Predição de desempenho em ambientes computacionais para turmas de programação: um Mapeamento Sistemático da Literatura

Filipe Dwan Pereira[3], Linnik Maciel de Souza[2], Elaine H. T. Oliveira[2], David B. F. Oliveira[2], Leandro S. G. Carvalho[2]
[1] Universidade Federal de Roraima (UFRR) – Boa Vista, RR, Brazil
[2] Universidade Federal do Amazonas (UFAM) – Manaus, AM, Brazil
filipe.dwan@ufrr.br,{elaine,david,leandro,linnik}@icomp.ufam.edu.br

Abstract. *Programming classes have a high failure rate and, as a result, many studies has been conducted to predict student performance, to help in decision making. The present work carried out a systematic mapping of studies from 2009 to 2019 to address six research questions. In total, we analysed 911 publications and filtered 70 works related to performance prediction in virtual environments, especially online judges. In general, the publications accepted explore Machine Learning and Data Mining techniques to make inferences through the modelling of students' programming profiles. Results point for research gaps and open questions for this field.*

Resumo. *Turmas de programação têm um alto índice de reprovação e, por conta disso, muitos estudos vêm sendo conduzidos para realizar a predição do desempenho do aluno, para ajudar na tomada de decisão. Nesse caminho, o presente trabalho realizou um mapeamento sistemático da literatura dos anos de 2009 a 2019 com o intuito de caracterizar estudos que propuseram métodos relacionados com predição de desempenho em ambientes computacionais para turmas de programação. No total, 911 publicações foram exploradas, em que 70 foram aceitas para o fichamento. No geral, as pesquisas exploram técnicas de Aprendizagem de Máquina (AM) e Mineração de Dados (MD) para fazer inferências, através da modelagem de perfis de programação dos estudantes. Os resultados apontam lacunas e questões abertas para essa área.*

1. Introdução

Inferir o desempenho do estudante em turmas de programação é algo que vem sendo estudado há décadas. As abordagens e os métodos que tratam desse problema vêm evoluindo. Antes, grande parte desses estudos eram realizados baseados na análise estática do passado do discente, ou seja, antes do seu ingresso no curso de graduação [Ahadi et al. 2016, Pereira et al., 2020]. Entretanto, o comportamento do estudante é dinâmico, isto é, o aluno pode mudar sua postura ao longo do tempo. Nesse cenário, o número de pesquisas nas áreas de *Learning Analytics* (LA) e *Educational Data Mining* (EDM) vem crescendo. Assim, a grande quantidade de métodos preditivos apontados na literatura juntamente com o aprimoramento das técnicas para enfrentar este problema de predição, motivou a realização deste mapeamento da literatura sobre predição de desempenho em ambientes computacionais para turmas de programação.

Estudos da área de LA e EDM com dados de cursos de programação tendem a modelar o comportamento do estudante a partir de sua interação com os ambientes computacionais utilizados, seja Ambientes Virtuais de Aprendizagem (AVA) ou

DOI: 10.5753/cbie.sbie.2020.1673

Ambientes de Correção Automática de Código (ACAC), também conhecidos como juízes online. Tipicamente, emprega-se uma abordagem dirigida aos dados, que modela o comportamento do estudante enquanto ele(a) está resolvendo um exercício ou enquanto está consumindo um recurso computacional que é monitorado por componentes de software capazes de coletar os logs dessas interações. No presente artigo, esses comportamentos de programação, após pré-processados, serão chamados de *perfil de programação do aluno*. É com base nesses perfis que as pesquisas tentam inferir o desempenho do aluno ou inferir se o aluno está com dificuldade na disciplina ou com risco de reprovação. Tipicamente, essa inferência de desempenho é realizada no início do curso com o intuito de permitir uma intervenção rápida.

Nesse caminho, o presente trabalho realizou um mapeamento sistemático da literatura de estudos publicados entre 2009 e 2019 com o intuito de responder seis questões de pesquisa e caracterizar as publicações aceitas em relação a modelos de predição de desempenho de alunos em ambientes computacionais utilizados para apoiar as aulas de programação, especialmente ACACs. Desta forma, neste trabalho são investigadas e analisadas algumas pesquisas que exploram técnicas de Aprendizagem de Máquina (AM), estatística inferencial e Mineração de Dados (MD) para prever o desempenho dos estudantes em turmas de programação, a fim de melhorar o processo de ensino e aprendizagem dessas turmas.

