CBIE 2023
CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO
Uma escola para o futuro: Tecnologia e conectividade a serviço da educação

Analisando a relação de conceitos de programação com o comportamento ”abuso do sistema” em programadores novatos

Hemilis J. B. Rocha[1], Evandro de B. Costa[2], Patricia C. de A. R. Tedesco[3]

[1]Campus Viçosa– Instituto Federal de Alagoas
Viçosa – AL– Brasil

[2]Instituto de Computação
Universidade Federal de Alagoas– Maceió, AL – Brasil

[3]Centro de Informática
Universidade Federal de Pernambuco– Recife, PE – Brasil

hemilis.rocha@edu.ifal.br, pcart@cin.ufpe.br, evandro@ic.ufal.br

Abstract. *In interactive learning environments, learners may interact in various ways, particularly in problem-solving activities. In this context, it is observed that some learners resort to the system's support facilities with inappropriate behaviour, identified in the literature as gaming the system. Thus, in this paper, we present a solution to detect ”system abuse”behaviour using machine learning algorithms. Then, after detecting such behaviour, we analyse the relationship between 16 commonly covered introductory programming concepts. Our results show that the cases of ”system abuse”occur primarily in problems associated with flow control and that learners consider them more difficult.*

Resumo. *Em ambientes interativos de aprendizagem, os aprendizes podem escolher interagir de diversas maneiras, principalmente em atividades de resolução de problemas. Neste contexto, observa-se que, alguns aprendizes recorrem às facilidades de suporte do sistema com um comportamento inadequado , identificado na literatura, como gaming the system– optamos por traduzir para ”abuso do sistema”.Assim, neste artigo, apresentamos um modelo para detectar o comportamento de ”abuso do sistema”, utilizando algoritmos de aprendizagem de máquina. Assim, após a detecção de tal comportamento, analisamos a relação de 16 conceitos, geralmente, abordados de programação introdutória. Com isso, nossos resultados mostram que os casos de ”abuso do sistema”acontecem em maioria nos problemas associados a controle de fluxo e que os aprendizes consideram mais difíceis.*

1. Introdução

Os ambientes interativos de aprendizagem, particularmente os Sistemas Tutoriais Inteligentes (STI), foram desenvolvidos para diferentes domínios, a fim de atender aprendizes com diferentes necessidades [Ghaleb et al. 2018]. Assim, em sistemas como esses, para ajudar os aprendizes a realizarem as atividades com sucesso, é essencial fornecer andaimes, como dicas e feedback[Cavalcanti et al. 2020]. No estudo [d Baker et al. 2008], constatou-se que, embora o uso do STI tenha aumentado o envolvimento e o esforço

dos aprendizes na sala de aula [d Baker et al. 2008], alguns aprendizes responderam à ajuda, feedback e suporte do STI com um comportamento inadequado chamado *gaming the system* [Baker et al. 2004b] – em português traduzimos para ”abuso do sistema”. A manifestação do comportamento ”abuso do sistema”pode acontecer, em diferentes situações, quando o aprendiz: i) pede ajuda de forma rápida e repetidamente até que o tutor dê ao aprendiz a resposta correta [Aleven and Koedinger 2000]; ii) insere respostas rápidas e sistematicamente até que o tutor identifique a resposta correta permitindo que o aprendiz avance abusando das provas [Aleven et al. 2006]. Tal comportamento tem sido relatado em vários estudos anteriores, por exemplo, em [d Baker et al. 2008, d Baker et al. 2010], incluindo, mais recentemente, um estudo no âmbito da educação em computação [Rocha et al. 2023].

A identificação desse tipo de comportamento tem se prestado a, potencialmente, incrementar a qualidade da informação contida em um modelo de aprendiz, que também inclui o comportamento ”abuso do sistema”. Para realizar essa identificação, foram criadas soluções que usam técnicas de engenharia de conhecimento [Aleven et al. 2006, Johns and Woolf 2006] e aprendizado de máquina [d Baker et al. 2008, d Baker et al. 2010]. Outros autores, em trabalho anterior [Gong et al. 2010b] descobriram que o comportamento do ”abuso do sistema”está mais relacionado às características do aprendiz do que às características de habilidade do domínio do conhecimento. Neste artigo, temos como objetivo analisar a manifestação do comportamento de ”abuso do sistema”, por aprendizes, em um domínio ainda pouco estudado: programação de computadores. Assim, apresentamos um modelo para detectar o comportamento de ”abuso do sistema”, utilizando algoritmos de aprendizagem de máquina. Após a detecção de tal comportamento, analisamos a relação de 16 conceitos, geralmente, abordados de programação introdutória. Com isso, nossos resultados mostram que os casos de ”abuso do sistema”acontecem em maioria nos problemas associados a controle de fluxo e que os aprendizes consideram mais difíceis. Dessa forma, nós abordamos as seguintes questões de pesquisa:

