O uso de estimativas de conhecimento do aluno em programação de computadores em modelos de detecção da emoção confusão livres de sensores

Tiago R. Kautzmann[1], **Gabriel de O. Ramos**[1], **Patrícia A. Jaques**[2]

[1]Programa de Pós-Graduação em Computação Aplicada (PPGCA)
Universidade do Vale do Rio dos Sinos (UNISINOS) – São Leopoldo – RS – Brasil

[2]Programa de Pós-Graduação em Informática (PPGInf)
Universidade Federal do Paraná (UFPR) – Curitiba – PR – Brasil

tkautzmann@gmail.com, gdoramos@unisinos.br, patricia@inf.ufpr.br

Abstract. *Detecting student confusion allows the computer-based learning environment to perform actions that help student regulate their confusion and benefit from it. The article presents evidence on the effects of using data on student knowledge estimates and student interaction with the environment on the performance of sensor-free models of student confusion detection in computer programming tasks. Machine learning models were trained with samples of data collected from 62 students during five months in programming classes. The results presented positive evidence supporting the study approach. The article also describes scenarios where the approach is most advantageous.*

Resumo. *Detectar a confusão do aluno permite ao ambiente computacional de aprendizagem realizar ações que ajudem o aluno a regular sua confusão e a se beneficiar dela. O artigo apresenta evidências sobre os efeitos de usar dados sobre estimativas de conhecimento do aluno, além de dados sobre a interação dele com o ambiente, no desempenho de modelos de detecção da confusão do aluno livres de sensores em tarefas de programação de computadores. Modelos de aprendizado de máquina foram treinados com amostras compostas por dados coletados de 62 alunos, durante cinco meses, em turmas de programação. Os resultados apresentaram evidências positivas que suportam a abordagem do estudo. O artigo também descreve cenários onde a abordagem é mais vantajosa.*

1. Introdução

As emoções possuem papel importante na aprendizagem de alunos, pois afetam processos cognitivos como a memorização e a tomada de decisão, e podem tanto beneficiar como prejudicar o processo de aquisição de conhecimentos [Pekrun 2011]. Computadores têm sido cada vez mais utilizados para promover a aprendizagem e as emoções também têm sido consideradas neste contexto, o que inclui desenvolver computadores que detectam e respondem às emoções dos seus usuários [Picard 2010].

O presente estudo tem interesse em modelos computacionais que detectam a confusão do aluno durante tarefas de aprendizagem de programação de computadores. A confusão é uma emoção propensa de ocorrer neste contexto [Bosch and D'Mello 2017,

DOI: 10.5753/sbie.2022.225768

Bosch et al. 2013, Coto et al. 2021], pois o aluno precisa entender o problema subjacente à tarefa de programação, e mobilizar esforços cognitivos para identificar e relacionar conhecimentos e estratégias de resolução de problemas [Lehman et al. 2013, Chi and Ohlsson 2005]. Quando o aluno está confuso, ele entra em um estado de desequilíbrio cognitivo, resultante de contradições, conflitos ou informações erradas apresentadas a ele [D'Mello and Graesser 2014]. Quando este desequilíbrio ocorre, além de entrar em confusão, o aluno poderá experimentar um conjunto de emoções negativas para a aprendizagem, como a frustração e o tédio, que o levam ao desengajamento [D'Mello et al. 2014, Silvia 2010]. Quando resolvida, a confusão pode levar a aprendizagem a níveis mais profundos, pois o desequilíbrio cognitivo pode levar o aluno a mobilizar esforços cognitivos [D'Mello and Graesser 2014]. Ambientes computacionais de aprendizagem que identificam a confusão do aluno, podem tomar decisões instrucionais importantes, como ajudar o aprendiz a regular sua confusão [Arguel et al. 2019].

O estudo também tem interesse em modelos livres de sensores, que não utilizam sensores conectados ao computador, como *webcams*. Utilizam apenas dados sobre a interação com a interface gráfica, através de teclado e *mouse* [Botelho et al. 2017]. Modelos livres de sensores não são tão intrusivos e são mais fáceis de serem aplicados em escala [Yang et al. 2019, Arroyo et al. 2009] e em escolas com limitações econômicas.

Uma revisão da literatura encontrou três trabalhos relacionados que apresentaram modelos livres de sensores de detecção da emoção confusão em tarefas de programação [Felipe et al. 2012, Lee et al. 2011, Vea and Rodrigo 2017]. Estes trabalhos apresentaram restrições importantes. Um deles [Felipe et al. 2012] descreveu um modelo detector de confusão que utilizou dados derivados do pressionamento de teclas durante a codificação, como velocidade média de digitação e quantidade de uso de *backspaces*. Apesar de ter obtido um bom desempenho (*kappa* = 0,856), este trabalho utilizou dados de uma quantidade pequena de alunos (6) no treinamento dos modelos. Outro trabalho [Vea and Rodrigo 2017] também utilizou dados sobre pressionamento de teclas, além de dados derivados de ações com o *mouse*. Este trabalho obteve desempenho moderado (*kappa* = 0,572 e precisão = 0,74), ainda abaixo de uma concordância esperada para juízes humanos (*kappa* = 0,6) [Ocumpaugh 2015]. Um terceiro trabalho [Lee et al. 2011] buscou detectar a confusão usando dados sobre compilações de códigos, como a quantidade de erros na compilação. Apesar de ter encontrado um bom desempenho (*kappa* = 0,86), o detector deste trabalho não identifica a confusão durante a codificação, pois dados sobre compilações são obtidos somente após a codificação. A detecção da confusão, neste contexto, poderia ser tardia. A revisão de literatura mostrou que modelos de detecção de confusão livres de sensores em tarefas de programação têm sido pouco explorados.

