Um modelo para detecção automática do comportamento de tentativa e erro em STI baseado em passo

Fábio M. Castilhos[1], **Felipe de Morais**[1], **Otávio B. Azevedo**[1], **Patricia A. Jaques**[1]

[1]Programa de Pós Graduação em Computação Aplicada (PPGCA)
Universidade do Vale do Rio dos Sinos (UNISINOS), São Leopoldo – RS – Brasil

{fcastilhos,felipmorais,obazevedo}@edu.unisinos.br, pjaques@unisinos.br

Abstract. *This paper presents a machine learning-based model for automatic detection of try-step abuse behavior in a step-based Intelligent Tutoring System. We have trained this model based on data obtained through the observation and analysis of videos and the system interaction logs. After the development of an annotation protocol, we have labeled the students' actions with the occurrence or not of the try-step abuse behavior. We submitted the model to unseen data, in which the model achieved a 0.684 Kappa.*

Resumo. *Este artigo apresenta um modelo baseado em aprendizado de máquina para detecção automática do comportamento de tentativa e erro em um Sistema Tutor Inteligente baseado em passos. Esse modelo foi treinado com base nos dados obtidos através da observação e análise de vídeos e dos logs de interação do aluno com o sistema. Após o desenvolvimento de um protocolo de anotação, rotulamos as ações dos alunos com a ocorrência ou não do comportamento de abuso de tentativa e erro. Submetemos o modelo a dados não usados previamente, nos quais o modelo alcançou um Kappa de 0,684.*

1. Introdução

Os Sistemas Tutores Inteligentes (STIs) são programas de computador que auxiliam os estudantes durante o processo de aprendizado, através de assistência individualizada às características dos alunos [Woolf 2007]. Apesar da comprovada eficácia desses sistemas na área da educação, alguns estudantes não utilizam todo o potencial do sistema e demonstram comportamentos indesejados que comprometem o seu aprendizado [VanLehn 2011]. Dentre esses comportamentos pode-se citar o *gaming the system*. *Gaming the system* (GTS) pode ser definido como a tentativa de ter sucesso em um ambiente computacional de aprendizagem explorando propriedades do sistema (como ajuda) ao invés de aprender o material. O GTS pode ser dividido entre o abuso por tentativa e erro (do inglês *try-step abuse*) e o abuso de dicas (*help abuse*).

Os STIs baseados em passos possuem conhecimento para auxiliar os alunos na resolução de tarefas, fornecendo *feedback* mínimo (certo ou errado) a cada passo de resolução inserido pelo aluno. No comportamento de tentativa e erro, o estudante introduz as respostas de forma rápida e sistemática, sem refletir na resolução da tarefa e se beneficiando deste *feedback* mínimo fornecido pelo tutor para identificar uma resposta correta [Baker 2006]. Esse comportamento também representa situações em que o aluno tenta resolver apressadamente um passo e erra, seja quando detém ou não tal conhecimento

DOI: 10.5753/cbie.sbie.2020.1062

[Aleven et al. 2004]. No caso do *help abuse*, o aluno pede ajuda rapidamente e repetidamente, sem raciocinar sobre a possível solução do problema, até que o tutor forneça a resposta [VanLehn 2011]. Esses comportamentos podem ser identificados e prevenidos, o que pode impactar positivamente na aprendizagem dos estudantes [Baker 2006].

Visando aprimorar os STIs baseados em passos e reduzir os efeitos negativos de comportamentos indesejados dos alunos, o objetivo deste trabalho é detectar de forma automatizada o comportamento de tentativa e erro. Essa detecção será realizada através de um modelo baseado em mineração de dados de interação (*logs*) entre o aluno e o STI PAT2Math. Tal detecção tem sido denominada como livre de sensores. Para o desenvolvimento deste tipo de modelo de detecção, são utilizados algoritmos de classificação supervisionados. Na fase de treinamento, os *logs* são utilizados como dados de entrada, onde cada *log* possui um rótulo identificando a presença ou não do comportamento de tentativa e erro. Os rótulos de treinamento foram obtidos através da análise de vídeos dos alunos utilizando o PAT2Math (tela e rosto com áudio), seguindo um novo protocolo, descrito neste artigo. Dessa maneira, como contribuição adicional, esse trabalho apresenta a definição deste protocolo para anotação de comportamentos de tentativa e erro.

