Adaptação de um método preditivo para inferir o desempenho de alunos de programação

Samuel C. Fonseca[1], **Elaine H. T. Oliveira**[1], **Filipe D. Pereira**[2], **David Fernandes de Oliveira**[1], **Leandro S. G. Carvalho**[1]

[1]Instituto de Computação – Universidade Federal Amazonas (UFAM)

[2]Departamento de Ciência da Computação - Universidade Federal de Roraima (UFRR)

{scf,elaine,david,galvao}@icomp.ufam.edu.br,filipe.dwan@ufrr.br

Abstract. *The objective of this work is to evaluate the generalizability of a method for predicting the performance of CS1 students in an online judge. To achieve this goal, we present an adaptation of the method to a dataset generated through a educational context different from the one in which the method was originally evaluated. As a result, we observed that the prediction method, using the new dataset, achieved an accuracy of 82% in the task of predicting student performance as early as possible.*

Resumo. *Este trabalho tem por objetivo avaliar a capacidade de generalização de um método de predição de desempenho de alunos de turmas de Introdução à Programação de Computadores (IPC) em ambientes de correção automática de código (ACAC). Para tanto, é apresentado um processo de adaptação do método à uma base de dados gerada a partir de um contexto educacional diferente daquele em que o método foi avaliado originalmente. Como resultado, observou-se que o método de predição, adaptado à nova base, atingiu uma acurácia de 82% na tarefa de detecção precoce de alunos com altas chances de reprovação.*

1. Introdução

Turmas de IPC, em geral, são numerosas [Pereira et al. 2019]. Com isso, atender individualmente alunos dessas turmas se torna um grande desafio. Segundo [Pereira 2018], para os estudantes iniciantes em disciplinas introdutórias de programação, estas habilidades são de um alto nível de complexidade: compreensão do problema, planejamento de solução, redução de erros no algoritmo e escrita do código no plano de solução.

Somado a isso, ainda é possível mencionar a alta taxa de reprovação em disciplinas introdutórias de programação [Pereira et al. 2019]. Segundo um grupo de pesquisadores, um terço dos estudantes de IPC não consegue obter o mínimo para a aprovação na disciplina [Watson and Li 2014].

Diante do exposto, a publicação de [Dwan et al. 2017] apresenta um modelo preditivo que utiliza algoritmos de aprendizagem de máquina para predizer os resultados do desempenho do aluno, informando se o aluno corre risco de reprovação ou não na disciplina de IPC. Além disso, os dados desse modelo foram obtidos através de um ACAC chamado CodeBench[1]. Para a validação do método de predição proposto, foram coletados

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

DOI: 10.5753/cbie.sbie.2019.1651

dados de nove turmas de IPC da Universidade Federal do Amazonas (UFAM), durante o período de 05/06/2016 a 13/09/2016. No total, 486 alunos resolveram sete listas usando a linguagem de programação Python. Eles podiam realizar um número ilimitado de tentativas de submissão, desde que atendesse ao prazo máximo estipulado para a resolução da lista de exercícios.

Entretanto, esse método preditivo foi construído para uma base de dados específica, a qual foi fornecida pelo juiz on-line CodeBench. Contudo, a comunidade científica estimula que tais estudos sejam reproduzidos em outros contextos educacionais a fim de verificar se eles são generalizáveis a outras bases de dados educacionais. Segundo [Estey et al. 2017], em um grupo de trabalho do ITiCSE (Annual Conference on Innovation and Technology in Computer Science Education), [Ihantola et al. 2015] identificaram a necessidade crítica de estudos de validação e de replicação para melhor compreensão dos fatores e das razões que contribuem para os resultados obtidos. Com efeito, o foco deste trabalho foi adaptar e aplicar o método preditivo proposto por [Dwan et al. 2017] à base de dados disponibilizada por [Estey et al. 2017].

Para apresentar a adaptação proposta, o presente artigo foi dividido em 5 seções. A Seção 2 apresenta os trabalhos relacionados que também realizaram predição de desempenho usando dados coletados a partir de turmas de IPC. A Seção 3 descreve a adaptação do modelo preditivo. A Seção 4 apresenta a metodologia utilizada na condução dos experimentos junto aos resultados e a Seção 5 apresenta as considerações finais.

