Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Departamento de Engenharia Elétrica

Framework de mineração de dados educacionais em ambiente de cursos a distância governamental

Iure Vieira Brandão

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. -Ing João Paulo Carvalho Lustosa da Costa

Brasília
2018

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Departamento de Engenharia Elétrica

Framework de mineração de dados educacionais em ambiente de cursos a distância governamental

Iure Vieira Brandão

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. -Ing João Paulo Carvalho Lustosa da Costa (Orientador)
ENE/UnB

Prof. Dr. Rafael Timóteo de Sousa Júnior
ENE/UnB

Dr. Marco Antonio Marques Marinho
ENE/UnB

Prof. Dr. José Edil Guimarães de Medeiros
Coordenador do Curso de Engenharia da Computação

Brasília, 13 de novembro de 2018

Dedicatória

Dedico este trabalho a todos os meus amigos e familiares, principalmente, ao meu padrinho, que onde quer que ele esteja, sempre vai me vigiar e querer o melhor para mim.

Agradecimentos

Primeiramente, agradeço a Deus por sempre me abençoar, me guiar e me mostrar os passos que precisam ser seguidos para completar cada etapa da minha vida. Em segundo lugar, agradeço imensamente a toda minha família, em especial, aos meus pais e meu irmão por serem a base da minha vida, pois sem eles, eu não chegaria aonde estou.

Agradeço também ao Prof. João Paulo pela oportunidade da realização deste trabalho e, principalmente, pelo apoio e confiança prestados em mim; ao Juliano Prettz e Bruno Justino por sempre me darem suporte e ensinamentos quando precisei e, que ao final, se tornaram grandes amigos meus; ao Giovanni Almeida e Vinícius Coelho, por terem me ajudado a desenvolver este trabalho.

Agradeço especialmente a Camila França por todo o amor, carinho e apoio durante todo esse tempo. Aos meus grandes amigos de curso, Gabriel, Ismael, Eduardo, Fábio, Gustavo, Yuri, Victor Araújo e Victor Dantas, pelos grandes ensinamentos e apoio durante todo os 5 anos de curso.

Agradeço imensamente, também, a todos meus grandes amigos que Deus me deu, Junqueira, Vitor, Matheus Fernando, Fernanda, Amanda, Caio, Iago, Vinícius, Luiza e Stefhanny, que independentemente do momento, sempre estenderam as suas mãos para me ajudar e apoiar.

Por fim, agradeço a Enap (TED 083/2016), como um todo, por ter permitido a realização deste trabalho ao confiar em mim e fornecer sua base de dados para realização deste estudo.

Resumo

A Mineração de Dados Educacionais, do inglês *Educational Data Mining* (EDM), é a área de pesquisa preocupada com o uso de métodos de mineração de dados e seu desenvolvimento, com o objetivo de explorar conjuntos de dados coletados em plataformas educacionais. A EDM tem um potencial considerável para melhorar a qualidade das metodologias de ensino-aprendizagem e a sua literatura científica possui uma vasta quantidade de trabalhos relacionados a este tema, no entanto é uma área que ainda precisa expandir significativamente, pelo fato de ser um ramo de pesquisa relativamente novo. Neste trabalho de conclusão de curso é proposto um *framework* de EDM que visa modelar as variáveis do Moodle da base de dados da Escola Nacional de Administração Pública (Enap). Essa base de dados é bastante extensa por possuir cerca de 76 mil alunos distribuídos em 144719 conclusões de cursos, que foram ofertados durante os anos de 2015 e 2016. Em seguida, é aplicada as técnicas de EDM para compreender a influência das variáveis padrão do Moodle e as variáveis propostas nas taxas de desempenho e abandono dos participantes. A estrutura do *framework* inclui as seguintes abordagens de mineração de dados: árvore de decisão, árvore de decisão baseada em Adaboost, random forest e k-means. O algoritmo Adaboost gerado supera todas as outras abordagens com 89 % de precisão, 88 % de revocação e 88% de medida de F1, em termos de avaliação de desempenho dos participantes. Dessa forma, foi possível concluir que o modelo proposto alcançou ótimos resultados e que foi capaz de gerar indicadores relacionados as características dos participantes, que são suscetíveis de utilização afim de diminuir o índice de reprovação e de desistência nos cursos à distância, ofertados pela Enap.

Palavras-chave: Mineração de Dados Educacionais, Análise Preditiva, Classificação supervisionada, Clusterização, Aprendizado de Máquina.

Abstract

Educational Data Mining (EDM) is the research area concerned with the use of data mining methods and its development in order to explore data sets collected in educational platforms. EDM has a considerable potential to improve the education quality and its scientific literature have a lot of projects, however this area still needs to be more expanded, because it is, relatively, a new research area. In this work, it was modeled the Moodle variables of the National School of Public Administration (Enap) database. And the Enap database has about 76 thousand of attendees distributed in 144719 course completions, which these courses were offered during the years, 2015 and 2016. Next it is applied EDM to understand the standard Moodle variables influence and the proposed variables in the attendees performance and dropout rates. The framework proposed includes the following data mining approaches: decision tree, Adaboost based decision tree, random forest and k-means. The obtained Adaboost based decision tree outperforms the other approaches with 89 % of Precision, 88 % of Recall and 88 % of F1 score in terms of the attendees performance evaluation. Thus, the proposed Framework reached great results and it generated some indicators related to the attendees characteristics, which can be used in order to reduce the failure and the dropout rates in the distance learning courses offered by Enap.

Keywords: Educational Data Mining, Predictive analysis, Supervised classification, Clustering, Machine Learning

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AM Aprendizado de Máquina (AM), do inglês *Machine Learning*.

AVA Ambiente Virtual de Aprendizagem.

CART Árvores de Classificação e Regressão, do inglês *Classification and Regression Trees*.

CSV Valores separados por vírgula, do inglês *Comma-separated values*.

EAD Educação a Distância.

EDM Mineração de Dados Educacionais, do inglês *Educational Data Mining*.

Enap Escola Nacional de Administração Pública.

FN Falso negativo, do inglês *False negative*.

FP Falso positivo, do inglês *False positive*.

KDD Descoberta de conhecimento em bancos de dados, do inglês *Knowledge Discovery in Database*.

MER Modelo entidade relacionamento.

SQL Linguagem de consulta estruturada, do inglês *Structured Query Language*.

SSE Soma do erro quadrado, do inglês *sum of the squared error*.

TN Verdadeiro negativo, do inglês *True negative*.

TP Verdadeiro positivo, do inglês *True positive*.

Capítulo 1

Introdução

A utilização da Educação a Distância (EAD) vem crescendo cada vez mais e é destaque em várias áreas de sua aplicação, pois contribui para uma grande abertura no atendimento das demandas por aprendizagem. Essa abertura se deve ao fato de ser uma modalidade de educação mediada por tecnologias em que discentes e docentes estão separados espacialmente, isto é, não estão fisicamente presentes em um ambiente presencial de ensino-aprendizagem [1].

Nesta circunstância, a Escola Nacional de Administração Pública (Enap) utiliza a EAD com o objetivo de capacitar os servidores públicos para que tenham condições de aumentar a capacidade de governo na gestão de políticas públicas. Para isso, a Enap utiliza um Ambiente Virtual de Aprendizagem (AVA) capaz de registrar em suas bases de dados todas as interações dos participantes com os cursos, tutores e todas as atividades realizadas por eles.

Esses registros geram bases de dados enormes e, devido ao nível de informações e de detalhes, a tarefa de analisá-las e processá-las se torna bastante árdua sem a ajuda da mineração de dados. Logo, este contexto torna bastante atrativo a aplicação de técnicas de mineração de dados afim de melhorar a qualidade dos cursos à distância.

1.1 Motivação

O termo mineração de dados, do inglês *Data Mining*, refere-se à área de conhecimento que tem o intuito de descobrir informações relevantes, até então não descobertas, através da análise de grandes quantidades de dados. Essas informações relevantes fazem referência ao processo de identificar relações entre dados que podem produzir novos conhecimentos e gerar novas descobertas. As informações sobre a relação entre dados e, posteriormente, a descoberta de novos conhecimentos, podem ser muito úteis para realizar atividades de tomada de decisão em diversas áreas [2].

Dessa forma, na área da educação, é possível utilizar a mineração de dados para verificar a relação entre uma abordagem pedagógica e o aprendizado do aluno, por exemplo. Através desta informação, não só o professor, mas a gestão educacional poderia compreender se a sua abordagem realmente está ajudando o aluno e, consequentemente, desenvolver novos métodos de ensino mais eficazes. Recentemente, com a expansão dos cursos a distância [1], muitos pesquisadores da área de Mineração de Dados Educacionais, do inglês *Educational Data Mining* (EDM), têm mostrado interesse em utilizar mineração de dados para investigar perguntas científicas nessa área, como por exemplo, quais são os fatores que afetam a aprendizagem dos alunos ou como desenvolver sistemas educacionais mais eficazes.

