Minerando dados de um juiz on-line para prever a evasão de estudantes em disciplinas introdutórias de programação

Dayvson Silva[1], **Sergio Cleger**[1,3], **Marcela Pessoa** [1,2], **Fernanda Pires** [1,2], **David B. F. Oliveira** [2], **Elaine H. T. Oliveira** [2], **Leandro S. G. Carvalho** [2]

[1]Escola Superior de Tecnologia – Universidade do Estado do Amazonas (UEA)
Manaus – AM – Brasil

[2]Programa de Pós-graduação em Informática
Instituto de Computação – Universidade Federal do Amazonas
Manaus – AM – Brasil

[3]Sidia - Instituto de Ciência e Tecnologia
Manaus – AM – Brasil

{ddss.snf19,stamayo,msppessoa,fpires}@uea.edu.br

{david,elaine,galvao}@icomp.ufam.edu.br

Abstract. *Dropout in CS1 (Computer Science 1) presents itself as a common problem at several universities. One of the ways to deal with this scenario is to look for mechanisms that identify the probability that a student will give up. In view of this, in this paper, 5 evasion prediction models are proposed in basic programming disciplines, using data mining techniques applied to the logs of an online Judge system. 38 attributes were extracted, from 2010 students from the years 2016 to 2019. The results show that the best model presented an average accuracy of at least 85.61 % in the first two weeks of the course and reached 91.96 % after six weeks of class, obtaining higher values in some classes.*

Resumo. *A evasão em disciplinas de Introdução à Programação se apresenta como um problema comum em várias universidades. Uma das formas de lidar com esse cenário, é buscar mecanismos que identifiquem as chances de um aluno desistir. Diante disto, neste artigo são propostos 5 modelos de previsão de evasão em disciplinas básicas de programação, utilizando técnicas de mineração de dados aplicadas aos logs de um sistema Juiz on-line. Foram extraídos 38 atributos, de 2010 estudantes dos anos 2016 a 2019. Os resultados mostram que o melhor modelo apresentou acurácia média de 85,61% nas duas primeiras semanas do curso e atingiu 91,96% após seis semanas de aula, obtendo valores superiores em algumas turmas.*

1. Introdução

As Instituições de Ensino Superior, especialmente nas áreas de Ciência Exatas, há anos enfrentam o problema de evasão escolar. A evasão ocorre quando alunos não completam cursos ou programas de estudo, podendo ser considerados como evadidos aqueles que se matriculam e desistem antes mesmo de iniciar o curso [Maia and Meirelles 2005].

Em um contexto mais específico, a evasão escolar em disciplinas introdutórias de programação ou *Computer Science 1* (CS1) assim chamadas na literatura, principalmente

em cursos que não têm a computação como fim (*non-majors*), representa um grande desafio para muitas Instituições de Ensino Superior. Segundo Watson et al. (2014), um dos motivos dos altos índices de desistência dessas disciplinas é o fato de que o aprendizado de programação requer uma grande capacidade de abstração por parte do aluno, além de exigir um conhecimento matemático razoável. Adicionalmente, as turmas iniciais de programação tendem a ser numerosas, de forma que é difícil o controle individual por parte do professor [Khosravi and Cooper 2017].

Uma forma de lidar com esse problema é buscar mecanismos para tentar identificar, o mais cedo possível, alunos com alta probabilidade de desistir da disciplina. Uma vez que o professor sabe que um dado aluno possui altas chances de evadir, ele poderá se valer de estratégias pró-ativas para tentar minimizar as chances da evasão de fato acontecer. Nesse sentido, muitos pesquisadores têm recorrido a técnicas de mineração de dados educacionais para tentar identificar alunos com alta probabilidade de evasão. Por exemplo, existem trabalhos que usam tais técnicas em bases de dados extraídas de sistemas acadêmicos [Manhães et al. 2012, Junior et al. 2019] e em bases de dados contendo os perfis socioeconômicos dos alunos [Pereira et al. 2019].

Neste trabalho, propomos um modelo de previsão de evasão em disciplinas básicas de programação que aplica técnicas de mineração de dados nos *logs* de sistemas Juízes on-line. Juízes on-line são sistemas de correção automática de códigos-fonte comumente empregados como ferramentas de apoio pedagógico em disciplinas de programação [Galvão et al. 2016]. À medida que os alunos usam um sistema Juiz on-line para submeter as soluções dos exercícios de programação propostos pelo professor, os *logs* desse sistema podem armazenar informações úteis sobre o engajamento e sobre o desempenho dos alunos. Uma vez disponíveis, essas informações abrem possibilidades para novas pesquisas [Dwan et al. 2017], especialmente na área de mineração de dados educacionais. Neste trabalho, propomos o uso dessas informações para a tarefa de previsão de evasão.