Questão de Pesquisa 01: Qual algoritmo de aprendizagem de máquina apresenta o melhor desempenho na classificação do comportamento ”abuso do sistema”em programadores novatos em atividades de a resolução de problemas de programação?

Questão de Pesquisa 02: Quais são os conceitos de programação associados aos problemas que têm relação com a manifestação do comportamento ”abuso do sistema”em programadores novatos em atividades de a resolução de problemas de programação?

Para isso, desenvolvemos um modelo para detectar o comportamento de ”abuso do sistema”em aprendizes de programação de computadores em atividades de resolução de problemas. O desenvolvimento do detector envolveu o treinamento de algoritmos de aprendizado de máquina simples e combinados para classificar o comportamento do sistema e obter modelos com melhor precisão, incluindo explorar, avaliar e comparar diferentes algoritmos de classificação única, como árvore de decisão, K vizinhos mais próximos (KNN), rede neural e Máquinas de vetores de suporte (SVM). Além disso, consideramos também combinação classificadores (ensemble) como Random Forest (RF) [Liu et al. 2012], Gradient Boosting Machine (Gradient Boosting), [Friedman 2001] Adaptive Boosting (AdaBoost) [Hastie et al. 2009] e XGBoost [Chen and Guestrin 2016]. Como um dos resultados deste estudo, avaliamos os classi-

ficadores usando seis métricas: curva ROC, acurácia, cobertura, precisão, *F-measure* e Kappa. Por fim, nossos resultados mostram que os casos de ”abuso do sistema”acontecem em maioria nos problemas associados a controle de fluxo e que os aprendizes consideram mais difíceis. Além disso, existe uma forte relação entre o fato de os aprendizes cometerem erros em problemas associados a conceitos de controle de fluxo e o comportamento de ”abuso do sistema”. Assim, em relação aos classificadores, notamos que considerando a média de todos os resultados do algoritmo referentes a todas as medidas, o classificador ensemble XGBoost obteve o melhor desempenho.

2. Contextualização e Trabalhos Relacionados

O comportamento de ”abuso do sistema”geralmente é manifestado, quando o aprendiz está pedindo ajuda rápida e repetidamente até que o tutor dê ao aprendiz a resposta correta [Aleven and Koedinger 2000] ou inserindo as respostas rápida e sistematicamente [Baker et al. 2004b]. Assim, este comportamento tem sido associado a uma pior aprendizagem [Baker et al. 2004b, Baker et al. 2005] e para os autores [Gong et al. 2010a] quando os aprendizes manifestam o comportamento ”abuso do sistema”, não aprendem quase nada. Além disso, os autores avaliaram se o comportamento de ”abuso”é associado a um aprendizado imediato mais pobre, aplicando o método de decomposição de aprendizado [Beck and Mostow 2008], onde o desempenho em uma determinada habilidade em um determinado momento é previsto com base no número de vezes que o aprendiz já se envolveu em comportamento ”abusos do sistema”nessa habilidade. Eles descobriram que o número de comportamentos de ”abusos do sistema”anteriores está associado a menos aprendizado. Assim, este comportamento tem sido observado em vários ambientes de aprendizagem, desde jogos educativos a newsgroups online [Cheng and Vassileva 2005], e tem sido repetidamente documentado em sistemas tutores inteligentes [Baker et al. 2004b, Mostow et al. 2002].