Dentro deste contexto, a EDM é definida como a área de pesquisa que tem como principal foco o desenvolvimento de métodos para explorar conjuntos de dados coletados em ambientes educacionais [3]. Portanto, é possível compreender de forma mais eficaz e adequada os alunos, como eles aprendem, o papel do contexto na qual a aprendizagem ocorre, além de outros fatores que influenciam a aprendizagem. Da mesma forma, é possível identificar se o aluno está desmotivado, as principais razões de evasão dos cursos e quando elas ocorrem, com o intuito de personalizar o ambiente e os métodos de ensino para oferecer melhores condições de aprendizagem.

1.2 Definição do Problema

Como a EDM é uma área recente de pesquisa, ela vem se estabelecendo como uma forte e consolidada linha de pesquisa que possui grande potencial para melhorar a qualidade do ensino de cursos presenciais e à distância [4]. Apesar dos esforços de pesquisadores brasileiros, essa área ainda é pouco explorada no país e tem um enorme potencial de crescimento [5].

Além disso, existem muitos métodos utilizados na EDM que são originalmente da área de mineração de dados. Contudo, muitas vezes estes métodos precisam ser modelados para que sejam aplicados no contexto da EAD. Outro aspecto é que existe uma falta de independência estatística nos tipos de dados encontrados ao coletar informações em ambientes educacionais [6]. Por causa disso, diversos algoritmos e ferramentas utilizadas na área de mineração de dados precisam ser modificados para analisar dados educacionais.

E a partir da oferta de cursos de capacitação de curta duração na modalidade instrucional (com tutoria) à distância da Escola Nacional de Administração Pública (Enap), tem-se a necessidade de construir uma análise dos dados de todos os alunos que fizeram esses cursos de 2015 a 2016 com o intuito de adquirir novos conhecimentos sobre as ca-

racterísticas do perfil de cada aluno e, consequentemente, encontrar maneiras de evoluir o ensino à distância e torná-lo cada vez mais atrativo e utilizável por diversos alunos.

1.3 Objetivos

A partir da motivação apresentada na seção 1.1, o objetivo principal desse trabalho foi sumarizado na seção 1.3.1. Para alcançar este objetivo, foi delimitado um conjunto de objetivos específicos listado na seção 1.3.2.

1.3.1 Objetivo Principal

Desenvolver um *framework*, com as técnicas de mineração de dados em âmbito educacional, especificamente na Enap, para modelar as variáveis do Moodle a fim de compreender melhor a influência delas nas performances dos participantes e em uma eventual chance de desistência do curso.

1.3.2 Objetivos Específicos

- Analisar e propor consultas para extração dos dados relacionados as informações dos cursos, interações e atividades dos alunos e dos tutores.
- Pré-processar os dados a fim de transformá-los em um formato apropriado para execução dos algoritmos de aprendizado de máquina.
- Analisar e comparar os resultados a partir de técnicas de classificação supervisionada e não-supervisionada com árvores de decisão, métodos ensemble e clusterização.
- Estudar e comparar os resultados obtidos sob a perspectiva de avaliação da performance dos algoritmos.
- Visualizar os resultados gerados a fim de entender e analisar as características dos participantes dos cursos.
- Extrair os dados e analisar as desistências nos cursos com o objetivo de identificar as principais causas para tal, com que frequência e quando elas ocorrem.

1.4 Trabalhos Relacionados

O *framework* de mineração de dados educacionais apresentado neste trabalho têm diversas aplicações relacionadas, por exemplo, o artigo [5], que apresenta as possibilidades de aplicação de técnicas de EDM no Brasil. Neste trabalho, os autores demonstram as

diversas tarefas de mineração de dados que podem ser aplicadas no contexto educacional e mostram uma análise de diversos trabalhos que demonstram como essa nova área de pesquisa pode contribuir com a melhor compreensão dos processos de ensino e aprendizagem, com o objetivo de motivar os alunos que utilizam a EAD no cenário da educação brasileira.

Em [7], é apresentado um estudo relacionado ao desenvolvimento de um plugin específico para um Ambiente Virtual de Aprendizagem (AVA), notadamente o *Moodle*, visando o combate à evasão nos cursos a distância da Enap. Esse plugin teve como funcionalidade principal o aperfeiçoamento da comunicação da instituição com os alunos, reduzindo as taxas de evasão através do envio automático de mensagens, de acordo com regras específicas definidas e com o nível de intervenção desejado.

O trabalho realizado em [8] concentrou-se, a partir do contexto de educação à distância no ambiente corporativo governamental, em minerar os dados gerados pela interação dos alunos com o Ambiente Virtual de Aprendizagem (AVA). O *Moodle* foi novamente utilizado durante a oferta de um curso de grande importância na formação dos servidores públicos, no ano de 2015, o curso de Gerência de Projetos: Teoria e Prática. Neste trabalho, foram analisados os dados dos *logs* referentes a 698 alunos deste curso e geradas (7) sete variáveis a partir desses dados, sob a perspectiva de identificar os perfis de interação dos alunos com o AVA.

Diferentemente dos trabalhos citados acima e, apesar dos trabalhos [7] e [8] também estarem relacionados à Enap, este trabalho propõe um novo *framework* de EDM, objetivando analisar os dados de todas as atividades e interações dos 76551 alunos de todos os 351 cursos ofertados por essa instituição durante os anos de 2015 e 2016 e, dessa forma, modelar as variáveis do *Moodle* a partir da base de dados da Enap. E, novamente, o *Moodle* foi o AVA utilizado pela Enap durante esse período. Portanto, este trabalho propõe a modelagem de 20 novas variáveis com o escopo de aplicar novas técnicas de mineração, como árvore de decisão, clusterização e métodos *ensemble*, para entender melhor a influência dessas variáveis nas performances dos participantes dos cursos, além de suas respectivas taxas de evasão e razões para tal.

1.5 Descrição dos capítulos

Este trabalho é apresentado nos capítulos restantes da seguinte forma:

O capítulo 2 - é designado a explicar os conceito teóricos de EAD, de mineração de dados e mineração de dados aplicada ao âmbito educacional, além de explicar todo o processo de aprendizado de máquina, com a descrição de seus algoritmos e as métricas

de avaliação deles. Tudo isso com o intuito de explicar o ciclo completo do processo de mineração de dados.

No capítulo 3 - é apresentado o *framework* proposto com todos os passos bem definidos e explicados, de forma a analisar e entender os dados de cursos à distância, com o objetivo de encontrar variáveis que possam influenciar as performances dos participantes dos cursos, permitindo uma perspectiva de melhora na qualidade da EAD da Enap.

O capítulo 4 - contém a apresentação dos resultados a partir da aplicação do *framework* proposto nos dados de EAD da Enap nos 351 cursos oferecidos de 2015 a 2016, composto por 76551 alunos. Inicialmente, são discutidos os resultados de cada algoritmo utilizado durante a aplicação do *framework* e, posteriormente, é feita a visualização dos dados gerados a partir destes algoritmos. Ademais, é mostrada a análise da taxa de desistência nestes cursos oferecidos e em qual período dos cursos elas ocorrem, afim de entender e analisar os perfis dos desistentes.

Por fim, no capítulo 5 é apresentado as conclusões para os estudos do *framework* proposto, as implicações dos resultados gerados e os trabalhos a serem realizados no futuro.

Capítulo 2

Referencial Teórico

Neste capítulo é apresentado o embasamento teórico utilizado para que fosse possível o desenvolvimento e a realização deste trabalho. Nele, são apresentados os conceitos relacionados ao ensino à distância, a descoberta de conhecimento em base de dados educacionais, abordando assuntos de como a mineração de dados pode ser utilizada nesse meio, que é quando surge o termo mineração de dados educacionais. Ademais, serão apresentados os algoritmos de aprendizado de máquina utilizados na mineração de dados educacional, e por conseguinte, as métricas de avaliação e performance desses algoritmos.

2.1 Educação a Distância

A EAD é definida como um processo de aprendizagem e ensino em ambientes virtuais, mediado por tecnologias interativas, que fornecem a base para o processo de educação através da interação e interlocução entre todas as pessoas envolvidas [9].

Uma das características marcantes dos cursos EAD é o uso da tecnologia. É através da internet que os alunos podem se comunicar com professores e colegas de turma, acessar o conteúdo do curso, assistir às aulas, tirar dúvidas e, principalmente, fazer avaliações. Portanto, um dos seus principais requisitos é que seus usuários tenham acesso a um computador conectado à internet e, principalmente, que eles tenham a noção básica de como manusear um computador, isto é, tenha uma alfabetização tecnológica básica.