Nos experimentos apresentados neste trabalho, foram usados 38 atributos obtidos a partir da interação de 2010 estudantes com o Juiz on-line *CodeBench*. Como resultado, o modelo proposto foi capaz de alcançar uma acurácia média na previsão de evasão de 91,96% nas seis primeiras semanas de aula.

Este artigo está organizado da seguinte forma: na Seção 2 são apresentados os trabalhos relacionados, na Seção 3 é apresentada a metodologia utilizada, na Seção 4 é apresentada a análise experimental e os resultados, seguido da conclusão, na Seção 5.

2. Trabalhos relacionados

O uso da mineração de dados educacionais em trabalhos científicos tem crescido nos últimos anos. Uma das razões para isso é o aumento da demanda por cursos de educação à distância e do crescente uso de ferramentas de aprendizagem que possibilitem a coleta e armazenamento de dados dos estudantes [Schlemmer and Portal 2016].

Dwan et al. (2017) propõe um modelo de previsão que usa técnicas de aprendizagem de máquina para identificar se um dado aluno será aprovado ou não em disciplinas de introdução à programação. Foram construídos modelos usando *Support Vector Machine* (SVM), *Random Forest* (RF), *AdaBoosting* (AB), *Decision tree* (DT) e *K-Nearest Neighbours* (KNN). Os modelos foram avaliados através de uma base de dados balanceada,

obtendo uma acurácia de 78,3%. Foram usados 22 atributos numéricos, que representam o comportamento dos alunos ao utilizar um Juiz on-line, especialmente relacionados à quantidade de submissões e erros que os códigos produziram em testes e submissões.

Seguindo a linha de investigação de abandono de estudos, Manhães et al. (2012) usaram os dados do sistema acadêmico da Universidade Federal do Rio de Janeiro, do curso de Engenharia Civil, de 1994 a 2005, com 543 alunos concluintes, 344 não concluintes e 887 calouros. Os atributos são informações das disciplinas do primeiro período (notas e situação final). Foram realizados três experimentos, com dez algoritmos de aprendizagem de máquina em cada. O melhor modelo apresentou acurácia de 80% em dizer se um calouro vai desistir do curso, a partir das notas inicias.

Na mesma linha, Junior et al. (2019) usam mineração de dados educacionais para identificar perfis de alunos propensos a desistir do curso de Sistemas de Informação, da Universidade Federal do Rio Grande do Norte. Foram feitos experimentos com KNN e J48, usando dados pessoais, dados de recebimento de auxílio e de disciplinas cursadas por 216 alunos. O trabalho apresenta precisão na classificação variando entre 90% e 95%.

Pereira et al. (2019) tentam identificar precocemente se estudantes *non-majors* de turmas de programação podem desistir da disciplina, usando dados socioeconômicos, do Juiz on-line e de controle acadêmico da universidade. Foram usados treze atributos, na maioria qualitativos e características socioeconômicas dos alunos. A base de dados tinha 1016 registros de alunos de três semestres e foi divida em 66% para treino e 34% para teste. Usando o *Weka*, foram utilizados os algoritmos CfsSubsetEval, para selecionar os atributos e o C4.5 para ordenar os atributos mais relevantes. Além dos algoritmos de aprendizagem de máquina *Neural Network*, AB, KNN e RF para criar os modelos de classificação dos discentes. Os resultados mostraram que o melhor modelo foi o criado com o algoritmo AB, que apresentou uma acurácia de 70,1% no conjunto de teste. Apesar do trabalho que está sendo proposto ter o mesmo objetivo de Pereira et al. (2019), a diferença é que eles fazem uso de variáveis sociodemográficas, enquanto esta proposta utiliza somente dados do comportamento do estudante no Juiz on-line.

Os trabalhos mencionados apresentam estratégias para identificar discentes com tendência para desistir. Este trabalho se difere por propor o uso de técnicas de aprendizagem de máquina combinadas com evidências extraídas das interações dos estudantes em sistemas Juízes on-line. Embora o trabalho de Dwan et al. (2017) tenha proposto o uso de *logs* de Juízes on-line para prever as notas dos alunos em disciplinas de programação, não foram encontrados trabalhos que usam tais *logs* para o problema de previsão de evasão.