De uma forma geral, as técnicas de previsão de desempenho encontradas na literatura são baseadas em dois passos: (i) gerar modelos de perfis de programação dos alunos à medida que esses desenvolvem suas soluções para os problemas de programação propostos, e (ii) usar técnicas de estatística inferencial, MD ou AM para explorar as dimensões dos perfis de programação e prever o desempenho dos estudantes a partir dos modelos gerados. Os perfis de programação podem ser originados a partir de variados tipos de evidências, dentre as quais destacamos: o modo como o estudante lida com o erro [Jadud 2006]; com os prazos, [Auvinen 2015, Pereira et al. 2019a]; o quão resiliente e determinado o estudante é durante suas tentativas de solucionar as questões propostas [Ahadi et al. 2016], e a análise sintática do grafo de fluxo de controle dos códigos submetidos [Otero et al. 2016, Pereira et al. 2019c]. Todos esses trabalhos analisam como esses perfis de programação do aluno se relacionam com a nota.

2. Metodologia do Mapeamento

A presente pesquisa foi conduzida em 3 macroetapas: Planejamento, Execução (condução) e construção de um Relatório Técnico do MSL reportando os resultados. No planejamento, foram definidos o objetivo do MSL, as questões de pesquisa, os filtros de seleção das bibliotecas digitais, os filtros de inclusão e exclusão dos artigos e os campos que deveriam ser extraídos dos artigos que fossem aceitos. Tais itens foram discriminados no protocolo do MSL[1]. Após isso, foi executado o protocolo, isto é, foi conduzido o mapeamento, quando foram identificados e selecionados artigos sistematicamente e foram extraídas as informações dos artigos em ficha, a fim de responder às questões de pesquisa. Finalmente, os dados dos fichamentos foram tabulados, sintetizados e analisados para a produção de um relatório técnico do MSL. Antes do planejamento foram identificados artigos de controle através de uma pesquisa exploratória. Além disso, este

[1] Protocolo, fichamentos, relatório técnico, lista de artigos aceitos e teste de Kappa disponível em: https://www.dropbox.com/sh/dofhlkaeu6whsi8/AAAn0Og8i8BbQDuBvFdPAQbSa?dl=0

trabalho foi realizado em pares. Assim, cada etapa e escolha de critério foi revisada pelo primeiro autor e por um colaborador externo deste artigo. Apenas a seleção dos artigos e extração dos dados foram realizadas somente por um dos autores deste MSL. Entretanto, nos momentos de indecisão em relação à seleção e extração, um coautor realizava uma revisão, a fim de que artigos e dados importantes não fossem excluídos. Além disso, os autores atuam nas áreas de ensino de programação e de informática na educação. Especificamente, a maioria dos autores deste artigo são professores do ensino superior e ministram aulas para turmas de programação.

2.1. Planejamento

Este trabalho tem por intento **analisar** publicações científicas utilizando um estudo baseado em um Mapeamento Sistemático da Literatura (MSL) **com o propósito de** caracterizar pesquisas **com relação** à coleta de dados e modelos para predição de desempenho de alunos **no contexto** de ambientes computacionais para turmas de programação, especialmente ACAC. Seis Questões de Pesquisa (QP) foram estabelecidas para a orientação do processo de seleção de artigos (Tabela 1).

Tabela 1: Questões de Pesquisa do MSL.

Questões	Descrição
QP1	Quais são as ferramentas e algoritmos utilizados para a predição de desempenho de estudantes de turmas de programação e qual o cenário educacional?
QP2	Quais foram os atributos usados nos modelos preditivos?
QP3	Como foi mensurada a relação dos atributos com a nota do aluno e como foram avaliados os modelos preditivos utilizados?
QP4	Quais os objetivos e resultados apresentados?
QP5	Até que ponto os modelos de predição apresentados são confiáveis e válidos?
QP6	Quais são as tendências da área?

Foram realizadas pesquisas nas bibliotecas digitais *IEEE Xplore* e *Scopus*. Além disso, foi realizada uma busca manual em português nas seguintes fontes: Simpósio Brasileiro de Informática na Educação (SBIE) e Revista Brasileira de Informática na Educação (RBIE). Para executar a pesquisa nas bibliotecas digitais foi elaborada uma *string* de busca, usando palavras-chave extraídas de artigos de controle encontrados através de uma busca exploratória. Assim, as palavras-chave foram categorizadas em população, intervenção e resultado, conforme apresentado na Tabela 2.

