras de cada classe. Em cada uma das dobras foi aplicada a técnica de hiperparametrização descrita no item 4) e os melhores modelos gerados foram utilizados para classificar o conjunto de teste. Para garantir coerência nos resultados obtidos, todos os algoritmos foram submetidos à mesma parte do conjunto de treino e teste e às mesmas técnicas de reamostragem.

Além disso, para cada método de reamostragem, os parâmetros são os recomendados pela biblioteca utilizada (imbalanced-learn) e, portanto, os adotados neste estudo experimental.

V. RESULTADOS

Os experimentos e resultados podem ser encontrados de forma detalhada no repositório do GitHub[6], e as discussões aqui apresentadas são referentes às médias obtidas das 30 interações para cada uma das técnicas de reamostragem de dados. A medida F1 foi a métrica específica usada para gerar os *boxplots* apresentados na Figura 2.

É possível observar, em relação a abordagem sem reamostragem, apresentada na Figura 2 (a), que todas as medianas ficaram entre 54% e 62%, sendo a mais baixa a do MNB e a mais alta a do LR. Existem ainda, nos algoritmos ET, LR, MNB e SGD valores atípicos também conhecidos como *outliers*.

Na abordagem de subamostragem Tomek Links, Figura 2 (b), é perceptível a queda de F1 de em média 3% em todos os algoritmos. Esta variação na medida F1 deve-se à diminuição na quantidade de dados, que é resultado da técnica aplicada e que neste cenário não causou impacto positivo. Já na técnica de sobre-amostragem SMOTE, apresentada na Figura 2 (c), é possível perceber melhorias em relação ao conjunto original. Alguns casos que demostram essa diferença são o ET que obteve um ganho de F1, saindo de uma mediana que estava próxima a 56% para 62%, e o MLP, PA e SGD, que obtiveram um ganho de F1 de até 2%. O MNB e o SVM foram os únicos em que o SMOTE não surtiu efeito positivo, com 2% e 3% de queda em média, respectivamente.

Na Figura 2 (d) se encontram os resultados do BD-SMOTE, no qual o comportamento dos algoritmos de aprendizagem de máquina foi semelhante ao obtido no SMOTE. Neste cenário, o ET foi um dos mais impactados positivamente, com um ganho de aproximadamente 5%, em contraste com os algoritmos MNB e SVM, que apresentaram uma queda de desempenho de 2% e 5%, respectivamente. O MLP, em relação ao conjunto de dados sem reamostragem, obteve um aumento 2%. Outro ponto de destaque, é a presença de *outliers* nos algoritmos ET, LR e SGD.

Já no ADASYN, outra variante do SMOTE, presente na Figura 2 (e), o algoritmo ET alcançou uma mediana superior aos demais algoritmos, com um valor que está entre 60% e 65%, enquanto que o LR, a abordagem com a melhor mediana nas outras técnicas de sobre-amostragem e de subamostragem ficou um pouco abaixo do ET com F1 de aproximadamente 62% e gerou um *outlier* abaixo de 58%.

[6]https://github.com/MachineResearchGroup/Research2021

Finalmente, os algoritmos no SMOTE-TL, Figura 2 (f), apresentaram um desempenho semelhante ao que foi visto nas abordagens de sobre-amostragem em relação às medianas, que ficaram entre 52% e 62,5%. Além disso, foi a abordagem de reamostragem que apresentou o maior número de *outliers*, mais precisamente nos algoritmos PA e SGD.

Tabela II: Resultados dos algoritmos no conjunto de dados sem reamostragem.

Algoritmo	Precisão (%)	Recall (%)	F1 (%)
ET	53	66	56
LR	60	67	61
MLP	59	66	60
MNB	55	60	55
PA	58	64	59
SGD	57	64	58
SVM	59	65	60

A Tabela II apresenta os resultados dos algoritmos no conjunto de dados sem a aplicação das técnicas de reamostragem. Dentre todos, o algoritmo LR obteve o melhor desempenho, com precisão de 60%, *recall* de 67% e F1 de 61%, e o algoritmo ET apresenta o pior desempenho, com precisão de 53%, *recall* de 66% e F1 de 56%.

Tabela III: Melhores resultados por algoritmos nos conjuntos reamostrados.

Algoritmo	Reamostragem	Precisão (%)	Recall (%)	F1 (%)
ET	ADASYN	65	65	63
LR	ADASYN	61	68	62
MLP	SMOTE	61	67	62
MNB	BD-SMOTE	53	59	54
PA	SMOTE	60	65	60
SGD	BD-SMOTE	60	65	60
SVM	Tomek Links	57	63	58

Os melhores resultados por algoritmo de aprendizado de máquina podem ser visualizados na Tabela III. O ADAYSN, SMOTE e BD-SMOTE foram as técnicas mais recorrentes, enquanto a subamostragem Tomek Links aparece apenas uma vez, combinada ao algoritmo SVM, que não obteve melhoras com nenhuma das reamostragens de dados.