3. Metodologia aplicada

Este trabalho visa identificar, o mais cedo possível, estudantes de Introdução à Programação de Computadores (IPC) com alta probabilidade de desistir da disciplina. Para isso, foram adotadas técnicas de mineração de dados em uma base de logs gerada a partir da interação dos estudantes com o juiz on-line *CodeBench*, que é utilizado como apoio a professores e alunos da Universidade Federal do Amazonas nas turmas de IPC.

3.1. Descrição dos dados

O Juiz on-line *CodeBench* foi usado como apoio em 14 cursos de Ciências Exatas e Engenharias, no primeiro período letivo dos anos de 2016 a 2019. Foram avaliados os se-

mestres ímpares por haver um padrão entre o número de turmas e por essas turmas serem compostas, em sua maioria, por alunos que estão cursando a disciplina pela primeira vez.

Os dados extraídos são de avaliações divididas em dois tipos: *homework* e *exam*, ambas obrigatórias para todas as turmas. As avaliações do tipo *homework* são compostas por, em média, 10 exercícios e as do tipo *exam* por, em média, 2 exercícios. Os atributos foram extraídos separadamente para cada tipo de avaliação, de forma a tornar os dados mais atômicos e livres para os experimentos.

Foram selecionados 38 atributos de 2010 estudantes das 37 turmas, tais como: identificadores do aluno, da turma e do semestre, quantidade de *logins* que o estudante realizou em um intervalo de tempo, a média de tempo ativo no Juiz on-line (cada interação possui uma data e hora, caso o intervalo de tempo inativo seja maior que três minutos, esse intervalo é descartado, caso seja menor, é acumulado para extração da média). Além desses, foram extraídos atributos relacionados aos dois modelos de avaliação: *exam* e *homework*. A seguir é realizada uma descrição desses atributos. Dois atributos são extraídos de cada descrição, relacionados aos dois tipos de avaliação.

- **Média de submissões corretas:** média da quantidade de submissões com as respostas corretas das questões das avaliações do tipo *exam* e *homework*;
- **Média de submissões compiladas e não corretas:** média da quantidade de submissões que compilaram, mas não produziram a resposta correta, das avaliações do tipo *exam* e do tipo *homework*, dentro do intervalo de tempo, por questão;
- **Média da quantidade de submissões com erro:** média da quantidade de submissões que produziram algum tipo de erro na compilação das avaliações do tipo *exam* e *homework*, dentro do intervalo de tempo por questão;
- **Média do tempo de resolução:** o tempo utilizado para resolver cada questão é acumulado, tanto para a tipo *exam*, quanto para *homework*, depois é extraída a média do tempo de resolução de cada questão. Este atributo guarda a média das médias de resolução das questões;
- **Média da quantidade de testes de cada questão:** guarda a média da quantidade de testes de cada questão das avalições do tipo *exam* e do tipo *homework*;
- **Média da quantidade de testes com erro:** média da quantidade de testes que produziram erro, para cada questão das avaliações do tipo *exam* e *homework*;
- **Média das médias acumuladas de tempo entre os testes:** média das médias acumuladas de cada intervalo de tempo entre os testes das questões das avaliações do tipo *exam* e do tipo *homework*;
- **Média de linhas de código válidas:** média da quantidade de linhas válidas de código escrito de cada questão das avaliações do tipo *exam* e do tipo *homework*. Foram consideradas linhas válidas, qualquer linha que não seja comentário ou linha em branco no código;
- **Total de submissões corretas:** soma de todas as submissões que produziram resposta corretas de cada questão das avaliações *exam* e *homework*;
- **Total de submissões incorretas:** soma de todas as submissões incorretas de cada questão das avaliações do tipo *exam* e do tipo *homework*;
- **Total de submissões que não compilam:** soma de todas as submissões que não compilaram de cada questão das avaliações do tipo *exam* e do tipo *homework*;
- **Total de tempo entre submissões:** soma das médias dos intervalos de tempo entre cada submissão de cada questão das avaliações do tipo *exam* e do tipo *homework*;