2. Gaming the System e Trabalhos Relacionados

[Baker 2006] analisou os resultados da aprendizagem quando os alunos praticaram GTS. Antes e depois do experimento, os estudantes foram avaliados a partir de um pré e um pós-teste, respectivamente, de forma a determinar o seu nível de conhecimento. Dentre os comportamentos observados, o GTS foi o comportamento que apresentou uma maior correlação negativa com os resultados do pós-teste. Já os estudantes que não praticaram GTS obtiveram resultados significativamente mais altos no pós-teste, atingindo 68% (contra apenas 44% dos estudantes que apresentaram o GTS).

Artigos mais recentes sobre o tema tratam a detecção do comportamento GTS em diferentes tutores, tal como o trabalho de [Peters and et al. 2018] no tutor *Math Spring*. Nesse trabalho, os autores verificam, através de um tutor para resolver problemas matemáticos, se os comportamentos de GTS são oriundos das características dos estudantes ou do design do STI. O trabalho utiliza um detector, que verifica a ocorrência de GTS, analisando a ação efetuada pelo estudante e o tempo gasto por ele, após a ação do tutor. Eles concluíram que, tanto as características do aluno quanto o problema que eles resolvem, são preditores comparáveis do GTS. Já no trabalho de [Paquette and et al. 2018], os autores desenvolveram um modelo de detecção baseado no *Cognitive Tutor Algebra*, com o objetivo de adaptar-se a novos contextos, buscar a melhor forma de intevir nas ocorrências do GTS e aumentar a adoção desses detectores. O modelo foi validado, também, nos tutores *Cognitive Tutor Middle School* e *ASSISTments*.

O trabalho de [Azevedo et al. 2018] trata da detecção e prevenção do comportamento de GTS (*try-step* e *help abuse*) e *help refusal* através de técnicas de gamificação do STI PAT2Math. Os resultados mostram uma redução significativa dos comportamentos indesejados nos ambientes gamificados e/ou parcialmente gamificados, comparados ao sistema sem gamificação. Porém, não foi realizada uma detecção automática do GTS.

O principal diferencial do modelo de detecção automático livre de sensores, proposto neste artigo, é que ele não emprega o conhecimento do aluno para a detecção do comportamento de tentativa e erro. Assim, o objetivo é verificar a possibilidade de detec-

tar, com uma boa taxa de sucesso, a presença ou ausência do comportamento de tentativa e erro apenas pelas ações dos estudantes, sem saber o seu conhecimento prévio. A vantagem de tal modelo de detecção é que ele poderia ser adaptado para ambientes computacionais de aprendizagem que não são capazes de detectar o conhecimento do aluno.

3. Modelo

Este trabalho propõe um modelo para detecção automática do comportamento de tentativa e erro [Aleven et al. 2004]. A detecção ocorre através da análise dos dados de interação dos alunos com um sistema tutor inteligente baseado em passos. Neste trabalho, todas as ocorrências do comportamento de tentativa e erro e os *logs* de interação dos alunos com o sistema foram obtidos do STI PAT2Math.

O modelo proposto segue uma abordagem de classificação (aprendizagem supervisionada) que, através de dados de *logs* do PAT2Math e rótulos alvo do comportamento, aprende a identificar o comportamento de tentativa e erro. Os dados de *logs* contém as ações realizadas pelos alunos no sistema (passos de resolução, pedidos de dicas, erros, etc). Os rótulos alvo de comportamento foram anotados manualmente por pesquisadores em Informática na Educação, seguindo um protocolo desenvolvido para a identificação do comportamento GTS (descrito na Seção 3.2). As anotações realizadas foram sincronizadas com dados de *log*, permitindo que esses conjuntos de dados (dados de *log* rotulados com os comportamentos) possam ser usados como amostras de treinamento dos algoritmos de aprendizagem de máquina supervisionados para a detecção do comportamento.