Ao contrário do modelo tradicional de educação, onde os dados dos alunos são registrados em papéis ou até mesmo em sistemas básicos de armazenamento de dados, o modelo de curso EAD tem como sua principal característica ter todas as informações de atividades, notas, interações com o ambiente entre outras coisas, gravadas em bancos de dados de ambientes virtuais de aprendizagem. Desta forma, a mineração desses dados pode construir modelos analíticos que permitem a descoberta de perfis e padrões dos perfis dos alunos, dos tutores, dos cursos e, até mesmo, dos seus conteúdos [10].

Apesar da popularidade da EAD crescer rapidamente na última década no ensino superior, muitas questões fundamentais de ensino-aprendizagem ainda estão em debate [11]. Uma dessas questões é a lacuna que existe nas sociedades, no qual muitos têm limitações ou simplesmente não têm acesso à este tipo de educação, o que, consequentemente, gera uma falta de inclusão deste tipo de aprendizagem para essas pessoas. Outra questão importante é a falta de alfabetização tecnólogica que pode gerar um impacto negativo nos indivíduos que têm seus primeiros contatos com a EAD e, consequentemente, uma taxa de evasão desses cursos muito alta [12].

Porém, apesar destes problemas que circulam este modelo de aprendizagem, a utilização dele junto ao AVA está complementamente ligado à geração de uma quantidade massiva de dados de interações e de dados de alunos que são armazenados nas bases de dados do próprio AVA. Isto abre portas para a utilização de mineração de dados educacionais com o intuito de gerar novos conhecimentos relevantes acerca de todo o processo de ensino, o que auxilia a melhoria da qualidade de cursos à distância [13].

2.2 Mineração de Dados

A mineração de dados é o processo de descobrir automaticamente informações úteis em grandes repositórios de dados. Técnicas de mineração de dados são implantadas para vasculhar grandes bancos de dados, afim de encontrar padrões novos, relevantes e únicos, que poderiam permanecer desconhecidos. Por isso, que o termo mineração de dados também é conhecido como Descoberta de conhecimento em bancos de dados, do inglês *Knowledge Discovery in Database* (KDD). As técnicas de mineração, além de vasculharem as bases, também fornecem recursos para prever o resultado de uma observação futura, como por exemplo, prever se um cliente recém-chegado gastará mais de 100 reais em uma loja de departamentos [14].

Nem todas as tarefas de descoberta de informações são consideradas como mineração de dados. Por exemplo, procurar registros individuais usando um sistema de gerenciamento de banco de dados são tarefas relacionadas à área de recuperação de informações. Embora tais tarefas sejam importantes e possam envolver o uso de algoritmos sofisticados e de estruturas de dados, elas dependem de técnicas tradicionais de ciência da computação e características óbvias dos dados para criar estruturas de índice para organizar e recuperar informações com eficiência. No entanto, técnicas de mineração de dados têm sido usadas para melhorar os sistemas de recuperação de informação [15].

A mineração de dados é parte integral da KDD, que é o processo geral de conversão de dados brutos em informações úteis, conforme pode ser visto na Figura 2.1, que fora adaptada de [14]. Esse processo consiste em uma série de etapas de transformação, desde

o pré-processamento dos dados até o pós-processamento dos resultados da mineração de dados.

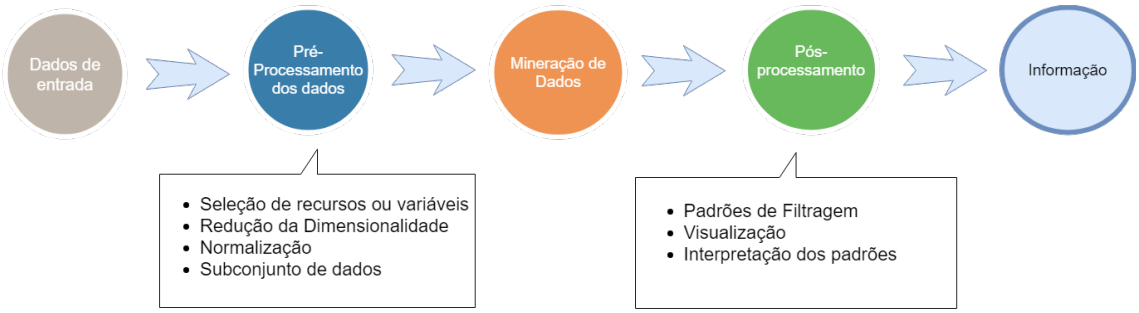


Figura 2.1: Processo de KDD.

Os dados de entrada podem residir em um repositório de dados centralizado ou distribuído e eles são armazenados em uma quantidade variada de formatos, como por exemplo, arquivos simples, planilhas ou, principalmente, em tabelas relacionais. A finalidade do pré-processamento é transformar os dados de entrada brutos em um formato apropriado para análise subsequente. As etapas envolvidas no pré-processamento de dados incluem a fusão de dados de várias origens, a limpeza de dados para remover ruídos e observações duplicadas e, principalmente, a seleção de registros e recursos relevantes para a tarefa de mineração de dados em questão.

Devido às muitas maneiras pelas quais os dados podem ser coletados e armazenados, o pré-processamento de dados é talvez a etapa mais trabalhosa e demorada no processo geral de descoberta de conhecimento, de acordo com [15]. Fechar o ciclo se refere ao processo de integrar os resultados da mineração de dados em sistemas de apoio à tomada de decisão. Essa integração requer uma etapa de pós-processamento que garanta que somente os resultados válidos e úteis sejam incorporados ao sistema de suporte à decisão. Um exemplo de pós-processamento importantíssimo é a visualização, que permite aos analistas de dados explorarem os dados e o resultado da mineração de uma quantidade vasta de pontos de vista.

2.2.1 Motivações para a evolução da Mineração de Dados

As técnicas tradicionais de análise de dados têm encontrado frequentemente dificuldades práticas para enfrentar os desafios colocados pelos novos conjuntos de dados e, por este motivo, a área de mineração de dados vêm sendo motivada a evoluir cada vez mais com o intuito de resolver essas adversidades encontradas [16].

O primeiro deste conjunto de problemas é a escalabilidade. Devido aos avanços na geração e coleta de dados, conjuntos de dados com tamanhos de *gigabytes*, *terabytes* ou até

petabytes tornaram-se bem comuns. Para os algoritmos de mineração de dados lidarem com esses enormes conjuntos de dados, eles devem ser escalonáveis. Muitos algoritmos de mineração de dados empregam estratégias de pesquisa especiais para lidar com problemas de pesquisa exponenciais. A escalabilidade também pode exigir a implementação de novas estruturas de dados para acessar registros individuais de maneira eficiente. Por exemplo, certos algoritmos podem ser necessários ao processar conjuntos de dados que não podem caber na memória principal. A escalabilidade também pode ser melhorada usando amostragem ou desenvolvimento de algoritmos paralelos e distribuídos [17].

O segundo aspecto é a alta dimensionalidade. Atualmente é comum encontrar conjuntos de dados com centenas ou milhares de atributos, ao invés de conjuntos de dados comuns com poucos atributos, como os de algumas décadas atrás [18]. Um exemplo disso, são os conjuntos de dados com componentes temporais ou espaciais, com tendência a ter alta dimensionalidade por terem diversos atributos atrelados a cada informação [19]. Técnicas tradicionais de análise de dados que foram desenvolvidas para dados de baixa dimensão, muitas vezes não funcionam bem para dados com dimensões tão elevadas. Além disso, para alguns algoritmos de análise de dados, a complexidade computacional aumenta rapidamente à medida que a dimensionalidade (número de recursos) aumenta [18].

O terceiro ponto se deve ao fato de certos dados serem heterogêneos e complexos. Os métodos tradicionais de análise de dados geralmente lidam com conjuntos de dados contendo atributos do mesmo tipo, contínuos ou categóricos. Como o papel da mineração de dados nos negócios, na ciência e na medicina têm crescido, também aumenta a necessidade de técnicas que possam lidar com atributos heterogêneos [20]. Recentemente, houve o surgimento de objetos de dados mais complexos e, exemplos para tal, incluem coleções de dados de DNA com estrutura sequencial e tridimensional [14]. Dessa forma, as técnicas desenvolvidas para mineração desses objetos complexos devem levar em consideração as relações dos dados, como por exemplo, a autocorrelação temporal e espacial.

A quarta e última motivação é a propriedade dos dados e a sua distribuição. Às vezes, os dados necessários para uma análise não são armazenados em um local ou de propriedade de uma organização. Ao invés disso, os dados são distribuídos geograficamente entre os recursos pertencentes a várias entidades. Isso requer o desenvolvimento de técnicas distribuídas de mineração de dados [14]. Entre os principais desafios enfrentados pelos algoritmos de mineração de dados distribuídos estão primeiramente em, como reduzir a quantidade de comunicação necessária para executar o computador distribuído. Em segundo, como consolidar efetivamente os resultados de mineração de dados obtidos de várias fontes e, em terceiro, como resolver os problemas da segurança destes dados a serem minerados [21].