2. Trabalhos relacionados

O trabalho de [Pereira 2018] procurou obter um modelo preditivo para inferir o desempenho dos discentes a partir de algoritmos de aprendizagem de máquina. Para isso, ele contou com uma seleção de 20 atributos extraídos a partir da interação dos alunos com um juiz on-line. Entre os atributos, podemos citar a média de linhas de comentários, a média de submissões, a média das linhas de log, etc. Vale ressaltar que os atributos adotados por [Pereira 2018] foram selecionados através de um processo de engenharia de variáveis (*feature selection*), e que cada atributo representa uma característica do perfil de programação dos alunos, o qual será definido mais adiante. Como resultado, atingiu-se 74,44% de acurácia na tarefa de identificar se os alunos iriam ser reprovados ou aprovados usando os dados das duas primeiras semanas de aula em uma base de dados balanceada. Destaca-se ainda que, a partir da oitava semana de aula, o método atingiu acurácias entre 85% e 90,62%.

No trabalho produzido por [Estey and Coady 2016] também foram propostos alguns atributos que podem ser usados para identificar alunos de IPC com risco de reprovação, como consumo de dicas, compilações, submissões, etc. Esse trabalho também apresentou um modelo preditivo para inferir o desempenho dos discentes. Vale ressaltar que a coleta de dados desse trabalho foi realizada pelo BitFit[2], uma ferramenta desenvolvida pelo autor para que alunos de turmas de IPC possam fazer exercícios semanais. Essa ferramenta, além de ser uma ACAC, disponibiliza uma série progressiva de dicas para cada atividade. Com isso, foi construído um modelo preditivo que fazia uma classificação binária para identificar se o estudante iria passar ou não na disciplina. O modelo preditivo apresentado obteve uma acurácia média de 81% ao final do curso,

[2]https://github.com/ModSquad-AVA/BitFit

entretanto com uma taxa de identificação de 30% dos alunos que reprovaram nas duas primeiras semanas de aula.

Além de [Pereira 2018] e [Estey and Coady 2016], outros trabalhos da literatura procuraram identificar atributos úteis para o processo de predição de desempenho dos estudantes em disciplinas de programação. Por exemplo, [Ahadi et al. 2016] mostram que a quantidade de submissões de códigos, independentemente dos códigos submetidos estarem corretos ou não, possui correlação com a nota do aluno nas provas.

No estudo de [Auvinen 2015] foi investigado se os alunos exibem hábitos de estudo indesejáveis em termos de prática de gerenciamento de tempo e comportamento de tentativa e erro. O autor relata que começar a estudar perto do prazo final das atividades está relacionado a um baixo desempenho. Somado a isso, observou-se em alguns discentes sinais de resolução de impasses por tentativa e erro e isso também está correlacionado com um desempenho baixo nos exercícios e na avaliação.

3. Adaptação do Modelo Preditivo

Nesta seção serão descritos o contexto educacional e a base de dados nos quais a adaptação proposta neste trabalho foi aplicada. Além disso, será apresentado o processo de adaptação do modelo preditivo proposto por [Dwan et al. 2017].

3.1. Contexto Educacional

Os dados deste estudo foram coletados por [Estey et al. 2017] em um curso de IPC de 13 semanas ministrado em Java. No total, foram abordados 10 tópicos de programação durante o curso: variáveis, fluxo de controle, métodos, condicionais, loops, E/S, arrays, algoritmos de busca, classificação e objetos. O curso consiste em 2,5 horas de aula por semana, com 10 laboratórios de programação de duas horas, onde os alunos recebem atividades de aprendizagem baseadas em problemas e tarefas de programação semanais. As avaliações parciais acontecem durante a quinta e a nona semana, e há uma avaliação final no encerramento do semestre.

Nesse contexto, foi introduzido o BitFit, onde as atividades realizadas nesse ACAC são voluntárias e não afetam as notas dos discentes. Esse juiz on-line foi introduzido como um recurso de prática suplementar, que oferece exercícios semelhantes aos apresentados durante os laboratórios semanais. Além disso, o BitFit é uma ferramenta de programação on-line, de código aberto, onde os alunos escrevem código no navegador. Botões para compilar, executar, submeter uma solução, obter uma dica, e fazer uma pergunta são todos instrumentados para coletar padrões de interação do aluno.

Os resultados de compilação e execução são exibidos para o usuário enquanto eles trabalham com o problema atual. Uma solução é considerada correta se todos os casos de teste forem aprovados. Vale ressaltar que não há restrição do número de vezes que um estudante pode compilar, executar ou submeter um exercício.