3.1. Material: logs e anotações

O PAT2Math é um Sistema Tutor Inteligente baseado em passos que possui um ambiente *web* com um editor de equações algébricas, o qual possibilita aos alunos resolverem equações de 1º grau passo-a-passo [Jaques and et al. 2013]. No STI PAT2Math, as equações estão agrupadas em planos (ou fases) que, por sua vez, estão agrupados em cinco níveis de dificuldade, variando do básico ao avançado. Ao acessar o sistema, o aluno começa sua trajetória resolvendo as equações do primeiro plano. Os planos subsequentes são desbloqueados assim que o aluno resolver todas as equações do plano anterior. Dentro de cada fase, o sistema permite ao estudante escolher a ordem em que deseja resolver as equações. Ao iniciar a resolução de uma equação, o tutor avalia cada passo submetido pelo estudante, indicando se o mesmo está correto ou não. Se o passo estiver incorreto, o sistema fornece um *feedback* de erro e permite que uma nova solução seja submetida para o passo atual. Caso o passo esteja correto, o sistema provê um *feedback* mínimo e permite que o aluno insira o próximo passo de resolução, até que a resposta final seja inserida. A qualquer momento, o aluno pode solicitar uma dica, que será fornecida especificamente para o passo atual que o aluno está resolvendo.

As dicas oferecidas pelo STI PAT2Math seguem uma abordagem baseada em níveis de abstração e são apresentadas ao aluno de acordo com o seu nível de conhecimento. Ajudas de nível mais baixo (1 a 3) contextualizam o aluno em relação ao problema em questão. Ajudas de nível mais alto (4 e 5) apresentam dicas mais pontuais, sendo que o último nível apresenta a solução do problema. Quando o usuário requisita uma ajuda, a de nível mais baixo lhe é fornecida; caso ele decida requisitar outra ajuda, sobre o mesmo passo, o nível seguinte é exibido e assim sucessivamente até que não hajam mais dicas

[Seffrin and et al. 2012]. Todas as ações dos alunos, obtidas por um coletor de *logs*, são armazenadas na base de dados do STI PAT2Math e, por isso, puderam ser recuperadas e utilizadas nas fases de treinamento do modelo. Cada um dos passos de resolução inseridos ou dicas solicitadas pelo aluno no tutor foi considerado como uma ação, ou seja, um novo *log* de interação. Para cada *log*, foram calculadas diferentes características, levando em consideração as informações específicas da ação do aluno. Esta abordagem segue o padrão proposto por [Baker and et al. 2013].

Os rótulos alvo de comportamento utilizados neste trabalho foram anotados manualmente a partir da análise de 18 vídeos com duração entre 30 e 45 minutos, obtidos da interação dos estudantes com o tutor inteligente PAT2Math. Cada vídeo completo contém três tipos de informação: o rosto dos alunos, o áudio ambiente e a tela do computador, mostrando a interação do aluno com o sistema. Os vídeos foram gravados simultaneamente, assim foi possível sincronizá-los para também serem vistos de forma simultânea. Desta forma, foi possível analisar manualmente cada ação do aluno no sistema e inferir se ele estava apresentando um comportamento de tentativa e erro ou não, sendo este o rótulo alvo das ações (*logs*). Essa tarefa de identificar o comportamento atual do aluno é chamada de anotação dos rótulos de comportamento. Este trabalho utilizou um protocolo próprio para a anotação dos rótulos de comportamentos, descrito na Seção 3.2.

3.2. Protocolo de anotação dos rótulos de comportamento

Os algoritmos de aprendizagem supervisionada necessitam de rótulos robustos para as amostras de treinamento desses algoritmos, ou seja, que representem os comportamentos da forma mais fidedigna possível. Para a identificação dos rótulos do comportamento de tentativa e erro que foram utilizados na fase de treinamento do modelo de detecção, foi elaborado um protocolo para treinamento de anotadores. O protocolo desenvolvido é uma adaptação de [Morais and et al. 2019], que segue quatro etapas: desenvolvimento de materiais, treinamento e teste dos avaliadores e fase de anotação. Para cada ação realizada pelos estudantes, é possível identificar o comportamento de tentativa e erro com base nas observações de vídeos. Estes vídeos contém informações dos rostos dos alunos, áudio ambiente e da tela do sistema enquanto o aluno utiliza o ambiente educacional, que neste caso é o STI PAT2Math. Para a execução do protocolo desenvolvido, três avaliadores, com experiência de 2, 6 e 8 anos na utilização e análise de comportamentos dos alunos em tutores inteligentes, foram treinados para identificar especificamente as ocorrências do comportamento de tentativa e erro durante a interação dos estudantes com o tutor.