Tabela 2: Palavras-chave categorizadas.

Categoria	Palavras-chave
População	**“novice programmers”, “programming student”, “introductory programming”**;
Intervenção	**“machine learning”, “data mining”, “learning analytics”, “edm”, “classifier algorithms”, “predictive models”**;
Resultado	**“performance”, “grading”, “automated assessment”, “difficulty level”**;

Dessa forma, a *string* de busca é uma composição das três categorias apresentadas na Tabela 2, conforme segue: (*"novice programmers" OR "programming student" OR "introductory programming") AND ("machine learning" OR "data mining" OR "learning analytics" OR “edm” OR "classifier algorithms" OR "predictive models") AND (“performance” OR “grading” OR "automated assessment" OR "difficulty level"*).

Para a seleção sistemática dos artigos nas bibliotecas digitais foram utilizados critérios de inclusão e exclusão nos artigos. Os critérios estão expostos na Tabela 3.

3. Execução do MSL

O MSL foi executado em duas etapas de seleção. Na primeira, os critérios foram aplicados após a leitura dos títulos, lista de palavras-chave e resumo dos artigos. A partir dos artigos selecionados, uma segunda seleção foi feita, agora aplicando os critérios após a leitura da introdução e conclusão destes. Uma vez que o artigo era aceito, ele era lido na íntegra e posteriormente fichado. A Tabela 4 mostra a quantidade de artigos analisados por biblioteca digital. A coluna *Resultado Inicial* apresenta o número de publicações retornadas e as duas colunas seguintes mostram a quantidade de artigos aceitos nas Seleções 1 e 2. No total, 70 artigos foram fichados.

Tabela 3: Critérios de Inclusão e Exclusão.

Critério de Inclusão	Critério de Exclusão
CI1. Aplica técnicas de predição de desempenho dentro de um contexto de um ambiente de estudo para turmas de programação; CI2. Revisões ou mapeamentos sistemáticos da literatura ou surveys relacionados à predição de desempenho de alunos de turmas de programação; CI3. Está disponível em forma de artigo completo.	CE1. Aborda o processo de ensino-aprendizagem em disciplinas gerais sem relacionamento com as de programação; CE2. Trata o problema de inferência de desempenho exclusivamente com métodos pedagógicos; CE3. Não atende a nenhum dos critérios de inclusão elencados no filtro de inclusão; CE4. Não faz coleta de dados em ambientes computacionais para turmas de programação.

Tabela 4: Resultados das etapas de seleção do MSL.

Biblioteca Digital	Resultado Inicial	1ª Seleção	2ª Seleção
IEEE Xplore	247	24	12
Scopus	563	80	50
SBIE	84	10	7
RBIE	17	3	1
Total	911	117	70

Tabela 5: Relacionamento entre as questões de pesquisa e os campos de extração.

QP1	QP2	QP3	QP4	QP5	QP6
C2, C3, C4, C5, C6, C8	C7	C9, C10	C1, C11	C12	C13

A fim de sistematizar a extração das informações necessárias para responder às QP expostas na Tabela 1, foram realizados fichamentos dos artigos aceitos. Especificamente, foram extraídos os seguintes campos dos 70 artigos analisados: (C1) resumo e objetivo do artigo; (C2) tamanho da amostra: número de estudantes envolvidos no experimento; (C3) nível de ensino (médio, graduação ou pós-graduação) e disciplina(s) em que o experimento foi realizado; (C4) linguagem de programação utilizada pelos alunos; (C5) ambiente computacional utilizado na(s) turma(s) de programação (juiz online, AVA, MOOC, etc.); (C6) algoritmo(s) de predição utilizado(s) nos experimentos; (C7) atributo(s) utilizado(s) para construção do modelo preditivo; (C8) ferramenta(s) utilizada(s) para implementação do(s) modelo(s) preditivo(s); (C9) técnica(s) utilizada(s) para avaliar a relação entre um atributo ou característica extraída dos logs dos estudantes e o seu desempenho; (C10) técnica(s) utilizada(s) para avaliar o desempenho do modelo preditivo; (C11) resultados obtidos, qual algoritmo se mostrou mais eficiente e qual a sua precisão; (C12) limitações e ameaças a validade; (C13)

trabalhos futuros e tendências da área. A Tabela 5 mostra quais campos extraídos estão relacionados com cada QP.