Em cada semestre em que o BitFit foi usado, havia mais de 80 perguntas distribuídas pelos tópicos do curso. Dentro de cada tópico há de seis a dez perguntas, ordenadas por dificuldade. Os discentes não são obrigados a resolver corretamente as questões mais fáceis antes de tentar resolver os exercícios mais difíceis dentro de um determinado tópico, e são capazes de iniciar em qualquer tópico que escolherem.

Os dados considerados para este estudo incluem o número de tentativas, quantidade de dicas consumidas, compilações, compilações sem erros, execuções, soluções e tempo de resolução. Vale ressaltar que todos esses dados coletados foram medidos em relação às notas da avaliação final.

3.2. Contexto dos dados

É necessário entender a organização da base de dados disponibilizada no GitHub[3] por [Estey et al. 2017]. Os dados foram coletados durante 4 semestres, e para cada semestre é possível acessar as notas finais dos alunos, bem como os atributos gerados pelos alunos para cada questão resolvida. No caso, o semestre 1 teve a participação de 155 alunos, o semestre 2 apenas 54, o semestre 3 teve 273 alunos, e por último, o semestre 4, teve 174 alunos. Cada tentativa de resolução de uma dada questão tinha as seguintes informações: o ID do usuário, o ID do tópico, o ID da questão, o tempo inicial da resolução da questão, o número de compilações, o número de execuções, o número de dicas consumidas, o número total de tentativas, o número total de tentativas corretas, o tempo final da resolução da questão e o número de compilações sem erro.

3.3. O Modelo Preditivo

A Figura 1 ilustra o processo de construção do modelo preditivo proposto por [Dwan et al. 2017]. Primeiramente, os autores realizaram o pré-processamento dos dados, onde os códigos submetidos e os logs dos alunos no ACAC foram analisados a fim de gerar os valores numéricos que compõem o perfil de programação de cada aluno. Esses valores são representados por médias de uma lista de exercícios. Para exemplificar, um atributo trabalhado pelos autores foi a média de tentativas, que era representado pela quantidade total de testes na tentativa de solucionar todas as questões da lista dividido pelo número total de questões da lista. Depois disso, foi realizado um ciclo entre seleção de atributos, escolha do algoritmo de aprendizagem de máquina e ajuste de hiperparâmetros até se obter um modelo que maximizasse a acurácia na predição do desempenho.

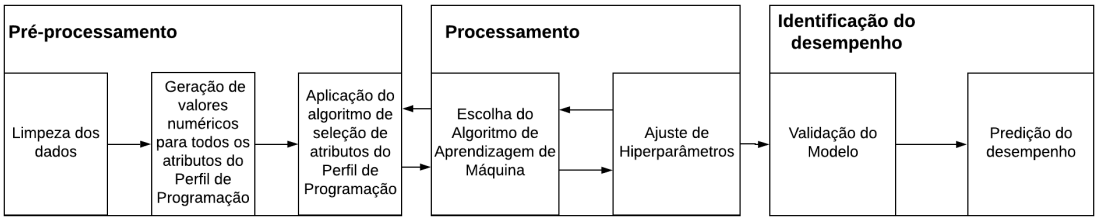

Figura 1. Fluxograma de aplicação do método de [Dwan et al. 2017]

A Figura 2 ilustra o processo de adaptação do método preditivo proposto por [Dwan et al. 2017] na base de dados disponibilizada por [Estey et al. 2017]. Primeiramente, como os dados do modelo proposto por [Dwan et al. 2017] estavam organizados por médias de listas de exercícios e não por questões, foi necessário realizar uma média dos atributos por sessão com os dados disponibilizados por [Estey et al. 2017], onde cada sessão é composta pelos dados dos discentes a cada semana de aula.