Na primeira etapa do protocolo (desenvolvimento de materiais), foram selecionados alguns vídeos em que o comportamento de tentativa e erro ocorreu. Estes comportamentos e as ações dos alunos foram listadas e identificadas em uma planilha. Para cada ação foram descritos o tempo de início (inserção do primeiro caractere na caixa de texto), tempo final (quando o estudante submete a resposta), a avaliação do tutor (correto ou incorreto), a equação inicial da tarefa e o último passo de resolução submetido pelo estudante. Para cada um dos avaliadores foram disponibilizados: (*i*) a planilha, (*ii*) os vídeos, (*iii*) uma descrição com as regras do protocolo, contendo a conceituação e exemplos de ocorrência do comportamento de tentativa e erro[1], (*iv*) a explicação sobre as fases de treinamento e de teste dos codificadores e (*v*) orientações gerais, visando permitir a correta classificação das ações. Tais orientações são descritas como:

[1] A descrição das regras do protocolo pode ser encontrada no link https://bit.ly/3j7nwJ4.

1. Uma ação representa o período no qual o estudante insere o primeiro caractere até o momento em que ele submete sua resposta. Geralmente, representa um passo de resolução de uma equação;
2. Os erros de digitação ou dupla submissão, quando por engano, não são considerados tentativa e erro;
3. Geralmente, o comportamento ocorre após a submissão de, ao menos, uma ação incorreta. Dessa forma, a ocorrência de dois ou mais erros seguidos pode ser um bom indicador do comportamento;
4. Vários erros em sequência nem sempre indicam tentativa e erro. Portanto, um fator determinante para essa análise é o tempo entre uma submissão e outra, visto que intervalos muito curtos sugerem que o aluno não leu o *feedback* de erro ou não revisou adequadamente os motivos do seu erro, podendo representar GTS;
5. Sequências numéricas, inversões de sinais ou de numerador e denominador, somas e subtrações seguidas, são exemplos comuns desse comportamento.

Após o desenvolvimento dos materiais, foi realizada uma reunião inicial entre os avaliadores, também chamados de codificadores, de forma a unificar o entendimento de todos sobre quando o comportamento de tentativa e erro ocorre. A segunda etapa, fase de treinamento dos codificadores, foi realizada em dois ciclos. Em cada ciclo, os avaliadores assistiram separadamente vídeos selecionados, classificando cada ação como presença ou ausência do comportamento de tentativa e erro. Após cada ciclo de anotação, os avaliadores discutiram os resultados e diferenças de interpretação, analisando a consistência das regras iniciais e, também, criando novas regras quando necessário.

Os rótulos anotados pelos avaliadores foram comparados entre si, utilizando a métrica *Kappa*. O coeficiente Kappa K é uma medida estatística de concordância entre dois avaliadores para valores categóricos nominais. Considera-se o *Kappa* mais robusto que um simples cálculo de percentual de concordância, já que K leva em conta a concordância que ocorre por acaso. Na comparação realizada nas fases de treinamento e teste, especificamente, foi utilizada uma variação do *Kappa* proposta por [Randolph 2005], chamada *Free-Marginal Multirater Kappa*, indicada quando os avaliadores não são forçados a atribuir um certo número de casos para cada categoria/rótulo e o número de avaliadores é maior que dois. Assim, dos ciclos de treinamento das anotações, foi obtido um índice de $89,15\%$ de concordância entre os avaliadores, sendo um *Free-Marginal Kappa* de $0,78$.

Na terceira etapa, fase de testes dos codificadores, cada avaliador testou o conhecimento adquirido na fase de treinamento, realizando novas anotações, de forma individual, não havendo discussões entre codificadores. Para a aprovação na fase de testes foi estabelecido que o valor de *Kappa* entre os codificadores deveria ser superior a $0,6$. Esse valor foi definido de acordo com o disposto em [Landis and Koch 1977], que estabelece que valores de *Kappa* superiores a $0,6$ possuem concordância substancial e valores acima de $0,8$ concordância quase perfeita.

Na etapa de testes, os avaliadores atingiram um nível concordância de 90% e o *Free-marginal Kappa* de $0,80$, ou seja, um excelente resultado. Na última etapa, de anotações, apenas um dos avaliadores aprovados nas fases anteriores rotulou cada um dos vídeos utilizados no trabalho. Tal abordagem tem sido empregada por boa parte dos trabalhos que envolvem anotação de rótulos de emoções [Morais and et al. 2019, Ocumpaugh 2015]. Essa tarefa envolve a análise de cada uma das ações realizadas pelos

estudantes, identificando o comportamento do aluno e transpondo as anotações que serão utilizadas como rótulos alvos no treinamento do modelo final.