4. Resultados e Discussão

O enfoque dos 70 artigos analisados era melhorar o processo de ensino e aprendizagem em turmas de programação com o uso de modelos preditivos, entretanto apenas dois trabalhos [Azcona et al. 2018, Quille, K., Bergin 2019] realizaram de fato a predição em turmas reais, isto é, os trabalhos normalmente se limitavam a construção do modelo, sem realizar a intervenção. Os modelos preditivos eram construídos em um esquema similar ao da Figura 1. Boa parte das turmas tratadas nos experimentos usavam um *Integrated Development Environment* (IDE) incorporado ao recurso de correção automática dos códigos submetidos pelos estudantes. Os professores selecionavam questões de programação, enquanto os estudantes tentavam resolvê-las. Os discentes podiam desenvolver, compilar, depurar e executar (alguns trabalhos adotavam o termo testar no mesmo sentido que executar) o seu código numa IDE (online ou local), antes de fazer a submissão que normalmente era avaliada automaticamente por um juiz online.

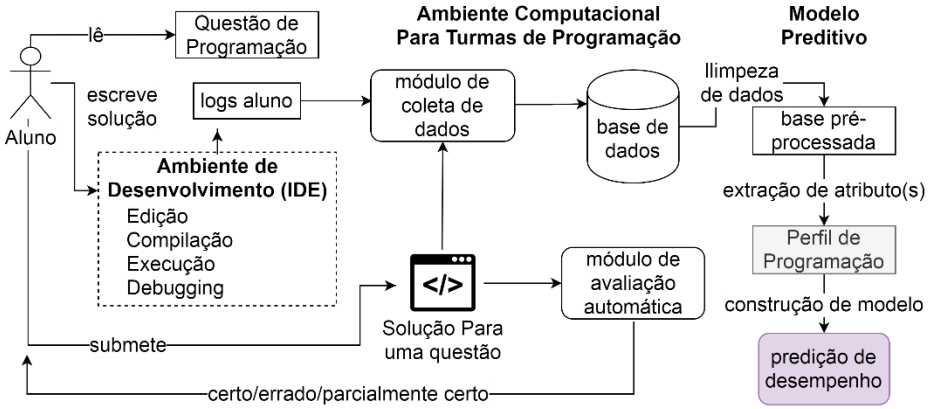

Figura 1: Esquema típico para a predição de desempenho de alunos a partir de dados coletados em ambientes computacionais para turmas de programação.

Enquanto o estudante resolvia a questão de programação, além do código-fonte submetido para avaliação, o processo de codificação do aluno era registrado em diferentes níveis de granularidade através de um módulo de coleta de dados que poderia ficar embutido no próprio ambiente de apoio para as turmas de programação. Tais dados eram utilizados pelos estudos para alimentar algoritmos de estatística inferencial, MD ou AM. Primeiramente, os dados eram pré-processados e, depois, extraiam-se atributos para formar um perfil de programação utilizado pelo modelo preditivo para inferir o desempenho do aluno. Dito isso, a seguir serão respondidas as questões de pesquisa.

Respondendo a QP1. A frequência de uso das ferramentas utilizadas nos 70 artigos para a construção dos modelos preditivos pode ser vista na Figura 2a, onde o weka, scikit-learn e o R foram os recursos mais utilizados. As ferramentas que foram usadas apenas em um estudo (Tableau, WizRule, SAS Enterprise Miner, Liblinear, Cluto, EMAx tool, UnBBayes e gcov) não foram plotadas. Ainda, muitos trabalhos (N=35) não apontaram as ferramentas utilizadas. A Figura 2b mostra os algoritmos de AM utilizados nos estudos. O uso de regressão se dá em função de alguns trabalhos (N=12) realizarem a predição da nota do estudante, que é um valor contínuo. Por outro lado, a maioria (N=37) empregou algoritmos de classificação. Para tanto, foram discretizadas as notas dos alunos, por