[3]www.github.com/aestey/phd

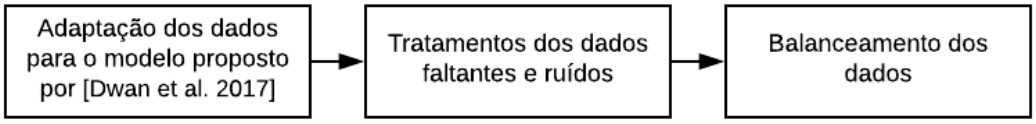

Figura 2. Fluxograma da adaptação da base de dados disponibilizada por [Estey et al. 2017] para o modelo proposto por [Dwan et al. 2017]

Em seguida, foram removidos os dados dos discentes que não usaram a ferramenta BitFit ao longo dos 4 semestres, e foi realizado um balanceamento na base utilizando uma técnica de subamostragem baseada em clusterização. Essa técnica realiza subamostragem através da geração de centroides baseadas em métodos de *clustering*. Neste trabalho, foi utilizado o algoritmo *K-means* (com inicialização de centroides usando *K-means++*) no processo de clusterização da classe majoritária. Note que o processo de clusterização é realizado por similaridade a fim de preservar a informação. Posteriormente, os dados são removidos alternadamente de cada cluster e não de forma aleatória, o que minimiza as chances de perda de informações relevantes na base de dados nesse processo de subamostragem. Utilizaram-se 8 clusters, visto que esse valor minimizava a entropia nos dados (*elbow method*) [Kodinariya and Makwana 2013].

Em relação à seleção de atributos, cabe dizer que não há necessidade de usar algoritmos para encontrar os subconjuntos de atributos mais relevantes, uma vez que [Estey et al. 2017] coletou apenas 7 atributos no total. O número de atributos adotados por [Dwan et al. 2017] é superior a 20, e por isso os autores optaram pelo procedimento de seleção de atributos.

Além disso, para o ajuste de hiperparâmetros - parâmetros dos algoritmos de classificação dos modelos preditivos - foi utilizado o GridSearchCV (GSCV), que é um algoritmo de busca exaustiva com todas as possibilidades sobre os valores específicos de um modelo preditivo [Pedregosa et al. 2011], onde o retorno dessa função são os melhores hiperparâmetros encontrados para o estimador.

De acordo com [Dwan et al. 2017], os melhores resultados em sua predição foram obtidos, em geral, pelos algoritmos *ensembles* baseados em árvore de decisão. Com isso, os algoritmos de classificação utilizados para a construção dos modelos preditivos foram o *Random Forest* (RF) e *ExtraTreesClassifier* (ETC). Em contribuição ao modelo proposto por [Dwan et al. 2017], neste trabalho também foi utilizado o algoritmo de classificação *XGBoost* (XGB) com *Early Stopping*, que é uma abordagem para treinar modelos complexos de aprendizagem de máquina para evitar overfitting, e que monitora o desempenho do modelo que está sendo treinado em um conjunto de dados de teste separado. Caso o desempenho desse conjunto não apresente melhoras após um número fixo de iterações de treinamento, então ela interrompe o procedimento de treinamento.

Para validar o modelo preditivo, foi adotado o método de validação cruzada com 10 partições (*folds*). Esse método divide a base em k partições, usando k-1 para treino e 1 para teste. Após isso, calcula-se a acurácia na partição de teste. Esse processo é repetido k vezes, até que todas as partições tenham sido usadas como teste. Finalmente, computa-se

a média das acurácias obtidas nos testes.

3.4. Perfis de Programação

O perfil de programação é um conjunto de atributos usados para descrever o comportamento do aluno durante suas tentativas de solucionar os exercícios de programação disponibilizados pelo professor. Neste trabalho, o perfil de programação dos alunos será composto pelos mesmos atributos adotados por [Estey et al. 2017] durante seus experimentos. São eles:

1. `compiles` (E1): Número de compilações;
2. `correct_attempts` (E2): Número de tentativas corretas;
3. `error_free_compiles` (E3): Compilações sem erro;
4. `hints` (E4): Quantidade de dicas que o aluno usou;
5. `runs` (E5): Quantidade de execuções;
6. `total_attempts` (E6): Números de todas as tentativas, corretas ou não;
7. `time` (E7): Tempo de resolução;

No entanto, conforme dito anteriormente, para usar esses atributos no modelo preditivo proposto por [Dwan et al. 2017], será necessário calcular a média de seus valores para cada um dos 10 tópicos de programação ministrados por [Estey et al. 2017] durante seu curso de IPC (vide Seção 3.1).

4. Experimentos

Após a adaptação e balanceamento da base, além do ajuste dos hiperparâmetros, foi realizada uma série de experimentos com o objetivo de identificar a acurácia do modelo preditivo. A Figura 3 ilustra a metodologia adotada na condução dos experimentos.