2.1. Trabalhos relacionados

No trabalho de [Baker et al. 2005], os autores concentram-se em verificar se há uma maior manifestação de comportamento de ”abuso do sistema”nas etapas mais difíceis do problema. Após o desenvolvimento e aplicação do modelo, o resultado foi a classificação dos aprendizes em três grupos. No primeiro grupo estão os aprendizes que nunca apresentaram comportamento de ”abuso do sistema”; no segundo, aprendizes que apresentaram comportamento de ”abuso do sistema”mas tiveram desempenho ruim, denominados ”GAMED-HURT”, e, aprendizes que apresentaram comportamento de ”abuso do sistema”e tiveram bom desempenho, denominados ”GAMED NO-HURT.”Em outro trabalho, os autores estudam os objetivos, atitudes, comportamento e aprendizado de aprendizes do ensino médio em um tutor de matemática [Baker et al. 2005]. Para o estudo, duas fontes de dados foram combinadas: um questionário sobre as motivações e crenças dos aprendizes e registros das ações de cada aprendiz com o tutor. Além disso, as instâncias de ”abuso do sistema”foram divididas em dois grupos: ”abuso do sistema”sem impacto e ”abuso”com impacto negativo na aprendizagem. Os autores descobriram que a frequência do comportamento não se correlaciona com uma medida conhecida de metas de desempenho; em vez disso, o comportamento de ”abuso do sistema”está relacionado a não gostar de computadores e do sistema tutor.

Usando o sistema tutor de álgebra cognitiva, no trabalho de [Paquette et al. 2014] foi construído um modelo cognitivo baseado em dados do sistema e conhecimento. Os

dados continham informações sobre as ações dos aprendizes durante o uso do sistema, como tempo, dicas, contexto do problema, entrada que o usuário inseriu e avaliação do sistema, e com isso, eles podem identificar se o sistema está sendo ”abusado”ou não. Nesse sistema, os sistemas tutores fornecem dicas, que podem ser dadas sempre que o aprendiz precisar de ajuda. Assim, quanto mais o estudante pergunta, mais específicas são as dicas. O sistema de Pontos Decimais tem como foco a matemática para aprendizes do 5º e 6º ano e tem como objetivo auxiliar no desenvolvimento dos aprendizes por meio de alguns mecanismos de ”abuso do sistema”[Richey et al. 2021].

Na pesquisa de [Paquette et al. 2014], são apresentados 13 padrões para detecção do comportamento e em [Paquette and Baker 2017], os autores utilizam os padrões para construir um detector do comportamento ”abuso do sistema”. Este comportamento também foi estudando em [Rocha et al. 2023], onde os autores desenvolveram um detector do comportamento ”abuso do sistema”analisando a influência de variáveis como (i) o nível de crença do aprendiz e o nível do sistema de crença no nível de dificuldade do problema em detectar o ”abuso do sistema”; e (ii) submissão parcial do problema associado à completa submissão, em forma de programa, na detecção de manipulação do sistema.

3. Materiais e Métodos

Inicialmente, nessa seção, discutimos as características do conjunto de dados utilizado, em seguida apresentamos os conceitos de programação analisados e por fim, os algoritmos utilizados na classificação e as métricas aplicadas na avaliação.

3.1. Dados

Neste trabalho analisamos o mesmo conjunto de dados utilizado no estudo em [Rocha et al. 2023]. Este conjunto foi obtido por meio de um sistema tutor cognitivo aplicado em um curso de universitário durante o semestre de 2022. O conjunto de dados é composto por dados sobre tentativas de resolução de problemas de programação, sendo formado por 5.307 instâncias. Assim, este conjunto está dividido em duas classes: classe 0: se não há manifestação de comportamento de ”abuso de sistema”; classe 1: se há manifestação de comportamento de ”abuso de sistema”. A Tabela 1 mostra a quantidade de tentativas de resolução de problemas por classe.

Tabela 1. Distribuição do conjunto de dados por classe

Classe	Descrição	Quantidade
0	não há comportamento de ”abuso do sistema”	4.044
1	há comportamento de ”abuso do sistema”	1.263
	Total	**5.307**

3.2. Conceitos de programação

Na Tabela 2, elencamos os 16 conceitos de programação que analisamos a relação com o comportamento "abuso do sistema". Para isso, em nosso conjunto de dados os conceitos estão associados aos problemas respondidos pelos aprendizes em sessões de resolução de problemas. Assim, cada conceito foi associado a mesma quantidade de problemas e um problema poderia ser associado a mais de um problema.