2.3 Mineração de Dados Educacionais

A Mineração de dados educacionais é uma área emergente de pesquisa, preocupada com o desenvolvimento de métodos para exploração dos conjuntos de dados únicos provenientes dos ambientes educacionais, com o objetivo de utilizar estes métodos, para compreender melhor os alunos e as características de como eles aprendem. Atualmente ela vem se estabelecendo como uma forte e consolidada linha de pesquisa que possui grande potencial para melhorar a qualidade do ensino [22].

A EDM é definida por [23], como um campo que explora algoritmos estatísticos, de aprendizado de máquina, Ciência da Computação e, principalmente, de análise de aprendizado e de mineração de dados sobre os diferentes tipos de dados educacionais (baseado em informática ou não). O agrupamento desses campos pode ser visualizado na Figura 2.2, adaptada de [24]. O principal objetivo desse agrupamento é analisar esses tipos de dados, com o intuito de resolver questões de âmbito educacional de forma embasada por esses campos de pesquisa [23].

Figura 2.2: Campo explorados com a EDM.

O trabalho [5] registrou, no Brasil, as possibilidades de oportunidades de aplicação dos conceitos de EDM. Este trabalho pode ser considerado como um grande incentivo para a aplicação de técnicas de mineração de dados em ambientes educacionais no país, em especial na oferta da educação em modalidade a distância.

Para a possibilidade da oferta de EAD, foi necessária a criação de sistemas de ambientes virtuais de aprendizagem, o que culminou no aumento tanto do *software* educacional instrumental quanto dos bancos de dados desses sistemas com informações dos alunos [25]. Da mesma forma, o uso da Internet na educação criou um novo contexto conhecido como educação baseada na web, em que grandes quantidades de informações sobre interação de ensino-aprendizagem são infinitamente geradas e todas essas informações fornecem uma mina de ouro de dados educacionais [26].

Sistemas educacionais adaptados e inteligentes baseados em *web* têm sido vistos como uma solução para ambientes de aprendizado individualmente mais ricos. Esses sistemas tentam oferecer educação personalizada aos alunos, construindo um modelo dos objetivos, referências e conhecimentos do indivíduo [27]. Logo, em [23] foi definido o ciclo de aplicação de mineração de dados em sistemas educacionais. Este ciclo pode ser visualizado na Figura 2.3, que foi adaptada de [23].

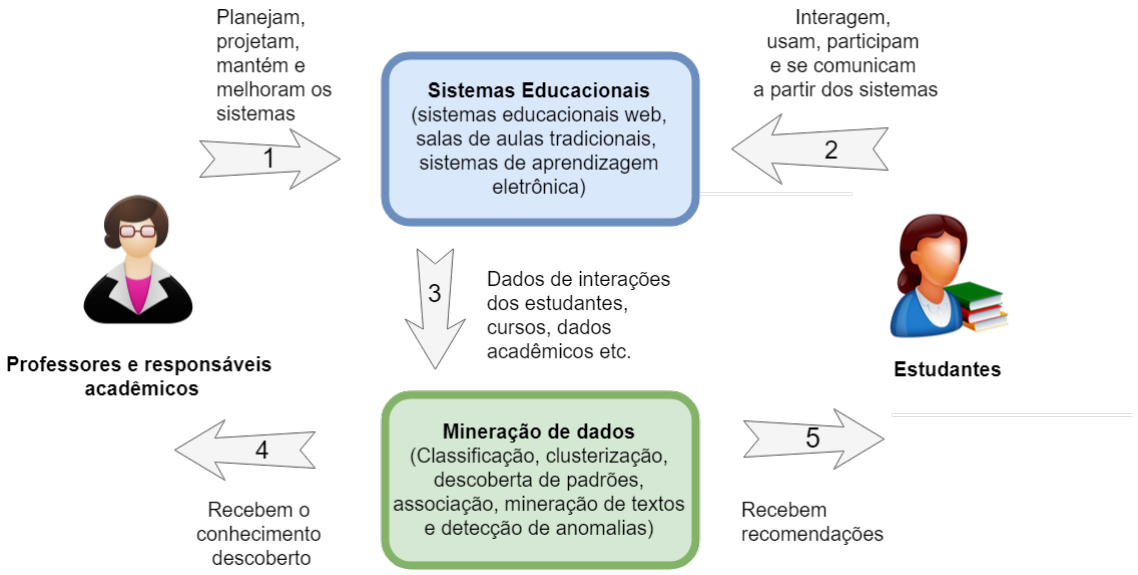


Figura 2.3: Ciclo de aplicação de mineração de dados em sistemas educacionais.

O ciclo começa, na seta indicada pelo número 1, no qual os professores e responsáveis acadêmicos planejam, projetam e mantém os sistemas educacionais. Com os sistemas educacionais prontos, os estudantes interagem e participam de atividades acadêmicas a partir deles, na seta indicada pelo número 2. Na seta indicada por 3, os dados das interações dos estudantes e dos cursos são armazenados nos bancos de dados para serem

utilizados na mineração de dados. Nesta etapa, é gerada a seta de número 4 que consiste nos professores e responsáveis acadêmicos receberam o conhecimento descoberto após a mineração. Com estas novas informações, os gestores podem melhorar os sistemas educacionais como um todo e reiniciar todo o ciclo. Nesta mesma etapa, se dá a origem da seta 5 que indica que a partir das minerações, é possível fazer recomendações aos estudantes de como melhorar o estudo em certas áreas, como se aperfeiçoar em certas matérias etc.

2.3.1 Tarefas da EDM

As tarefas de mineração de dados educacionais são geralmente divididas em duas maiores categorias [14]: tarefas preditivas e tarefas descritivas. No qual o objetivo das tarefas de predição é prever o valor de um determinado atributo com base nos valores de outros atributos no âmbito educacional. O atributo a ser previsto é comumente conhecido como variável-alvo ou dependente, enquanto que os atributos usados para fazer a previsão são conhecidos como variáveis explicativas ou independentes. Já o objetivo das tarefas descritivas é derivar padrões (correlações, tendências, clusters, trajetórias e anomalias) que resumem os relacionamentos subjacentes nos dados educacionais. Tarefas descritivas de EDM são constantemente de natureza exploratória e frequentemente requerem técnicas de pós-processamento para validar e explicar os resultados. A partir destas duas categorias, a Figura 2.4 (adaptada de [14]) ilustra quatro das principais tarefas da EDM.

A modelagem preditiva refere-se à tarefa de construir um modelo para a variável-alvo como uma função das variáveis explicativas. Existem dois tipos de tarefas de modelagem preditiva: classificação, que é usada para variáveis-alvo discretas, e regressão, que é usada para variáveis-alvo contínuas. Por exemplo, prever se um usuário do sistema educacional fará uma atividade em um módulo do curso é uma tarefa de classificação porque a variável de destino é de valor binário. Por outro lado, prever a nota futura de um aluno é uma tarefa de regressão, pois o preço é um atributo de valor contínuo [28]. O objetivo de ambas as tarefas é aprender um modelo que minimize o erro entre os valores previstos e verdadeiros da variável de destino [14].

A análise de associação é usada para descobrir padrões que descrevem características fortemente associadas aos dados. Os padrões descobertos são normalmente representados na forma de regras de implicação ou subconjuntos de recursos. Devido ao tamanho exponencial de seu espaço de pesquisa, o objetivo da análise de associação é extrair os padrões mais interessantes de maneira eficiente. Aplicações úteis de análise de associação incluem encontrar grupos de alunos que têm atividades parecidas, identificar alunos que precisam de tratamento especial ou entender as relações entre cursos e alunos [29].