3.3. Características

Foram extraídas várias informações de cada um dos *logs* gerados pelos alunos, chamadas de características. Este trabalho emprega 13 características que tem o objetivo de representar as ações dos estudantes. Essas características são utilizadas durante o treinamento do modelo de detecção automática de comportamento de tentativa e erro. As características escolhidas foram classificadas em quatro grupos: detalhamento da ação, tempo, interações prévias e comportamento do tutor.

Detalhamento da ação: Este grupo contém três características que detalham a ação corrente do aluno. Como os dados foram coletados do STI PAT2Math, que é um sistema tutor baseado em passos, as ações são geralmente passos (na resolução de uma equação) ou pedidos de ajuda pelos alunos. Assim, elas foram classificadas como: (1) a avaliação do passo fornecido pelo aluno, cujos possíveis valores são correta, incorreta ou pedido de dica; (2) uma característica indicando se a ação é a primeira tentativa (primeira tentativa de resolução no passo corrente) ou se é o primeiro pedido de dica no passo corrente (o estudante pode solicitar mais de uma dica no mesmo passo), (3) característica identificando se a resposta submetida pelo aluno é igual a resposta submetida na ação anterior. **Tempo:** Grupo relacionado ao tempo utilizado na execução das ações. Neste grupo foram incluídas duas características: (1) o tempo da ação, em segundos, contado a partir da inserção dos primeiros caracteres até a submissão do passo pelo aluno; (2) o tempo da ação, em segundos, contado a partir da conclusão do último passo até a próxima submissão. **Interações prévias:** As interações prévias mantém o histórico das ações imediatamente anteriores, realizadas pelos estudantes, sejam elas ações do mesmo passo ou de passos anteriores. Para isso, foram adicionadas mais seis características: (1) número de erros cometidos no passo atual; (2) número de dicas solicitadas no passo atual; (3) número de ações, dentre as cinco últimas, que referem-se ao passo atual; (4) número de erros cometidos nas últimas cinco ações; (5) número de dicas solicitadas nas últimas oito ações e (6) número de comportamentos de tentativa e erro que foram identificados nas últimas cinco ações. Os parâmetros referentes ao número de ações verificadas, seguem o modelo proposto por [Baker 2006] e foram fixados de forma a incluir a maior parte dos exemplos e aprimorar a performance. **Comportamento do tutor:** O último grupo de características refere-se ao comportamento do sistema tutor durante a interação dos estudantes. Nesse grupo foram incluídas duas características: o nível de dificuldade da equação que o estudante está resolvendo e outra característica indicando se o tutor forneceu a resposta do passo ao aluno, no *feedback* de erro.

As características geradas passaram por uma etapa de análise de qualidade, utilizando a ferramenta RapidMiner. Assim, cada uma das características foi avaliada de acordo com quatro critérios: (1) Correlação: mede o quanto a característica espelha os resultados do rótulo alvo; (2) *ID-ness*: indica o quanto os valores das características são diferentes entre si; (3) Estabilidade: aponta o quanto os valores das características são idênticos entre si; (4) *Text-ness*: reporta o quanto a característica parece conter texto livre.

A Tabela 1 descreve a lista de características e os resultados da análise da qualidade de cada uma delas. Após a análise, o RapidMiner apresenta um indicativo sobre

Tabela 1. Análise da Qualidade das Características.

Grupo	Característica	Correl.	ID -ness	Estab.	Text -ness
Detalhamento da ação	Avaliação da ação	16,17%	0,19%	83,65%	3,31%
	Primeira Tentativa?	2,05%	0,13%	69,18%	1,38%
	Resposta igual a última?	40,70%	0,13%	97,99%	1,38%
Tempo	Tempo da ação	0,30%	5,32%	8,37%	0,00%
	Tempo desde a última ação	0,11%	7,07%	4,93%	0,00%
Interações prévias	Número de erros	18,71%	0,58%	74,69%	0,00%
	Número de dicas	2,10%	0,13%	98,83%	0,00%
	Últimas 5 ações no passo	8,79%	0,39%	30,56%	0,00%
	Erros nas últimas 5 ações	13,72%	0,39%	53,73%	0,00%
	Dicas nas últimas 8 ações	1,35%	0,13%	97,40%	0,00%
	Tentativa e erro nas últimas 5 ações	13,46%	0,39%	87,87%	0,00%
Comportamento do tutor	Nível	0,43%	0,19%	45,10%	3,68%
	Resposta no feedback?	16,43%	0,19%	98,57%	1,40%

