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

Avaliando influência de características de desempenho na predição resultado acadêmico em disciplinas de programação

Luciano de Souza Cabral[1,4], Filipe Dwan Pereira[2,4] e Rafael Ferreira Mello[3,4]

[1] Campus Jaboatão dos Guararapes - Instituto Federal de Pernambuco (IFPE)
CEP: 54080-000 - Jaboatão dos Guararapes - PE - Brasil

[2] Campus Boa Vista - Universidade Federal de Roraima (UFRR)
CEP: 69310-000 - Boa Vista - RR - Brasil

[3] Campus Recife - Universidade Federal Rural de Pernambuco (UFRPE)
CEP: 55.292-901 - Recife - PE - Brasil

[4] Centro de Estudos Avançados do Recife (CESAR)
CEP: 50030-390 - Recife - PE - Brasil

{lscabral, filipedwan, rafaelflmello}@gmail.com

Abstract. *This paper describes an assessment of reducing important features in prediction/performance classification problems in programming disciplines. It is common in machine learning problems to utilize the most relevant features to enhance the model. However, we analyze a dataset from Automated Code Correction Environment (ACCE) of a Brazilian university. We identify the features, assess their significance, and evaluate the model's performance when using or not using the key features from the dataset. Our findings indicate that the reduction does not significantly affect the model's performance.*

Keywords*: Machine Learning, Feature Reduction, Artificial Intelligence, Education.*

Resumo. *Este artigo aborda uma análise da redução de características essenciais em cenários de predição de desempenho nas disciplinas de programação. Embora seja comum em aprendizado de máquina utilizar as características mais relevantes para aprimorar o modelo, este estudo se concentra em examinar um conjunto de dados proveniente de um Ambiente de Correção Automática de Código (ACAC) de uma universidade brasileira. O estudo identifica as características, avalia sua importância e investiga o impacto no desempenho do modelo ao utilizar ou não as características principais do conjunto de dados. Os resultados demonstram que a redução das características não exerce um efeito substancial no desempenho do modelo.*

Palavras-chave*: Aprendizagem de Máquina, Redução de características, Inteligência Artificial, Educação.*

1. Introdução

A previsão do desempenho dos estudantes em turmas de programação é um tópico que tem sido objeto de estudo por várias décadas. As abordagens e métodos que abordam

essa questão estão em constante evolução, apesar de parte substancial desses estudos serem baseados na análise estática do desempenho prévio estudantil, até de antes de sua matrícula na universidade [Ahadi et al., 2016, Pereira et al., 2020a, Pereira et al., 2020b]. Entretanto, embora haja casos padrões, também existem casos que é possível observar a dinamicidade comportamental estudantil ao longo do semestre, ou seja, o estudante que inicia a disciplina mal pode terminar bem.

A literatura mostra que algoritmos de aprendizagem de máquina sofrem redução de desempenho quando o conjunto de dados para construção do classificador contém muitos atributos [Pereira et al., 2020b]. Técnicas de redução de dimensionalidade podem melhorar o desempenho do modelo, reduzir custos computacionais e tornar os resultados mais compreensíveis [Pereira et al., 2020b]. Para esse fim, utiliza-se técnicas que se dividem em duas categorias, agregação e seleção. A primeira, combina atributos originais usando funções lineares, resultando em menos atributos; já a segunda mantém apenas atributos relevantes, descartando os irrelevantes [Broder, 2021]. O aprendizado por transferência envolve aplicar conhecimento adquirido de uma tarefa para auxiliar em outra relacionada, sendo especialmente valioso em situações com conjuntos de dados pequenos ou limitados [Foresti, 2023]. Vale ressaltar que muitas abordagens de seleção de características resultam na identificação de atributos altamente correlacionados com a classe alvo que pode gerar um overfitting no modelo desenvolvido

Diante deste contexto, este estudo propõe a utilização de dados oriundos do Ambiente de Correção Automática de Código (ACAC) denominado CodeBench[1] da Universidade Federal do Amazonas para treinamento e análise de desempenho de modelos de aprendizagem de máquina, avaliando em dois diferentes momentos, um contendo as características mais relevantes e outro retirando a feature de maior importância encontrada. Neste sentido, o principal objetivo deste trabalho é apresentar uma discussão sobre as características que deveria ser consideradas em estudos de predição de desempenho.