- **Total de testes:** antes de submeter a questão para ser avaliada pelo Juiz on-line, o aluno pode testar o código. Este atributo soma a quantidade de testes de cada questão das avaliações do tipo *exam* e do tipo *homework*;
- **Total de testes com erro:** soma o total de testes que produziu algum erro de cada questão das avaliações do tipo *exam* e do tipo *homework*;
- **Soma das média do tempo entre cada teste:** soma das médias dos intervalos de tempo entre cada teste, de cada questão do tipo *exam* e do tipo *homework*
- **Soma das linhas de código válidas:** soma das linhas de códigos válidas das questões do tipo *exam* e do tipo *homework*;

3.2. Composição das bases de dados

Um período letivo é dividido em 7 módulos, onde cada módulo tem uma lista de exercícios (*homework*) que deve ser resolvida pelos estudantes. Ao final do módulo é realizada uma avaliação (*exam*). Dessa forma, para tentar identificar a evasão de forma mais precoce possível, foram criadas três bases:

- **Base de Dados 1 (1AP):** informações de utilização do Juiz on-line do primeiro módulo, ou seja, informações de resolução do *homework* até a primeira prova (*exam*), o que equivale à segunda semana de aula;
- **Base de Dados 2 (2AP):** informações desde o início até o segundo módulo, ou seja, informações de resolução do *homework* até a segunda prova (*exam*), aproximadamente o primeiro mês de aula;
- **Base de Dados 3 (3AP):** informações até o terceiro módulo, ou seja, informações de resolução do *homework* até a terceira prova (*exam*), aproximadamente os primeiros 45 dias após o início das aulas;

3.3. Algoritmos de mineração de dados educacionais

Os atributos apresentados na seção 3.1 refletem o comportamento dos alunos durante as atividades no Juiz on-line. Esses dados evidenciam que os valores divergem muito entre os alunos, confirmando a diversidade de tipos de alunos e dos métodos de aprendizagem.

Para conduzir os experimentos de mineração em torno desses dados, foram selecionados um algoritmo de aprendizagem de máquina com base probabilística e três baseados em árvore de decisão, que são métodos preditivos de alta estabilidade, acurácia e facilidade de interpretação. Além disso, de acordo com Dwan et al. (2017), os algoritmos baseados em árvore mostraram os melhores resultados em bases de dados extraídas de Juízes on-line. Desta forma, os algoritmos selecionados foram: *Random Forest* (ARF), *Extra tree classifier* (ETC), *Xgboost* (XGB), *Xgboost with early stopping* (XES) e *Gaussian naive bayes* (GNB).

O *Random Forest* é um dos algoritmos de aprendizagem de máquina mais utilizados [Hamoud et al. 2018], visto que sua flexibilidade e facilidade de uso garantem bons resultados, na maioria das vezes. O uso de um grande número de árvores (florestas) facilita a avaliação e seleção das melhores evidências para a tarefa de predição. Já o *Extra tree classifier (ETC)* é um método baseado em árvores de decisão [Jauhari and Supianto 2019] que, tal qual o método anterior, randomiza as decisões sobre os subconjuntos de dados para minimizar o aprendizado e o ajuste excessivo dos dados, característica que pode medir a aprendizagem. Este método introduz mais variações no conjunto.

Xgboost é um modelo em crescente utilização, fato decorrente de sua acurácia e otimização computacional [Chen et al. 2015]. É baseado em árvores de decisão e utiliza uma estrutura de *Gradient Boosting*. Por possuir essas características, é muito utilizado em competições de ciência de dados. Diferente dos anteriores, o algoritmo *Gaussian Naive Bayes* é um classificador probabilístico [Wasif et al. 2019], utilizado normalmente quando os atributos que descrevem as instâncias são condicionalmente independentes.

Com exceção do *Gaussian Naive Bayes (GNB)*, todos os modelos escolhidos são baseados em estratégias de árvore sobre os atributos. A escolha desses algoritmos se deve à tipologia dos dados e à facilidade de experimentação, visualização e interpretação [Cieslak and Chawla 2008]. No caso de Xgboost (XGB), utiliza-se uma estrutura *Gradient Boosting* com e sem *Early Stooping*, estratégia utilizada para evitar *Overfitting*.

Os algoritmos citados nesta seção são usados nos experimentos e, em todos os resultados, foi utilizado o teste estatístico de Friedman, conforme descrito a seguir.