Os trabalhos relacionados treinaram modelos de aprendizado de máquina supervisionada para a detecção da confusão de alunos. As amostras utilizadas no treinamento desses modelos foram compostas por dados sobre a interação do aluno com o ambiente de programação, como dados sobre movimentos de *mouse* e pressionamento de teclas. Estes dados não possuem relação causal conhecida com a emoção confusão. Discussões sugerem que modelos de detecção de emoções podem se beneficiar de uma abordagem mista que usa dados sem relação causal conhecida com a emoção e dados com relação causal conhecida [D'Mello and Graesser 2014, D'Mello 2020]. Nesse sentido, **o primeiro e principal objetivo do presente estudo é verificar o desempenho de modelos de detecção da**

emoção confusão livres de sensores quando são considerados, além de atributos de dados sobre interação do aluno com o ambiente, ***atributos de dados sobre estimativas de conhecimento do aluno em programação de computadores***.

A abordagem apresentada no presente estudo, de investigar os efeitos de usar **estimativas de conhecimento do aluno** na predição de confusão, é justificada em teorias cognitivas de emoções, que consideram que as emoções possuem um componente cognitivo chamado *appraisal* (avaliação) [Scherer 2005]. Nesta perspectiva, a confusão é uma emoção que surge a partir de uma avaliação cognitiva (*appraisal*) de uma falta de correspondência entre a informação recebida pelo indivíduo e o conhecimento do indivíduo [Silvia 2010, D'Mello and Graesser 2014]. Em outras palavras, a confusão surge quando o indivíduo é confrontado com uma contradição, como quando a informação que chega não pode ser integrada ao mapa mental existente, ou quando o aluno está incerto do que fazer na sequência de uma tarefa [D'Mello and Graesser 2014]. A confusão está associada com *appraisals* de novidade (algo não familiar) e de baixa compreensibilidade [Silvia 2010], e há evidências de uma correlação da confusão com a falta de conhecimento [D'Mello et al. 2009, Lee et al. 2011, Rodrigo et al. 2010] e que a confusão é um bom preditor de resultados de testes de conhecimento [Graesser et al. 2007].

Um segundo objetivo do estudo é verificar quanto tempo de observação do comportamento do aluno no ambiente de programação é necessário para o modelo inferir adequadamente que o aluno está confuso ou não. Os trabalhos relacionados que consideraram janelas de observação baseadas em tempo, utilizaram janelas de até 20 segundos [Felipe et al. 2012, Vea and Rodrigo 2017]. Este objetivo busca reforçar as evidências encontradas na literatura sobre o tamanho adequado da janela de observação do comportamento do aluno. Neste trabalho, foram verificados diferentes tamanhos de janelas de observação de tempo fixo (5, 10, 20, 40, 60, 90, 120, 180, 240 e 360 segundos).

O presente estudo apresenta uma abordagem original não investigada nos trabalhos relacionados (livres de sensores) e nem em trabalhos que utilizaram sensores [Tiam-Lee and Sumi 2019, Tiam-Lee and Sumi 2018, Bosch et al. 2014, Grafsgaard et al. 2011]. São descritos os primeiros passos de uma série de estudos que busca verificar os efeitos de usar atributos de dados sobre estimativas de conhecimento do aluno na detecção da confusão de alunos em tarefas de aprendizagem de programação.

2. Método

Para alcançar os objetivos do estudo, foram treinados e validados diversos modelos de aprendizado de máquina supervisionada que buscam identificar a confusão (e a não confusão) do aluno em amostras de dados que refletem o comportamento do estudante em um ambiente computacional de programação, enquanto realiza tarefas de programação. Foram gerados diversos modelos, e não somente um, pois cada modelo utiliza um diferente algoritmo de aprendizado de máquina e um diferente conjunto de amostras que reflete o tamanho da janela de observação do comportamento do aluno. O estudo verifica o desempenho dos modelos em diferentes métricas (primeiro objetivo) e em diferentes tamanhos de janelas de observação (segundo objetivo).

As amostras utilizadas no treinamento e validação dos modelos foram compostas por atributos de dados sobre estimativas de conhecimento do aluno em programação, juntamente de atributos que representam a abordagem dos trabalhos relacionados (atributos

sobre o uso de *mouse* e teclado), entre outros. Para cada amostra foi vinculado um rótulo de emoção ("confusão"ou "não confusão"), de forma que cada amostra representa um momento da interação do aluno com o ambiente em que estava confuso ou não estava confuso. Espera-se que os modelos de aprendizado de máquina supervisionada aprendam padrões relacionados aos estados de "confusão"e "não confusão", presentes nas amostras.