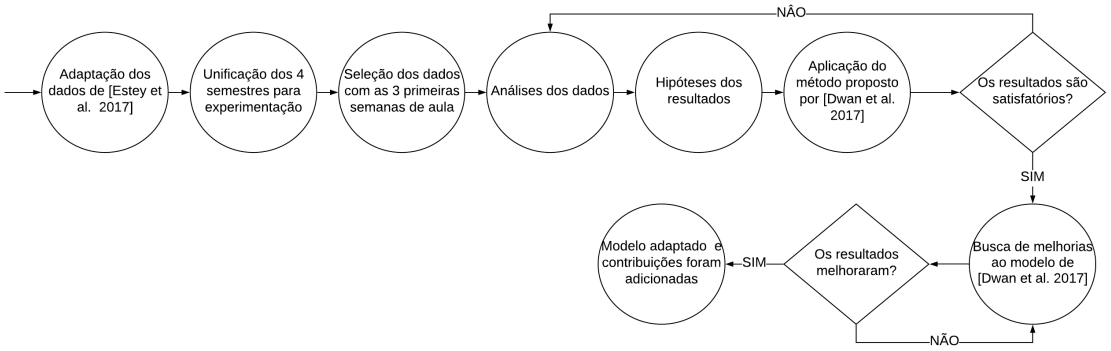

Figura 3. Fluxograma da condução dos experimentos

Vale frisar que os experimentos foram conduzidos de tal forma que seja possível comparar os resultados obtidos com aqueles publicados por [Estey et al. 2017]. Diante do exposto, para comparar os resultados, foram realizados experimentos com todos os semestres em conjunto, utilizando os dados das 3 primeiras semanas de aula. Foram conduzidos experimentos com a base desbalanceada e com a base balanceada.

4.1. Experimentos com a base de dados desbalanceada

A Tabela 1 mostra que o algoritmo de classificação RF aliado ao GSCV obteve um *recall* R_{ap} para os alunos aprovados[4] igual a 70%, e precisão P_{ap} para os aprovados[5] igual a 84%. Entretanto, o algoritmo obteve um *recall* R_{rep} para os alunos reprovados igual a 50% com uma precisão P_{rep} para o mesmo grupo de 30%. Assim, o método preditivo proposto por [Dwan et al. 2017] obteve uma acurácia Acc igual a 66% na base desbalanceada disponibilizada por [Estey et al. 2017].

Tabela 1. Resultado com todos os semestres para as três primeiras semanas de aula na base de dados desbalanceada.

Resultados	R_{ap}	P_{ap}	R_{rep}	P_{rep}	Acc
Adaptação de [Dwan et al. 2017]	70%	84%	50%	30%	66%
[Estey et al. 2017]	89%	92%	67%	57%	85%

A Tabela 1 mostra ainda que [Estey et al. 2017] obteve, com os dados das 3 primeiras semanas de aula, um *recall* R_{ap} de 89% para os alunos aprovados com uma precisão P_{ap} de 92% para o mesmo grupo. Por outro lado, obteve-se um *recall* R_{rep} de 67% para os alunos reprovados com uma precisão P_{rep} de 57% após aplicar um filtro de trajetória. Assim, o método de [Estey et al. 2017] atingiu uma acurácia Acc de 85%, obtendo melhores resultados que os atingidos com o método de [Dwan et al. 2017]. Como a base de dados estava desbalanceada, o algoritmo de classificação RF ficou com viés, ou seja, ele tendia a inferir que a grande maioria dos alunos seriam aprovados (classe majoritária) e, por conseguinte, obteve resultados não satisfatórios.

4.2. Experimento com a base de dados balanceada

A Tabela 2 mostra os resultados com a base balanceada através da técnica de Cluster Centroids, onde o algoritmo de classificação RF aliado ao GSCV obteve um *recall* R_{ap} para os alunos aprovados igual a 92%, e precisão P_{ap} para os aprovados igual a 80%. Por outro lado, o mesmo algoritmo obteve um *recall* R_{rep} de 71% para os alunos reprovados e precisão P_{rep} de 87%. Nesse experimento, o método de [Dwan et al. 2017] atingiu uma acurácia Acc de 82%.

Tabela 2. Resultado com todos os semestres para as três primeiras semanas de aula na base de dados balanceada.