3.4. Teste de Friedman

O teste de Friedman é um teste não-paramétrico utilizado para comparar dados amostrais vinculados, ou seja, quando o mesmo indivíduo é avaliado mais de uma vez [Hastie et al. 2009]. Nos experimentos deste trabalho, são considerados os resultados individuais obtidos de cada turma ao aplicar cada uma das técnicas mencionadas.

O teste de Friedman não utiliza diretamente os dados numéricos, mas sim os postos ocupados por eles (turmas) após a ordenação feita para cada grupo (classificador) separadamente. O teste é muito utilizado para validar desempenhos significativos, ou não, de determinadas técnicas sobre conjuntos de dados, destacando o ganho de uma determinada técnica sobre as outras consistentemente.

4. Análises experimentais e resultados

Esta seção apresenta como os experimentos foram conduzidos e os resultados alcançados. Foram feitas duas análises, uma que avalia a performance por turma e outra que avalia o desempenho geral, considerando as 37 turmas analisadas.

4.1. Performance por turmas

Nos experimentos feitos para cada uma das turmas, tentou-se verificar a performance das técnicas em turmas com números reais de alunos (aproximadamente 40 alunos por turma), e serve como referência para os demais experimentos. Nesta etapa, optou-se pelo método de validação cruzada com dez partições (*folds*). Esse processo garante que todos os discentes de uma turma sejam testados com um dos dez modelos gerados. A métrica utilizada para a avaliação dos resultados é a acurácia. Ao calcular as médias dos resultados de cada uma das turmas para as diferentes técnicas (Tabela 1), pode-se observar que o algoritmo ETC (88,62% e 91,63% de acurácia) se comportou melhor que os demais, destacando diferenças significativas ($\rho < 0.05$) nesses resultados na primeira e terceira avaliações. Nessa tabela, cada linha (1AP, 2AP e 3AP) representa os resultados para cada uma das bases comentadas na Seção 3.2.

Na matriz de confusão apresentada na Tabela 2, é possível observar que, em média, o número de alunos desistentes classificados como não-desistentes é pequeno (dois).

Tabela 1. Média da acurácia dos modelos por turma.

	ARF	ETC	XGB	XES	GNB
1AP	0,8683	**0,8862**	0,8705	0,8558	0,8111
2AP	0,9059	**0,9163**	0,9074	0,8881	0,8758
3AP	**0,9317**	0,9314	0,9116	0,9083	0,9148

Por outro lado, os alunos não-desistentes classificados como desistentes (seis), é um número um pouco maior, que não representa impacto negativo uma vez que, se esses alunos forem considerados desistentes e receberem tratativa especial, não há problema pois eles não tendem a desistir.

Tabela 2. Matriz de confusão dos modelos ETC por turma, após segunda avaliação.

		Predição 2AP	
		Desistente	Não-desistente
Real	Desistente	14,08 $\pm$ 6,42	0,89 $\pm$ 0,92
	Não-esistente	3,70 $\pm$ 1,69	35,65 $\pm$ 5,61

O algoritmo GNB teve o pior desempenho quando comparado aos demais. Esse resultado já era esperado e o método foi propositalmente escolhido para ser a base de referência (*baseline*) dos experimentos. Em geral, os resultados melhoram com o incremento de dados (ou seja, 15 dias, 30 dias e 45 dias de utilização do Juiz on-line). Os resultados obtidos, neste primeiro experimento, podem ser considerados positivos, porém, o viés do conhecimento a priori da classificação podem reduzir a sua generalização.

4.2. Performance para generalizar

Com o intuito de generalizar os modelos inferidos sobre os dados de turmas individuais, realizaram-se experimentos onde, cada modelo treinado com uma turma foi testado com as demais. Pode-se inferir que, nestes experimentos não é utilizado o conhecimento a priori dos discentes da mesma turma para classificar, e sim o conhecimento de outras turmas. Portanto, utiliza-se um conhecimento a priori de uma turma para predizer o comportamento das outras. A Tabela 3 apresenta o comportamento médio dos experimentos turma a turma. É possível verificar que se mantém o padrão dos experimentos anteriores (comportamento médio dos modelos utilizados para cada uma das turmas), destacando diferenças significativas entre os modelos, com $\rho < 0.01$ nas duas primeiras (1AP e 2AP) e $\rho < 0.05$ nas seis semanas de aula (3AP).

Tabela 3. Médias do desempenho de todos os experimentos.