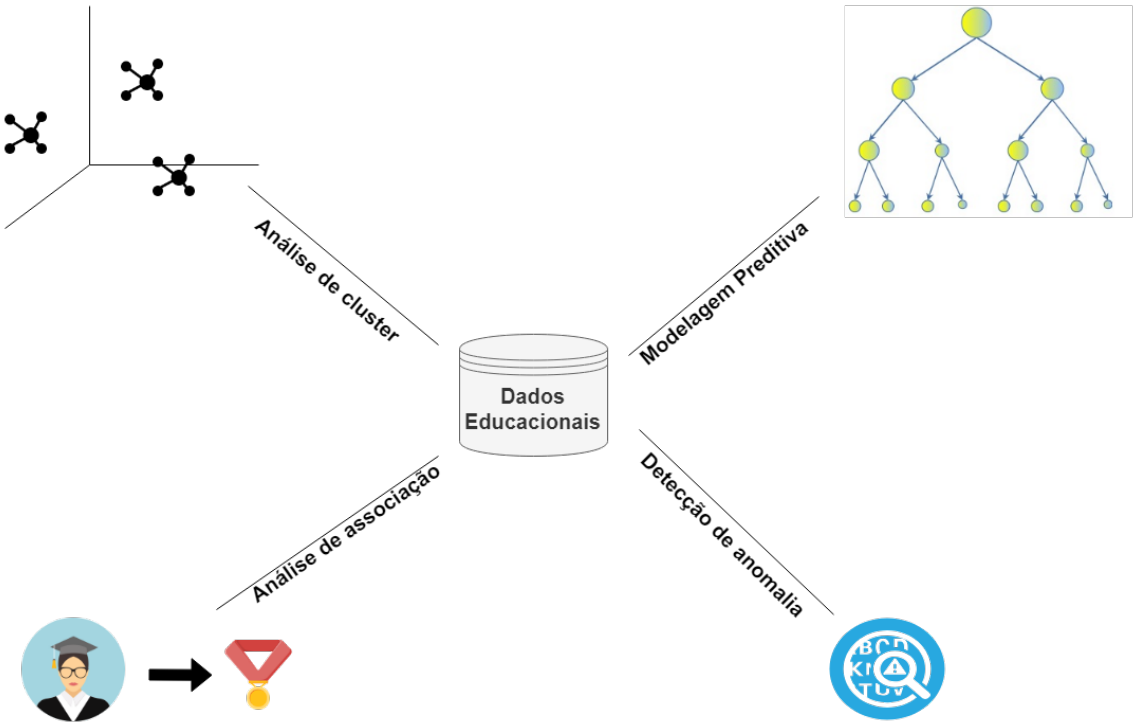


Figura 2.4: Quatro das principais tarefas da EDM.

A análise de cluster procura encontrar grupos de observações intimamente relacionadas, de modo que as observações que pertencem ao mesmo agrupamento (cluster) sejam mais semelhantes entre si do que as observações que pertencem a outros agrupamentos. O agrumento tem sido usado para reunir conjuntos de alunos com atividades relacionadas e notas parecidas [30].

Detecção de anomalia é denominado a tarefa de identificar observações cujas características são significativamente diferentes do restante dos dados. Tais observações são conhecidas como anomalias, do inglês "outliers". O objetivo de um algoritmo de detecção de anomalias é descobrir as anomalias reais e evitar rotular os objetos normais como anômalos. Em outras palavras, um bom detector de anomalia deve ter uma alta taxa de detecção e uma baixa taxa de alarme falso. As aplicações de detecção de anomalias incluem a detecção de fraudes nos sistemas educacionais, padrões incomuns de performances e de atividades dos alunos [31].

2.4 Aprendizado de Máquina

O Aprendizado de Máquina (AM), do inglês *Machine Learning*, é definido em [32], como um procedimento de análise que automatiza o desenvolvimento de modelos analí-

ticos usando algoritmos, que aprendem interativamente a partir de dados de entrada. A característica interativa do aprendizado de máquina é essencial, pois conforme os modelos são expostos a novos dados, eles se tornam capazes de se adaptarem de forma independente, isto é, aprendem com os modelos e cálculos feitos anteriormente para produzir decisões e resultados confiáveis sobre esse novo conjunto de dados [33].

Inicialmente, para resolver um problema em um computador (máquina), é preciso de um algoritmo. E um algoritmo é uma seqüência de instruções que devem ser realizadas para transformar a entrada em saída. Porém, existem muitas aplicações para as quais não se tem um algoritmo, mas existem dados de exemplo. E com o avanço da tecnologia, atualmente tem-se a capacidade de armazenar e processar grandes quantidades de dados, bem como acessá-los de locais fisicamente distantes em uma rede de computadores. Os dados armazenados só se tornam úteis quando analisados e transformados em informações que podem ser usadas, por exemplo, para fazer previsões. É muito improvável identificar o processo no qual o dado foi concebido pelas diferentes características de interações dos usuários, mas é possível construir uma boa e útil aproximação desse processo. Dessa forma, é possível detectar certos padrões ou regularidades nestes dados. E é exatamente este o nicho do AM [33].

A aplicação de métodos de AM em grandes base de dados é chamada de mineração de dados e, consequentemente, dessa forma essas duas áreas estão bastante atreladas [34]. Porém, o AM não é somente um problema de dados, é também uma parte do ramo da inteligência artificial. Para ser inteligente, um sistema que esteja em um ambiente em constante mudança deve ter a capacidade de aprender e se o sistema pode aprender e se adaptar a essas mudanças, o projetista do sistema não precisará prever nem fornecer soluções para todas as situações possíveis. Logo, o AM é bastante essencial para achar soluções para vários problemas em diversas áreas além da mineração, como por exemplo, problemas de visão, reconhecimento de fala e robótica [35].

Segundo [33], o AM pode ser divido em 3 tipos de aprendizado: o aprendizado supervisionado (seção 2.4.1), o aprendizado não-supervisionado (seção 2.4.2) e o aprendizado por reforço (seção 2.4.3).

2.4.1 Aprendizado Supervisionado

A aprendizagem supervisionada, do inglês *supervised learning*, é uma técnica de AM com o objetivo de aprender a mapear a partir de dados de entrada, uma saída, cujos os valores corretos do mapeamento são fornecidos por um supervisor. A saída deste mapeamento pode prever um rótulo de classe do objeto de entrada, em problemas de classificação, ou pode ser um valor contínuo, em problemas de regressão [36].

O aprendizado supervisionado é caracterizado pela capacidade de construir modelos que aprendem com base em observações ou conjunto de dados existentes, o que permite a replicação desse aprendizado na previsão de futuros cenários. Neste tipo de aprendizado, os algoritmos possuem a capacidade de generalizar a partir de regularidades constatadas apoiado em um determinado conjunto de treinamento, isto é, eles utilizam um conhecimento prévio do domínio, previamente estabelecido, para orientar a generalização de situações futuras [33].

Dentro do aprendizado supervisionado, existem duas tarefas principais, que são: a classificação e a regressão. Ambas estão relacionados aos tipos de dados utilizados e, conforme pode ser visto na Figura 2.5 (adaptada de [14]), as duas tarefas possuem a função de mapear um atributo de entrada x para um dos rótulos de classe predefinido, y. Porém, a característica principal que distingue classificação de regressão é que na regressão, uma tarefa de modelagem preditiva y é um atributo contínuo, enquanto que na classificação é um atributo discreto [14].

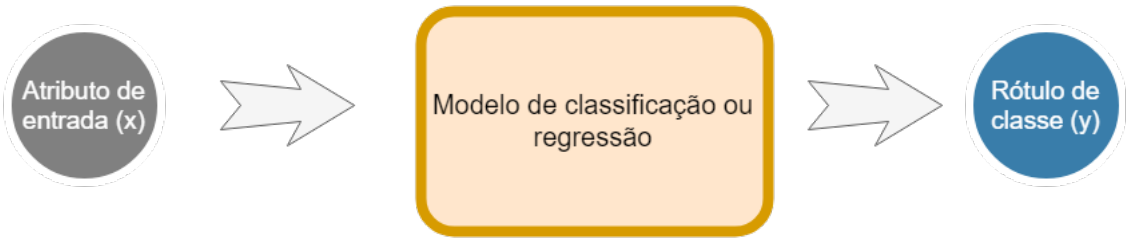


Figura 2.5: Classificação ou regressão como tarefa de mapear um atributo x em um rótulo de classe y.

Um modelo de classificação pode servir como uma ferramenta explicativa para distinguir entre objetos de diferentes classes e também pode ser usado para prever o rótulo da classe de registros desconhecidos. Para isso, tem-se os atributos preditivos e o atributo-alvo. Os atributos preditivos designam as características que serão determinantes para que se classique um dado em um atríbuto-alvo. Logo, o atríbuto-alvo é a característica que se deseja predizer com base nas características dos atributos preditivos e ele é categórico com os rótulos que representam as classes dos dados.

Os exemplos de classificadores, ou técnica de classificação, incluem classificadores baseados em árvore de decisão, classificadores baseados em regras, redes neurais, máquinas de vetores de suporte e classificadores *naive Bayes*. Cada técnica emprega um algoritmo de aprendizado para identificar um modelo que melhor se ajusta ao relacionamento entre o conjunto de atributos e o rótulo de classe dos dados de entrada. O modelo gerado por um algoritmo de aprendizado deve ajustar os dados de entrada bem e prever corretamente os rótulos de classe de registros que ele nunca viu antes. Portanto, um objetivo chave

do algoritmo de aprendizado é construir modelos com boa capacidade de generalização, isto é, modelos que prevêem com precisão os rótulos de classe de registros anteriormente desconhecidos [37].

Uma abordagem geral para resolver problemas de classificação é definir, primeiramente, um conjunto de treinamento, que consiste em registros cujos rótulos de classe são conhecidos. Após isso, o conjunto de treinamento é usado para criar um modelo de classificação, que é aplicado posteriormente ao conjunto de testes, que consiste em registros com rótulos de classe desconhecidos [33]. Entre as tarefas de classificação, a técnica baseada na construção de árvores de decisão se destaca quando é necessária a identificação de padrões descritivos e preditivos, além de ser uma técnica muito utilizada e ter um potencial de performance significativo [38].