Para isso, este artigo está dividido em: estado da arte envolvido e os trabalhos relacionados na Seção 2; a metodologia utilizada para os experimentos na Seção 3; a discussão e os resultados na Seção 4; as considerações finais, limitações e implicações futuras do estudo na Seção 5; e por fim registra-se as referências utilizadas

2. Contextalização

Na revisão sistemática [Pereira et al., 2020b], os autores analisaram cerca de 70 artigos com foco na melhoria do processo de ensino e aprendizagem em turmas de programação utilizando modelos de aprendizagem de máquina, mas apenas dois trabalhos [Azcona et al. 2018, Quille, K., Bergin 2019] realizaram as predições utilizando dados de turmas reais, demais apenas se limitavam a construir o modelo, sem maiores intervenções reais.

Os dados que foram utilizados pelos modelos no cenário supracitado, geralmente são oriundos de um Ambiente de Correção Automática de Código (ACAC), onde a comunidade estudantil interage com conteúdos e atividades na área de programação. Assim, na resolução das atividades, são coletados dados dos estudantes a partir de logs da interação deles no Ambiente de Desenvolvimento Integrado (do inglês, *Integrated Development Environment* - IDE) incorporados aos Ambientes.

[1] https://codebench.icomp.ufam.edu.br/

No processo de resolução das atividades, os discentes codificam, executam, depuram, testam, e ao final submetem o código desenvolvido para ser avaliado por um docente ou por um juiz online, e, ao longo dessa jornada, uma série de atributos são coletados automaticamente para enriquecimento do banco de dados.

Diversos algoritmos de Aprendizado de Máquina foram empregados para classificação, incluindo *ensembles*, árvores de decisão, modelos probabilísticos (como redes bayesianas ou naive bayes), máquinas de vetores de suporte (SVM) e redes neurais. Além disso, 17 estudos utilizaram algoritmos de clusterização para agrupar alunos com perfis semelhantes e avaliar sua relação com o desempenho. A preferência por modelos interpretáveis, como árvores de decisão e clusterização, destaca-se, embora possam oferecer resultados menos robustos. Redes neurais, embora mais robustas, tendem a ser modelos caixa-preta. A interpretação de algoritmos de aprendizado de máquina ou questão de justiça algorítmica foram sub-exploradas [Pereira et al., 2020b].

3. Metodologia e Experimentos

Nesta seção, explana-se sobre a metodologia utilizada, incluindo-se o dataset, cenário educacional que foi aplicado os métodos e técnicas de aprendizagem de máquina, métricas associadas a avaliação dos modelos.

3.1. Dataset

O dataset utilizado neste trabalho é oriundo de um projeto da UFAM denominado CodeBench, no qual possui seu dataset disponível[2]. Deste modo, utilizou-se duas versões do dataset: a primeira continha dados comportamentais de 1655 estudantes de 6 semestres (2016.1-2018.2), incluindo informações sobre o uso do ambiente de correção automática de código (ACAC), índices de procrastinação, uso da IDE, atalhos de teclado, acesso ao sistema, variáveis e operadores utilizados, entre outros. A segunda versão continha apenas dados sociodemográficos, excluindo informações pessoais de acordo com a Lei Geral de Proteção de Dados. Esses dados incluíam gênero, idade, turno, disponibilidade de PC e internet em casa, compartilhamento do computador com outros membros da família, experiência anterior em programação, conclusão de cursos técnicos na área, entre outros. É importante destacar que os dados comportamentais foram coletados na semana 11 dos cursos, quando apenas a primeira nota de dois exames estava disponível, em geral, aplicados na disciplina de Lógica de Programação.

Em virtude das versões terem compatibilidade parcial, foram geradas outras versões do dataset, efetuando operações de união, intersecção, limpeza de itens nulos, remoção de duplicatas, mantendo a versão mais atualizada, entre outras operações.

