Trabalho de Conclusão de Curso

Estudo e desenvolvimento de um detector automático de comportamento Gaming the System em resolução de problemas de programação

Julios Suruagil Lins da Rocha
jslr@ic.ufal.br

Orientador:

Evandro de Barros Costa

Co-orientadora:

Hemilis Joyse Barbosa Rocha

Maceió, Dezembro de 2022

Julios Suruagil Lins da Rocha

Estudo e desenvolvimento de um detector automático de comportamento Gaming the System em resolução de problemas de programação

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientadores:

Evandro de Barros Costa

Co-orientadora:
Hemilis Joyse Barbosa Rocha

Maceió, Dezembro de 2022

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária: Taciana Sousa dos Santos – CRB-4 – 2062

R672e Rocha, Julios Suruagil Lins da.
Estudo e desenvolvimento de um detector automático de comportamento Gaming the System em resolução de problemas de programação / Julios Suruagil Lins da Rocha. – 2022.
32 f. : il. color.

Orientador: Evandro de Barros Costa.
Coorientadora: Hemilis Joyse Barbosa Rocha.
Monografia (Trabalho de Conclusão de Curso em Ciência da Computação) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2022.

Bibliografia: f. 30-32.

1. Programação (Computadores). 2. Resolução de problemas. 3. Algoritmos de aprendizagem. 4. Aprendizagem de máquina. I. Título.

CDU: 004.42

Agradecimentos

Meus sinceros agradecimentos a todos os amigos que fiz no curso, aos que fazem parte do instituto de computação e nos dão suporte, e em especial a Kelly Bianca e Hemilis Joyse, que foram fundamentais para conclusão dessa jornada. Agradeço também ao professor Evandro Costa e à servidora Ana Ferreira por todo apoio.

Resumo

Este trabalho está relacionado a um estudo sobre um tipo de comportamento, considerado indesejável, identificado em alguns estudantes, diante de situação de resolução de problemas de programação. Assim, foi investido no desenvolvimento de uma solução para detectar automaticamente a presença do mencionado comportamento. Deste modo, primeiramente foi desenvolvido um ambiente apropriado para coleta de dados, seguindo-se pela preparação desses dados para serem utilizados por algoritmos de aprendizagem de máquina supervisionada, considerando uma tarefa de classificação, permitindo a detecção do comportamento. O desenvolvimento do detector envolveu explorar, avaliar e comparar diferentes algoritmos classificadores. Como um dos resultados, verificamos que o algoritmo MLP apresentou um melhor desempenho, com uma acurácia de 0.84.

Palavras-chave: Aprendizagem de máquina supervisionada, Preparação de dados, Mineração de Dados Educacionais

Abstract

This work is related to a study on a type of behavior, considered undesirable, identified in some students, when faced with a situation of solving programming problems. Thus, we invested in the development of a solution to automatically detect the presence of the mentioned behavior. Thus, firstly, an appropriate environment for data collection was developed, followed by the preparation of these data to be used by supervised machine learning algorithms, considering a classification task, allowing the detection of behavior. The development of the detector involved exploring, evaluating and comparing different classification algorithms. As one of the results, we found that the MLP algorithm performed better, with an accuracy of 0.84.

Keywords: Supervised machine learning, Data preparation, Educational data mining

Lista de Figuras

Lista de Tabelas

Conteúdo

1

Introdução

Na área de Informática na Educação se faz uso de ambientes computacionais de apoio ao ensino e aprendizagem, tendo como exemplo os sistemas tutores inteligentes (ITS), os juízes online, os ambientes virtuais de aprendizagem (AVA). Em todos estes tipos de ambiente, há opções para se configurar um espaço para resolução de problemas, tendo disponíveis recursos de apoio ao estudante, por exemplo dicas e feedback, no momento da solução de problemas. Em particular, os ITS permitem tais facilidades para aprendizagem baseada na resolução de problemas, tendo sido utilizados por alunos com diferentes necessidades, preferências e características, sendo aplicados a uma diversidade de domínios Ghaleb et al. (2018).

Embora o uso de software de tutor cognitivo tenha aumentado o envolvimento e o esforço dos alunos na sala de aula Schofield (1995), alguns alunos responderam à ajuda, feedback e suporte de software de tutoria cognitiva, com um conjunto de estratégias não orientadas para o aprendizado. Esse conjunto de estratégias é chamado de "jogar o sistema"Baker et al. (2004b). Esse comportamento é definido na literatura como “tentando ter sucesso em um ambiente educacional explorando as propriedades do sistema em vez de aprender o material e tentar usar o conhecimento para responder corretamente"Baker and de Carvalho (2008). Além disso, está relacionado a resultados ruins na aprendizagem do aluno (Cocea et al. (2009); Pardos et al. (2013);

O comportamento de jogar o sistema tem sido relatado em vários estudos d Baker et al. (2008, 2010). Além disso, os logs de interação dos alunos com os ITS têm sido utilizados para gerar modelo de aluno com comportamento jogando com o sistema. Tais modelos foram criados usando técnicas de engenharia do conhecimento Aleven et al. (2006); Gong et al. (2010) e aprendizagem de máquina d Baker et al. (2008, 2010). No entanto, um detector ideal deve atender aos seguintes critérios d Baker et al. (2008): i) identificar com precisão uma categoria (ou categorias) de comportamento que é conhecida por estar associada a uma diferença significativa na experiência ou nos resultados do aluno; ii) prever não apenas quais alunos se envolvem no

comportamento, mas quando o fazem; iii) ajudar os pesquisadores a entender melhor esses comportamentos; e iv) generalizar para novas aulas de tutores, ou melhor ainda, tutores inteiramente novos.

Baker et al. e seus colegas realizaram um estudo em 2008 no sentido de comparar os detectores, até então desenvolvidos, utilizando os critérios citados acima. Outros trabalhos têm investido no desenvolvimento de detectores do comportamento *gaming the system*. O trabalho aqui apresentado segue nessa linha, sendo que no domínio de programação, tendo como objetivo geral um estudo desse problema comportamental e desenvolvimento de uma solução para detecção automática da presença desse tipo de comportamento *gaming the system*, considerado indesejável, identificado em alguns estudantes, diante de situação de resolução de problemas de programação.

Para realizar o desenvolvimento do detector, primeiramente tivemos que desenvolver um ambiente apropriado para coleta de dados, criando e anotando uma base de dados, seguindo-se pela preparação desses dados para serem utilizados por algoritmos de aprendizagem de máquina supervisionada, considerando uma tarefa de classificação, permitindo a detecção do comportamento. O desenvolvimento do detector envolveu treinar algoritmos de aprendizagem de máquina para classificação do comportamento *gaming the system* e obter os modelos com melhor acurácia, incluindo explorar, avaliar e comparar diferentes algoritmos classificadores, a exemplo de árvore de decisão, K-NN, rede neural e SVM.

Como um dos resultados deste estudo, verificamos que a rede neural MLP apresentou um melhor desempenho, com uma acurácia de 0.84.

Esta monografia está estruturada do seguinte modo. No capítulo 2, foram apresentados alguns assuntos de apoio ao entendimento deste trabalho. No capítulo 3, discutimos alguns trabalhos relacionados e modelos implementados em outros sistemas. No capítulo 4 é abordado sobre os estudos de *gaming the system* trazendo uma revisando da literatura e fazendo a discussão destes trabalhos. No capítulo 5 falamos sobre as variáveis utilizadas em experimentos semelhantes. No capítulo 6 é falado sobre o sistema que foi desenvolvido e por último o capítulo de conclusão onde é abordado os resultados e as conclusões obtidas por este estudo.

2

Fundamentação Teórica

Neste capítulo abordaremos conceitos de aprendizagem de máquina, incluindo pré-processamento de dados e avaliação de modelos.

2.1 Aprendizagem de máquina

Monard and Baranauskas (2003) explica o conceito de aprendizagem de máquina como uma área da inteligência artificial tendo como propósito a implementação de sistemas com a capacidade de tomar decisões de forma automática. Tradicionalmente, a aprendizagem é dividida em três categorias, sendo: Aprendizagem supervisionada, Aprendizagem não supervisionada e aprendizagem por reforço. Resumidamente, cada uma delas pode ser caracterizada, assim:

- Aprendizagem supervisionada: são treinados por meio de exemplos rotulados, no qual a saída é conhecida.
- Aprendizagem não supervisionada: aqui não tem rotulos, o programa vai analisar os exemplos e com base nisso ele vai determinar se eles podem ser agrupados, formando os agrupamentos.
- Aprendizagem por reforço: é chamado de aprendizagem por reforço pois o algoritmo faz tentativas para chegar em uma resposta.

2.1.1 Aprendizagem supervisionada

Nesta seção vamos falar um pouco mais detalhado sobre a aprendizagem supervisionada, explicando alguns conceitos e definições.

A abordagem de aprendizado supervisionado consiste em utilizar uma coleção de exemplos ou instâncias, já classificados (rotulados), para induzir um modelo que seja capaz de classificar

novas instâncias de forma precisa, com base no aprendizado obtido na fase de treinamento. Nessa abordagem tem-se a figura de um supervisor ou "professor externo", o qual rotula as saídas, considerando uma coleção de exemplos na forma entrada-saída. Neste caso, o algoritmo de AM é treinado a partir de coleções de exemplos rotulados com o objetivo de aprender uma função desejada. Norvig and Russell (2013).

2.2 Análise exploratória

Uma maneira imprescindível de investir na qualidade da descoberta do conhecimento por parte dos algoritmos é a execução de técnicas que pré-processem os dados. Ausência de valores, dados ruidosos, valores inconsistentes, atributos de natureza distintas e redundância nos dados podem facilmente fazer com que toda uma análise fracasse (da Silva et al., 2017).