2.4.1.1 Árvore de Decisão

A árvore de decisão é um classificador supervisionado que pode ser definido como uma estrutura de dados hierárquica que implementa a estratégia de dividir e conquistar por meio de nós com bordas direcionadas. Existem diversos algoritmos e todos eles constroem uma árvore a partir de uma dada amostra de treinamento devidamente rotulada com as classes de interesses [39].

A ideia deste classificador é fazer uma série de perguntas cuidadosamente elaboradas sobre os atributos do registro de teste. Para cada possibilidade de resposta, uma pergunta de acompanhamento é feita até chegarmos a uma conclusão sobre o rótulo de classe do registro. Essa série de perguntas e respostas pode ser organizada em uma forma de uma árvore e, dessa forma, a árvore de decisão tem três tipos de nós [14]:

- O nó raiz, que não tem arestas de entrada, e zero ou mais de saída.
- O nó interno, que contém exatamente uma aresta de entrada e duas de saída.
- O nó folha, que contém uma aresta de entrada e nenhuma de saída.

Em uma árvore de decisão, cada nó folha é atribuído a um rótulo de classe. Os nós não-terminais, que incluem a raiz e outros nós internos, contêm condições de teste de atributo para separar registros que possuem características diferentes. E o caminho com o maior número de níveis até chegar ao nó folha de uma árvore, denomina a sua profundidade. Um exemplo simples de árvore de decisão para classificar animais vertebrados em mamíferos e não mamíferos, pode ser visto na Figura 2.6 (adaptada de [14]). No qual, o atributo de tempetura do corpo é utilizado como nó raiz para separar animais vertebrados de sangue quente e frio. Como todos os vertebrados de sangue frio não são mamíferos, um nó folha identificado como não mamíferos é criado como nó filho à direita do nó raiz. Se o

vertebrado é de sangue quente, um atributo subsequente, dá à luz, é usado para distinguir mamíferos de outras criaturas de sangue quente, que são principalmente as aves.

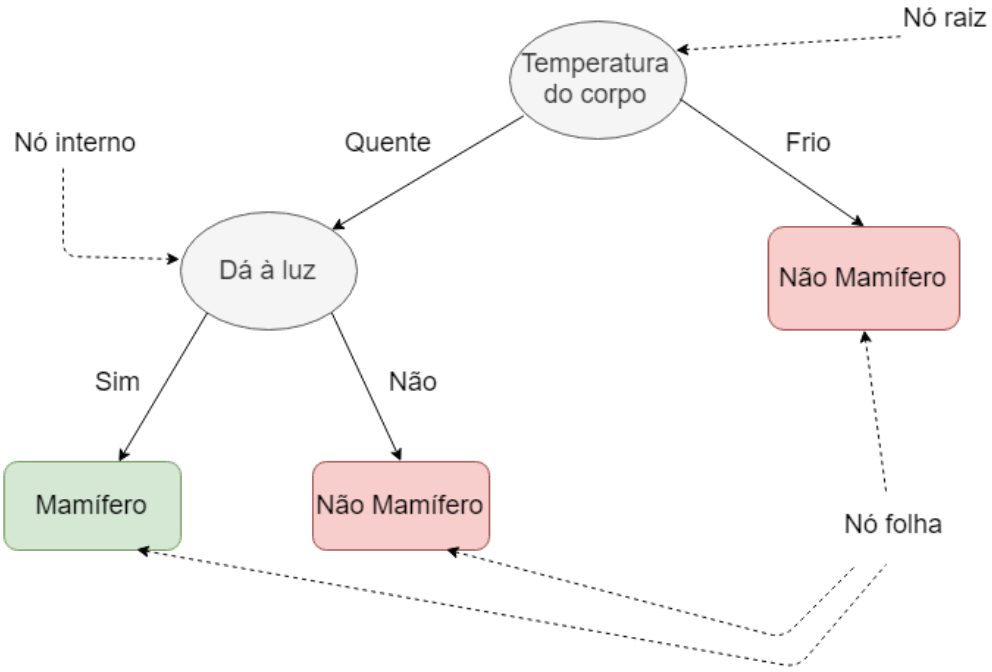


Figura 2.6: Exemplo simples de árvore de decisão.

Classificar um registro de teste é simples quando uma árvore de decisão é construída. A partir do nó raiz, é aplicada a condição de teste ao registro e é seguida a ramificação apropriada com base no resultado do teste. Isso levará a outro nó interno, para o qual uma nova condição de teste é aplicada ou a um nó folha. O rótulo de classe associado ao nó folha é então atribuído ao registro e o dado é classificado.

2.4.1.1.1 Algoritmo CART

Árvores de Classificação e Regressão, do inglês *Classification and Regression Trees* (CART) é um algoritmo de árvore de decisão muito utilizado por ser a evolução de diversos algoritmos como o ID3, C4.5 e, consequentemente, possui uma performance significativamente elevada em relação a esses algoritmos anteriores [40]. Neste algoritmo, uma árvore de decisão cresce de uma maneira recursiva particionando os registros de treinamento em subconjuntos sucessivamente apropriados. Para ilustrar a definição dessa recursão, considere D_t um conjunto de treinamento que estão associados a um nó t e $y = \{y_1, y_2, ..., y_c\}$ sejam os rótulos de classe. Dessa forma, a recursão leva em consideração dois passos [41]:

1. Se todos os registros do conjunto D_t pertencem a mesma classe y_t, logo t é um nó folha rotulado como y_t.

2. Se o conjunto D_t contém registros que pertencem a mais de uma classe, um atributo de condição de teste é selecionado para particionar estes registros em subconjuntos menores. Um nó filho é então criado para cada possibilidade de resultado da condição de teste e os registros em D_t são distribuídos nos nós filhos conforme estes resultados.

Dessa forma, o algoritmo CART ainda precisa resolver o problema de como escolher o atributo de condição de teste para particionar o conjunto de treinamento em subconjuntos da melhor forma. Para isso, existem várias métricas e todas elas são definidas em termos da distribuição das classes dos atributos antes e depois da partição. O mais utilizado, é a impureza de *Gini*, por escolher um critério de partição mais eficiente, gerando árvores mais precisas e menos complexas [42].

Para demonstrar como a impureza de *Gini* é calculada e depois poder defini-la, considere que $p(i|t)$ denote a fração de registros que pertencem a classe i em um nó t. Para um problema de duas classes, a distribuição de classes em qualquer nó pode ser definida como (p_0, p_1), onde $p_1 = 1 - p_0$ [14]. A partir da distribuição calculada com base nos nós de uma árvore, a medida de impureza de *Gini* e o erro de classificação são dados por:

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1}[p(i|t)]^2, \tag{2.1}$$

$$\text{Erro de classificação}(t) = 1 - \max_i[p(i|t)], \tag{2.2}$$

onde c é o número de classes. Logo, a impureza de Gini é uma medida da frequência com que um elemento escolhido aleatoriamente do conjunto seria rotulado incorretamente se fosse rotulado aleatoriamente de acordo com a distribuição de rótulos no subconjunto, isto é, ela mede o quão impuro o nó é, da forma que se fosse rotular uma classe para um registro a partir de um nó, qual seria a frequência com que o classificador erra. Já o erro de classificação é dado pela máxima proporção das distribuições em um nó, que determina a taxa de erro de uma classificação. Dessa forma, quanto maior a impureza, maior a chance de classificar erradamente um registro [14].

Levando tudo isso em consideração, o algoritmo de CART utiliza essa medida para escolher iterativamente o atributo de condição de teste que deixa menor o grau de impureza dos nós filhos, para que dessa forma seja criada uma árvore com a maior precisão possível de classificação [43]. Dessa forma, considerando vetores de treinamento $x_i \in R^n$, $i = 1, \ldots, l$, e um vetor com os rótulos de saída $y \in R^l$, o espaço é dividido recursivamente de forma que as amostras com os mesmos rótulos sejam agrupadas. Ao considerar os dados

em um nó m, que é representado pela letra Q. Para cada candidato de atributo de teste $\theta = (j, t_m)$ consistindo em um característica j e um limiar t_m, particionar os dados em $Q_{esquerda}$ e $Q_{direita}$ nós filhos, pode ser escrito da seguinte forma [41]:

$$\begin{aligned} Q_{esquerda}(\theta) &= (x, y)|x_j <= tm, \\ Q_{direita}(\theta) &= Q \backslash Q_{esquerda}(\theta), \end{aligned} \tag{2.3}$$

onde no nó filho à esquerda ($Q_{esquerda}$), em um conjunto de dados (x) e rótulos (y), o conjunto de dados que passa para ele, deve ter a sua característica j menor ou igual ao um limite tm. E para o nó filho à direita ($Q_{direita}$), vão todos os dados que não foram para o nó filho à esquerda, pois não satisfizeram as suas características j com o limite t_m. Porém, com a Equação 2.1, é necessário escolher o candidato de atributo de teste que gere o menor nível de impureza para o nó filho à esquerda [41]:

$$\theta^* = \arg\min_{\theta} \text{Gini}(Q, \theta) \tag{2.4}$$

Com toda a árvore modelada e feita, é possível calcular a importância de cada variável dela, isto é, será possível mensurar o quão importante cada característica é, do conjunto de dados, para que os desempenhos da classificação e predição possam ser alterados significativamente. Essa medição da importância é chamada de importância de Gini e ela é calculada como a soma das reduções da impureza de gini de uma variável para os seu nós filho durante toda a árvore, isto é, a cada vez que a variável é escolhida como atributo de condição de teste na árvore, é calculado o quanto essa variável reduziu a sua impureza, proporcionalmente, para as impurezas dos seus nós filhos. Esse é calculo é feito da forma a seguir [41]:

$$\text{Importância de Gini} = \sum_{j=1}^{k} \frac{N_{Tj}}{N} \cdot \left[I_j - \left(\frac{N_{TRj}}{N_{Tj}} \cdot I_{Dj} \right) - \left(\frac{N_{TLj}}{N_{Tj}} \cdot I_{Lj} \right) \right], \tag{2.5}$$

onde K é igual ao número de vezes que a variável é utilizada como critério na árvore, N é o número total de amostras, N_{Tj} é o número de amostras no nó atual (j), N_{TLj} e N_{TRj} são o número de amostras do nó filho à esquerda e à direita, respectivamente, I_j é a impureza de Gini do nó atual e, por fim, I_{Dj} e I_{Lj} são as impurezas dos nós à direita e à esquerda, respectivamente.

2.4.1.1.2 Exemplo de cálculo da impureza de Gini

Para exemplicar o cálcula da impureza de Gini, na Figura 2.7, este único nó raiz (N1) tem um número igual de registros para ambas as classes, que é igual a 2. Logo, a

distribuição antes da partição para p_0 é o número de registros que pertencem à classe 0, divido pelo número de registros do nó. Dessa forma $p_0 = 2/4 = 0.5$, $p_1 = 1 - 0.5 = 0.5$ e a distribuição é (0.5, 0.5). Depois da partição, para o nó filho à esquerda (N2), a distribuição é (1,0), pois só existem membros pertences à classe 0. Da mesma forma, para o nó filho à direita (N3), a distribuição é (0,1), pois só existem membros pertences à classe 1.

No exemplo da Figura 2.7, a impureza de *Gini* e o erro de classificação seriam:

$$\text{Gini (Nó N1)} = 1 - (2/4)^2 - (2/4)^2 = 0.5$$
$$\text{Erro (Nó N1)} = 1 \text{ - max } [2/4, 2/4] = 0.5$$
$$\text{Gini (Nó N2)} = 1 - (2/2)^2 - (0/2)^2 = 0$$
$$\text{Erro (Nó N2)} = 1 \text{ - max } [0/2, 2/2] = 0$$
$$\text{Gini (Nó N3)} = 1 - (0/2)^2 - (2/2)^2 = 0$$
$$\text{Erro (Nó N3)} = 1 \text{ - max } [2/2, 0/2] = 0$$

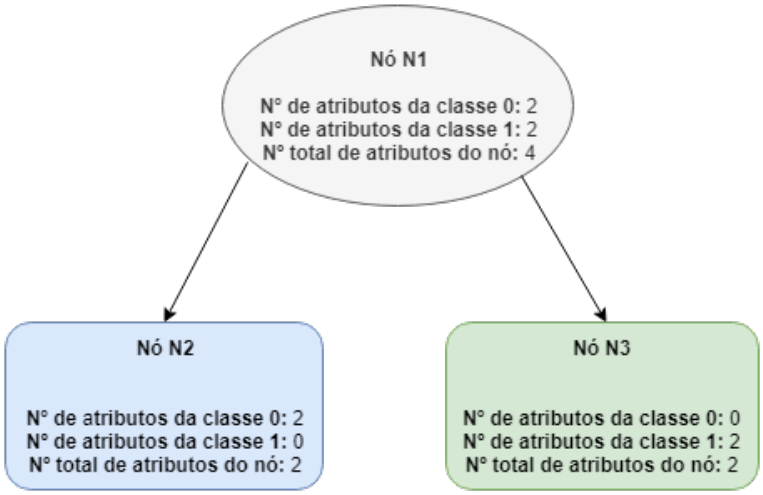


Figura 2.7: Árvore de decisão com distribuição de classes.

Dessa forma, pode-se observar que as impurezas dos nó 2 e 3 tinham o valor mais baixo possível, pois para cada nó, só há como atribuir um rótulo para um registro de entrada e, consequentemente, pela Equação 2.2 não há taxa de erro. Já para o nó 1, a impureza era a mais alta possível, pois a distribuição de classe era uniforme, o que leva a uma taxa de erro de 0.5 (mais alta possível). No gráfico da Figura 2.8 (adaptado de [14]), é possível observar que quanto maior a distribuição p das classes, maior a impureza de *Gini* e o erro de classificação. O inverso também acontece, quanto menor a distribuição das classes, menor a impureza e o erro.

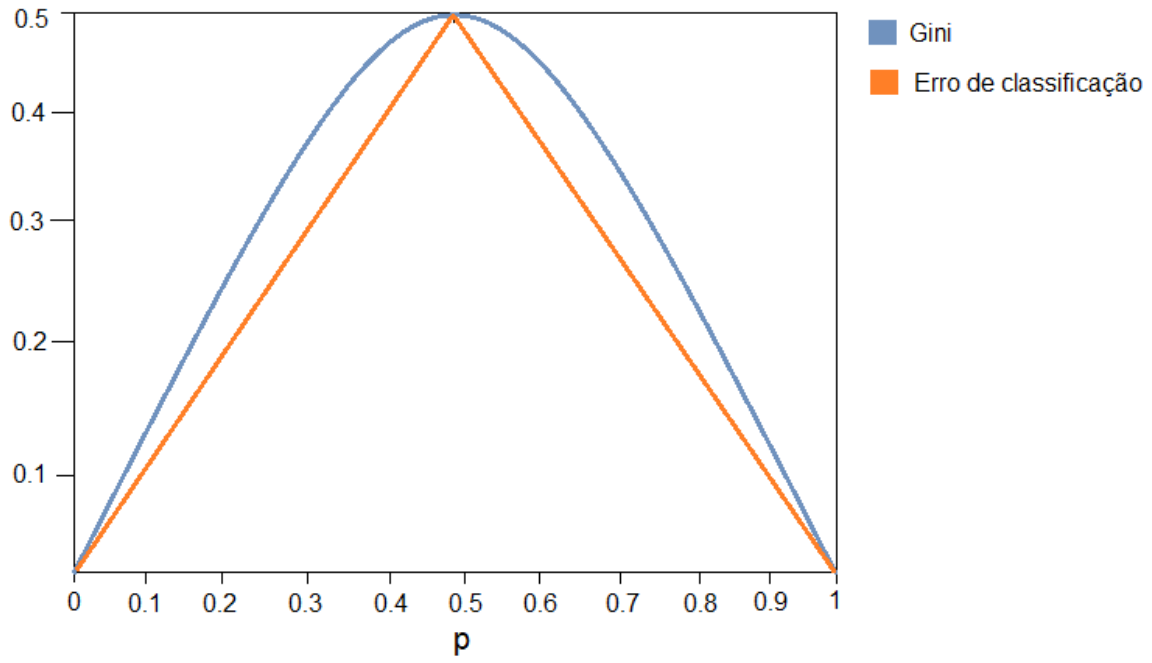


Figura 2.8: Medida de impureza de Gini e Erro de classificação de acordo com a distribuição de classes.

2.4.1.2 Métodos *ensemble*

Outra técnica de classificação bastante utilizada são os métodos *ensemble*, por ter uma melhor performance do que classificadores simples, como a árvore de decisão. Esta técnica tem um melhor desempenho, pois agregam as predições de múltiplos classificadores, ao invés de predizer os rótulos de classe de exemplos desconhecidos usando um único classificador induzido por dados de treinamento. Um método *ensemble* ou método de combinação de classificador, constrói um conjunto de classificadores base a partir de dados de treinamento e executa a classificação, dando as devidas importâncias as previsões feitas por cada classificador base [44].

O classificador *ensemble* prevê o rótulo de classe de um exemplo de teste, obtendo uma votação majoritária sobre as previsões feitas pelos classificadores de base. Logo, se os classificadores base forem idênticos, o conjunto classificará erroneamente os mesmos exemplos previstos incorretamente pelos classificadores de base, da mesma forma que classificadores simples. Por outro lado, se os classificadores base forem independentes, isto é, se os erros não estiverem correlacionados, o conjunto fará uma previsão incorreta somente se mais da metade dos classificadores base preverem incorretamente. Este é o principal fator que faz com que a performance desse método seja melhor do que a dos classificadores simples [45].