Resultados	R_{ap}	P_{ap}	R_{rep}	P_{rep}	Acc
Adaptação de [Dwan et al. 2017]	92%	80%	71%	87%	82%
[Estey et al. 2017]	89%	92%	67%	57%	85%

Comparando, a precisão P_{ap} para alunos aprovados de [Estey et al. 2017] foi superior a de [Dwan et al. 2017]. O último obteve uma precisão P_{ap} de 80% enquanto o primeiro atingiu 92%. Em relação à acurácia Acc, os resultados são relativamente próximos,

[4]O *recall* dos aprovados é o número de alunos que foram classificados corretamente como aprovados dividido pela quantidade total de alunos realmente aprovados. Analogamente, pode-se quantificar o *recall* dos alunos reprovados.

[5]A precisão dos aprovados é o número de alunos que foram classificados corretamente como aprovados dividido pelo número total de alunos classificados como aprovados. Analogamente, pode-se quantificar a precisão dos alunos reprovados.

com uma pequena superioridade do trabalho apresentado por [Estey et al. 2017], visto que a adaptação de [Dwan et al. 2017] atingiu 82% de acurácia Acc, enquanto [Estey et al. 2017] chegaram a 85%.

Apesar disso, note que o objetivo principal desses modelos é identificar alunos com alta probabilidade de reprovação, conforme defendido pelo próprio trabalho de [Estey et al. 2017]. A importância desse objetivo deve-se ao fato de que, uma vez identificados tais alunos, pode-se tomar um conjunto de medidas proativas para minimizar as chances de tais alunos realmente reprovarem. Nesse sentido, o trabalho de [Dwan et al. 2017] foi superior, pois ele obteve um *recall* R_{rep} e precisão P_{rep} maiores do que os obtidos por [Estey et al. 2017]. Mais especificamente, [Dwan et al. 2017] obteve um *recall* R_{rep} de 71% para os alunos reprovados com uma precisão P_{rep} de 87%, enquanto [Estey et al. 2017] chegaram a um *recall* R_{rep} de 67% com uma precisão P_{rep} de 57%. Em outras palavras, a adaptação do método de [Dwan et al. 2017] consegue identificar alunos com alta probabilidade de reprovação com um nível de confiança maior. Além disso, [Estey et al. 2017] obteve um *recall* R_{ap} de 89% para os alunos aprovados, enquanto a adaptação do método de [Dwan et al. 2017] atingiu 92%.

4.2.1. Perfil dos discentes em uma base de dados balanceada

Por fim, tentou-se identificar as características que diferem alunos com alta probabilidade de aprovação dos alunos com alta probabilidade de reprovação. Note que essas diferenças são importantes para os docentes, uma vez que podem trazer *insights* sobre o que precisa ser feito para reverter a alta probabilidade de reprovação de alguns alunos. Nesse sentido, a Figura 4 apresenta um gráfico detalhando as principais diferenças entre o perfil de programação de alunos aprovados do perfil de alunos reprovados.

Figura 4. Radar de atributos dos discentes aprovados em laranja, e reprovados em azul

Conforme pode ser visto na figura, alunos aprovados passam mais tempo resolvendo as atividades, assim como possuem mais compilações sem erro. Além disso, eles tentam e acertam mais. Por outro lado, os alunos reprovados consomem mais dicas. Em relação a execução e compilações, não há uma grande diferença.

Ainda, é possível comparar esse padrão de comportamento com o obtido por [Estey et al. 2017]. Nesse artigo, os autores relatam que os discentes aprovados compi-

lam mais e consomem menos dicas, enquanto os alunos reprovados compilam menos e consomem mais dicas. Neste trabalho foi obtido o inverso em relação ao número de compilações, ou seja, alunos reprovados compilavam um pouco mais que os alunos aprovados, o que pode ser explicado pela subamostragem realizada na base de dados. Note que esse padrão de mais compilações/execuções por parte de alunos reprovados também foi reportado no trabalho de [Dwan et al. 2017].

Em poucas palavras, com exceção da divergência em torno da quantidade de compilações supracitada, o que se nota é que os comportamentos de programação de alunos relatados por [Dwan et al. 2017] são similares aos citados nesta seção relacionados ao contexto educacional e dados disponibilizados por [Estey et al. 2017]. Isso pode sugerir que esses padrões são generalizáveis, isto é, multi-institucionais e multiculturais, visto que os autores lidam com contextos educacionais e culturais diferentes.