Nesta sessão discutiremos os procedimentos de pré-processamento utilizados para refinamento dos dados: limpeza, integração e transformação de dados.

2.2.1 Limpeza de dados

A limpeza de dados tem como o objetivo eliminar os dados ausentes ou dados ruidosos que possam atrapalhar a etapa de processamento de dados. As seguintes seções abaixo dentro deste tópico foram observadas a partir do livro da Silva et al. (2017).

Ausência de valores

Para o problema de dados ausentes existem algumas formas de identificar, a primeira é quando os dados não são encontrados dentro de um conjunto de dados, e a segunda é quando eles tem valores associados ao mesmo atributo.

Tem maneiras de resolver esta falha de dados, como:

- Remoção dos outros dados que estão relacionados ao dado que está faltante, como por exemplo remover a linha em um conjunto de dados que existem dados ausentes nela;
- Preenchimento manual dos dados desta maneira ocorre tentativas de encontrar uma resposta baseada em um especialista, no entanto se tiver uma grande quantidade de dados em falta, esse processo pode demorar bastante e pode ocorrer periogo de existir viés nos dados;
- Preenchimento automático é feito baseado na determinação de um valor da constante, em uma média ou mediana dos valores que já existe ou de acordo com o resultado de algum método de análise preditiva, da mesma forma que no preenchimento manual, também ocorre o risco de ter viés nesse caso.

Valores ruidosos

Os valores ruidosos são normalmente chamados de outliers, consistem em dados que foram alterados em relação ao seu dado primário, e que são resultados de erros de medidas, é possivel identifiar quando vocë já sabe o resultado esperável mas surgem valores que não fazem parte da mesma classificação.

Para resolver este tipo de problema, são realizadas as seguintes técnicas:

- Inspeção e correção manual com a análise exploratória é possivel identificar os ruidos e com isso corrigi-los manualmente, porém se a quantidade de dados que estão com esse problema for alta fica desvantojoso fazer essa correção, desta forma se a quantidade de dados for alta é melhor aplicar alguns procedimentos de suavização;
- Identificação e limpeza automática são aplicados algoritmos que são utilizados para suavizar ou anular os ruidos dentro de um grupo de dados, alguns algoritmos fazem a resolução do problema de predição e agrupamento conseguindo fazer a separação dos dados ruidosos de um conjunto de dados.

2.2.2 Integração de dados

São ações que permitem integrar os dados que vem de fontes de dados diferentes, a principal motivação de realizar a integração dos dados é devido a inconsistência e aos valores redundantes, como esses dados vem de fontes distintas os dados vem muito diferentes e é preciso fazer essa integração de forma que é preciso fazer uma formatação. Uma das soluções visa fazer o tratamento das inconsistências que faz a retirada do valor incosistente, a correção manual e a análise dos conjuntos de dados (da Silva et al., 2017).

2.2.3 Transformação de dados

É feito pelo uso das técnicas de normalização e conversão dos dados, o método de normalização mais utilizado é a de min-max e a por z-score, ambas fazem a padronização dos dados. Já a conversão é feita de duas maneiras, sendo elas: discretização que faz o uso dos algoritmos para fazer a criação de categorias e codificação, neste último caso o algoritmo vai analisar os números determinando um padrão entre eles (da Silva et al., 2017).

2.3 Análise preditiva

A análise preditiva é uma tecnica que faz a identificação de padrões em um conjunto de dados para que eles possam ser agrupados em classes de acordo com suas características. Estudar os atributos dos dados e entender como eles se relacionam é um passo fundamental para a divisão desses dados em grupos baseados em suas características, onde esses grupos serão as classes

trabalhadas. Nesse contexto, uma vez que esse relacionamento é descoberto, chamamos de rotulação de dados o processo de inclui-los nessas classes (da Silva et al., 2017).

Em geral, quando um modelo que faz uso do conhecimento obtido pelo processo de dados classificados é chamado modelo preditivo e quando o processo faz utilização de aprendizagem de máquina dizemos que este aprendizado é supervisionado. Uma vez que o modelo está determinado, podemos utilizá-lo para predizer o rótulo de dados não antes conhecidos, ou seja, novos dados. Chamamos esse passo de generalização do modelo, que é a capacidade de fazer previsões bem sucedidas em dados não antes observados.

2.3.1 Classificação

A seguir apresentamos alguns dos classificadores supervisionados mais populares na literatura.

Rede Neural

Uma rede neural se define por uma função que estrutura as entradas e saidas da Silva et al. (2017), ou segundo Haykin (2001) ele diz que uma rede neural é uma máquina que foi projetada para modelar a forma que o cérebro realiza uma atividade ou função. Para o processo de aprendizagem de uma rede neural é necessário ser aplicado um algoritmo de aprendizagem que tem por principio fazer a alteração dos pesos sinápticos ordenadamente de forma que ele obtenha o resultado esperado.

Support Vector Machine(SVM)

O SVM é um algoritmo que visa adicionar um separador entre os objetos de exemplo que adicione a maior distância possivel entre eles, o que o ajuda na generalização. Segundo Norvig and Russell (2013) os SVMs criam uma separação linear em hiperplano, mas têm a capacidade de incorporar os dados em um espaço de dimensão superior, usando o assim chamado truque de kernel. Muitas vezes, os dados que não são separáveis linearmente no espaço de entrada original são facilmente separáveis em um espaço de dimensão superior. O separador linear de dimensão superior é realmente não linear no espaço original. Isso significa que o espaço de hipótese é expandido em relação aos métodos que usam representações estritamente lineares.

Árvore

Arvore de decisão é composta por um conjunto de nós internos e folhas organizados de forma hierárquica. No contexto da tarefa de classificação, a arvore de decisão representa o modelo capaz de guiar a tomada de decisão sobre determinada classe a qual um exemplar pertence (da Silva et al., 2017). Em uma arvore de decisão, cada nó representa um atributo e dá acesso a uma subárvores de acordo com os valores que o atributo pode assumir. Cada nó folha representa uma decisão de classificação.

KNN

O K-NN ou K-vizinhos mais próximos é um algoritmo para classificação baseado em métricas de distância. A idéia desse classificador é armazenar o conjunto de treinamento e a cada nova entrada compara-la com esta base de dados. Cada novo exemplar é classificado com base na "votação da maioria", onde um numero k é estabelecido previamente e a classe atribuida a esse exemplar é a classe da maioria dos exemplares já classificados próximos a ele (da Silva et al., 2017). A distância entre dois exemplares pode ser calculada utilizando distância euclidiana.

2.4 Avaliação

Existem duas formas de avaliação, a primeira sendo por classificadores e a segunda por modelos preditivos. A Avaliação por classificadores normalmente é feita por meio da acurácia e a por modelos é verificado por via de uma função de perda contínua que visa fazer a medição da resposta do erro entre a resposta que foi recebida e a já se sabia.

Nesta seção os assuntos abordados foram baseados no livro de (da Silva et al., 2017).

2.4.1 Acurácia

A acurácia, ou taxa de classificações corretas, é dada por:

Acurácia = $\quad \text{mod}\,(y - f(x) = 0)$

Onde:

- $\text{mod}\,(z)$ => contagem de vezes que z é verdadeiro
- f => modelo preditivo
- x => subconjunto de dados
- $f(x)$ => classificação feita pelo modelo
- y => classe resposta esperada

2.4.2 Matriz de confusão

A matriz de confusão permite a visualização do desempenho de um algoritmo de classificação. É especialmente útil para analisar o tipo de erro que o modelo pode estar cometendo. Em um classificador binário, onde há apenas duas classes de saída, as classes devem ser definidas como "classe positiva"e "classe negativo".

Onde:

- Verdadeiro positivo (TP): objeto foi classificado como positivo e realmente pertence a classe positiva.

Classe Esperada (y)		Classe Predita (f(x))	
		positivo	negativo
	positivo	Verdadeiros positivos (VP)	Falsos negativos (FN)
	negativo	Falsos Positivos (FP)	Verdadeiros negativos (VN)

Figura 2.1: Fonte: extraída do Livro da Silva et al. (2017)

- Falso positivo (FP): objeto foi classificado como positivo porém pertence a classe negativa.
- Verdadeiro negativo (TN): objeto classificado como negativo e realmente pertence a classe negativa.
- Falso negativo (FN): objeto classificado como negativo porém pertence a classe positiva.

Para correta avaliação de um modelo, uma série de medidas podem ser tiradas da matriz de confusão para o classificador binário. São elas:

- Recall: também chamado de taxa de verdadeiros positivos ou sensibilidade é a porcentagem de verdadeiros positivos sobre todos os exemplos positivos $(VP/(VP+FN))$
- Precision: porcentagem de verdadeiros positivos sobre todos os classificados como positivos $(VP/(VP+FP))$
- F-mesure: relaciona precisão e recall

$$\frac{2}{(\frac{1}{recall})+(\frac{1}{precision})}$$

- ROC: a curva ROC é um recurso que relaciona as taxas de verdadeiros positivos e falsos positivos. A área abaixo dessa curva tem variação entre 0.5 e 1, e mostra a habilidade de um modelo realizar classificação corretas e erradas. s

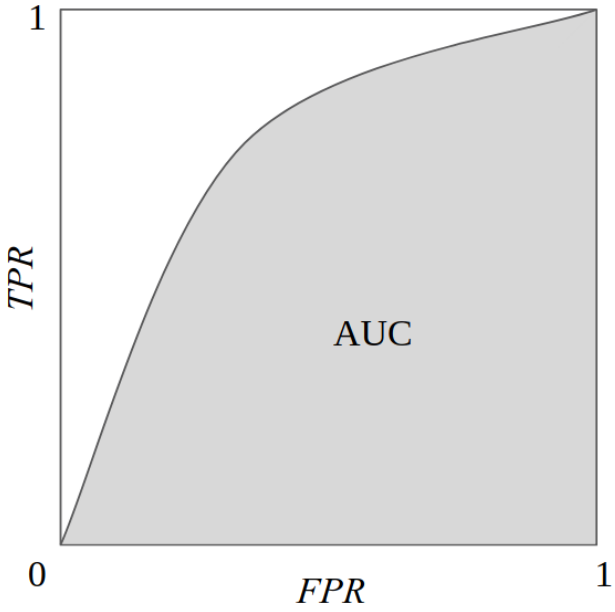


Figura 2.2: Fonte: extraída da internet Paulino

3

Trabalhos Relacionados

Neste Capítulo, inicialmente apresentamos uma discussão a respeito de alguns modelos mais genéricos de busca por ajuda (Seção 3.1) e em seguida abordarem os modelos que visam caracterizar um comportamento de abuso de ajuda, o *gaming the system* (Seção 3.2).