exemplo, atribuindo um rótulo *passed* para alunos aprovados e *failed* para os reprovados para uma posterior classificação binária do desempenho. Os algoritmos de AM utilizados para classificação foram *ensembles*, árvores de decisão, modelos probabilísticos (como redes bayseanas ou naive bayes), máquinas de vetores de suporte (SVM) e redes neurais. Vale ressaltar que 17 trabalhos aplicavam algoritmos de clusterização para agrupar estudantes com perfil de programação similar e depois analisar a relação dos grupos com o desempenho dos estudantes. O que se observou pela escolha dos algoritmos de AM é que modelos baseados em árvores de decisão e de clusterização são mais viáveis para interpretação, o que possibilita uma análise dos fatores que podem levar o aluno à reprovar ou a ser aprovado, algo crucial no campo da informática na educação. No entanto, devido à simplicidade desses modelos, os resultados podem ser menos satisfatórios. De outra forma, as redes neurais tendem a ser mais robustas, mas os modelos preditivos são caixas pretas. Destaca-se ainda a área de interpretação de algoritmos de aprendizagem de máquina que pouco foi explorada nos artigos fichados.

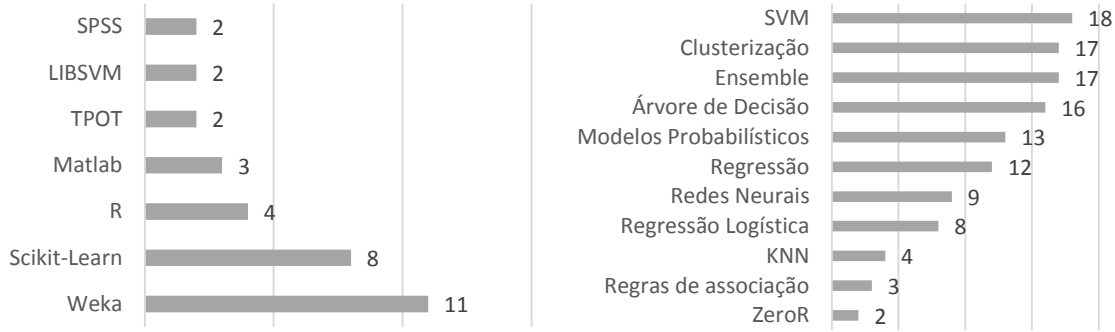

Figura 2: Ferramentas (a) e algoritmos (b) utilizados nos 70 estudos analisados. Perceba que alguns estudos usaram mais de um algoritmo/ferramenta.

A distribuição do número de alunos participantes dos experimentos é apresentada na Figura 3. Observe que um número baixo de exemplos para treino pode gerar um baixo poder de predição dos algoritmos de AM. Nesse sentido, apenas em 4 pesquisas foram realizados experimentos com mais de 3000 discentes. Uma quantidade alta de instâncias torna mais viável o uso de algoritmos de aprendizagem profunda, por exemplo. Quanto ao nível de ensino, mais de 90% dos estudos analisados foram conduzidos em instituições de ensino superior. Além disso, em geral, trabalhou-se exclusivamente com disciplinas de introdução à programação de computadores. Por fim, quanto aos ambientes de apoio utilizados nas turmas de programação, os trabalhos usaram tipicamente juízes online ou plugins instalados no IDE do estudante, ambos em conformidade com a Figura 1. De fato, apenas 7 trabalhos registravam os dados dos discentes em AVA ou MOOC e 3 trabalhos eram revisões da literatura. Por fim, poucos estudos longitudinais (com dados coletados por mais de um ano) foram encontrados. Além disso, apenas um estudo longitudinal era multi-institucional, o que é algo importante para validar o poder de generalização do modelo preditivo.

Respondendo a QP2. Os atributos utilizados nos modelos preditivos variavam em nível de granularidade. Em geral, usavam-se evidências *data-driven* baseadas em teclas digitadas, edição de linha de código, edição de arquivo, compilação, execução, submissão. Tais atributos eram então utilizados para formar um perfil de programação de cada aluno. Para ilustrar, existem estudos que modelaram: a forma como o estudante lida com o erro analisando pares de compilação [Jadud 2006]; com os prazos, [Auvinen 2015,