a inclusão ou exclusão de cada uma das características. Das 13 características, quatro foram indicadas como de baixa qualidade: *número de dicas*, *dicas nas últimas 8 ações*, *resposta igual a última*, e *resposta no feedback*. Dessas quatro, todas possuíam uma alta estabilidade, ou seja, muitos valores repetidos, sendo que as características relacionadas às dicas apresentaram poucas ocorrências. A característica *resposta igual a última* ainda possuía uma alta correlação com os rótulos alvo, o que poderia levar a um *overfitting* do modelo, ou seja, ele poderia não generalizar bem para novos dados. No entanto, optamos por deixar essa característica, pois uma análise manual dos vídeos realizada pelos autores indica que ela é uma forte preditora do comportamento de tentativa e erro. A partir das recomendações do RapidMiner, foram realizados testes com subconjuntos das características. Dados os resultados obtidos, também foram mantidas as características *resposta igual a última* e *número de dicas* e apenas o atributo *dicas nas últimas 8 ações* foi excluído. Já no grupo das características que o RapidMiner recomendou a manutenção, foi realizada a exclusão da característica *nível*, visto que esta não teve muita relevância e apresentou um baixo impacto nos resultados de classificação. Desta forma, o modelo final contou com apenas 11 características. As duas características excluídas são listadas em vermelho, na Tabela 1.

4. Avaliação e Resultados

Para a avaliação do modelo proposto neste trabalho, utilizamos os dados obtidos pelos *logs* e pela observação dos vídeos das interações de alunos com o STI PAT2Math. Os dados de interação e vídeos são provenientes de duas turmas do sétimo ano de uma escola particular que utilizaram o PAT2Math entre os meses de maio e agosto de 2018. Os alunos que participaram do experimento assinaram um termo de consentimento, junto com seus responsáveis, autorizando a utilização de tais dados neste trabalho.

No total, foram analisadas 1.541 ações distintas de 18 alunos. Desse total, em 74 ações, os estudantes apresentaram o comportamento de tentativa e erro, contra 1.467 ações em que o comportamento **não** foi identificado como tentativa e erro. Os dados foram divididos em dois grupos distintos. O primeiro, com 80% das ações, foi utilizado para treinamento e validação do modelo e o segundo, com as ações restantes, utilizado para teste do modelo em dados não vistos.

Para o experimento e obtenção de resultados preliminares, foram selecionados sete

algoritmos de classificação: *Decision Tree*, *Deep Learning*, *Generalized Linear Model*, *Gradient Boosted Trees*, *Logistic Regression*, *Naïve Bayes* e *Random Forest*. Cada um deles foi utilizado para treinar os modelos de classificação, utilizando a validação cruzada com 10 partições (10 *fold cross-validation*) como estratégia de avaliação do desempenho destes modelos. Os resultados obtidos através do experimento são apresentados na Tabela 2, na coluna de validação (dados desbalanceados). Para cada algoritmo, são listados o valor de *Kappa* e a métrica *F1*, que traz um equilíbrio entre o número de Falsos Positivos e Falsos Negativos, medidos pelas métricas *Precision* e *Recall*, respectivamente.

Os modelos resultantes do treinamento foram testados com os dados não vistos, cuja performance também foi medida através das métricas *Kappa* e *F1*, mais o percentual de erros verificados na aplicação dos dados aos modelos. Os resultados obtidos também são apresentados na Tabela 2, na coluna de testes referentes aos dados desbalanceados. Na fase de validação, o algoritmo com melhores resultados foi o *Gradient Boosted Trees*, alcançando um *Kappa* de $0,765$ e um *F1* de $77,55\%$. Já na fase de testes, os melhores resultados foram atingidos pelo algoritmo *Decision Trees*, que atingiu um *Kappa* de $0,701$, *F1* de $71,43\%$ e um percentual de erro de $2,60\%$, na classificação de novas ações.

Tabela 2. Resultados dos Modelos de Classificação.