3.2. Cenário educacional

O cenário educacional ao qual este estudo se propõe a investigar é a predição de desempenho de estudantes na disciplina de Lógica de Programação, permitindo intervenções oportunas para melhorar o ensino e a aprendizagem em tempo hábil. Tal estudo justifica-se pelo fato de que turmas de programação possuem alto índice de reprovação, motivando pesquisas, como predição do desempenho do aluno, para orientar

[2] https://codebench.icomp.ufam.edu.br/dataset/

a tomada de decisões [Pereira et al., 2020b].

3.3. Métricas de Avaliação e Algoritmos

Neste primeiro estudo, avaliar-se-ão os desempenhos dos modelos por meio de métricas convencionais, como matriz de confusão (precisão, cobertura, *f1-score*, acurácia), juntamente com a métrica estatística *Cohen Kappa*, amplamente utilizada na área de Inteligência Artificial na Educação (AIED), a qual mede a concordância entre observadores em problemas de classificação. Para esse fim, empregaremos o algoritmo *Decision Tree Classifier*, cujo objetivo é ter um sistema de IA explicável.

4. Experimentos e Resultados

Ao trabalhar com a base de dados, temos as seguintes características originais:

Tabela 1.Versões iniciais dos datasets

Versão	Linhas × Colunas	Descrição	Descrição
Comportamentais Original	2036 × 21	Dados dos semestres 2016.1, 2016.2, 2017.1, 2017.2, 2018.1, e 2018.2, sem dados nulos ou -1s	Id, watWinScore, procrastination, amountOfChange, event_activity, attempts, comments, lloc, systemAccess, firstExamGrade, deleteAvg, events, correctness, correctnessCodeAct, copyPaste, syntaxError, ideUsage, keystrokeLatency, errorQuotient, countVar, finalGrade
Sociodemográficos Original	2058 × 23	Dados dos semestres 2016.1, 2016.2, 2017.1, 2017.2, 2018.1, e 2018.2, contém dados nulos e -1s	Id, course name, high school name, school type, shift, graduation year, has a PC at home, share this PC with other people at home, Internet, previous experience of any computer language, worked or interned before the degree, company name, year started working, year stopped working, started other degree programmes, degree course, institution name, year started this degree, year stopped this degree, sex, age, civil status, have kids

Neste sentido era necessário efetuar um tratamento dos dados, neste sentido, efetuou-se uma verificação de colunas que seriam mais úteis para computação (numéricas) e colunas categóricas como turno, gênero, e estado civil.

Efetuou-se o respectivo tratamento (*OneHot*) além do tratamento dos dados nulos, dos -1s encontrados, no qual, é provável terem sido inseridos pela falta de preenchimento de dados por parte dos estudantes. Outra questão observável é a existência de idades inferiores a 17-18 anos, idade mínima requerida para se matricular em uma universidade, idades abaixo dessa faixa (2,3,4,7) foram eliminadas da base.

Utilizou-se a coluna *Id* para efetuar o *merge*, e identificar duplicadas, depois efetuar sua remoção, após isso a coluna foi removida. A coluna alvo foi a *finalGrade*, que foi normalizada para 0 (reprovações) e 1 (aprovações). Ao final selecionou-se 19 colunas da base comportamental (Todas da tabela 1, exceto *Id* e *finalGrade*) e 11 colunas da base sociodemográfica (*school type, shift, graduation year, has a PC at home, Internet, previous experience of any computer language, worked or interned before the degree, sex, age, civil status, have kids*), restando assim as variações a seguir:

Tabela 2.Versões geradas após o tratamento dos dados

Versão	Linhas × Colunas	%	Descrição
Merge_v1	3082 × 30	100%	Com duplicadas, com dados nulos e -1s.
Merge_v2	1655 × 30	53,69%	Sem duplicadas, com dados nulos e -1s.
Merge_v3	1113 × 30	36,11%	Sem duplicadas, sem dados nulos e -1s.
Merge_v4	1063 × 30	34,49%	Sem duplicadas, sem dados nulos e -1s, e idades corrigidas.