Tabela 2. Conceitos de programação

Identificador	Descrição
[C01]	Comentários
[C02]	Tipos de dados
[C03]	Declaração de variável
[C04]	Manipulação de variáveis
[C05]	Escopo da variável
[C06]	Constantes
[C07]	Incremento de variável
[C08]	Decréscimo de variável
[C09]	Operadores relacionais
[C10]	Operadores booleanos
[C11]	Operadores aritméticos
[C12]	Estrutura condicional simples
[C13]	Estrutura condicional aninhada
[C14]	Estrutura de repetição
[C15]	Variáveis Complexas: Vetores
[C16]	Variáveis Complexas: Matrizes

3.3. Processamento de Dados, Classificação e Avaliação

Quando utiliza-se algoritmos classificadores de aprendizagem de máquina, para evitar o problema da obtenção de modelos superestimados, deve-se dividir o conjunto de dados em dois conjuntos treinamento e teste [Friedman et al. 2001]. Por isso, subdividimos nosso conjunto de dados em dois: treinamento (75%) e teste (25%). Além disso, de acordo a Tabela 1, há mais exemplos na classe 0 que na classe 1, ou seja as classes estão desbalanceados. Um conjunto de dados é dito desbalanceado, quando existem muito menos casos de algumas classes do que de outras. No entanto, os algoritmos classificadores são muito sensíveis a dados desbalanceados e tendem a valorizar as classes predominantes ignorando as classes menos representativas[Phua et al. 2004]. Por isso, para lidar com as classes desbalanceadas, utilizamos o algoritmo SMOTE da biblioteca imblearn[1], que consegue simular os dados em uma quantidade menor utilizando vizinhos próximos para esse fim, evitando o modelo de ser tendencioso em uma classe.

3.3.1. Algoritmos

Para desenvolver os modelos classificadores, analisamos o desempenho de dois tipos de classificadores: simples e ensemble. Para os classificadores simples, aplicamos os algorit-

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

mos de árvore de decisão, como os autores utilizaram em [Baker and de Carvalho 2008], como em SVM [d Baker et al. 2010], KNN e redes neurais (MLP). Em termos de algoritmos de aprendizado ensemble [Yang et al. 2013], o uso de *random forest* tem aparecido na literatura entre as abordagens mais utilizadas para tarefas de classificação de aprendizes em outros contextos [Alamri et al. 2021]. Assim, para construir nosso modelo, empregamos vários métodos de ensemble concorrentes, como segue: *Random Forest* (RF) [Liu et al. 2012], *Gradient Boosting Machine* (*Gradient Boosting*), [Friedman 2001] *Adaptive Boosting* (AdaBoost) [Hastie et al. 2009] e XGBoost [Chen and Guestrin 2016] para prosseguir com a análise exploratória.

3.3.2. Avaliação

Avaliamos os desempenhos dos classificadores por meio de seis métricas: curva ROC, acurácia, precisão, cobertura, *F-measure* e Kappa, no nível de significância $(p < 0{,}05)$. Primeiro, a área da curva ROC é um recurso que relaciona taxas de verdadeiros positivos e falsos positivos. A área sob esta curva varia entre 0,5 e 1 e mostra a capacidade de um modelo realizar classificações corretas e incorretas [Gonçalves et al. 2014]. Essa medida revela a probabilidade de que, se o detector comparar duas sequências, uma envolvendo ”abuso do sistema”e outra não envolvendo ”abuso do sistema”, ele identificará corretamente qual sequência é qual.

A precisão indica o desempenho geral do modelo. A cobertura, também chamada de taxa ou sensibilidade de verdadeiros positivos, é a porcentagem de exemplos positivos gerais de verdadeiros positivos. A precisão é a porcentagem de verdadeiros positivos sobre todos os classificados como positivos. A medida *F-measure* refere-se à precisão e cobertura. Então, usamos a segunda, terceira, quarta e quinta medições para avaliar se o detector estava se equilibrando adequadamente entre identificar ”abuso do sistema”e evitar falsos positivos. O *Kappa* avalia se o detector determina as sequências de ação corretas com ”abuso do sistema”em vez do acaso. Um Kappa de 0 indica que o detector funciona aleatoriamente e um *Kappa* de 1 indica que o detector funciona perfeitamente.

4. Resultados e Discussões

Uma das principais contribuições deste trabalho é fornecer *insights* sobre as principais conceitos de programação que levam programadores novatos a manifestar comportamento de ”abuso do sistema”ao resolver problemas de programação. Com isso, podemos direcionar maior atenção a tais conceitos e traçar intervenções personalizadas para aprendizes que manifestam tal comportamento.