A análise dos resultados possibilitou apresentar outras informações interessantes: qual algoritmo de aprendizado de máquina possui melhor desempenho no contexto do estudo? quais atributos de dados foram mais significativos nos melhores modelos treinados? As próximas subseções descrevem como foi realizada a coleta de dados e a geração das amostras, além de características dos modelos treinados e a validação deles.

2.1. Coleta de dados

Os modelos foram treinados com amostras geradas a partir de dados de *log* coletados pela interação dos alunos participantes da pesquisa com um ambiente de programação, enquanto realizavam exercícios de programação. Uma versão do software *BlueJ*[1] foi adaptada pela pesquisa para coletar dados de *log* de interação do aluno com o ambiente, coletar relatos de confusão, exibir os enunciados dos exercícios ao aluno e avaliar as soluções do aluno. Esta versão adaptada é chamada a partir daqui como *BlueJ* Afetivo.

Os dados de *log* foram coletados a partir da interação com o *BlueJ* Afetivo de 62 alunos participantes de três turmas de ensino superior e duas turmas de ensino técnico, ambas turmas de introdução a programação de computadores. Os alunos participantes retornaram assinado termos de consentimento e de assentimento[2]. Os alunos utilizaram o *BlueJ* Afetivo como a principal ferramenta de programação dos exercícios atribuídos pelos professores, durante os 5 meses de aula, de fevereiro a julho de 2021.

No *BlueJ* Afetivo, os alunos visualizavam os enunciados de exercícios atribuídos pelos professores. O enunciado de cada exercício foi apresentado ao aluno como uma lista de objetivos. Cada objetivo de exercício foi atrelado a um ou dois componentes de conhecimento (CC primário e/ou secundário) de programação (criação de classe, criação de método, estrutura de seleção *if...else*, estrutura de repetição, entre outros). Isso possibilitou o acompanhamento de quais componentes de conhecimento o aluno estava engajado em determinado momento. Para cada objetivo de exercício, o aluno poderia submeter sua solução para avaliação de um juiz online[3] integrado ao *BlueJ* Afetivo. Os *feedbacks* do juiz online (solução de objetivo correta ou incorreta) foram utilizados por um modelo de conhecimento do aluno *Bayesian Knowledge Tracing* (BKT) [Badrinath et al. 2021], também implementado na pesquisa, que infere diversas estimativas sobre o conhecimento do aluno nos componentes de conhecimento atrelados aos objetivos de exercício. As estimativas geradas são as seguintes: probabilidade do aluno dominar o CC (*pL*); probabilidade do aluno aplicar corretamente o CC na próxima oportunidade (*pCorreto*); probabilidade de aprender o CC (*pT*); probabilidade de esquecer o CC (*pF*); probabilidade de

[1]O *BlueJ* é um ambiente de desenvolvimento de software projetado para ser utilizado no ensino de programação de computadores [Kölling et al. 2003]. Disponível em https://www.bluej.org/.

[2]A presente pesquisa está registrada na Plataforma Brasil sob o CAAE (Certificado de Apresentação para Apreciação Ética) de número 24435519.2.0000.5344.

[3]O juiz online recebe a solução do aluno para um objetivo de exercício e aplica testes unitários que avaliam se a solução foi corretamente implementada. Uma solução é considerada correta quando todos os testes unitários atribuídos ao objetivo de exercício tiveram seus testes satisfeitos.

cometer um deslize no CC (*pS*) e probabilidade de adivinhar o CC (*pG*). Estas estimativas foram utilizadas na geração dos atributos sobre estimativas de conhecimento do aluno.

Cada registro de *log* foi gerado a partir de um dos seguintes eventos no *BlueJ* Afetivo: a) tecla pressionada ou liberada pelo aluno; b) clique do *mouse*; c) movimento do *mouse* (início ou parada); d) erro de sintaxe durante a codificação; e) submissão de solução para avaliação do juiz online; e f) resposta da avaliação do juiz online. No total, foram coletados 13,526664 milhões de registros de *log*. Estes registros foram utilizados na geração dos atributos de dados sobre interação do aluno com o ambiente de programação.

Os rótulos de emoção (“confusão”e “não confusão”), atrelados às amostras dos modelos de aprendizado de máquina supervisionada, foram obtidos pelo método de autorrelato. Sempre que o aluno se sentia confuso durante as tarefas, ele deveria apertar um botão na interface do *BlueJ* Afetivo. Foram coletados 1147 relatos de confusão.

2.2. Amostras

Foram gerados 10 conjuntos de amostras. Cada conjunto representa uma configuração de janela de observação de tempo fixo (5, 10, 20, 40, 60, 90, 120, 180, 240 e 360 segundos). Cada amostra é composta por um conjunto de 65 atributos de dados. Cada atributo sintetiza um determinado comportamento observado no tempo da janela. Os atributos gerados foram classificados nos seguintes tipos: a) teclado, derivados dos registros de *log* sobre pressionamento de tecla (8 atributos); b) *mouse*, derivados dos registros de *log* sobre cliques e movimentos de *mouse* (6 atributos); c) erro de sintaxe, derivado dos registros de *log* sobre erros de sintaxe na codificação (1 atributo); d) submissão de solução, derivados dos registros de *log* sobre submissões ao juiz online (5 atributos); e) interações gerais com a interface gráfica (3 atributos); e f) conhecimento do aluno, derivados das estimativas de conhecimento do aluno, inferidas pelo modelo de conhecimento do aluno BKT (42 atributos). A Tabela 1 mostra alguns dos 65 atributos de dados gerados.