3.1 Modelos de busca por ajuda

A busca por ajuda, por parte dos estudantes, é um comportamento rotineiro nas salas de aulas presenciais e a distância. Ela acontece depois que um aluno percebe que precisa de ajuda e é uma decisão proativa que permite que os alunos continuem progredindo no processo de resolução de problemas Arroyo et al. (2014). Sendo considerada também uma habilidade metacognitiva, a busca por ajuda tem sido estudada em contextos sociais como salas de aula Karabenick (2012). Nesse contexto, é considerada um importante atributo de autorregulação Newman (1998); Butler and Winne (1995) que, em vez de sinalizar a dependência de um aluno, pode ser instrumental no desenvolvimento de outras habilidades como, por exemplo a auto regulação Newman (2002).

Quando o estudante solicita um feedback ou ajuda do professor, tutor ou colega, ele pode fazer um bom uso refletindo na conclusão com êxito na finalização da atividade e positivamente processo de aprendizagem Bartholomé et al. (2006) ou um mal uso (improdutivo) visando apenas conseguir a resposta. Nesse último, há evidências de que o uso improdutivo de um feedback se correlaciona negativamente com a aprendizagem Aleven et al. (2006).

Tal situação não é diferente quando o estudante está na atividade de resolução de problemas interagindo com um ambiente virtual de aprendizagem. Uma vez que, algumas observações são preocupantes como, por exemplo, altas frequências, de 73% e 64%, referentes a mau uso da ajuda de um sistema de tutoria Aleven et al. (2006). Além disso, há evidências de que os alunos tendem a não usar efetivamente as facilidades de ajuda oferecidas pelos ambientes virtuais

de aprendizagem Bartholomé et al. (2004). Os estudantes frequentemente usam os feedbacks providos pelo sistema para obter respostas, sem tentar entender como as respostas são derivadas ou as razões por trás das respostas Aleven and Koedinger (2000). Diante deste cenário, várias iniciativas têm sido desenvolvidas para modelar o comportamento do estudante na busca por ajuda e feedback.

Na pesquisa de Aleven et al. (2004) há um modelo de comportamento de busca de ajuda visando identificar comportamento produtivo e improdutivo de busca. Esse modelo foi projetado para funcionar em um tutor no domínio de gemetria. Este oferece ao estudante dois tipos diferentes de ajuda: dicas contextuais em vários níveis de detalhes e outra na forma de um Glossário descontextualizado. O modelo foi implementado com 57 regras de produção. Trinta e duas das regras são "regras de bugs" que refletem desvios do comportamento ideal de busca de ajuda e permitem que a busca de ajuda tutor para fornecer feedback aos estudantes sobre tais desvios. Duas informações-chave são avaliadas pelo modelo: (1) se o aluno levou tempo suficiente para considerar sua ação, (2) se o aluno usou apropriadamente, ou não, facilidades de ajuda no momento determinado no processo de resolução de problemas. Além do modelo, os autores criaram uma taxonomia de erros que inclui quatro categorias principais: abuso de ajuda, abuso de teste, evitar ajuda e erros diversos.

Os autores validaram o modelo usando um conjunto de dados existentes. Algumas das principais observações dos autores foram: i) os alunos frequentemente evitavam usar ajuda quando era provável que fosse benéfico e muitas vezes agiam de maneira rápida e possivelmente não deliberada; ii) O comportamento de busca de ajuda dos alunos foi responsável por tanta variação em seus ganhos de aprendizagem quanto em seu desempenho no nível cognitivo indicando que o modelo precisa de ajustes.

Depois que obtiveram esses resultados Aleven et al. (2006), os autores continuam o processo de aprimoramento do modelo. Apesar de continuar com a mesma quantidade de regras de produção, 57, de acordo com os autores, houve mudança nas decisões do fluxograma. Por exemplo, no fluxograma do modelo, ao decidir qual ação tomar em seguida: tentar a etapa, solicitar uma dica ou inspecionar um item do Glossário, três informações principais são consideradas: (1) O tempo necessário para a ação deliberada; (2) A(s) habilidade(s) envolvida(s) na etapa e a probabilidade de que o aluno a domine; (3) O que o aluno já fez em relação a esta etapa (por exemplo, o número de tentativas anteriores – e sem sucesso – e o número de dicas).

3.2 Modelos para detecção de *gaming the system*

Focando no comportamento indesejado na busca por ajuda ou feedback, alguns autores têm criado modelos para detectar automaticamente um comportamento chamado *gaming the system*.

Baker et al. (2004a) apresentam e discutem um Modelo de Resposta Latente (LRM) de aprendizagem de máquina para discernir quais alunos frequentemente jogam o sistema de uma

maneira que está correlacionada com baixo aprendizado. Este modelo corrobora a hipótese de Baker et al 2004 de que os alunos que jogam o sistema são mais propensos a fazê-lo nas etapas mais difíceis. Combinam três fontes de dados sobre o desempenho e o comportamento do aluno em uma aula de tutor cognitivo ensinando sobre a geração do gráfico de dispersão. Todos os dados foram extraídos de um grupo de 70 alunos usando aquela aula de tutor cognitivo como parte de seus currículos normais de matemática. A primeira fonte de dados foi um registro de todas as ações que cada aluno executou enquanto usava o tutor. Cada aluno realizou entre 71 e 478 ações dentro do tutor e foram armazenadas 24 variáveis dos arquivos de log.

Como o foco do trabalho é verificar se o estudante joga mais nos momentos difíceis do problema, as variáveis são focadas em caracterizar tais informações. Apresentaremos as variáveis consideradas no Quadro 5.1. A segunda fonte de dados foi o conjunto de observações codificadas por humanos do comportamento do aluno durante a aula obtendo o a proporção aproximada de tempo que cada aluno gastou jogando no sistema. E a terceira fonte de dados foram os resultados da aprendizagem dos alunos. Os estudantes foram divididos em três grupos: Estudantes que nunca apresentaram comportamento *gaming the system*, estudantes que apresentaram o comportamento de jogo, mas tiveram um desempenho ruim, chamados "GAMED-HURT"e os estudantes que apresentaram o comportamento *gaming the system* e tiveram um bom desempenho, chamados "GAMED NOT-HURT".

No artigo de Baker et al. (2005) estudaram os objetivos, atitudes, comportamento e aprendizado 103 estudantes do ensino médio em um tutor de matemática. Estudaram esses estudantes duas aulas de tutor cognitivo sobre geração e interpretação de gráficos de dispersão. Para o estudo, foram combinadas duas fontes de dados: um questionário sobre as motivações e crenças dos alunos, registros das ações de cada estudante dentro do tutor (analisados tanto na forma bruta quanto por meio de um detector de jogos), e dados de teste/pós-teste. Além disso, as instâncias de *gaming the system* foram divididas em dois casos: *gaming* sem impacto no ganho pré-teste/pós-teste e *gaming* com impacto negativo no ganho pré-teste/pós-teste. Os autores descobriram que a frequência de jogar o sistema não se correlaciona com uma medida conhecida de metas de desempenho; em vez disso, jogar está relacionado a não gostar de computadores e do tutor.

Após a realização de um experimento para identificar comportamentos fora da tarefa por estudantes, em Walonoski and Heffernan (2006), os autores desenvolveram um modelo para detectar um comportamento indesejado na busca por feedback, o *gaming the system*. Durante o experimento, foram observados seis comportamentos fora da tarefa: jogando, na tarefa (conversando), na tarefa (papel ou professor), na tarefa (tutor), fora da tarefa (falando) e fora da tarefa (inativo) Walonoski and Heffernan (2006). O modelo foi elaborado utilizando observações coletadas e os logs das ações dos estudantes usando um STI no domínio da matemática. Para isso, vários conjuntos de dados foram gerados por meio de filtragem de ação não supervisionada usando janelas de tempo. Os conjuntos de dados tinham 1430 atributos e 1 valor de classificação (jogos, verdadeiro ou falso).

O trabalho de Paquette et al. (2014) foi feito utilizando o cognitive tutor de álgebra, fazendo a construção do modelo cognitivo com base nos dados do sistema e utilizando o conhecimento de um especialista. Os dados continham informações sobre as ações dos alunos durante a utilização do sistema como: o tempo, dicas, contexto do problema, entrada que o usuário inseriu, avaliação do sistema e com isso eles conseguem identificar se existe o *gaming the system* ou não. Nesse sistema os tutores fornecem dicas, as mesmas podendo ser dadas a qualquer momento que o aluno necessitar de ajuda, quanto mais ele pede mais específica ficam as dicas. O modelo cognitivo fez a separação dos dados em duas partes, a primeira identifica a ação do aluno e a segunda identifica os padrões, eles utilizaram o Kappa de Cohen para medir a confiabilidade entre avaliadores.