Pereira et al. 2019a]; o quão resiliente e determinado o estudante se encontra durante a resolução das questões propostas, que é calculado com base no número de tentativas e na corretude dos códigos [Ahadi et al. 2016, Fonseca et al. 2019]; qual o padrão de digitação do(a) estudante quando ele(a) está programando [Leinonen et al. 2016, Pereira et al. 2019b]; a análise sintática do grafo de fluxo de controle dos códigos submetidos [Otero et al. 2016, Pereira et al. 2019c]; mudança de código entre submissões e comportamentos de procrastinação [Edwards et al. 2009]; análise estática dos códigos submetidos [Dwan et al. 2017, Azcona et al. 2018]. Os 7 trabalhos que foram conduzidos em MOOC ou AVA usavam como atributo a frequência de uso de recursos e atividades. Mais ainda, 18 pesquisas usaram características demográficas e históricas dos alunos para realizar as inferências, combinando-as ou não com os atributos baseados em evidências *data-driven*.

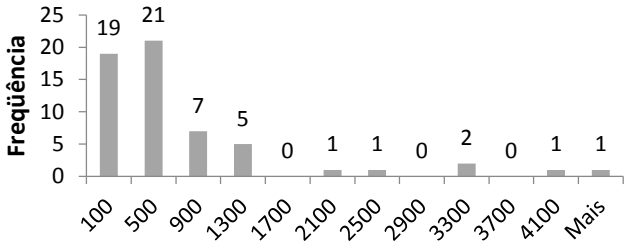

Figura 3: Histograma dos tamanhos de amostras (2 trabalhos não relataram esse valor).

Respondendo a QP3. Em estudos como Jadud (2006), foi mensurada a correlação de uma única variável independente com a nota do aluno. Quando se tratavam de variáveis contínuas, foi utilizada a correlação de *Pearson* ou a correlação de *Spearman*. Quando se tratavam de variáveis dicotômicas, usou-se a correlação de *phi*, como em Ahadi, Vihavainen e Lister (2016). Nesse caso, a nota era representada por 0 para os alunos que tiraram menos que 5, ou por 1, caso contrário. Para mensurar a precisão de classificadores, normalmente os pesquisadores adotam a forma de cálculo tradicional: isto é, a razão entre o número de itens corretamente avaliados pelo total de itens (acurácia). Em apenas ~45% dos casos em que houve classificação, métricas foram usadas para mensurar a precisão do modelo em cada classe com técnicas como a precisão, revocação e *f1-score*. Como a natureza das bases de dados tende a ser desbalanceada, normalmente com mais alunos reprovados do que aprovados, é essencial apresentar sempre os valores das métricas (*f1-score*, *revocação*, etc.) por classe, a fim de não reportar resultados de modelos preditivos tendenciosos. Já no caso da regressão, usou-se tipicamente o método dos mínimos quadrados e o coeficiente de determinação. Notou-se artigos com falta de rigor estatístico, por exemplo, ao deixar de apresentar a significância estatística das correlações ou regressões. Ademais, em caso de regressões lineares, em geral, não se mostrou se os dados tinham uma natureza linear ou se foram realizados testes de diagnóstico de homocedasticidade ou de normalidade residual.

Respondendo a QP4. Em geral, os trabalhos objetivavam explorar o relacionamento entre o desempenho nos exercícios realizados pelos estudantes e a média final ou a subsequente nota em uma prova. Usavam-se técnicas de estatísticas inferencial, AM ou MD para fazer inferências através da modelagem de perfis de programação desses estudantes. Apesar da grande quantidade de métricas comportamentais relacionadas com o desempenho dos alunos (veja reposta a QP2), acredita-se que existe uma necessidade de verificar o poder preditivo delas juntas e em outros contextos educacionais (diferentes metodologias de ensino, linguagem de programação, universidade, etc.), diferentes dos utilizados nas pesquisas dos autores. Faz-se necessário, bem como, objetivos de estudos

voltados para a intervenção no ambiente de ensino com uso dos perfis de programação e modelos inferenciais. De fato, apenas 2 [Azcona et al. 2018, Quille, K., Bergin 2019] dos 70 estudos realizaram a intervenção como o modelo preditivo numa turma real, algo abaixo do esperado, visto que a intenção principal de se criar um modelo preditivo para estimar o desempenho é de usá-lo para detectar precocemente alunos com risco de reprovação e desistência, a fim de prevenir que isso aconteça.