Tabela 1. Exemplos de atributos de dados gerados para as amostras

Atributo de dado	**Tipo**
Quantidade de teclas pressionadas	Teclado
Quantidade de movimentos de *mouse*	*Mouse*
Quantidade de erros de sintaxe no código	Erro de sintaxe
Quantidade de submissões de soluções de objetivos com erro	Submissões
Tempo ocioso total	Interações
Média das probabilidades de aplicar corretamente os componentes de conhecimento do objetivo em andamento	**Conhecimento**
Média das probabilidades de dominar os componentes de conhecimento do objetivo em andamento	**Conhecimento**
Média das probabilidades de transitar do estado “não aprendido”para “aprendido”nos componentes de conhecimento do objetivo em andamento	**Conhecimento**

A Figura 1 mostra exemplos de registros do banco de dados com fragmentos de amostras, que ilustra como as amostras foram formadas. A primeira linha é sobre um registro de amostra referente a uma janela de observação de 20 segundos, no qual o aluno

relatou estar confuso. O registro mostra que, num intervalo de 20 segundos anteriores ao relato de confusão, houveram 14 teclas digitadas e cinco movimentos de *mouse*. Além disso, a média das probabilidades de o aluno acertar os componentes de conhecimento no objetivo de exercício em andamento estava em 82%, e a probabilidade de o aluno dominar os componentes de conhecimento do objetivo de exercício em andamento estava em 71%.

aluno	janela	confuso	nao_confuso	quant_teclas	quant_movimentos_mouse	media_pcorreto_ccs	media_pln_ccs
495	20.000	1	0	14.000	5.000	0.8201698320	0.7175748379
495	20.000	1	0	0.000	15.000	0.8201698320	0.7175748379
495	20.000	1	0	0.000	19.000	0.6786934689	0.5287970711
495	20.000	0	1	11.000	20.000	0.7806274778	0.8552131155
495	20.000	0	1	32.000	8.000	0.8981124742	0.8848881743
495	20.000	0	1	2.000	18.000	0.9139540613	0.9675229613

Figura 1. Fragmento de amostras para uma janela de 20 segundos.

A Tabela 2 mostra a quantidade de amostras geradas nas diferentes janelas (Jan.). A Tabela mostra um balanceamento na quantidade de amostras com os rótulos de "confusão"(C) e "não confusão"(NC), uma característica importante para o treinamento adequado dos modelos de aprendizado de máquina em cada classe de saída [Raschka 2018].

Tabela 2. Quantidade de amostras geradas para cada configuração de janela

Jan.	C	NC	Total
5	963	963	1926
10	949	949	1898
40	933	933	1866

Jan.	C	NC	Total
60	924	924	1848
90	902	902	1804
180	828	828	1656

Jan.	C	NC	Total
240	771	771	1542
360	689	689	1378

2.3. Modelos

A Figura 2 mostra o esquema de modelos gerados pelo estudo. Cada modelo é gerado para um determinado algoritmo e um determinado conjunto de amostras. Foram utilizados 10 diferentes algoritmos de aprendizado de máquina supervisionado (listados à direita da Figura 2) e 10 diferentes conjuntos de amostras (listados à esquerda da Figura 2), resultando em 100 modelos. Na Figura 2, cada linha representa um modelo.

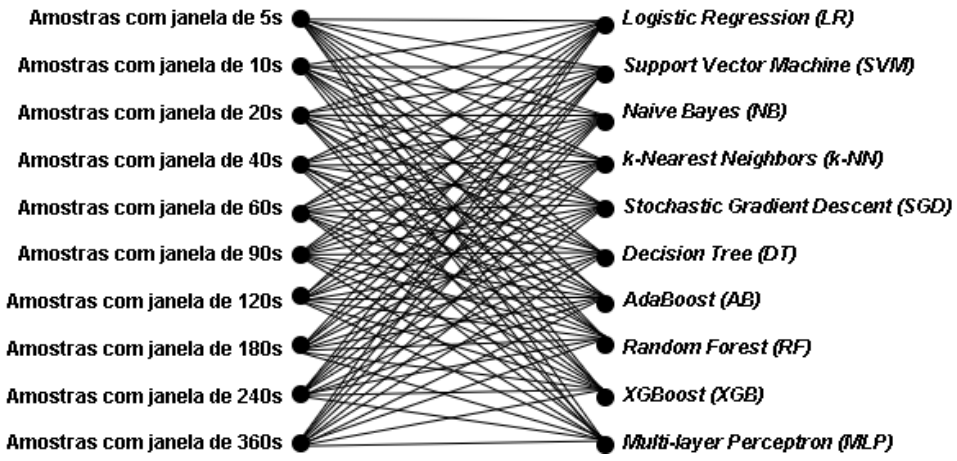

Figura 2. Esquema dos modelos gerados na pesquisa.