4.1. Detectanto comportamento de ”Abuso do sistema”

Tendo desenvolvido um detector razoavelmente prático de comportamento de ”abuso do sistema”em geral, discutimos nossas tentativas de diferenciar os tipos de comportamento de ”abuso do sistema”nesta seção. Em todos os casos, o mesmo método de validação cruzada foi usado como acima. Para isso, apresentamos na Tabela 3 todos os valores das medidas recuperadas de cada algoritmo.

Diferenciando todos os algoritmos, de acordo com os valores da Tabela 3 para precisão, o nível de precisão dos resultados obtidos pela aplicação de cada classificador,

Tabela 3. Análise comparativa entre as pontuações obtidas em cada modelo de aprendizagem de máquina

	Acurácia	Precisão	Recall	*F-measure*	Kappa	AUC
Decision tree	0,84	0,88	0,85	0,84	0,34	0,79
K-NN	0,77	0,78	0,78	0,79	0,41	0,79
SVM	0,62	0,71	0,67	0,64	0,31	0,71
MLP	0,85	0,84	0,84	0,83	0,53	0,76
RF	0,90	0,91	0,89	0,90	0,49	0,81
GB	0,91	0,91	0,90	0,91	0,65	0,81
AdaBoost	0,89	0,89	0,89	0,89	0,60	0,83
XGBoost	0,91	**0,92**	0,91	0,90	0,47	0,82

os algoritmos XGBoost e Random forest também tiveram a melhor eficiência média de 91%. No entanto, considerando a precisão, cobertura e F-measure, o modelo foi mais preciso em capturar o comportamento relacionado ao ”abuso do sistema”foi o XGBoost (Precision=92%, Recall = 90% e F-measure =91%), apresentando desempenho um pouco melhor. Considerando os casos de falsos positivos e verdadeiros positivos, o AdaBoost (AUC=0,83) teve melhor desempenho. O *Gradient Boosting* obteve um melhor desempenho considerando o *Kappa* (0,63).

4.2. A relação dos conceitos de programação de computadores

Após a etapa de desenvolvimento do detector do comportamento ”abuso do sistema”, analisamos as tentativas de resolução associadas a cada problemas e assuntos. Com isso, percebemos que, de acordo com a Figura 1, a maioria (38.55%) das tentativas relacionadas a comportamento de ”abuso do sistema”estão associadas a problemas com os assuntos sobre controle de fluxo. $[C12]$ estrutura condicional simples (12.5%), $[C13]$ estrutura condicional aninhada (10.25%), $[C14]$ estrutura repetitiva (15.8%). Em 22.8% das tentativas estão relacionadas à variáveis complexas: $[C15]$ Vetores (12.5%) e $[C16]$ Matrizes (10.3%).

Além disso, cinco dos erros mais comuns cometidos por programadores novatos em Java estão associados a conceitos de controle de fluxo [Brown and Altadmri 2017]. Exemplos: ”Parênteses não balanceados, colchetes ou chaves e aspas, ou usando esses diferentes símbolos de forma intercambiável”; ”Ponto-e-vírgula incorreto imediatamente após uma condição *if* ou após a condição for ou while, que involuntariamente forma o corpo *if/loop*”; ”Separadores incorretos em loops for (usando vírgulas em vez de ponto e vírgula)”; ”Inserindo a condição de uma instrução *if* entre chaves em vez de parênteses”[Brown and Altadmri 2017]. Assim, existe uma forte relação entre o fato de os aprendizes cometerem erros nesses problemas associados às disciplinas $[C12]$,$[C13]$,$[C14]$,$[C15]$ e $[C16]$ na Tabela 2, considere o problema e opte por manifestar o comportamento de ”abuso do sistema”. Essa relação pode ser parcialmente observada em pesquisas anteriores, no domínio da matemática, que sugerem baixo conhecimento prévio combinado com a dificuldade do problema como causa do comportamento ”abuso do sistema”. Além disso, para outros autores, os aprendizes apresentam tal comportamento nas etapas mais difíceis do problema [Baker et al. 2004a].