	ARF	ETC	XGB	XES	GNB
1AP	0,8136	**0,8561**	0,8035	0,8035	0,7691
2AP	0,8698	**0,8959**	0,8627	0,8627	0,8351
3AP	0,9187	**0,9194**	0,9057	0,9057	0,9022

Na matriz de confusão da Tabela 4 pode-se observar que, em média, o número de alunos desistentes classificados como não-desistentes representa menos de 10% do número total de desistentes. A Tabela 5 apresenta o comportamento médio das turmas, ou seja, a média de acurácia das turmas como generalizadoras ao tentar predizer as demais.

Nas últimas colunas foram registrados os melhores e piores resultados dentre modelos testados. Analisando individualmente, os melhores resultados estão entre 88,71% e 94,73%, e os piores oscilaram entre 45,65% e 48,76%, conforme apresentado na última coluna.

Tabela 4. Matriz de confusão dos modelos ETC de todas as turmas como generalizadoras após a terceira avaliação.

		Predição 2AP	
		Desistente	Não Desistente
Real	Desistente	512,51 $\pm$ 38,01	31,59 $\pm$ 24,80
	Não Desistente	127,70 $\pm$ 42,63	1283,86 $\pm$ 22,30

Tabela 5. Desempenho médio de todos os experimentos com as melhores e as piores desempenhos.

	ARF	ETC	XGB	XES	GNB	Melhor	Pior
1AP	0,8136	**0,8543**	0,7845	0,8035	0,7691	0,8871	0,4565
2AP	0,8678	**0,8913**	0,8360	0,8627	0,8351	0,9166	0,4770
3AP	0,9147	**0,9196**	0,9016	0,9057	0,9022	0,9473	0,4876

Em uma análise mais detalhada, é possível identificar as melhores e piores turmas como generalizadoras. A Tabela 6 apresenta um resumo, das cinco melhores e cinco piores performances de modelos (independente do algoritmo de Aprendizado de Máquina), treinados com dados de dez das 37 turmas analisadas. Os modelos foram treinados com os dados de uma turma e testados com os dados das demais 36 turmas.

Tabela 6. Acurácias médias das melhores e piores turmas como generalizadoras.

	1AP	**2AP**	**3AP**
1 º Melhor	0,8742	0,9034	0,9351
2 º Melhor	0,8666	0,8998	0,9339
3 º Melhor	0,8637	0,8985	0,9334
4 º Melhor	0,8617	0,8977	0,9329
5 º Melhor	0,8597	0,8960	0,9323
5 º Pior	0,7451	0,8223	0,8913
4 º Pior	0,7392	0,8164	0,8902
3 º Pior	0,6722	0,7715	0,8563
2 º Pior	0,6112	0,6738	0,8003
1 º Pior	0,5787	0,6591	0,7431

Como pode-se inferir, o número de alunos em cada turma pode ser diferente e, portanto, o número de instâncias nos conjuntos de dados pode variar em cada experimento. Com base nisso, tentou-se verificar o número de alunos classificados de maneira incorreta, com foco nos alunos desistentes (classificação original), ou seja, aqueles que são desistentes mas não foram classificados desta forma (falso negativo). O foco nos desistentes justifica-se uma vez que, este é o interesse principal da pesquisa e, acredita-se que, os alunos que não tendem a desistir e foram classificados como desistentes (falso positivo), não influenciarão na evasão escolar das instituições.

Ao analisar os resultados do método ETC, em média os modelos tiveram acurácia de 69,7% na identificação dos alunos desistentes (640 alunos), ao terminar a primeira

avaliação. Este número foi incrementado para 74,5% e 80,1% nas datas da segunda e terceira avaliações, respectivamente. Considerando esses resultados, pode-se avaliar como positiva a performance dos três modelos baseados em árvore de decisão, uma vez que conseguiu identificar no mínimo 7 de cada 10 alunos desistentes nas primeiras semanas de atividades, o que pode facilitar o trabalho e a atenção personalizada sobre esses discentes.

5. Conclusão e trabalhos futuros

Este trabalho teve como objetivo utilizar modelos de mineração de dados para identificar, nas primeiras semanas de aula, a possibilidade de um aluno desistir da disciplina de Introdução à Programação. Foram utilizados dados da utilização do Juiz on-line *CodeBench* de 2010 estudantes, divididos em 37 turmas, no período de 2016 a 2019. Para os experimentos foram utilizados 38 atributos, na maioria relacionados às atividades de exercícios e provas. As bases compreendem três períodos de tempo distintos: as primeiras duas semanas de aula, as primeiras quatro e as primeiras seis semanas de aula.