Richey et al. (2021) utiliza o sistema Decimal Point, sistema que é focado em matemática para os alunos de 5 e 6 série que tem como objetivo auxiliar no processo de desenvolvimento dos alunos utilizando alguns mecanismos de jogos. Foram desenvolvidos detectores com base nos logs utilizando replays de texto, o jogo tem personagens que causam uma fantasia ao usuário que está fazendo uso deste sistema. Durante a realização deste experimento os alunos (174 estudantes) foram submetidos a realização de pré teste, pós teste e pós teste atrasado, em cada teste tinham 24 questões. A codificação foi dividida em três fases e após terem uma confiabilidade aceita, os dados eram adicionados em algoritmos de aprendizagem de máquina para entender os codificadores, foi utilizado o Kappa para medir a confiabilidade.

No trabalho anterior de Paquette et al. (2014) foram identificados 13 padrões, neste estudo atual (Paquette and Baker, 2017) foi feito a construção de um modelo baseado nesses padrões que visa detectar o uso do estudante na plataforma, conseguindo detectar a ordem em que elas são realizadas. Nesta pesquisa foi feita a distinção do comportamento de jogo em três ferramentas (Cognitive Tutor Álgebra, Tutor Middle School, ASSISTments). O modelo teve um desempenho no kappa de 0,3, que demonstra a melhoria da identificação dos comportamentos de jogos, no momento em que ele é aplicado em um conjunto de testes.

4

Metodologia

Nesta seção vão ser descritos os procedimentos que foram abordados durante a aplicação dos algoritmos de aprendizagem de máquina e o detalhamento das informações sobre o desenvolvimento do sistema de aprendizagem, tal como a Figura 4.1.

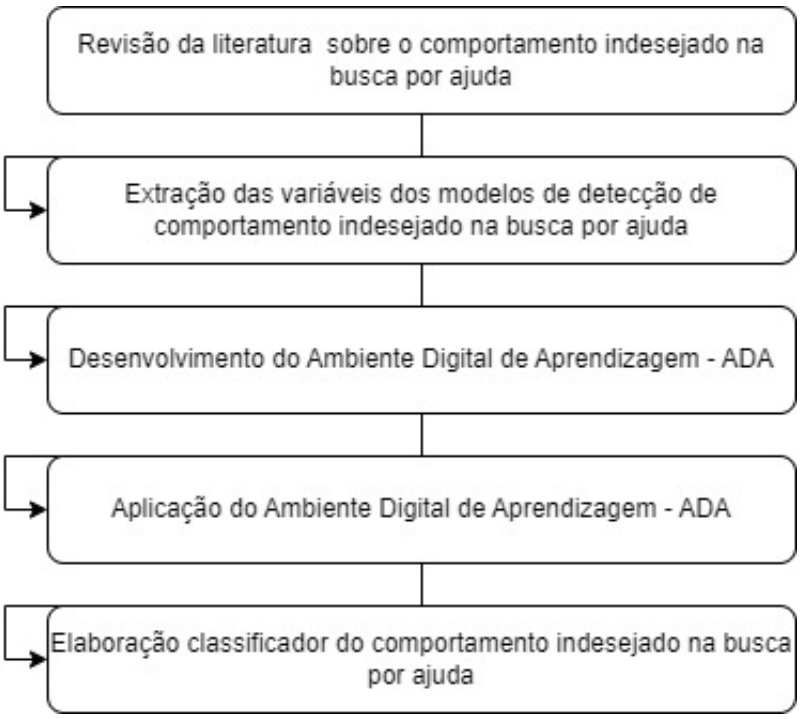


Figura 4.1: Metodologia

4.1 Revisão da Literatura

Realizamos uma revisão da literatura para identificar e analisar os métodos para detecção de comportamento indejado do estudante na busca por feedaback, verificando aqueles que apre-

sentam os melhores desempenhos. Com isso, verificamos o que já foi estudado sobre situação de comportamento do estudante de usar o feedback visando obter a resposta correta. Neste sentido, os trabalhos analisados estão com seus resultados mostrados em uma tabela comparativa descrita na tabela 5.1.

4.2 Ambiente para Coleta de Dados: Desenvolvimento do ADA

O sistema foi desenvolvido para ser um site de resolução de problemas de programação de multipla escolha, que internamente captura as váriaveis utilizadas na literatura para estudo do *gaming the system*. O sistema suporta problemas de computação cadastrados previamente em conjunto com as dicas, alternativas relacionadas e indicação de sua dificuldade. Os problemas foram classificados em sessões.

Os problemas de uma mesma sessão serão enviados em seguida uma vez que essa sessão seja selecionada para aplicação. O questionário é dado como encerrado quando todos os problemas forem respondidos corretamente, se um problema for respondido de forma incorreta volta a aparecer novamente no final da fila de questões.

4.3 Aplicação do sistema

Para realizar uma coleta de dados, o sistema ADA foi aplicado presencialmente com os alunos da turma 2022.1 de inteligência artifical do ensino superior na turma sob responsabilidade do professor Evandro de Barros Costa, docente do instituto de computação na Universidade Federal de Alagoas. 14 alunos participaram do experimento que durou aproximadamente 45 minutos.

4.4 Elaboração dos modelos de classificação

Para elaborar os modelos classificadores foram estudados os principais algoritmos de aprendizagem supervisionada e que são utilizados para este tipo de detecção na literatura. Os selecionados foram Árvore de decisão, K-NN, SVM e a Rede reural MLP. Também foram levantados os principais métodos de avaliação de modelos que são amplamente utilizados para aprendizagem de máquina. Todas as bibliotecas foram utilizadas em Python, sendo os algoritmos classificadores e métricas do pacote Scikit learn, o pacote imblearn foi utilizado para balanceamento dos dados. Além disso, utilizamos a biblioteca pandas para manipulação da base de dados.

5

Descoberta de Conhecimento

Neste capítulo apresentamos os estudos desenvolvidos no sentido de caracterizar o comportamento indesejado *gaming the system*. Para isso, discutimos a metodologia (Seção 5.1) utilizada para mapear as variáveis por detectores do comportamento *gaming the system*, relatados na literatura. Na Seção 5.1.1, apresentamos os padrões resultantes do mapeamento dos estudos selecionados. Na seção 5.1.2 discutimos os resultados de um estudo de revisão da literatura visando mapear as principais variáveis utilizadas por modelos detectores do comportamento de abuso de ajuda: modelos de busca por ajuda e detectores *gaming the system*.

5.1 Metodologia

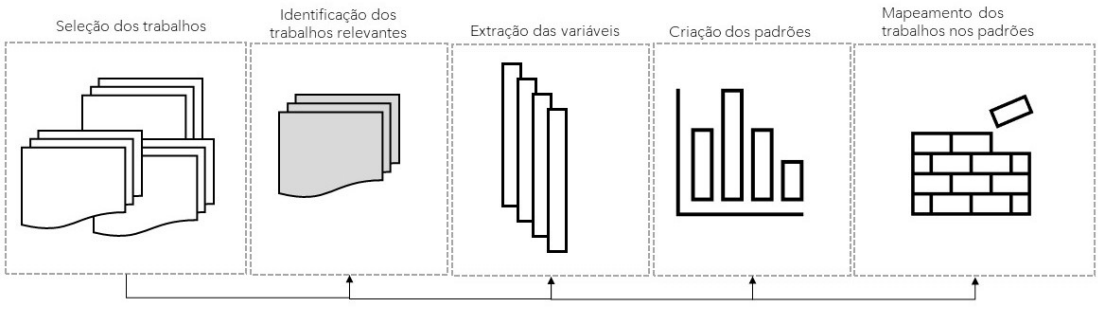


Figura 5.1: Descoberta de conhecimento

Nesta seção apresentaremos os resultados da análise da literatura sobre as principais variáveis utilizadas pelos modelos de detecção do comportamento indesejado *gaming the system*. Para isso, de acordo com a Figura 5.1 elaboramos este estudo nas seguintes etapas:

i) **Seleção dos trabalhos**: realizamos uma busca nas as seguintes fontes: IEEE[1], ACM Digital Library[2], Springer[3] and ScienceDirect[4].

ii) **Identificação dos trabalhos relevantes**: identificamos as pesquisas que relatam a modelagem para comportamento de abuso de ajuda;

iii) **Extração das variáveis**: em todos os trabalhos com modelos, foram extraídas as variáveis utilizadas;

iv) **Criação dos padrões**: analisamos as variáveis extraídas e percebemos padrões associados e então, elaboramos oito padrões apresentados na Seção 5.1.1;

v) **Mapeamento dos trabalhos nos padrões**: após identificação dos padrões, mapeamos as variáveis, das pesquisas encontradas, nos padrões elaborados. Apresentamos este mapeamento na Seção 5.1.2.

5.1.1 Padrões de variáveis

Após a busca, seleção e análise dos trabalhos, identificamos oito padrões de características:

- **P1-Respostas corretas:** variáveis destinadas ao armazenamento de informações relacionadas aos acertos dos estudante durante a realização da atividade. Por exemplo, medidas estatísticas e valores de tempo;
- **P2-Respostas incorretas:** variáveis que armazenam qualquer tipo de erro cometido pelo estudante durante a realização da atividade. Por exemplo, medidas estatísticas e valores de tempo;
- **P3-Bugs:** variáveis que armazenam qualquer tipo de bug do sistema com relação ao problema, podendo ser um bug conhecido ou desconhecido;
- **P4-Dicas:** variáveis que guardam informações sobre as dicas solicitadas pelo estudante ao sistema. Por exemplo, momento da solicitação da dica;
- **P5-Conhecimento prévio:** variáveis que guardam as informações relacionadas ao conhecimento prévio do estudante sobre o domínio de conhecimento em questão;
- **P6-Problema ou questão:** informações sobre o problema ou a questão que o estudante está resolvendo em um determinado momento no sistema. Por exemplo, número de tentativa em um mesmo problema;
- **P7-Proporções e combinações:** são variáveis que armazenam informações sobre a relação de duas ou mais combinações de outras variáveis. Por exemplo, número de vezes que o estudante demandou ajuda em uma habilidade do domínio de conhecimento;
- **P8-Uso do sistema:** são variáveis que armazenam informações relacionadas às funcionalidades onde o detector está implantado. Por exemplo, tipo de widget da interface envolvida na ação.