Respondendo a QP5. Regularmente, as ameaças à validade dos estudos e limitações encontradas eram relacionadas à validação interna e externa dos modelos preditivos e, consequentemente, ao poder de generalização dos modelos em outros contextos educacionais. O próprio tamanho da base de dados utilizada sugere isso, como pode ser visto na Figura 3, em que apenas 18 estudos (~26%) usaram dados de mais de 500 estudantes. Como ameaças internas à validade dos modelos, pode-se apontar a possibilidade de estudantes trabalharem em grupo, o que acarretaria uma mudança de comportamento individual na interação do usuário com o ambiente onde foram coletados os dados dos estudantes.

Além disso, no início do curso o aluno pode explorar os recursos do ambiente computacional para conhecê-los, gerando comportamentos fora de um padrão que deve se estabelecer depois desse processo de ambientação. Note que os alunos que já tiverem experiência em programação ou que reprovaram na disciplina poderão transpor essa etapa de ambientação com mais facilidade. Como um fator agravador, para possibilitar uma intervenção precoce, boa parte dos estudos tentam realizar a predição do desempenho dos alunos com os dados do início do curso e essas instabilidades comportamentais podem gerar tanto falsos positivos como falsos negativos. No mais, como a interação do aluno com a disciplina é dinâmica, o estudante pode começar bem (com bons hábitos de estudo e comportamentos efetivos), mas por motivos velados, mudar sua postara ao longo do curso e terminar reprovando. O contrário também é possível, isto é, o aluno começar mal, mas conseguir transpor os obstáculos e terminar sendo aprovado. Novamente, tais mudanças podem gerar falsos positivos e falsos negativos no processo de predição.

Finalmente, outro ponto visto em 9 trabalhos é a falta ou a não explicitação de que foi realizada uma separação a priori de uma parte de dados para realização dos testes no modelo preditivo. A parte da base separada para teste não deve ser usada no processo de seleção do modelo de predição. Ela só é empregada no momento de validação do modelo. Do contrário, o modelo poderá alcançar uma acurácia alta, mas sendo resultado de vários treinamentos e testes realizados na mesma base de dados. Além disso, 6 estudos reportam resultados com um simples separação de dados de teste e treino. Isso também não é desejável, uma vez que métodos como validação cruzada (por exemplo, com 10 partições) permitem a validação do modelo em toda a base de dados. Finalmente, um processo de reamostragem (com ou sem substituição) pode ser empregado, a fim de treinar o algoritmo muitas vezes (estatísticos tipicamente recomendam >100 vezes) para que seja possível identificar com nível de confiança e erro padrão qual foi o desempenho do modelo preditivo.

Respondendo a QP6, como trabalhos futuros, tendências da área e questões potencialmente abertas, destacam-se: inferir automaticamente o nível de dificuldade das questões de programação de juízes online; investigar novas evidências *data-driven* que possam ser utilizadas para compor perfis de programação com alto poder preditivo; gerar dicas automáticas para alunos que usam juízes online; criar um sistema de recomendações de atividades ou conteúdo baseado na similaridade dos perfis de programação dos

estudantes; replicar estudos que tratam de predição de desempenho em outras instituições de ensino, a fim de validar o poder de generalização do modelo preditivo proposto em outro contexto educacional; checar se a utilização de algoritmos de aprendizagem de máquina baseados em redes neurais profundas poderia melhorar o desempenho dos modelos preditivos apresentados nos artigos elencados neste trabalho.

Frisa-se ainda que, além da predição, na educação é importante entender quais fatores levam o aluno a alcançar notas baixas ou altas, ou seja, é essencial interpretar as decisões dos modelos preditivos a fim de dar suporte tanto aos professores como aos alunos, sobre quais comportamentos devem ser incentivados e quais devem ser melhorados. Além disso, modelos preditivos são essenciais, no entanto, faz-se necessário checar se eles funcionam nos ambientes educacionais, através da criação de ferramentas que possam dar suporte tanto para os professores quanto para os alunos. Perceba que tudo isso converge para o desenvolvimento de um ensino personalizado de acordo com as individualidades do aluno de programação.