Foram selecionados algoritmos de aprendizado de máquina (Figura 2) baseados em árvores (*Decision Tree*), a mesma abordagem dos trabalhos relacionados, e algoritmos que usam conjuntos de árvores (*Random Forest* e *XGBoost*), recomendados para dados tabulares [Shwartz-Ziv and Armon 2022], como os usados nesta pesquisa. Os demais são algoritmos de aprendizado de máquina tradicionais (*Logistic Regression*, *Support Vector Machines*, *Naive Bayes*, *k-Nearest Neighbors*, *Stochastic Gradient Descent*, *AdaBoost*) [Kubat 2017] e de rede neural (*Multilayer Perceptron*). As amostras utilizadas buscaram refletir os tamanhos de janelas de observação dos trabalhos relacionados (até 20 segundos), mas o trabalho também buscou investigar janelas maiores, de até 360 segundos.

Cada modelo foi gerado duas vezes. Na primeira, o modelo foi gerado sem a etapa de seleção de atributos de dados (do inglês *feature selection*) e na segunda, com esta etapa. O método de seleção de atributos de dados permite eliminar atributos irrelevantes que prejudicam o desempenho dos modelos [Kubat 2017]. No total, foram gerados 200 modelos. O método utilizado para a seleção dos atributos de dados foi o *Recursive Feature Elimination* (RFE) [Thi et al. 2008]. O RFE é um método iterativo que treina o modelo diversas vezes, em diversas iterações. A cada iteração, o RFE gera pesos de importância para cada atributo de dados e ranqueia os atributos, dos mais importantes aos menos importantes. Ao final de cada iteração, o método elimina das amostras os atributos de dados menos importantes. No final do processo, o RFE seleciona o conjunto de atributos de dados que obteve o melhor desempenho na métrica de acurácia. O método permitiu ao estudo identificar os atributos de dados mais relevantes nos melhores modelos.

Para gerar cada modelo, foi utilizado um pipeline[4] tradicional, composto pelos componentes de normalização de amostras, otimização de hiperparâmetros, seleção de atributos (em metade dos modelos) e validação dos modelos [Kubat 2017, Raschka 2018].

2.4. Validação dos modelos

A validação de um modelo de aprendizado de máquina verifica se, após o modelo ser treinado com determinado conjunto de amostras (amostras de treino), possui bom desempenho de predição em um outro conjunto de amostras (amostras de teste) que não foi usado no treinamento. O objetivo é verificar se o modelo generaliza bem para dados não vistos no treinamento [Raschka 2018]. O estudo utilizou o método de validação cruzada *k-fold* (k = 10) com amostras estratificadas. O método divide aleatoriamente as amostras de dados em k conjuntos (*folds*) com tamanho igual de amostras em cada conjunto. Então são realizadas k iterações de treinamento e teste. Em cada iteração, são utilizados k - 1 (9) conjuntos de amostras no treinamento do modelo, enquanto o conjunto restante de amostras é usado para o teste do modelo. Ao final das iterações, é calculada a média dos desempenhos nos testes, nas k iterações [Wong 2015], para determinada métrica. O estudo utilizou as seguintes métricas: acurácia, *AUC ROC*, *kappa*, precisão, *recall* e F1.

3. Resultados e discussões

A Tabela 3 apresenta os resultados de acurácia, *AUC ROC*, *kappa*, precisão, *recall* e F1 dos melhores modelos (com melhor acurácia), em cada configuração de janela (Jan.). É possível verificar que o melhor modelo encontrado foi obtido para o conjunto de amostras

[4]Um *pipeline* é uma sequência de componentes de processamento de dados, comumente usada em tarefas de aprendizado de máquina [Paper 2021].

referente a uma janela de observação de 10 segundos, com o algoritmo *Random Forest*. Este modelo obteve os melhores resultados de acurácia, *AUC ROC*, *kappa*, precisão e F1. O melhor resultado de *recall* foi obtido no melhor modelo de janela de 5 segundos.

Tabela 3. Desempenho dos melhores modelos nas diferentes janelas de dados

Jan.	Algoritmo	Acurácia	AUC ROC	*Kappa*	Precisão	*Recall*	F1
5	XGB	0,89197	0,94857	0,78394	0,87273	**0,92523**	0,89495
10	RF	**0,89725**	**0,95082**	**0,79452**	**0,87981**	0,92308	**0,89992**
20	XGB	0,8771	0,93528	0,75419	0,86296	0,91425	0,87957
40	XGB	0,84831	0,91918	0,69661	0,83298	0,87788	0,85218
60	XGB	0,84303	0,91161	0,68611	0,82421	0,87442	0,84813
90	XGB	0,82151	0,90439	0,64303	0,80924	0,85262	0,82502
120	RF	0,82274	0,90144	0,64546	0,81377	0,83801	0,82543
180	XGB	0,80798	0,88887	0,61597	0,80004	0,82252	0,81071
240	XGB	0,80221	0,86712	0,6044	0,79828	0,8093	0,80351
360	RF	0,7961	0,86917	0,5922	0,80007	0,79386	0,79504

Outro resultado interessante é que à medida que as janelas de observação aumentam de tamanho, o desempenho dos melhores modelos nas métricas de desempenho geral (acurácia, AUC ROC e *kappa*) diminui. Esse resultado sugere que modelos de detecção de confusão, no contexto do estudo, funcionam melhor quando o comportamento do aluno é observado em intervalos de tempo menores, como nos intervalos de 5 e 10 segundos.