Com dados variando entre 100% e 34% dos dados, efetuamos medições com objetivo de averiguar o impacto da redução no desempenho do modelo. Assim rodamos o *Decision Tree Classifier* com configurações simples (*random_state=0, max_depth=3*) para as seguintes versões do dataset: *ComportST (3082, 20):* dados comportamentais pós merge e sem tratamento; *SocioDemST (3082, 12*): dados sociodemográficos pós merge e sem tratamento; *Merge_v1 (3082, 30)*: Ambas, pós merge e sem tratamento.

Tabela 3. Resultados do Decision Tree Sem Tratamento nos Dados.

Versão	Linhas,Colunas	Precision	Recall	F1-score	Accuracy	Kappa
ComportST	3082, 20	0.82	0.80	0.80	0.80	0.575
SocioDemST	3082, 12	0.43	0.65	0.52	0.65	-0.003
Merge_v1	3082, 30	0.82	0.82	0.82	0.82	0.598

Observa-se que as bases sem tratamento, dentre as características mais importantes para o algoritmo neste experimento, o *firstExamGrade* se destaca para as versões do dataset *Comport e Merge*. Já para as versões *SocioDem* a coluna mais destacada foi *age,* conforme pode ser visualizado nas figuras a seguir.

Figura 1. Feature importance: ComportST.

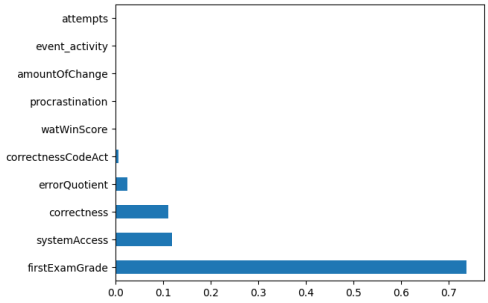


Figura 2. Feature importance: SocioDemST.

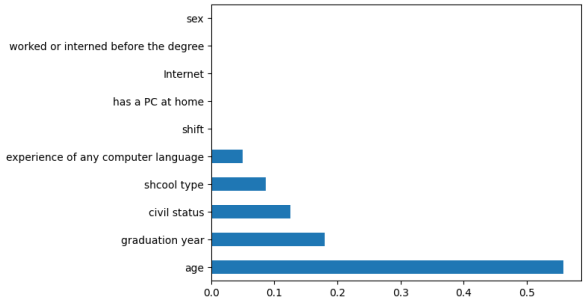
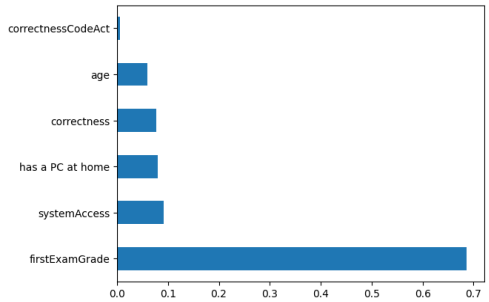


Figura 3. Feature importance: Merge_v1.



A seguir, outra versão dos experimentos, agora com as bases pós tratamento: *ComportCT (1063, 19):* dados comportamentais pós merge e tratamento; *SocioDemCT (1063, 11*): dados sociodemográficos pós merge e tratamento; *Merge_v4 (1063, 30)*: Ambas as bases, pós merge e tratamento. Os resultados foram coletados através da função *classification_report* do *SciKit Learn* para *Python*, através da média ponderada (*weighted avg*), e podem ser vistos nas tabelas a seguir.

Tabela 4. Resultados do Decision Tree Pós Tratamento nos Dados.

Versão	Linhas,Colunas	Precision	Recall	F1-score	Accuracy	Kappa
ComportCT	1063, 19	0.80	0.79	0.79	0.79	0.578
SocioDemCT	1063, 11	0.59	0.54	0.52	0.54	0.125
Merge_v4	1063, 30	0.80	0.79	0.79	0.79	0.578

Observa-se que utilizando apenas 34,49%, registra-se diminuição de apenas 0,01% de acurácia e até aumento de Kappa em 0,003 para os dados comportamentais, aumento na acurácia de 0,11%, na precisão em 0,16% e Kappa de 0,155 para os dados sociodemográficos, além de redução de apenas 0,02 na precisão e 0,03% nas métricas recall, f1, e acurácia para os dados na versão unificada (*merge*). Dentre as características mais importantes para o algoritmo neste experimento, o *firstExamGrade continua com destaque,* para as versões do dataset *Comport*, *Merge*. Já para a versão *SocioDem* a coluna mais agora é *graduation year,* conforme pode ser visualizado nas figuras a seguir.