5.1.2 Trabalhos mapeados

Na Tabela 5.1, apresentamos a comparação de alguns trabalhos sobre modelagem de pedido de ajuda ou detectores *gaming the system* dos estudantes, encontrados na literatura. Os dois primeiros trabalhos são modelos para pedidos de ajuda e os últimos são detectores de abuso de ajuda chamados de comportamento *gaming the system*.

Os trabalhos de Aleven et al. (2004) e Aleven et al. (2006) apresentam um modelo para caracterização de comportamento de busca de ajuda do estudante, visando identificar comportamento produtivo e improdutivo de busca. O modelo foi implementado com 57 regras de produção. Trinta e duas das regras são "regras de bugs" que refletem desvios do comportamento ideal de busca de ajuda e permitem a busca de ajuda do tutor para fornecer feedback aos estudantes sobre tais desvios. Para isso, os dois trabalhos mencionados consideram as mesmas variáveis para todos os padrões definidos na Seção 5.1.1. São eles:

- **P1-Respostas corretas:** ações corretas de cada estudante;
- **P2-Respostas incorretas:** ações erradas de cada estudante;
- **P4-Dicas:** momento de solicitação da dica;
- **P5-Conhecimento prévio:** estimativa de conhecimento para uma habilidade;
- **P6-Problema ou questão:** número de dicas disponíveis para cada problema; Habilidade envolvida no problema.

Os terceiro, quarto e sexto trabalhos da Tabela 5.1, respectivamente Baker et al. (2004a), Baker et al. (2005) e d Baker et al. (2008), utilizam as mesmas representações de dados: dados de log do uso de um sistema tutor por estudantes e dados de observação de codificadores. Utilizando esses dados, os autores classificam os estudantes em três grupos de comportamento: estudantes sem manifestação do comportamento *gaming the system*, estudantes com comportamento *gaming the system* que sofreram prejuízo na aprendizagem e estudantes com comportamento *gaming the system* que não sofreram prejuízo na aprendizagem. Eles consideram as seguintes variáveis associadas aos padrões identificados na Seção 5.1.1:

- **P1-Respostas corretas:** ações corretas e as primeira tentativas;
- **P2-Respostas incorretas:** ações incorretas e as primeira tentativas;
- **P3-Bugs:** bug conhecido e bug desconhecido;
- **P4-Dicas:** número de vezes que pediu ajuda;
- **P5-Conhecimento prévio:** probabilidade de o aluno saber a habilidade (algoritmo bayesiano de rastreamento de conhecimento Corbett and Anderson (1994)); Segundos gastos em cada oportunidade para praticar cada habilidade entre os problemas;

- **P6-Problema ou questão:** número real de segundos e os desvios padrão do tempo médio gasto por todos os alunos nesta etapa do problema, entre os problemas;
- **P7-Proporções e combinações:** ações incorretas, indicando um bug conhecido; incorreta com um bug desconhecido; incorreta com um pedido de ajuda; segundos foram gastos nas últimas 3 ações, ou 5 ações; Número de vezes que pediu ajuda em uma habilidade; número de vezes que errou uma resposta em uma habilidade; quantas vezes o aluno pediu ajuda nas últimas 8 ações; quantos erros o aluno cometeu nas últimas 5 ações;
- **P8-Uso do sistema:** tipo de widget de interface envolvido na ação: o aluno estava escolhendo em um menu suspenso, digitando uma string, digitando um número, traçando um ponto ou selecionando uma caixa de seleção.

O quinto trabalho da tabela, o trabalho de Walonoski and Heffernan (2006), assim como nos trabalhos discutidos anteriormente, os autores desenvolveram um detector de comportamento *gaming the system* baseado em dados oriundos de um experimento. Com isso, realizaram registro de observações em sala de aula dos alunos usando o software de tutoria e criação de conjunto de dados com base nessas observações. Os conjuntos de dados tinham 1430 atributos e 1 valor de classificação de comportamento *gaming the system* ou não. Para cada padrão que identificamos, os autores relataram as seguintes variáveis:

- **P1-Respostas corretas:** medidas estatísticas o tempo em milissegundos; velocidade de cada solicitação e primeiras tentativas;
- **P2-Respostas incorretas:** medidas estatísticas o tempo em milissegundos; velocidade de cada solicitação e primeiras tentativas;
- **P4-Dicas:** medidas estatísticas o tempo em milissegundos; velocidade de cada solicitação e solicitação de dicas de baixo pra cima;
- **P5-Conhecimento prévio:** probabilidade ter uma determinada habilidade e o nível naquela habilidade;
- **P6-Problema ou questão:** número total de perguntas de problema de nível superior; número total de perguntas de ajuda de acompanhamento; número total de perguntas que apresentavam uma interface de usuário de múltipla escolha; número total de perguntas que apresentavam uma interface de usuário de caixa de texto; porcentagem correta na primeira tentativa de todos os alunos; medidas estatísticas das dificuldades do problema e o número total de vezes que um problema foi repetido;
- **P7-Proporções e combinações:** número de tentativas por problema; número de tentativas corretas por problema; número de tentativas incorretas por problema; número de dicas por problema; número de dicas de fundo por problema; número de repetições por problema

de nível superior; e 1378 atributos separados representando os efeitos quadráticos entre quaisquer dois dos atributos.

O sétimo trabalho da tabela de Paquette et al. (2014), apresenta um modelo de detector baseado em como especialistas codificam o *gaming the system*, este modelo avalia as respostas do aluno à medida que ele realiza a tarefa. Neste trabalho foi utilizado um sistema tutor que continham atividades matemáticas de álgebra e o experimento foi aplicado em 59 alunos durante 1 ano escolar desses estudantes.

- **P1-Respostas corretas:** ações corretas de cada estudante;
- **P2-Respostas incorretas:** ações erradas de cada estudante;
- **P3-Bugs:** bug conhecido pelo sistema;
- **P4-Dicas:** momento em que a ajuda foi pedida;
- **P6-Problema ou questão:** número em segundos que levou para uma ação ser tomada ou entre ações.

No trabalho de Richey et al. (2021) foram implementados detectores baseados em logs utilizando codificadores de texto, durante a participação do estudo tinham 213 alunos da 5 e 6 série, eles fizeram o processo de pré teste (ocorre antes de utilizar o sistema), pós teste (após a finalização do sistema) e pós teste atrasado que consistia em fazer o experimento 1 semana após a finalização. Foi utilizado o sistema Decimal Point que constitui-se de problemas matemáticos. Esse trabalho também inclui alguns elementos de jogos como feedback e personagens que trazem um contexto de narrativa a ferramenta.

- **P1-Respostas corretas:** o usuário precisa acertar a questão para prosseguir para as próximas perguntas;
- **P4-Dicas:** ao errar uma questão o aluno é auxiliado com dicas até chegar em uma resposta correta;
- **P5-Conhecimento prévio:** o aluno precisa estar aprendendo sobre números decimais;
- **P6-Problema ou questão:** envolve problemas de organização, adição, sequência e classificação de números decimais.

Paquette and Baker (2017) utiliza o modelo cognitivo em cima de seu trabalho anterior em Paquette et al. (2014) no qual ele utilizou o sistema matemático que utilizava lógica juntamente com outras 2 ferramentas de aprendizagem, fazendo essa análise ele coletou 13 padrões nestes três sistemas que são denominados como jogos ou representações de jogos, esses padrões determinam as ações das pessoas que realizam uma atividade em um sistema. No geral, este

trabalho faz uma comparação dessas ferramentas levando em consideração o desenvolvimento dos alunos.

- **P2-Respostas incorretas:** padrão de entrada incorreta que os alunos inserem rapidamente mais de uma vez em diferentes partes do mesmo problema;
- **P3-Bugs:** o aluno dá entrada no mesmo bug mais de uma vez;
- **P4-Dicas:** aluno pede uma dica e rapidamente entra com uma tentativa de resposta.

Tabela 5.1: Padrões por detector

ID	Trabalho	P1	P2	P3	P4	P5	P6	P7	P8
1	Aleven et al. (2004)	X	X		X	X	X		
2	Aleven et al. (2006)	X	X			X	X		
3	Baker et al. (2004a)	X	X	X	X	X	X	X	X
4	Baker et al. (2005)	X	X	X	X	X	X	X	X
5	Walonoski and Heffernan (2006)	X	X		X	X	X	X	
6	d Baker et al. (2008)	X	X	X	X	X	X	X	X
7	Paquette et al. (2014)	X	X	X	X		X		
8	Richey et al. (2021)	X			X	X	X		
9	Paquette and Baker (2017)		X	X	X				

Fonte: Elaborada pelo autor (2022)

6

Modelos de classificação

6.1 Método

6.1.1 Ambiente Digital de Aprendizagem

O ambiente de aprendizagem online ADA foi desenvolvido visando capturar dados do estudante iniciante em programação durante a atividade de resolução de problemas

O ADA é um software desenvolvido com a linguagem de programação Python[1] no backend e no frontend ReactJS[2] que é uma biblioteca do JavaScript que faz a construção de interfaces de usuário[3]. Para o armazenamento dos dados foi utilizado o PostgreSQL[4] que é um sistema de gerenciamento de banco de dados relacional baseado no postgres.