Sobre os melhores algoritmos de aprendizado de máquina, o *XGBoost* (*eXtreme Gradient Boosting*) [Chen and Guestrin 2016] foi utilizado na maioria dos melhores modelos (7 modelos). O segundo algoritmo mais presente, em 3 modelos, foi o *Random Forest* [Breiman 2001]. Os resultados sugerem a implementadores de modelos de detecção de confusão, no contexto do estudo, que considerem estes algoritmos.

A estudo também verificou quais atributos de dados foram selecionados pelo método RFE nos melhores modelos de cada janela de observação, ou seja, quais atributos foram mais relevantes. Do total de 42 atributos sobre estimativas de conhecimento do aluno, 33 atributos (78,6% do total) foram selecionados nos melhores modelos de todas as configurações de janelas. Os demais atributos que também foram selecionados em todos os melhores modelos, sobre a interação do aluno com o ambiente, estão a quantidade de erro de sintaxe no código, a quantidade de cliques do *mouse* no código, a quantidade de movimentos de *mouse*, o tempo de movimentos de *mouse*, a quantidade de teclas digitadas, a quantidade de teclas de *backspace*, a quantidade de teclas de controle (setas de movimentação), a quantidade de intervalos de digitação e o tempo ocioso no sistema.

A Tabela 4 mostra alguns dos atributos de dados mais relevantes. A tabela mostra valores de média (M) e desvio padrão (DP) desses atributos de dados quando os alunos estavam confusos e quando não estavam confusos. O objetivo é observar o comportamento desses atributos de dados quando os estudantes estavam confusos e quando não estavam confusos. A aplicação de testes estatísticos não paramétricos *MannWhitney U* mostrou que a diferença dos valores médios para cada atributo de dados entre as amostras de confusão e de não confusão são todas significativas, com valor de p abaixo do nível de significância ($\alpha = 0,05$). Observa-se que, apesar de nos momentos de confusão os

alunos terem apresentado menos atividade de digitação, eles passaram mais tempo movimentando o *mouse*. Em relação aos dados sobre estimativas de conhecimento, quando não confusos, os alunos apresentaram maior probabilidade de dominarem e aplicarem corretamente os componentes de conhecimento (CCs) do objetivo do exercício em codificação. Os dados apresentados na Tabela 4 ajudam a explicar os efeitos desses atributos de dados nas classes de interesse ("confusão"e "não confusão").

Tabela 4. Informações estatísticas sobre alguns atributos de dados relevantes

	Confusão		**Não confusão**		***MW U***
	M	**DP**	**M**	**DP**	***p***
Média das probabilidades do aluno dominar os CCs do objetivo de exercício	0,68997	0,16661	0,89796	0,12073	$< 0,001$
Média das probabilidades do aluno aplicar corretamente os CCs do objetivo de exercício	0,71391	0,13421	0,87345	0,10695	$< 0,001$
Quantidade de teclas digitadas	1,40569	3,71375	10,39409	12,57671	$< 0,001$
Tempo total de movimentos de *mouse*	2,21374	1,48403	1,5699	1,46963	$< 0,001$

4. Conclusão

O artigo apresenta um estudo com as primeiras investigações sobre os efeitos de usar estimativas de conhecimento do aluno, justificado por teorias de emoções, juntamente de dados de interação do aluno com o ambiente, em modelos de detecção da emoção confusão livres de sensores, no contexto de tarefas de aprendizagem de programação. A abordagem do estudo, de utilizar atributos de dados sobre estimativas de conhecimento do aluno, é uma abordagem original não investigada em trabalhos relacionados que apresentaram modelos livres de sensores e nem em trabalhos que usaram sensores.

Os melhores resultados dos modelos foram encontrados quando os atributos de dados refletiram observações de 10 segundos do comportamento do aluno: acurácia de 0,89725; *AUC ROC* de 0,95082; *kappa* de 0,79452, precisão de 0,87981; e F1 de 0,89992. O melhor valor de *recall* (0,92523) foi obtido em uma janela de 5 segundos. Os resultados são interessantes, com um valor de *AUC ROC* próximo de 1,0 e valores de acurácia, precisão, *recall* e F1 próximos de 0,9. O melhor valor encontrado para *kappa* (0,79452) é um valor superior ao da concordância esperada para juízes humanos (*kappa* = 0,6) [Ocumpaugh 2015]. Além disso, os resultados mostraram que 78,6% dos atributos (33 atributos) sobre o conhecimento do aluno foram selecionados para composição das amostras de todos os melhores modelos de cada janela de observação, o que indica a relevância desses atributos. Os resultados sugerem que usar atributos de dados sobre o conhecimento do aluno, juntamente de atributos sobre interação com o ambiente, pode ser uma abordagem interessante e que mais investigação é justificada.