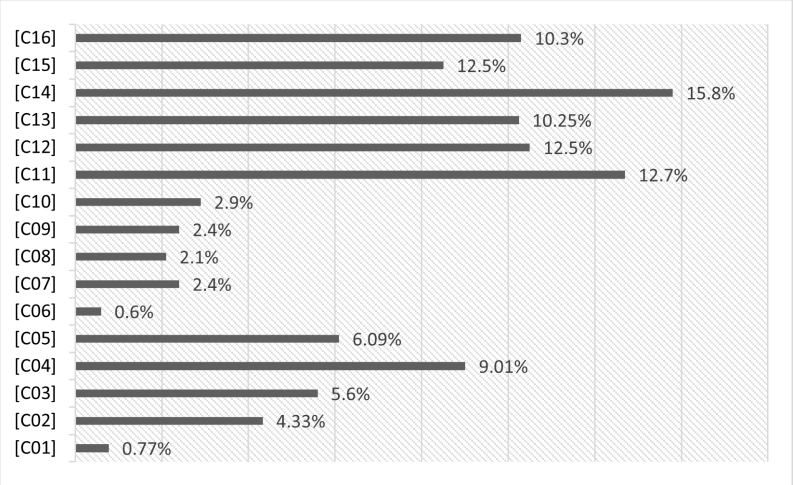

Figura 1. Percentual de tentativas com comportamento ”abuso do sistema”por conceitos de programação

5. Conclusão

Neste trabalho, desenvolvemos um estudo comparativo de algoritmos para então selecionarmos uma solução computacional usando técnicas de aprendizado de máquina supervisionado para uma tarefa de detecção automática de situações nas quais os aprendizes usam o comportamento ”abuso do sistema”, durante o processo de resolução de problemas de programação. Daí então, apresentamos um estudo sobre a relação de alguns recursos relacionados à identificação de ”abuso do sistema”no comportamento do sistema. Assim, os resultados indicaram que a criação de modelos preditivos eficazes para esta tarefa é viável e efetivo. Nesse sentido, a principal contribuição deste trabalho é a geração de modelos de aprendizado de máquina com bom desempenho para classificar a presença do comportamento ”abuso do sistema”relacionado aos conceitos de programação de computadores. Com relação aos classificadores, notamos que considerando a média de todos os resultados do algoritmo referentes a todas as medidas, o classificador do ensemble XGBoost obteve o melhor desempenho. Além disso, de acordo com todas as medidas, os classificadores ensemble tiveram melhor desempenho que os classificadores individuais. Além disso, percebemos que existe uma forte relação entre o fato dos aprendizes cometerem erros nesses problemas associados a alguns conceitos da Tabela 2, considerarem o problema e optarem por manifestar ”abusar do sistema”.

Referências

Alamri, A., Sun, Z., Cristea, A. I., Stewart, C., and Pereira, F. D. (2021). Mooc next week dropout prediction: weekly assessing time and learning patterns. In *Intelligent Tutoring Systems: 17th International Conference, ITS 2021, Virtual Event, June 7–11, 2021, Proceedings 17*, pages 119–130. Springer.

Aleven, V. and Koedinger, K. R. (2000). Limitations of student control: Do students know when they need help? In *International conference on intelligent tutoring systems*, pages 292–303. Springer.

Aleven, V., Mclaren, B., Roll, I., and Koedinger, K. (2006). Toward meta-cognitive tutoring: A model of help seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education*, 16(2):101–128.

Baker, R. and de Carvalho, A. (2008). Labeling student behavior faster and more precisely with text replays. In *Educational Data Mining 2008*.

Baker, R. S., Corbett, A. T., and Koedinger, K. R. (2004a). Detecting student misuse of intelligent tutoring systems. In *International conference on intelligent tutoring systems*, pages 531–540. Springer.

Baker, R. S., Corbett, A. T., Koedinger, K. R., and Wagner, A. Z. (2004b). Off-task behavior in the cognitive tutor classroom: When students"game the system". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 383–390.

Baker, R. S., Roll, I., Corbett, A. T., and Koedinger, K. R. (2005). Do performance goals lead students to game the system? In *AIED*, pages 57–64.

Beck, J. E. and Mostow, J. (2008). How who should practice: Using learning decomposition to evaluate the efficacy of different types of practice for different types of students. In *International conference on intelligent tutoring systems*, pages 353–362. Springer.

Brown, N. C. and Altadmri, A. (2017). Novice java programming mistakes: Large-scale data vs. educator beliefs. *ACM Transactions on Computing Education (TOCE)*, 17(2):1–21.