As principais funcionalidades do ADA são: cadastro, resolução de problemas, solicitação de dica, cadastro de nível de dificuldade do problema e cadastro de estado afetivo.

Cadastro

Nas Figuras 6.1 e 6.2, apresentamos as telas do ADA nas quais os estudantes devem fornecer suas informações relacionadas aos dados socioeconômicos.

Resolução de problema

Na Figura 6.3 há um exemplo de tela exibida aos estudantes durante a resolução de um problema. Cada problema foi elaborado seguindo a estrutura apresentada na Figura 6.3. No entanto, todos os problemas são de múltipla escolha e possuem quatro níveis de dicas.

[1] https://www.python.org/about/
[2] https://pt-br.reactjs.org/
[3] https://pt-br.reactjs.org/docs/getting-started.html
[4] https://www.postgresql.org/docs/current/intro-whatis.html

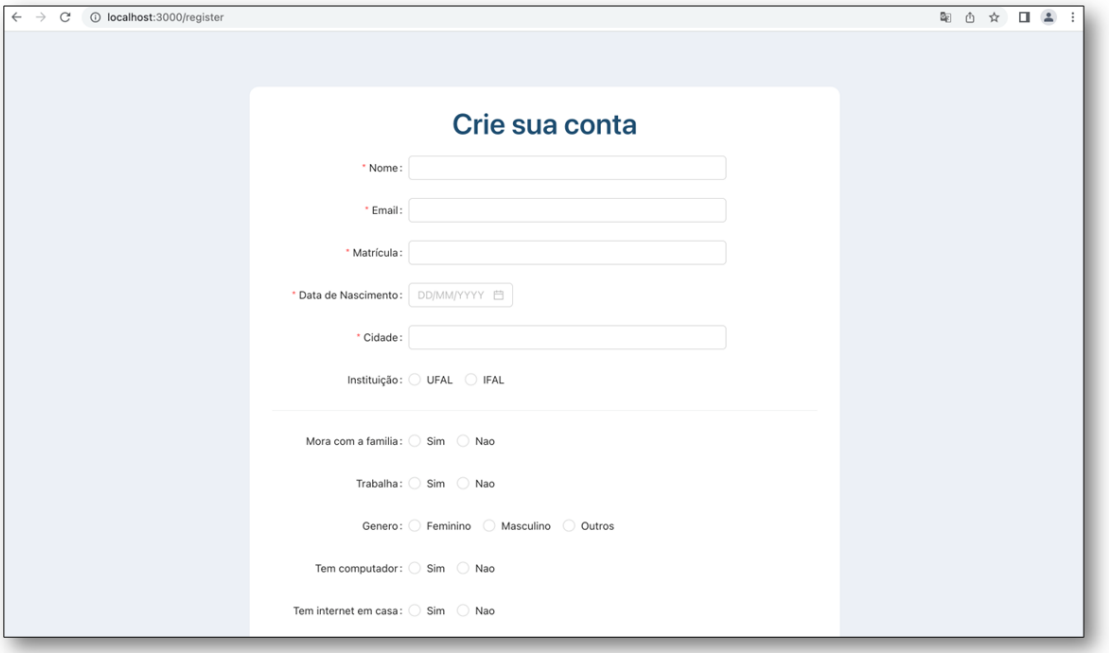


Figura 6.1: Criação de conta - Questionário socioeconômico 1

Como podemos observar a indicação do número 2 na Figura 6.3 há o espaço de exibição da dica que pode ser solicitada pelo estudante nos botões da indicação do número 3 na Figura 6.3. Além de solicitar dica, o estudante pode declarar seu estado afetivo a qualquer momento durante a resolução do problema, conforme a indicação do número 1 também na Figura 6.3.

Avaliação do nível do problema

Após o estudante submeter a solução co problema, o sistema solicita ao estudante que avalie o nível de dificuldade do problema indicando entre os níveis: muito fácil, fácil, médio, difícil e muito difícil, conforme a Figura 6.4.