5. Considerações Finais e Limitações

O que ficou evidente na condução do MSL e na análise das publicações investigadas é que ainda existem muitas questões de pesquisa em aberto e, consequentemente, muitas oportunidades. Destaca-se que ambientes computacionais para turmas de programação possuem suas especificidades e eles podem variar dependendo de vários fatores. Dessa forma, frisa-se que apesar de existirem muitos modelos de predição de desempenho, ainda é necessária a replicação desses estudos em outros cenários educacionais, com novas bases de dados. Tais modelos ainda podem ser aprimorados de várias formas como, por exemplo, combinando os atributos utilizados em perfis de programação de pesquisas diferentes, a fim de realizar uma predição de desempenho mais acurada ou uma análise de quais atributos são menos dependentes de contexto educacional. Além disso, é importante testar os modelos propostos com intervenções com potencial de aprimorar o processo de ensino e aprendizagem de programação.

Como limitação, existe a possibilidade de subjetividade na seleção dos artigos. Por mais que tenham sido usados critérios de inclusão e exclusão no processo de seleção de estudos primários, pode ter acontecido de pesquisas importantes não terem sido incluídas. A fim de contornar esse problema, nos momentos em que o autor principal tinha dúvida em relação à seleção ou não de uma determinada pesquisa, então um coautor era consultado, a fim de que houvesse um consenso. Além disso, foi realizado um teste de concordância com dois autores deste trabalho, com o intuito de analisar o grau de subjetividade do processo de seleção. Usou-se um amostra de 30 artigos e os resultados do teste apontaram um alto gral de concordância ($k = 0,83$, coeficiente Kappa). No mais, neste estudo não foram considerados alguns artefatos que possam conter resultados significativos como monografias, dissertações e teses.

Agradecimentos

Esta pesquisa, realizada no âmbito do Projeto Samsung-UFAM de Ensino e Pesquisa (SUPER), nos termos do artigo 48 do Decreto nº 6.008/2006 (SUFRAMA), foi parcialmente financiada pela Samsung Eletrônica da Amazônia Ltda., Nos termos da Lei Federal nº 8.387/1991, por meio dos convênios 001/2020 e 003/2019, firmados com a Universidade Federal do Amazonas e a FAEPI, Brasil. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

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.

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

Dwan, F., Oliveira, E., & 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).

Edwards, S.H., Snyder, J., Pérez-Quiñones, M.A., Allevato, A., Kim, D., Tretola, B., 2009. Comparing effective and ineffective behaviors of student programmers. Workshop on Computing education research, ACM. pp. 3– 14.

Fonseca, S., Oliveira, E., Pereira, F., Fernandes, D., de Carvalho, L. S. G. (2019). Adaptação de um método preditivo para inferir o desempenho de alunos de programação. Simpósio Brasileiro de Informática na Educação (SBIE).

Jadud, M. C. (2006). Methods and tools for exploring novice compilation behaviour. International Workshop on Computing Education Research, p. 73–84.

Leinonen, J., Longi, K., Klami, A. e Vihavainen, A. (2016). Automatic inference of programming performance and experience from typing patterns. ACM Technical Symposium on Computing Science Education, p. 132–137.

Otero, J., Junco, L., Suarez, R., Palacios, A., Couso, I. e Sanchez, L. (2016). Finding informative code metrics under uncertainty for predicting the pass rate of online courses. Information Sciences, 373:42–56.

Pereira, F.D., Oliveira, E. H. T., Fernandes, D., Cristea, A. (2019a). Early performance prediction for CS1 course students using a combination of machine learning and an evolutionary algorithm. in: 2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT), IEEE. pp. 183–184.

Pereira, F. D., Oliveira, E., Cristea, A., Fernandes, D., Silva, L., Aguiar, G., Ahmed, A., Alshehri, M. (2019b). Early dropout prediction for programming courses supported by online judges. International Conference on Artificial Intelligence in Education (pp. 67-72). Springer, Cham.

Pereira, F., Oliveira, E., Fernandes, D., de Carvalho, L. S. G., & Junior, H. (2019c). Otimização e automação da predição precoce do desempenho de alunos que utilizam juízes online: uma abordagem com algoritmo genético. Simpósio Brasileiro de Informática na Educação (SBIE).

Pereira, F. D., Oliveira, E. H., Oliveira, D. B., Cristea, A. I., Carvalho, L. S., Fonseca, S. C., Toda, A., Isotani, S. (2020). Using learning analytics in the Amazonas: understanding students' behaviour in introductory programming. British Journal of Educational Technology.

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.