Foram empregados cinco modelos nos experimentos. Destaque para o modelo ETC, por apresentar acurácia média de 88,6% e 91,6%, para treinamento e teste com turmas individuais, usando os dados coletados até a primeira e segunda provas de cada turma (duas a quatro semanas de aula). Além desse desempenho, o modelo apresenta a melhor acurácia média (85,6%, 89,6% e 91,9%) quando treinado com dados de uma turma e testado em outras turmas, para todas as bases de dados. Ao se observar apenas as classificações de alunos desistentes, que são o foco principal da pesquisa, o modelo ETC responde com precisão de 69,7% para a primeira base de dados, 74,5% para a segunda e 80,1% para a terceira. Esses resultados confirmam, assim, um bom índice de acerto, no treino e teste com turmas diferentes, em estudantes que correm o risco de evadir nas primeiras semanas de aula. Ressalta-se que, quanto mais cedo se detectar o risco de um estudante desistir, mais rápido podem ser tomadas ações para evitar a evasão.

Como trabalhos futuros pretende-se gerar experimentos retirando os dados de alunos que, de acordo com os atributos aqui utilizados, tiveram comportamento parecido com os demais nas bases de dados, mas desistiram depois das avaliações iniciais, e analisar as características que possuem maior relevância na tarefa de classificação preditiva.

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, além do apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001 e da PROPESP/UEA através do PBICT.

Referências

Chen, T., He, T., Benesty, M., Khotilovich, V., and Tang, Y. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, pages 1–4.

Cieslak, D. A. and Chawla, N. V. (2008). Learning decision trees for unbalanced data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 241–256. Springer.

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. In *Brazilian Symposium on Computers in Education (SBIE)*, volume 28, page 1507.

Galvão, L., Fernandes, D., and Gadelha, B. (2016). Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação. In *Brazilian Symposium on Computers in Education (SBIE)*, volume 27, page 140.

Hamoud, A., Hashim, A. S., and Awadh, W. A. (2018). Predicting student performance in higher education institutions using decision tree analysis. *International Journal of Interactive Multimedia and Artificial Intelligence*, 5:26–31.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

Jauhari, F. and Supianto, A. A. (2019). Building student's performance decision tree classifier using boosting algorithm. *Indonesian Journal of Electrical Engineering and Computer Science*, 14(3):1298–1304.

Junior, I. B., Rabelo, H., Naschold, A. M. C., Ferreira, A. M., Burlamaqui, A., de Souza Rabelo, D. S., and Valentim, R. (2019). Uso de mineração de dados educacionais para a classificação e identificação de perfis de evasão de graduandos em sistemas de informação. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 8, page 159.

Khosravi, H. and Cooper, K. M. (2017). Using learning analytics to investigate patterns of performance and engagement in large classes. In *Proceedings of the 2017 acm sigcse technical symposium on computer science education*, pages 309–314.

Maia, M. d. C. and Meirelles, F. d. S. (2005). Tecnologias de informação e comunicação e os índices de evasão nos cursos a distância. In *Proceedings of 12th International Congress of Distance Education*.

Manhães, L. M. B., Da Cruz, S. M. S., Costa, R. J. M., Zavaleta, J., and Zimbrão, G. (2012). Previsão de estudantes com risco de evasão utilizando técnicas de mineração de dados. In *Brazilian symposium on computers in education (sbie)*, volume 1.

Pereira, A. F. S., de Carvalho, L. S. G., and Souto, E. (2019). Predição de evasão de estudantes non-majors em disciplina de introdução à programação. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 8, page 178.

Schlemmer, E. and Portal, C. (2016). Estratégias para minimizar a evasão na educação a distância: o uso de um sistema de mineração de dados educacionais e learning analytics.

Wasif, M., Waheed, H., Aljohani, N. R., and Hassan, S.-U. (2019). Understanding student learning behavior and predicting their performance. In *Cognitive Computing in Technology-Enhanced Learning*, pages 1–28. IGI Global.

Watson, C. and Li, F. W. (2014). Failure rates in introductory programming revisited. In *Proceedings of the 2014 Conference on Innovation Technology in Computer Science Education*, ITiCSE '14, page 39–44, New York, NY, USA. Association for Computing Machinery.