Figura 4. Feature importance: ComportCT.

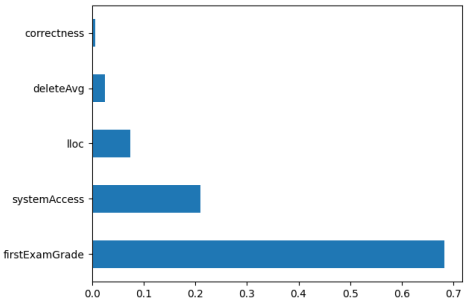


Figura 5. Feature importance: SocioDemCT.

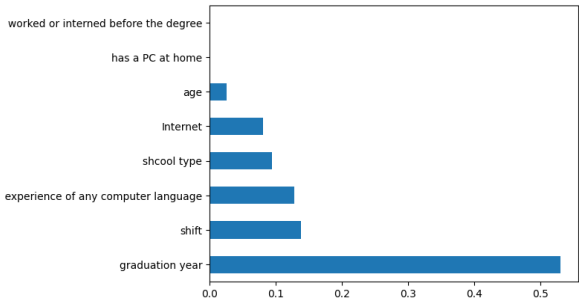
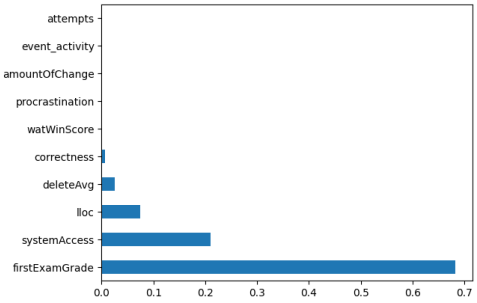


Figura 6. Feature importance: Merge_v4.



Ou seja, apenas com tratamentos adequados os modelos podem ter melhorias significativas no aspecto de justiça algorítmica, onde antes dava uma importância maior para idade, após o tratamento, embora continue dando importância para este dado, é

quase 10 vezes menor. Neste sentido, uma nova rodada de experimentos é vislumbrada, a retirada da coluna *firstExamGrade* para verificar o impacto dela no modelo.

Assim, tratou-se de avaliar a influência da coluna *firstExamGrade (FEG)* no desempenho do modelo, em uma simulação retirando-se a coluna para treinamento e predição do modelo, ao final avalia-se com as mesmas métricas para verificar a influência de sua remoção, como pode ser observado abaixo.

Tabela 5. Resultados do Decision Tree Pós remoção da coluna *FEG*.

Versão	Linhas,Colunas	Precision	Recall	F1-score	Accuracy	Kappa
ComportCT-FEG	1063, 18	0.81	0.79	0.79	0.79	0.582
Merge_v4-FEG	1063, 29	0.81	0.79	0.79	0.79	0.591

Como é possível atestar, ignorar a coluna *firstExamGrade (FEG)* melhorou levemente o desempenho do modelo, em 0.01% na precisão em ambos e no Cohen Kappa 0.004 para ComportCT-FEG e 0.013 para Merge_v4-FEG, respectivamente.

Agora para ambas as versões da base aparecem como fatores importantes para a decisão do modelo *lloc* (número de linhas lógicas no código), *systemAccess* (índice de acesso ao sistema pelo estudante), *procrastination* (índice de tempo que o estudante não interage com a aplicação), e *ideUsage* (uso da IDE contida no sistema de juiz online) para o ComportCT-FEG, já para Merge_v4-FEG contém os 2 primeiros, seguido de *age* (idade do estudante) quase empatado com o campo *procrastination,* manteve-se um padrão de fatores relevantes, sem campos sensíveis em termos de justiça algorítmica.