Cavalcanti, A. P., de Mello, R. F. L., de Miranda, P. B. C., and de Freitas, F. L. G. (2020). Análise automática de feedback em ambientes de aprendizagem online. In *Anais do XXXI Simpósio Brasileiro de Informática na Educaçao*, pages 892–901. SBC.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Cheng, R. and Vassileva, J. (2005). Adaptive reward mechanism for sustainable online learning community. In *AIED*, pages 152–159.

d Baker, R. S., Corbett, A. T., Roll, I., and Koedinger, K. R. (2008). Developing a generalizable detector of when students game the system. *User Modeling and User-Adapted Interaction*, 18(3):287–314.

d Baker, R. S., Mitrović, A., and Mathews, M. (2010). Detecting gaming the system in constraint-based tutors. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 267–278. Springer.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). The elements of statistical learning. vol. 1 springer series in statistics. *New York*.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Ghaleb, E., Popa, M., Hortal, E., Asteriadis, S., and Weiss, G. (2018). Towards affect recognition through interactions with learning materials. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 372–379. IEEE.

Gonçalves, L., Subtil, A., Oliveira, M. R., and de Zea Bermudez, P. (2014). Roc curve estimation: An overview. *REVSTAT-Statistical journal*, 12(1):1–20.

Gong, Y., Beck, J., Heffernan, N. T., and Forbes-Summers, E. (2010a). The impact of gaming (?) on learning at the fine-grained level. In *Proceedings of the 10th International Conference on Intelligent Tutoring Systems (ITS2010) Part*, volume 1, pages 194–203.

Gong, Y., Beck, J. E., Heffernan, N. T., and Forbes-Summers, E. (2010b). The fine-grained impact of gaming (?) on learning. In *International Conference on Intelligent Tutoring Systems*, pages 194–203. Springer.

Hastie, T., Rosset, S., Zhu, J., and Zou, H. (2009). Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360.

Johns, J. and Woolf, B. (2006). A dynamic mixture model to detect student motivation and proficiency. In *AAAI*, pages 163–168.

Liu, Y., Wang, Y., and Zhang, J. (2012). New machine learning algorithm: Random forest. In *Information Computing and Applications: Third International Conference, ICICA 2012, Chengde, China, September 14-16, 2012. Proceedings 3*, pages 246–252. Springer.

Mostow, J., Beck, J., Chalasani, R., Cuneo, A., Jia, P., Kadaru, K., et al. (2002). A la recherche du temps perdu, or as time goes by: Where does the time go in a reading tutor that listens? In *International conference on intelligent tutoring systems*, pages 320–329. Springer.

Paquette, L. and Baker, R. S. (2017). Variations of gaming behaviors across populations of students and across learning environments. In *Artificial Intelligence in Education: 18th International Conference, AIED 2017, Wuhan, China, June 28–July 1, 2017, Proceedings 18*, pages 274–286. Springer.

Paquette, L., de Carvalho, A. M., and Baker, R. S. (2014). Towards understanding expert coding of student disengagement in online learning. In *CogSci*.

Phua, C., Alahakoon, D., and Lee, V. (2004). Minority report in fraud detection: classification of skewed data. *Acm sigkdd explorations newsletter*, 6(1):50–59.

Richey, J. E., Zhang, J., Das, R., Andres-Bray, J. M., Scruggs, R., Mogessie, M., Baker, R. S., and McLaren, B. M. (2021). Gaming and confrustion explain learning advantages for a math digital learning game. In *Artificial Intelligence in Education: 22nd International Conference, AIED 2021, Utrecht, The Netherlands, June 14–18, 2021, Proceedings, Part I*, pages 342–355. Springer.

Rocha, H. J. B., de Azevedo Restelli Tedesco, P. C., de Barros Costa, E., and Rocha, J. S. (2023). An approach for detecting gaming the system behavior in programming problem-solving. In *International Conference on Intelligent Tutoring Systems*, pages 75–87. Springer.

Yang, D., Sinha, T., Adamson, D., and Rosé, C. P. (2013). Turn on, tune in, drop out: Anticipating student dropouts in massive open online courses. In *Proceedings of the 2013 NIPS Data-driven education workshop*, volume 11, page 14. Lake Tahoe, NV.