Não foi objetivo do estudo comparar diretamente os seus resultados com os apresentados pelos trabalhos relacionados, pois as amostras utilizadas em ambos os estudos são diferentes. No entanto, algumas considerações podem ser feitas. Um dos trabalhos

relacionados [Felipe et al. 2012] obteve *kappa* de 0,856, superior ao encontrado no presente estudo. No entanto, este trabalho utilizou apenas 6 alunos na coleta de dados. Outro trabalho [Vea and Rodrigo 2017] obteve um valor moderado de *kappa* (0,572) e precisão (0,74). Um terceiro trabalho relacionado [Lee et al. 2011] obteve um valor de *kappa* superior ao presente trabalho (0,86), mas utilizou atributos de dados de compilação de código, que não permitem a detecção da confusão durante a codificação. Os trabalhos relacionados não apresentaram resultados para as demais métricas (acurácia, *AUC ROC*, *recall* e *F1*), de forma que, nestas métricas, o presente trabalho apresenta os primeiros resultados no contexto de detecção de confusão por modelos livres de sensores, em tarefas de aprendizagem de programação de computadores.

Os resultados ainda sugerem que os cenários onde a abordagem do estudo é mais vantajosa é quando os atributos de dados refletem observações de comportamento do aluno em intervalos de tempo menores, como nos intervalos de 5 e 10 segundos. Os resultados também sugerem o uso dos algoritmos *XGBoost* e o *Random Forest*, que utilizam conjuntos de árvores de decisão, recomendados para problemas envolvendo dados tabulares [Shwartz-Ziv and Armon 2022], como os usados no estudo. A presente pesquisa reforça as evidências já encontradas na literatura [Shwartz-Ziv and Armon 2022] sobre o bom desempenho desses algoritmos quando são usados dados tabulares.

Os autores do presente estudo pretendem realizar uma série de trabalhos futuros. Um dos trabalhos pretende gerar modelos *baseline* que representam apenas a abordagem dos trabalhos relacionados (apenas dados de interação). Os resultados desses modelos serão comparados com a abordagem investigada (estimativas de conhecimento do aluno + dados de interação) através de testes estatísticos. Também será estendida a análise de relevância dos atributos de dados. Pretende-se também verificar os efeitos de utilizar janelas de dados de tempo variável. Outro estudo futuro pretende identificar o poder de generalização dos modelos para estudantes com características heterogêneas. Esse estudo é interessante, porque modelos que generalizam bem para estudantes com características heterogêneas não precisam ser novamente treinados sempre que são utilizados com alunos com características diferentes dos alunos usados no treinamento.

Apesar dos resultados interessantes encontrados pela pesquisa, com algoritmos de aprendizado de máquina, pode ser uma boa idéia verificar o desempenho da abordagem do estudo com algoritmos de *deep learning*. O mecanismo de detecção de confusão poderia se beneficiar de modelos que consideram o comportamento do aluno ao longo do tempo, como em modelos de redes neurais recorrentes.

Os pesquisadores também pretendem integrar o modelo de detecção de confusão em um ambiente computacional adaptativo de aprendizagem de programação. Ao detectar a confusão do aluno, o ambiente poderia tomar decisões instrucionais, como apresentar *feedbacks* sobre o conteúdo e diminuir as lacunas de conhecimento do estudante, além de ajudar o aluno a gerenciar a sua confusão, ensinando estratégias de regulação emocional e estratégias autorregulatórias [Arguel et al. 2019].

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, da FAPERGS (Processo 17/2551-0001203-8) e do CNPq (processo 306005/2020-4).

Referências

Arguel, A., Lockyer, L., Kennedy, G., Lodge, J. M., and Pachman, M. (2019). Seeking optimal confusion: a review on epistemic emotion management in interactive digital learning environments. *Interactive Learning Environments*, 27:200–210.

Arroyo, I., Cooper, D., Burleson, W., Woolf, B., Muldner, K., and Christopherson, R. (2009). Emotion sensors go to school. In *Frontiers in Artificial Intelligence and Applications*, number 1 in Frontiers in Artificial Intelligence and Applications, pages 17–24. IOS Press.

Badrinath, A., Wang, F., and Pardos, Z. (2021). pybkt: An accessible python library of bayesian knowledge tracing models. *International Educational Data Mining Society*.

Bosch, N., Chen, Y., and D'Mello, S. (2014). It's written on your face: Detecting affective states from facial expressions while learning computer programming. In Trausan-Matu, S., Boyer, K. E., Crosby, M., and Panourgia, K., editors, *Intelligent Tutoring Systems*, pages 39–44, Cham. Springer International Publishing.

Bosch, N. and D'Mello, S. (2017). The affective experience of novice computer programmers. *International Journal of Artificial Intelligence in Education*, 27(1):181–206.

Bosch, N., D'Mello, S., and Mills, C. (2013). What emotions do novices experience during their first computer programming learning session? In *International Conference on Artificial Intelligence in Education*, pages 11–20. Springer.

Botelho, A. F., Baker, R. S., and Heffernan, N. T. (2017). Improving sensor-free affect detection using deep learning. In André, E., Baker, R., Hu, X., Rodrigo, M. M. T., and du Boulay, B., editors, *Artificial Intelligence in Education*, pages 40–51, Cham. Springer International Publishing.

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Chen, T. and Guestrin, C. (2016). Xgboost. volume 42, pages 785–794. ACM.