Como observado na análise dos *boxplots* o ET foi o mais afetado pela sobre-amostragem do conjunto de dados, com uma melhora de 7% de F1 e 12% de precisão. Outro destaque pode ser feito ao SGD quanto a sua precisão, que foi de 57% para 60%. Por outro lado, os algoritmos MNB e SVM tiveram uma perda de desempenho de 1% e 2% de F1 respectivamente, não demostrando ganhos quando utilizados para classificação com os conjuntos de dados reamostrados. Os demais algoritmos tiveram ganhos discretos, entre 1% e 3% em todas as métricas.

Para verificar que há diferença significativa entre as médias de F1 dos algoritmos, o teste de significância estatística *Student's t-Test* foi aplicado. Considerando a hipótese nula (i. e. a diferença não é significativa) em relação ao conjunto de dados original e as demais bases reamostradas e o ponto de corte para o valor-p de 5%, os resultados obtidos são apresentados na Tabela IV e os comentários acerca dos resultados estatísticos são:

Tabela IV: Resultados do valor-p para cada algoritmo por técnica de reamostragem.

Algoritmo	Tomek Links	Adasyn	SMOTE	BD-SMOTE	SMOTE-TL
ET	0,000	0,000	0,000	0,000	0,000
LR	0,000	0,074	0,212	0,147	0,147
MLP	0,000	0,001	0,011	0,001	0,001
MNB	0,009	0,006	0,011	0,004	0,007
PA	0,000	0,026	0,025	0,001	0,009
SGD	0,000	0,000	0,000	0,000	0,000
SVM	0,000	0,000	0,000	0,000	0,000

1) A hipótese nula foi rejeitada quando o teste-t foi aplicado aos resultados obtidos pelo algoritmo ET. Desse modo, os valores alcançados por ele nos conjuntos de dados reamostrados e, em especial nas abordagens de sobre-amostragem, onde o classificador obteve uma melhora de em média 7%, diferem com um grau de confiança de 95%.
2) No algoritmo MLP, a hipótese nula também foi rejeitada em todas as abordagens de reamostragem. Com o SMOTE, o MLP obteve seus melhores ganhos e, embora, em menores proporções de F1, a rede neural também melhorou com outras técnicas de sobre-amostragem, já que as diferenças foram testadas.
3) Com o LR, apenas no Tomek Links a hipótese nula foi rejeitada. Todas as demais sobre-amostragens tiveram hipótese nula aceita, ou seja, as diferenças existentes entre suas médias nestes conjuntos de dados em relação ao original não foram significativas.
4) O MNB e SVM, mesmo com todas as hipóteses rejeitadas, os valores de F1 obtidos por eles quando aplicados aos conjuntos de dados reamostrados com as técnicas apresentadas não foram satisfatórios. Por outro lado, os algoritmos PA e SGD, apesar dos seus ganhos discretos de 1% e 2% de F1 com as sobre-amostragens, também tiveram todas as hipóteses rejeitadas.

VI. CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho, cinco técnicas de reamostragem de dados foram combinadas com sete algoritmos de aprendizagem de máquina supervisionada para classificar 11 tipos de requisitos não-funcionais presentes no PROMISE_exp de Lima et al. [7]. Mesmo a base sendo uma recente expansão do PROMISE, ela ainda possui o mesmo problema de desbalanceamento entre as classes que a compõe, com uma razão de desequilíbrio entre a classe majoritária SE e a minoritária PO de 10,42, por exemplo. Desse modo, a aplicação dos algoritmos de reamostragem e de classificação já mencionados pode contribuir para a escolha de um conjunto de técnicas que melhor possam resolver problemas relacionados com a classificação de requisitos de *software* não-funcionais. Além disso, o agrupamento desses atributos de qualidade aumenta a organização do processo de gerenciamento de *software*, a medida que podem ser priorizados os recursos que são considerados mais importantes para cada tipo de projeto.

Em relação aos experimentos, verificou-se que no conjunto de dados sem reamostragem o algoritmo LR apresentou o

melhor desempenho, com F1 de 61%, e no conjunto subamostrado com Tomek Links todos os algoritmos apresentaram queda no desempenho. Nos conjuntos de dados sobreamostrados o ET foi um dos mais impactados, chegando a alcançar uma melhoria de 7% de F1 e de 12% de precisão com ADASYN. Os algoritmos MLP, PA e SGD obtiveram ganhos entre 1% e 3% em todas as métricas. De acordo com os resultados obtidos conclui-se que as técnicas de sobreamostragem de dados se mostram promissoras em conjunto com alguns algoritmos de aprendizagem de máquina no cenário de classificação de requisitos não-funcionais.

Como trabalhos futuros, pretende-se:

- Explorar outras soluções para conjuntos de dados desbalanceados como, por exemplo, aprendizagem sensível ao custo;
- Realizar experimentos com outros conjuntos de dados com maior quantidade de requisitos de *software*;
- Realizar experimentos com outras técnicas de vetorização e investigar os seus efeitos.