Algoritmo	Dados Desbalanceados					Dados Balanceados				
	Validação		Teste			Validação		Teste		
	Kappa	F1	Kappa	F1	%Erro	Kappa	F1	Kappa	F1	%Erro
Random Forest	0,740	75,03%	0,654	66,67%	2,60%	0,699	71,68%	0,684	70,27%	3,57%
Gradient Boosted Trees	0,765	77,55%	0,670	68,75%	3,25%	0,689	70,64%	0,627	64,71%	3,90%
Deep Learning	0,735	74,64%	0,602	62,07%	3,57%	0,643	66,54%	0,622	68,29%	4,22%
Logistic Regression	0,756	76,65%	0,568	58,33%	3,25%	0,624	64,85%	0,581	60,87%	5,84%
Decision Trees	0,716	72,86%	0,701	71,43%	2,60%	0,604	62,57%	0,378	41,18%	6,49%
Generalized Linear Model	0,744	75,46%	0,568	58,33%	3,25%	0,602	62,77%	0,581	60,87%	5,84%
Naïve Bayes	0,526	55,49%	0,627	65,00%	4,55%	0,488	52,19%	0,540	57,14%	6,82%

Apesar dos bons resultados obtidos, tanto em relação ao *Kappa* quanto ao valor de *F1*, o conjunto de dados apresenta um desbalanceamento no número de ações por rótulo, ou seja, as classes não estão igualmente representadas. Das 1.233 ações analisadas na fase de validação, apenas 59 (cerca de $4,8\%$ do total das ações) apresentaram o comportamento de tentativa e erro. Apesar de comum, em conjuntos de dados reais, o desbalanceamento dos dados pode criar um viés na classificação, tornando mais difícil a identificação da classe minoritária, além de dificultar a aplicação de métricas mais usuais na avaliação da performance do modelo [Santos et al. 2018].

Para solucionar o problema de desbalanceamento dos dados, aplicamos uma técnica de replicação dos dados da classe minoritária. O *SMOTE (Synthetic Minority Oversampling Technique*), proposto por [Chawla et al. 2002], é um método usual de sobreamostragem, do inglês *oversampling*, que cria amostras sintéticas com base nos vizinhos mais próximos dos valores das características na classe minoritária. Optamos pelo *SMOTE* para balancear os dados do conjunto de treino e possibilitar que os resultados fossem mais confiáveis. A estratégia de *oversampling* é realizada durante a validação cruzada. Dessa forma, o conjunto de dados é primeiramente dividido entre 10 partições estratificadas e os dados sintéticos gerados pelo *SMOTE* são incluídos apenas no conjunto de treinamento, evitando que o conjunto de validação seja afetado ou visto pelo modelo de treinamento, permitindo uma avaliação adequada da capacidade de generalização do modelo [Santos et al. 2018].

Dessa forma, as 1.233 ações do grupo de validação são divididas em 10 partições, sendo que cada partição contém 1.110 (54 com rótulo de tentativa e erro) para treinamento e 123 (5 com rótulo de tentativa e erro) para validação. O *SMOTE* iguala as duas classes, cada uma com 1.056 instâncias, repetindo o processo a cada iteração e o modelo resultante é testado nos dados do conjunto de validação. Por fim, após a etapa de validação, o modelo é ainda aplicado aos dados do conjunto de testes, que contém as 308 (15 com rótulo de tentativa e erro) instâncias não vistas nas etapas de treinamento e validação. A Tabela 2 mostra os resultados alcançados com os modelos treinados com base no conjunto de dados balanceados. Tanto na fase de validação quanto na fase de testes, o algoritmo que atingiu os melhores resultados no grupo de dados balanceados foi o *Random Forest*. Na validação, o algoritmo atingiu um valor de *Kappa* de 0,699 e *F1 Score* de $71,68\%$. Na fase de testes, o *Random Forest* chegou a um *Kappa* de $0,684$ e um *F1* de $70,27\%$.

Ao analisar os resultados obtidos com o algoritmo *Random Forest*, é possível verificar que o modelo treinado com os dados balanceados apresentou uma diferença menor entre os resultados de teste e validação, quando comparada com o modelo treinado com os dados originais (desbalanceados). Isso indica que o modelo treinado com dados balanceados é mais apto a generalizar os resultados para novos dados, ou seja, dados não vistos ainda pelo modelo.