Chi, M. T. and Ohlsson, S. (2005). *Complex Declarative Learning*. Cambridge University Press.

Coto, M., Mora, S., Grass, B., and Murillo-Morera, J. (2021). Emotions and programming learning: systematic mapping. *Computer Science Education*, pages 1–36.

D'Mello, S., Person, N., and Lehman, B. (2009). Antecedent-consequent relationships and cyclical patterns between affective states and problem solving outcomes. *Frontiers in Artificial Intelligence and Applications*, 200:57–64.

D'Mello, S. K. (2020). Big data in the science of learning. In *Big data in psychological research.*, pages 203–225. American Psychological Association.

D'Mello, S., Lehman, B., Pekrun, R., and Graesser, A. (2014). Confusion can be beneficial for learning. *Learning and Instruction*, 29:153 – 170.

D'Mello, S. K. and Graesser, A. C. (2014). Confusion. In *International handbook of emotions in education*, pages 299–320. Routledge.

Felipe, D. A. M., Gutierrez, K. I. N., Quiros, E. C. M., and Vea, L. A. (2012). Towards the development of intelligent agent for novice c/c++ programmers through affective

analysis of event logs. In *Proc. Int. MultiConference Eng. Comput. Sci*, volume 1, page 2012. Citeseer.

Graesser, A., Chipman, P., King, B., McDaniel, B., and D'Mello, S. (2007). Emotions and learning with autotutor. In *Proceedings of the 2007 Conference on Artificial Intelligence in Education: Building Technology Rich Learning Contexts That Work*, page 569–571, NLD. IOS Press.

Grafsgaard, J. F., Boyer, K. E., and Lester, J. C. (2011). Predicting facial indicators of confusion with hidden markov models. In D'Mello, S., Graesser, A., Schuller, B., and Martin, J.-C., editors, *Affective Computing and Intelligent Interaction*, pages 97–106, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kubat, M. (2017). *An introduction to machine learning*, volume 2. Springer.

Kölling, M., Quig, B., Patterson, A., and Rosenberg, J. (2003). The bluej system and its pedagogy. *Computer Science Education*, 13:249–268.

Lee, D. M. C., Rodrigo, M. M. T., d Baker, R. S., Sugay, J. O., and Coronel, A. (2011). Exploring the relationship between novice programmer confusion and achievement. In *International Conference on Affective Computing and Intelligent Interaction*, pages 175–184. Springer.

Lehman, B., D'Mello, S., Strain, A., Mills, C., Gross, M., Dobbins, A., Wallace, P., Millis, K., and Graesser, A. (2013). Inducing and tracking confusion with contradictions during complex learning. *International Journal of Artificial Intelligence in Education*, 22:85–105.

Ocumpaugh, J. (2015). Baker rodrigo ocumpaugh monitoring protocol (bromp) 2.0 technical and training manual. *New York, NY and Manila, Philippines: Teachers College, Columbia University and Ateneo Laboratory for the Learning Sciences*.

Paper, D. (2021). *Introduction to Deep Learning*, pages 1–24. Apress, Berkeley, CA.

Pekrun, R. (2011). Emotions as drivers of learning and cognitive development. In *New perspectives on affect and learning technologies*, pages 23–39. Springer.

Picard, R. W. (2010). Affective computing: From laughter to ieee. *IEEE Transactions on Affective Computing*, 1:11–17.

Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning.

Rodrigo, M. M. T., Baker, R. S. J., and Nabos, J. Q. (2010). The relationships between sequences of affective states and learner achievement. *Proceedings of the 18th International Conference on Computers in Education*, pages 56–60.

Scherer, K. R. (2005). What are emotions? and how can they be measured? *Social science information*, 44(4):695–729.

Shwartz-Ziv, R. and Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90.

Silvia, P. J. (2010). Confusion and interest: The role of knowledge emotions in aesthetic experience. *Psychology of Aesthetics, Creativity, and the Arts*, 4:75–80.

Thi, H. A. L., Nguyen, V. V., and Ouchani, S. (2008). Gene selection for cancer classification using dca. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5139 LNAI:62–72.

Tiam-Lee, T. J. and Sumi, K. (2018). Adaptive feedback based on student emotion in a system for programming practice. In Nkambou, R., Azevedo, R., and Vassileva, J., editors, *Intelligent Tutoring Systems*, pages 243–255, Cham. Springer International Publishing.

Tiam-Lee, T. J. and Sumi, K. (2019). Analysis and prediction of student emotions while doing programming exercises. In *International conference on intelligent tutoring systems*, pages 24–33. Springer.

Vea, L. and Rodrigo, M. M. (2017). Modeling negative affect detector of novice programming students using keyboard dynamics and mouse behavior. In Numao, M., Theeramunkong, T., Supnithi, T., Ketcham, M., Hnoohom, N., and Pramkeaw, P., editors, *Trends in Artificial Intelligence: PRICAI 2016 Workshops*, pages 127–138, Cham. Springer International Publishing.

Wong, T. T. (2015). Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition*, 48:2839–2846.

Yang, T.-Y., Baker, R. S., Studer, C., Heffernan, N., and Lan, A. S. (2019). Active learning for student affect detection. *International Educational Data Mining Society*.