Figura 7. Feature importance: ComportCT-FEG.

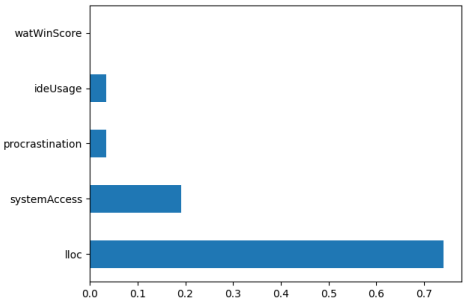
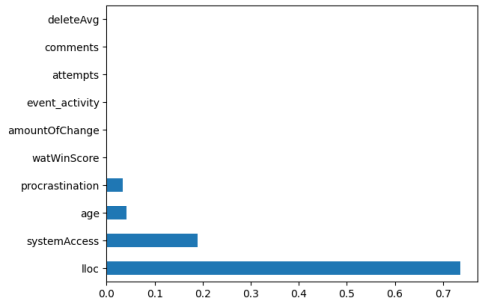


Figura 8. Feature importance: Merge_v4-FEG.



Para uma análise mais profunda do modelo, plotou-se a árvore de decisão gerada pelo algoritmo com as duas últimas versões das bases de dados discorridas no parágrafo anterior e com a análise dos fatores importantes nas figuras acima. A seguir apresenta-se as árvores de decisão do classificador usando ambas as bases.

A análise das árvores de decisão revela os fatores cruciais para o sucesso dos estudantes no sistema de juiz online. Na base de dados ComportCT-FEG (Fig. 9), esses fatores incluem a lógica em seu código, índice de acesso ao sistema, uso da IDE e controle da procrastinação. Na base Merge_v4-FEG (Fig. 10), os fatores são semelhantes, com a lógica no código, acesso ao sistema, controle da procrastinação e idade desempenhando papéis importantes. Vale notar que, apesar da inclusão da idade na segunda base, não há evidência de preconceito etário, uma vez que não foram encontrados alunos aprovados com 41 anos ou mais.

Figura 9. Decision tree: ComportCT-FEG.

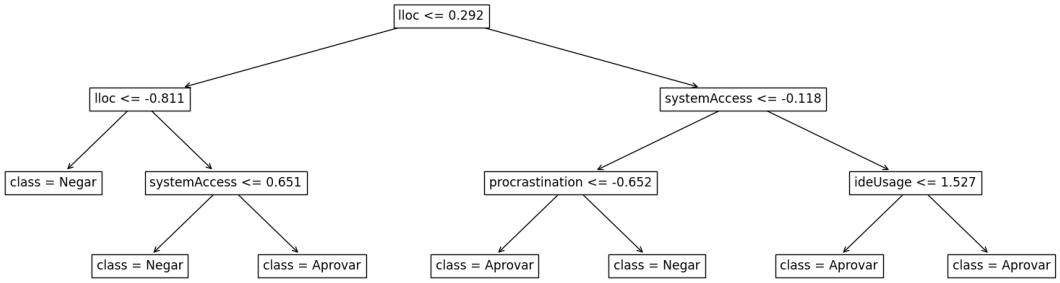
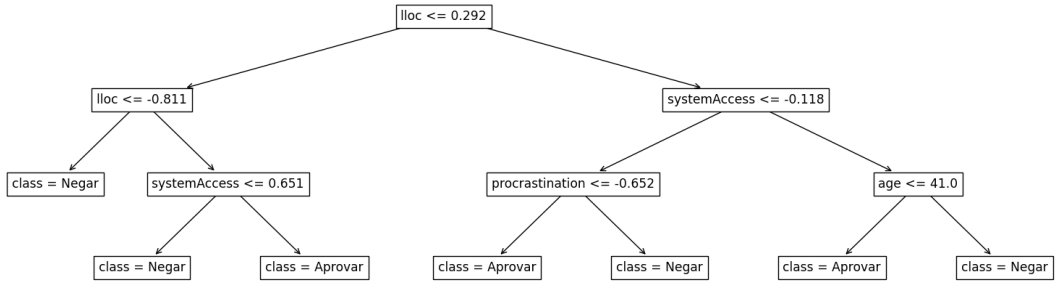


Figura 10. Decision tree: Merge_v4-FEG.