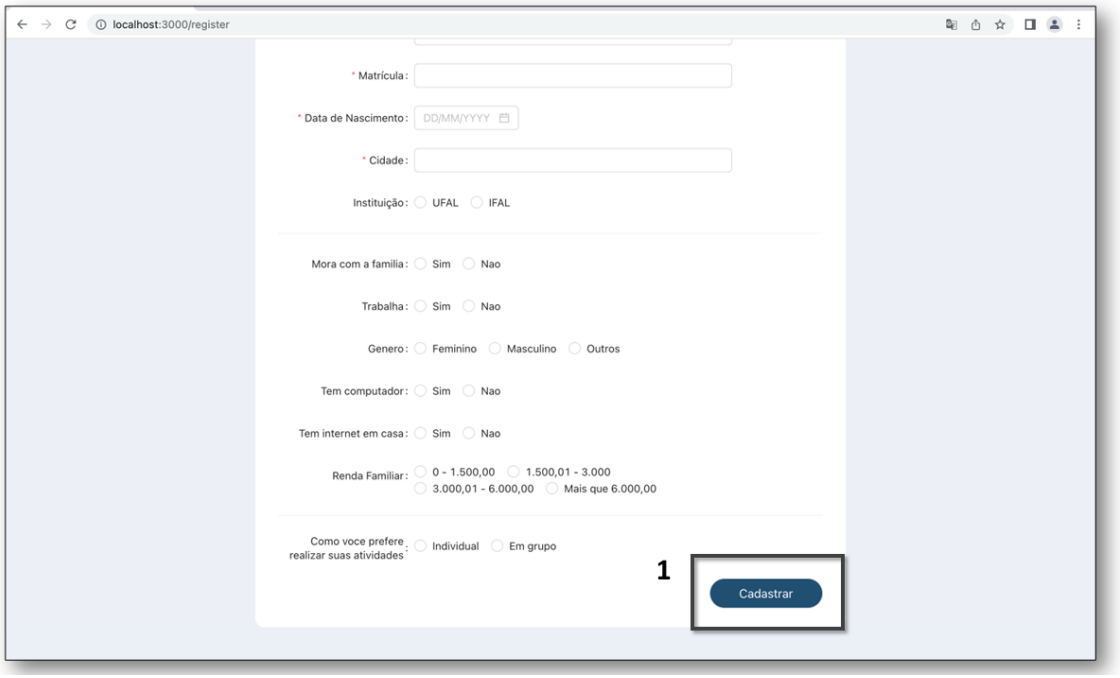


Figura 6.2: Criação de conta - Questionário socioeconômico 2

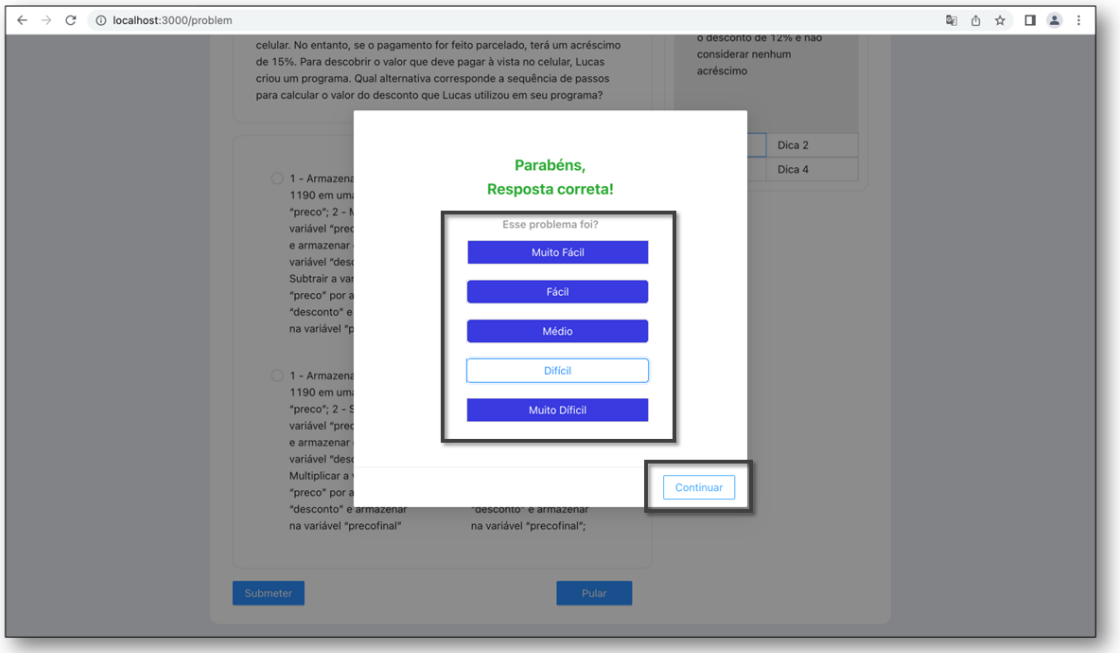


Figura 6.4: Avaliação do nível de dificuldade do problema

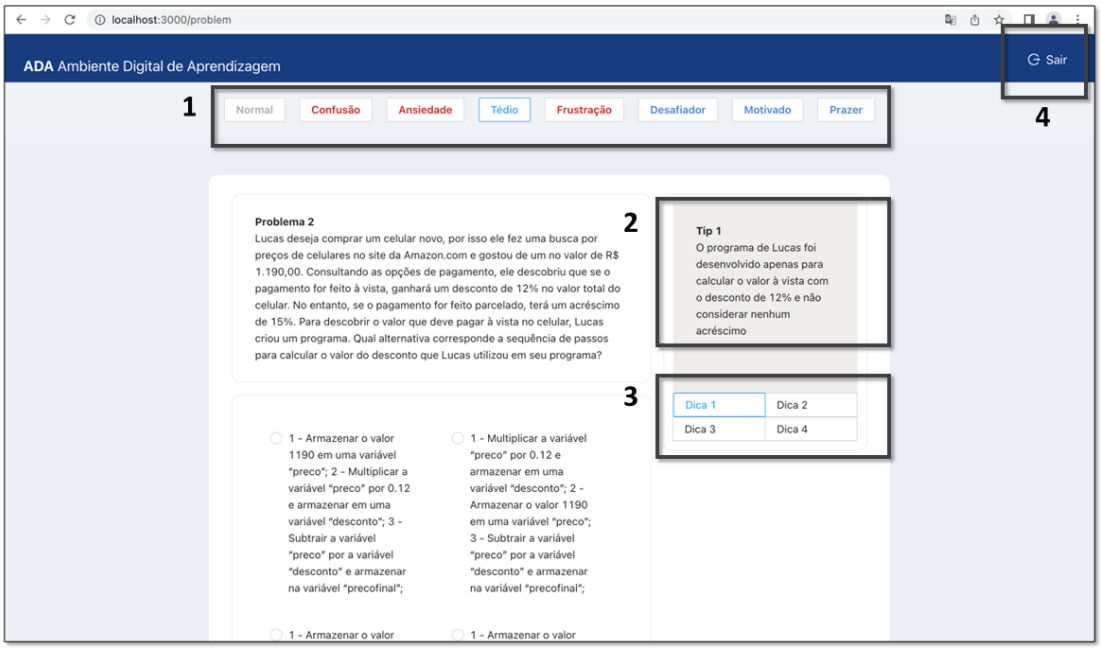


Figura 6.3: Resolução de problema

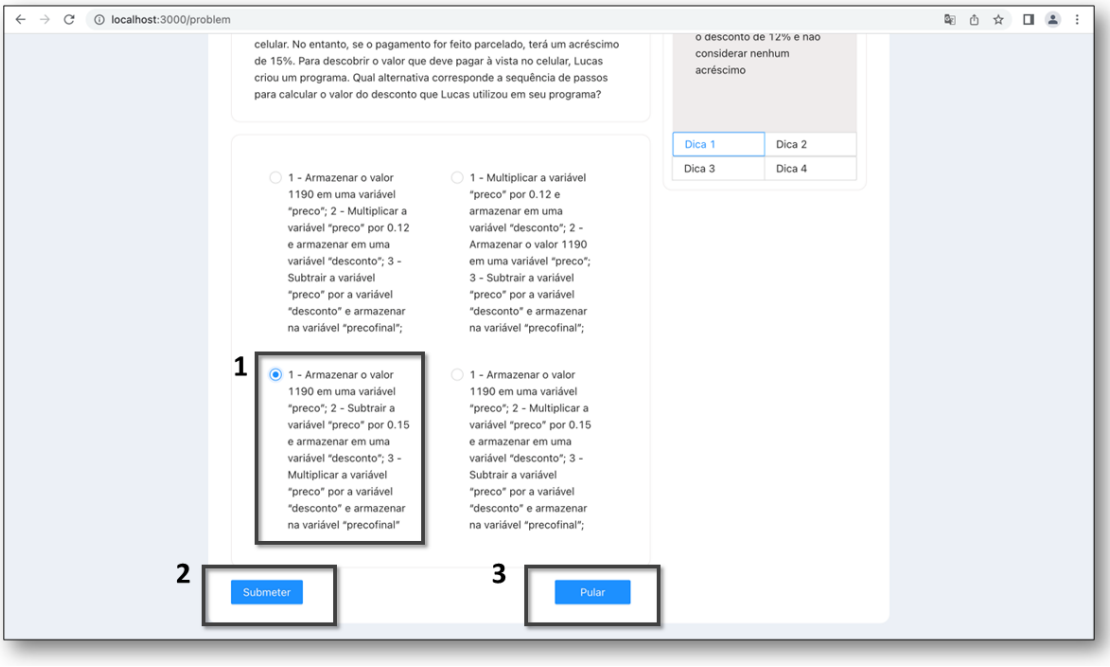


Figura 6.5: Pular o problema

Conclusão da resolução

Após concluir a resolução de todos os problemas, o estudante é direcionado a uma página de encerramento e o sistema exibe um link para que o estudante envie as soluções desenvolvidas para o problemas, conforme a tela apresentada na Figura 6.6.

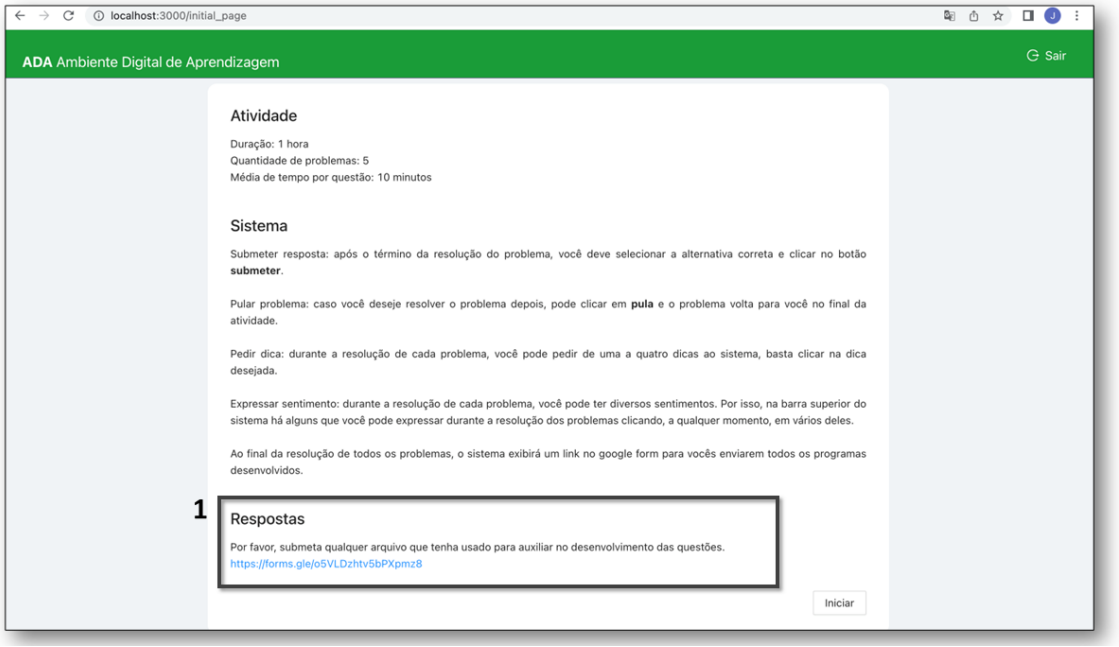


Figura 6.6: Encerramento da Resolução

6.1.2 Pré-processamentos da base de dados

Após a aplicação do experimento, obtenção e rotulação dos dados, iniciamos a fase de pré-processamento. Alguns problemas inerentes à natureza do experimento eram esperados, como dados faltosos, visto que os alunos podem não utilizar todos os recursos de interface do sistema. Uma vez que aplicamos o sistema em apenas uma turma, também tivemos uma quantidade de dados reduzida. Dessa forma, de modo a tentar alcançar uma maior qualidade final dos dados, fizemos utilização das técnicas descritas a seguir.

Primeiramente, fizemos a remoção dos atributos que carregam informações relativas ao sistema, como id do aluno, id do problema, hora do início de prova e etc. Em seguida, para selecionar dados numericos no mesmo formato, foi feito a conversão de formato do tipo hora, minutos e seguntos (hh:mm:ss) para armazenar apenas os segundos (ss) do dado que guardava o momento onde uma dica foi pedida. O mesmo ocorreu com o momento de submissão de uma resposta.

Em seguida, utilizando a biblioteca pandas do Python, foi realizado a conversão dos campos referentes a estados afetivos e dificuldade do problema de dado categórico para numérico. Para tratar a quantidade de dados foi utilizado o uso da biblioteca imblearn[5], que consegue fazer uma simulação dos dados em menor quantidade utilizando vizinhos próximos para esse fim, evitando que o modelo fique enviesado sobre uma classe.

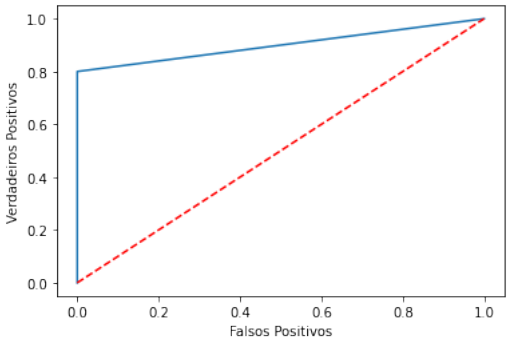
[5]https://imbalanced-learn.org/stable/

6.2 Resultados

Nesta sessão está sendo apresentado o desempenho dos algoritmos com a base de dados citada na seção anterior após os tratamentos realizados. Durante o treinamento dos dados foram aplicados os conjuntos de treinamento, sendo (70%) para o ajuste dos parâmetros de cada algoritmo, utilizando validação cruzada e os (30%) restantes foram utilizados para a avaliação dos modelos gerados. Os resultados obtidos estão apresentados a seguir.