5. Conclusão

Este trabalho apresentou um modelo para detecção automática do comportamento de tentativa e erro (*try-step abuse*) baseado em mineração de dados de *logs*. Para o treinamento dos modelos, foram utilizados os *logs* de interação entre o aluno e o STI PAT2Math. Cada *log* foi rotulado com o comportamento do aluno, obtido por meio de um novo protocolo de anotação, baseado na análise de vídeos do rosto dos alunos com áudio e da tela dos computadores durante a interação dos estudantes com o tutor. Os modelos foram treinados considerando um conjunto de algoritmos de classificação. Ainda, foi analisada a diferença da aplicação destes modelos nos dados originais e em dados balanceados. O modelo baseado em *Random Forest* obteve um *Kappa* de $0,699$ e um valor de *F1* de $71,68\%$, na fase de validação. Já na fase de testes, o modelo atingiu um *Kappa* de $0,684$ e um *F1* de $70,27\%$, ambos os resultados considerando o conjunto de dados balanceados. Esse resultado aponta que é possível realizar a detecção automática do comportamento de tentativa e erro via mineração de dados em STIs baseados em passos, sem considerar o conhecimento do estudante como característica.

O trabalho realizado contribui com a detecção automática de comportamentos indesejados dos estudantes, como a tentativa e erro, de forma a auxiliar no desenvolvimento de ações para mitigação dos efeitos de tais comportamentos no aprendizado dos alunos junto aos tutores inteligentes. Buscou-se detectar este comportamento sem considerar o conhecimento do aluno, tornando o modelo independente dessa informação. Essa estratégia permite que este modelo seja empregado em ambientes educacionais onde não se pode predizer de forma automática o conhecimento do aluno.

Como limitações deste trabalho, pode-se destacar a detecção de outros comportamentos indesejados, como o *help abuse* e o *help refusal* que ficaram fora do escopo, e o baixo número de ações avaliadas como *pedido de dica* que impossibilitaram a identificação desses comportamentos. Dessa maneira, a criação de um modelo que possa

identificar e prevenir mais comportamentos indesejados nos tutores inteligentes baseados em passos serão o foco principal de trabalhos futuros.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, da FAPERGS (Processo 17/2551-0001203-8) e do CNPq (processo 309218/2017-9).

Referências

Aleven, V., McLaren, B., Roll, I., and Koedinger, K. (2004). Toward tutoring help seeking. In *Intelligent Tutoring Systems*, pages 227–239, Berlin. Springer.

Azevedo, O., de Morais, F., and Jaques, P. A. (2018). Exploring gamification to prevent gaming the system and help refusal in tutoring systems. In *EC-TEL*, pages 231–244.

Baker, R. and et al. (2013). Modeling and studying gaming the system with educational data mining. In *Int. Handbook of Metacog. and Learn. Techn.*, pages 97–115. Springer.

Baker, R. S. (2006). *Designing Intelligent Tutors That Adapt to when Students Game the System*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA. AAI3241593.

Chawla, N., Bowyer, K., Hall, L., and Kegelmeyer, W. (2002). Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357.

Jaques, P. A. and et al. (2013). Rule-based expert systems to support step-by-step guidance in algebraic problem solving: The case of the tutor pat2math. *Expert Systems With Applications*, 40(14):5456–5465.

Landis, J. and Koch, G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Morais, F. and et al. (2019). EmAP-ML: A Protocol of Emotions and Behaviors Annotation for Machine Learning Labels. In *EC-TEL*, Netherlands. Springer.

Ocumpaugh, J. (2015). Baker rodrigo ocumpaugh monitoring protocol (bromp) 2.0 technical and training manual.

Paquette, L. and et al. (2018). A system-general model for the detection of gaming the system behavior in ctat and learnsphere. In *AIED*, pages 257–260. Springer.

Peters, C. and et al. (2018). Predictors and outcomes of gaming in an intelligent tutoring system. In *ITS*, pages 366–372. Springer.

Randolph, J. J. (2005). Free-marginal multirater kappa (multirater k [free]): An alternative to fleiss' fixed-marginal multirater kappa. *Online submission*.

Santos, M., Soares, J., Abreu, P., Araujo, H., and Santos, J. (2018). Cross-validation for imbalanced datasets: Avoiding overoptimistic and overfitting approaches. *IEEE*.

Seffrin, H. and et al. (2012). Dicas inteligentes no sistema tutor inteligente pat2math. In *Simpósio Brasileiro de Informática na Educação (SBIE)*, volume 23.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4):197–221.

Woolf, B. P. (2007). *Building Intelligent Interactive Tutors: Student-centered Strategies for Revolutionizing e-Learning*. Morgan Kaufmann.