Algo que foi refletido pelo modelo, por notoriamente não haver evidências positivas para o caso analisado, obviamente, ao longo do tempo e surgindo exemplos de sucesso na aprendizagem de lógica de programação para a universidade analisada, de candidatos ou candidatas com maior idade sendo aprovado, logo o modelo ajusta-se aos dados e deve retirar tal nó *age* em 3º nível na árvore.

6. Considerações finais, Limitações e Trabalhos Futuros

Em termos de *insights* pedagógicos, este estudo sugere que um sistema de classificação contínua de desempenho de estudantes ao longo do semestre, que pode ser valioso para orientar professores, permitindo intervenções personalizadas orientadas pelo sistema.

Pode-se ainda ser utilizado para personalizar o conteúdo do curso, destacando os pontos fortes de cada aluno. Por exemplo, alunos com bom uso da IDE e lógica de programação podem receber conteúdo mais prático, enquanto outros com um perfil mais teórico podem receber materiais diferentes. Além disso, sugere a exploração do active learning, um método que permite ao modelo melhorar seu desempenho ao selecionar porções estratégicas de dados para treinamento, tornando-o mais personalizável.

Neste sentido, registrando um mix de limitações e trabalhos futuros, este estudo reconhece a necessidade de abordar questões de viés e justiça algorítmica e nos dados, utilizando ou criando medidas próprias para tal. Além disso, sugere a exploração do active learning, um método que permite ao modelo melhorar seu desempenho ao selecionar porções estratégicas de dados para treinamento, tornando-o mais personalizável.

Agradecimentos

Agradecemos ao IFPE, UFRR, UFRPE e ao CESAR pelo incentivo à pesquisa.

Referências

Azcona, D., Hsiao, I. H., & Smeaton, A. F. (2018). Personalizing computer science education by leveraging multimodal learning analytics. IEEE Frontiers in Education Conference (FIE) (pp. 1-9).

Ahadi, A., Vihavainen, A. e Lister, R. (2016). “On the number of attempts students made on some online programming exercises during semester and their subsequent performance on final exam questions”. ACM Conference on Innovation and Technology in Computer Science Education, p. 218–223.

Broder, R. S. (2021). Análise de Desempenho em Algoritmos de Aprendizado de Máquina Treinados em Dados com Viés. Trabalho de Conclusão de Curso – Bacharelado em Ciência da Computação – Universidade Federal de São Paulo – Instituto de Ciência e Tecnologia, São José dos Campos, 2021.

Fonseca, S., Oliveira, E., Pereira, F., Fernandes, D., & Carvalho, L. (2019). Adaptação de um método preditivo para inferir o desempenho de alunos de programação. Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE), 30(1), 1651. doi:https://doi.org/10.5753/cbie.sbie.2019.1651

Foresti, T. (2023). “Machine Learning Types: Learn about the different types of machine learning, including supervised, unsupervised, semi-supervised, and reinforcement learning”. Publicado em 28 de junho de 2023. Awari, disponível em: https://awari.com.br/machine-learning-types-principais-tipos-de-aprendizado-em-machine-learning, último acesso 30 ago 2023.

Pereira, F. D., Oliveira, E. H. T., Oliveira, D. B., Cristea, A. I., Carvalho, L. S., Fonseca, S. C., Toda, A., Isotani, S. (2020a). “Using learning analytics in the Amazonas: understanding students’ behavior in introductory programming”. British Journal of Educational Technology.

Pereira, F. D., Souza, L. M., Souza, L. M.; Oliveira, E. H. T.; Oliveira, D. B. F.; Carvalho, L. S. G. (2020b). “Predição de desempenho em ambientes computacionais para turmas de programação: um Mapeamento Sistemático da Literatura”. In: Simpósio Brasileiro De Informática Na Educação, 31. Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2020.

Quille, K., Bergin, S. (2019). CS1: how will they do? How can we help? A decade of research and practice. Computer Science Education, 29(2-3), 254-282.