AGRADECIMENTOS

Os autores agradecem o incentivo a pesquisa oferecido pelo Programa Institucional de Bolsas de Iniciação Científica (PIBIC), a Fundação Antônio dos Santos Abranches (FASA) e a Universidade Católica de Pernambuco (UNICAP).

REFERÊNCIAS

[1] K. G. Sotelo, C. Baron, P. Esteban, C. G. Estrada, and L. J. L. Velázquez, "How to find non-functional requirements in system developments," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1573–1578, 2018.

[2] M. Asif, I. Ali, M. S. A. Malik, M. H. Chaudary, S. Tayyaba, and M. T. Mahmood, "Annotation of software requirements specification, extractions of nonfunctional requirements, and measurement of their tradeoff," *IEEE Access*, vol. 7, pp. 36 164–36 176, 2019.

[3] E. Dias Canedo and B. Cordeiro Mendes, "Software requirements classification using machine learning algorithms," *Entropy*, vol. 22, no. 9, p. 1057, 2020.

[4] M. Binkhonain and L. Zhao, "A review of machine learning algorithms for identification and classification of non-functional requirements," *Expert Systems with Applications*, vol. 1, p. 100001, 2019.

[5] M. Younas, K. Wakil, M. Arif, and A. Mustafa, "An automated approach for identification of non-functional requirements using word2vec model," *International Journal of Advanced Computer Science and Applications*, vol. 10, 2019.

[6] Z. Kurtanović and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 490–495.

[7] M. Lima, V. Valle, E. Costa, F. Lira, and B. Gadelha, "Software engineering repositories: Expanding the promise database," in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, 2019, pp. 427–436.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[9] C. Padurariu and M. E. Breaban, "Dealing with data imbalance in text classification," *Procedia Computer Science*, vol. 159, pp. 736–745, 2019.

[10] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[11] L. Hakim and S. Rochimah, "Oversampling imbalance data: Case study on functional and non functional requirement," in *2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*. IEEE, 2018, pp. 315–319.

[12] O. Loyola-González, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and M. García-Borroto, "Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases," *Neurocomputing*, vol. 175, pp. 935–947, 2016.

[13] C. F. Tsai, W. C. Lin, Y. H. Hu, and G. T. Yao, "Under-sampling class imbalanced datasets by combining clustering analysis and instance selection," *Information Sciences*, vol. 477, pp. 47–54, 2019.

[14] M. Younas, D. N. Jawawi, I. Ghani, and M. A. Shah, "Extraction of non-functional requirement using semantic similarity distance," *Neural Computing and Applications*, vol. 32, no. 11, pp. 7383–7397, 2020.

[15] "Promise." [Online]. Available: https://terapromise.csc.ncsu.edu/12/#repo/view/head/requirements/nfr

[16] Z. S. H. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe, and K. Schneider, "What works better? a study of classifying requirements," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 496–501.

[17] R. Navarro-Almanza, R. Juarez-Ramirez, and G. Licea, "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, 2017, pp. 116–120.

[18] I. Muhammad and Z. Yan, "Supervised machine learning approaches: A survey." *ICTACT Journal on Soft Computing*, vol. 5, no. 3, 2015.

[19] F. Machado Silva, M. Lucian Lenz, P. Henrique Chagas Freitas, and S. Cerqueira Bispo, *Fundamentos de Aprendizagem de Máquina*. Grupo A, 2020.

[20] Y. Chen and Q. Zhao, "Mineral exploration targeting by combination of recursive indicator elimination with the l2-regularization logistic regression based on geochemical data," *Ore Geology Reviews*, vol. 135, p. 104213, 2021.

[21] C. C. Aggarwal, "Neural networks and deep learning," *Springer*, vol. 10, pp. 978–3, 2018.

[22] Q. A. Shreda and A. A. Hanani, "Identifying non-functional requirements from unconstrained documents using natural language processing and machine learning approaches," *IEEE Access*, 2021.

[23] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *The Journal of Machine Learning Research*, vol. 7, p. 551–585, 2006.

[24] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[25] H. sui, B. Yang, Y. Zhai, W. Qu, Y. Zhai, and B. An, "The problem of classification in imbalanced data sets in knowledge discovery," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 9. IEEE, 2010, pp. V9–658.

[26] I. Tomek, "Two modifications of cnn," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 11, pp. 769–772, 1976.

[27] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International Conference on Intelligent Computing*. Springer, 2005, pp. 878–887.

[28] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 1322–1328.

[29] M. Marinho, D. Arruda, F. Wanderley, and A. Lins, "A systematic approach of dataset definition for a supervised machine learning using nfr framework," in *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*. IEEE, 2018, pp. 110–118.

[30] T. T. Joy, S. Rana, S. Gupta, and S. Venkatesh, "Fast hyperparameter tuning using bayesian optimization with directional derivatives," *Knowledge-Based Systems*, vol. 205, p. 106247, 2020.