5. Considerações Finais

A adaptação proposta obteve resultados expressivos. Para ilustrar, alcançou-se nas três primeiras semanas uma acurácia Acc de 82% em uma base de dados balanceada. Entretanto, o trabalho de [Estey et al. 2017] atingiu uma acurácia Acc de até 85% com os dados das mesmas três primeiras semanas de aula em uma base de dados desbalanceada. Apesar dessa diferença, deve-se notar que, conforme apontado por [Alamri et al. 2019], para minimizar o problema de alto índice de reprovação em turmas de IPC, é necessário focar mais na detecção precoce de alunos com altas chances de reprovação. Nesse sentido, a adaptação proposta neste estudo foi superior, isto é, na capacidade de identificar precocemente alunos com chances de reprovar com alta precisão. Destaca-se que o que propiciou essa melhora foi a adaptação proposta neste estudo ao método de [Dwan et al. 2017], onde foi realizada uma subamostragem utilizando uma técnica baseada em centroides de clusters. Por outro lado, [Estey et al. 2017] atingiu uma acurácia Acc maior em função de uma alta precisão P_{ap} nos verdadeiros positivos, isto é, o modelo de [Estey et al. 2017] é mais preciso para reconhecer alunos que provavelmente serão aprovados na disciplina.

Destaca-se que a principal contribuição desta pesquisa está no conjunto de evidências obtido através dos experimentos, que fortalece a ideia de uma possível generalização do modelo preditivo proposto por [Dwan et al. 2017]. Para exemplificar, no contexto educacional de [Estey et al. 2017], a linguagem Java era utilizada, enquanto no de [Dwan et al. 2017] era a linguagem Python. Com isso, acredita-se que a linguagem de programação não é um fator que influencia nas métricas propostas para o perfil de programação.

Frisa-se ainda que a generalização do modelo preditivo é relevante por diversas razões, dentre as quais ressalta-se: i) um único modelo preditivo generalizado poderia ser aplicado em vários contextos educacionais diferentes; ii) várias instituições e professores poderiam utilizar esse modelo preditivo, permitindo a identificação de discentes em risco de reprovação e iii) tendo um único modelo generalizado, vários especialistas poderiam otimizar o método.

Agradecimentos

Os autores agradecem o apoio prestado pela Fundação de Amparo à Pesquisa do Estado do Amazonas - FAPEAM por meio do Edital N. 002/2018 - Universal Amazonas.

Referências

Ahadi, A., Vihavainen, A., and 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*, pages 218–223.

Alamri, A., Alshehri, M., Cristea, A., Pereira, F. D., Oliveira, E., Shi, L., and Stewart, C. (2019). Predicting moocs dropout using only two easily obtainable features from the first week's activities. pages 163–173. Springer.

Auvinen, T. (2015). Harmful study habits in online learning environments with automatic assessment. *2015 International Conference on Learning and Teaching in Computing and Engineering*, pages 50–57.

Dwan, F., Oliveira, E., and Fernandes, D. (2017). Predição de zona de aprendizagem de alunos de introdução à programação em ambientes de correção automática de código. *Simpósio Brasileiro de Informática na Educação-SBIE*, 28(1):1507.

Estey, A. and Coady, Y. (2016). Can interaction patterns with supplemental study tools predict outcomes in cs1? *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '16*, pages 236–241.

Estey, A., Keuning, H., and Coady, Y. (2017). Automatically classifying students in need of support by detecting changes in programming behaviour. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*.

Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Börstler, J., Edwards, S. H., Isohanni, E., Korhonen, A., Petersen, A., Rivers, K., Rubio, M. , Sheard, J., Skupas, B., Spacco, J., Szabo, C., and Toll, D. (2015). Educational data mining and learning analytics in programming: Literature review and case studies. *ACM. Proceedings of the 2015 ITiCSE on Working Group Reports*, pages 41–63.

Kodinariya, T. M. and Makwana, P. R. (2013). Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Pereira, F. D. (2018). Uso de um método preditivo para inferir a zona de aprendizagem de alunos de programação em um ambiente de correção automática de código. *Dissertação (Mestrado em Ciência da Computação) – Instituto de Computação, Universidade Federal do Amazonas. Manaus*.

Pereira, F. D., Oliveira, E., Cristea, A., Fernandes, D., Silva, L., Aguiar, G., Alamri, A., and Alshehri, M. (2019). Early dropout prediction for programming courses supported by online judges. pages 67–72. Springer.

Watson, C. and Li, F. W. (2014). Failure rates in introductory programming revisited. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, pages 39–44.