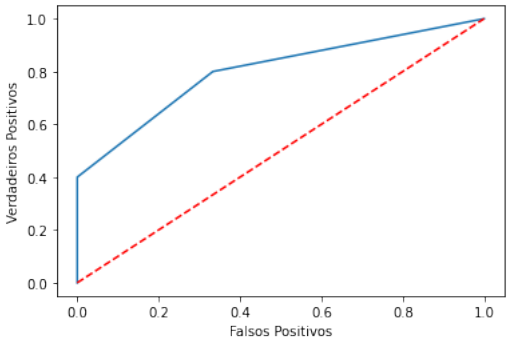
	Acurácia	Precision	Recall	F-Score
Árvore de decisão	0.84	0.88	0.85	0.84
K-NN	0.78	0.79	0.79	0.79
SVM	0.68	0.81	0.67	0.63
Rede Neural	0.84	0.85	0.84	0.84

Tabela 6.1: Resultados obtidos a partir do conjunto de teste



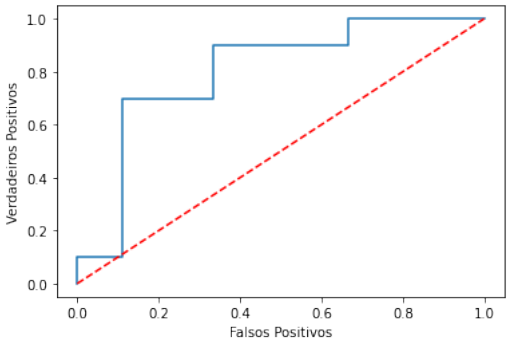
	Verdadeiro	Falso
Verdadeiro	9	0
Falso	3	7

Figura 6.7: Curva ROC e Matriz de confusão da Árvore de decisão



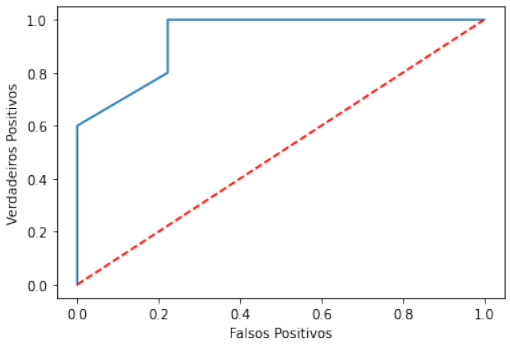
	Verdadeiro	Falso
Verdadeiro	7	2
Falso	2	8

Figura 6.8: Curva ROC e Matriz de confusão do KNN



	Verdadeiro	Falso
Verdadeiro	3	6
Falso	0	10

Figura 6.9: Curva ROC e Matriz de confusão do SVM



	Verdadeiro	Falso
Verdadeiro	7	2
Falso	1	9

Figura 6.10: Curva ROC e Matriz de confusão da Rede Neural

7

Conclusão

O propósito deste trabalho foi estudar e desenvolver uma solução computacional utilizando técnicas de aprendizagem de máquina supervisionada para tarefa de classificação, que automaticamente detecta quando estudantes fazem uso do comportamento *gaming the system*, durante o processo de solução de problemas de programação.

Os resultados obtidos indicam que é possível criar modelos de aprendizagem de máquina supervisionada para detectar automaticamente a presença de comportamento *gaming the system*, em contextos de resolução de problema. Neste sentido, entre as contribuições desse trabalho, estão: (1) desenvolvimento de um ambiente para coleta de dados; (2) a criação e anotação de uma base de dados com os dados; (3) geração dos modelos de aprendizagem de máquina para classificação da presença de comportamento *gaming the system*.

Este trabalho possui algumas limitações, já permitindo apontar algumas pesquisas futuras, de curto prazo, por exemplo analisar a ferramenta em uma turma em curso.

Referências bibliográficas

Vincent Aleven and Kenneth R Koedinger. Limitations of student control: Do students know when they need help? In *International conference on intelligent tutoring systems*, pages 292–303. Springer, 2000.

Vincent Aleven, Bruce McLaren, Ido Roll, and Kenneth Koedinger. Toward tutoring help seeking. In *International conference on intelligent tutoring systems*, pages 227–239. Springer, 2004.

Vincent Aleven, Bruce Mclaren, Ido Roll, and Kenneth Koedinger. Toward meta-cognitive tutoring: A model of help seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education*, 16(2):101–128, 2006.

Ivon Arroyo, Beverly Park Woolf, Winslow Burelson, Kasia Muldner, Dovan Rai, and Minghui Tai. A multimedia adaptive tutoring system for mathematics that addresses cognition, metacognition and affect. *International Journal of Artificial Intelligence in Education*, 24(4):387–426, 2014.

Ryan Baker and Adriana de Carvalho. Labeling student behavior faster and more precisely with text replays. In *Educational Data Mining 2008*, 2008.

Ryan Baker, Jason Walonoski, Neil Heffernan, Ido Roll, Albert Corbett, and Kenneth Koedinger. Why students engage in "gaming the system" behavior in interactive learning environments. *Journal of Interactive Learning Research*, 19(2):185–224, 2008.

Ryan Shaun Baker, Albert T Corbett, and Kenneth R Koedinger. Detecting student misuse of intelligent tutoring systems. In *International conference on intelligent tutoring systems*, pages 531–540. Springer, 2004a.

Ryan Shaun Baker, Albert T Corbett, Kenneth R Koedinger, and Angela Z Wagner. Off-task behavior in the cognitive tutor classroom: When students"game the system". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 383–390, 2004b.

Ryan Shaun Baker, Ido Roll, Albert T Corbett, and Kenneth R Koedinger. Do performance goals lead students to game the system? In *AIED*, pages 57–64, 2005.

Tobias Bartholomé, Elmar Stahl, and Rainer Bromme. Help-seeking in interactive learning environments: Effectiveness of help and learner-related factors in a dyadic setting. *CLS*, 4: 81–88, 2004.

Tobias Bartholomé, Elmar Stahl, Stephanie Pieschl, and Rainer Bromme. What matters in help-seeking? a study of help effectiveness and learner-related factors. *Computers in Human Behavior*, 22(1):113–129, 2006.

Deborah L Butler and Philip H Winne. Feedback and self-regulated learning: A theoretical synthesis. *Review of educational research*, 65(3):245–281, 1995.

Mihaela Cocea, Arnon Hershkovitz, and Ryan SJD Baker. The impact of off-task and gaming behaviors on learning: immediate or aggregate? In *Artificial Intelligence in Education*, pages 507–514. Ios Press, 2009.

Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

Ryan SJ d Baker, Albert T Corbett, Ido Roll, and Kenneth R Koedinger. Developing a generalizable detector of when students game the system. *User Modeling and User-Adapted Interaction*, 18(3):287–314, 2008.

Ryan SJ d Baker, Antonija Mitrović, and Moffat Mathews. Detecting gaming the system in constraint-based tutors. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 267–278. Springer, 2010.

Leandro Augusto da Silva, Sarajane Marques Peres, and Clodis Boscarioli. *Introdução à mineração de dados: com aplicações em R*. Elsevier Brasil, 2017.

Esam Ghaleb, Mirela Popa, Enrique Hortal, Stylianos Asteriadis, and Gerhard Weiss. Towards affect recognition through interactions with learning materials. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 372–379. IEEE, 2018.

Yue Gong, Joseph E Beck, Neil T Heffernan, and Elijah Forbes-Summers. The fine-grained impact of gaming (?) on learning. In *International Conference on Intelligent Tutoring Systems*, pages 194–203. Springer, 2010.

Simon Haykin. *Redes neurais: princípios e prática*. Bookman Editora, 2001.

Stuart A Karabenick. *Strategic help seeking: Implications for learning and teaching*. Routledge, 2012.

Maria Carolina Monard and José Augusto Baranauskas. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, 1(1):32, 2003.

Richard S Newman. Adaptive help seeking: A role of social interaction in self-regulated learning. 1998.

Richard S Newman. How self-regulated learners cope with academic difficulty: The role of adaptive help seeking. *Theory into practice*, 41(2):132–138, 2002.

Peter Norvig and Stuart Russell. Inteligência artificial. *Tradução: Regina Célia Simille de Macedo. Consultoria Editorial e Revisão técnica: Dr. Flávio Soares Corrêa da Silva, Dra. Leliane Nunes de Barros and Dra. Renata Wassermann*, 3:13–31, 2013.

Luc Paquette and Ryan S Baker. Variations of gaming behaviors across populations of students and across learning environments. In *International Conference on Artificial Intelligence in Education*, pages 274–286. Springer, 2017.

Luc Paquette, Adriana MJB de Carvalho, and Ryan Shaun Baker. Towards understanding expert coding of student disengagement in online learning. In *CogSci*, 2014.

Zachary A Pardos, Ryan SJD Baker, Maria OCZ San Pedro, Sujith M Gowda, and Supreeth M Gowda. Affective states and state tests: Investigating how affect throughout the school year predicts end of year learning outcomes. In *Proceedings of the third international conference on learning analytics and knowledge*, pages 117–124, 2013.

Arthur Paulino. Área abaixo da curva roc. https://medium.com/ensina-ai/%C3%A1rea-abaixo-da-curva-roc-15d2ae93a577. Accessed: 2018-10-06.

J Elizabeth Richey, Jiayi Zhang, Rohini Das, Juan Miguel Andres-Bray, Richard Scruggs, Michael Mogessie, Ryan S Baker, and Bruce M McLaren. Gaming and confrustion explain learning advantages for a math digital learning game. In *International Conference on Artificial Intelligence in Education*, pages 342–355. Springer, 2021.

Janet Ward Schofield. *Computers and classroom culture*. Cambridge University Press, 1995.

Jason A Walonoski and Neil T Heffernan. Detection and analysis of off-task gaming behavior in intelligent tutoring systems. In *International Conference on Intelligent Tutoring Systems*, pages 382–391. Springer, 2006.