Como citado anteriormente, a ideia básica desse método é construir múltiplos classificadores a partir dos dados originais de treinamento e depois agregar suas previsões ao classificar exemplos desconhecidos, a lógica básica é apresentada na Figura 2.9 (adaptada de [14]). O conjunto de classificadores pode ser construído ao manipular o conjunto de treinamento e é com base no tipo da manipulação que são construídos os algoritmos para esse método. Por terem um bom desempenho na construção de classificadores, os dois principais algoritmos deste método são o *Adaboost* e o *random forest* [45].

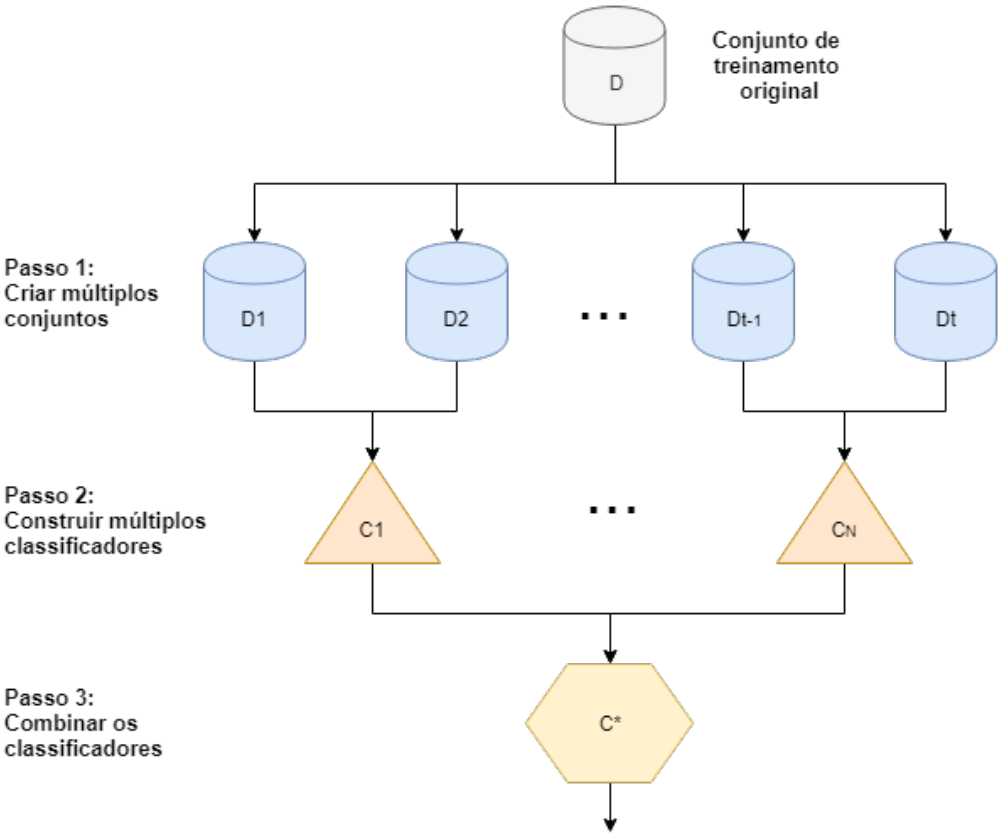


Figura 2.9: Lógica básica do método de aprendizagem *ensemble*.

2.4.1.2.1 Adaboost

O *Adaboost* é um algoritmo que utiliza a manipulação do conjunto de treinamento para construir a base de classificadores, conforme comentado anteriormente. Nessa abordagem, vários conjuntos de treinamento são criados a partir do conjunto de treinamento original. Dessa forma, é criado um procedimento iterativo usado para alterar de forma adaptativa a distribuição dos novos conjuntos de treinamento para que os classificadores de base se concentrem em exemplos difíceis de classificar. Este procedimento iterativo atribui um

peso a cada conjunto de treinamento e pode alterá-los de forma adaptativa no final de cada iteração. Inicialmente, para cada conjunto de treinamento é atribuído o peso de $1/N$, onde N é o número de conjuntos, para que todos sejam escolhidos para serem treinados inicialmente [46].

Para definir a importância de um classificador e o peso de um conjunto de treinamento, considere que $\{(x_j, y_j)|j = 1, 2, ..., N\}$ denote um número N de conjunto de treinamento, atributos de entrada x e rótulos de classe, y . A importância de um classificador base C_i depende da sua taxa de erro, que é definida por [46]:

$$\in_i = \frac{1}{N}[\sum_{j=1}^{N} w_j I(C_i(x_j) \neq y_j)], \tag{2.6}$$

onde $I(p) = 1$ se o predicado p for verdadeiro e 0, caso contrário. Isto é, se o classificador atribuir um rótulo ao atributo x que não é o rótulo certo y, o valor é 1 e seu erro não é zerado neste conjunto de treinamento. Então, a importância de uma classificador é dada da forma [46]:

$$\alpha_i = \frac{1}{2} ln(\frac{1- \in_i}{\in_i}), \tag{2.7}$$

Observe que α_i tem um valor positivo grande se a taxa de erro estiver próxima de 0 e um valor negativo grande se a taxa de erro estiver próxima de 1.

O parâmetro α_i também é usado para atualizar o peso dos conjuntos de treinamento. Para ilustrar, considere que $w_i^{(j)}$ denote o peso atribuído ao exemplo (x_i,y_i) durante j-ésima iteração. O mecanismo de atualização do peso é dado por [46]:

$$w_i^{j+1} = \frac{w_i^{(j)}}{Z_j} x \left\{ \begin{array}{c} exp^{-\alpha_j} \text{ if } C_j(x_i) = y_i \ , \\ exp^{\alpha_j} \text{ if } C_j(x_i) \neq y_i \end{array} \right\}, \tag{2.8}$$

no qual Z_j é o fator de normalização para garantir que $\sum_i w_i^{j+1} = 1$.

2.4.1.2.2 Exemplo de utilização do Adaboost

Para ilustrar como o *Adaboost* escolhe os conjuntos de treinamento a partir dos seus respectivos pesos, considere que a partir do conjunto de treinamento original foram gerados 5 novos conjuntos (de 1 a 5) e que foi escolhido para o procedimento iterativo ter 3 iterações. Dessa forma, os pesos dos conjuntos de treinamento são atualizados no final de cada iteração. Conjuntos que são classificados incorretamente terão seus pesos aumentados, enquanto aqueles que são classificados corretamente terão seus pesos diminuídos. A Tabela 2.1 mostra os conjuntos escolhidos após cada interação.

Tabela 2.1: Exemplo de conjuntos de treinamentos escolhidos a cada iteração.

Iteração 1	1	2	3	4	5
Iteração 2	4	4	3	3	1
Iteração 3	1	4	3	5	2

Inicialmente, todos os exemplos contêm o mesmo peso de 1/5 e são escolhidos para serem treinados pelo primeiro classificador base (primeira iteração). Após ser treinado, o classificador testa a classificação em cada conjunto de treinamento escolhido. E supondo que os conjuntos 4 e 5 são difíceis de serem classificados, eles tomam o lugar de conjunto que foram mais fáceis na segunda iteração. Na terceira iteração, é feita a mesma coisa e no final, temos um conjunto de classificadores que se juntados se tornam um único classificador forte. E cada classificador tem o seu peso de importância na hora da junção desses classificadores, logo a previsão final do classificador *ensemble* é obtida tomando uma média ponderada das previsões feitas por cada classificador de base [14].

2.4.1.2.3 Random Forest

O *random forest* também é um algoritmo que utiliza a manipulação do conjunto de treinamento para construir a base de classificadores, porém ele combina as predições feitas por múltiplas árvores de decisão, onde cada árvore é gerada com base nos atributos de conjuntos aleatórios. Esses conjuntos aleatórios são gerados de uma distribuição de probabilidade fixa, ao contrário da abordagem adaptativa do AdaBoost, onde a distribuição de probabilidade é variada para focar nos exemplos que são difíceis de classificar [47].

Nesta abordagem, a aleatoriedade é injetada no processo de construção do modelo escolhendo aleatoriamente N amostras, com substituição, do conjunto de treinamento original. A força de um conjunto de classificadores refere-se ao desempenho médio dos classificadores, no qual o desempenho é medido probabilisticamente em termos da margem do classificador [47]:

$$\text{margem}, M(X, Y) = P(Y_\theta = Y) - \max_{Z \neq Y} P(Y_\theta = Z), \tag{2.9}$$

onde Y_θ é a classe prevista de X de acordo com um classificador construído a partir de alguns conjuntos aleatórios. Quanto maior a margem, maior a probabilidade de que o classificador preveja corretamente um determinado exemplo X. Cada árvore de decisão usa um conjunto de treinamento aleatório. Para combinar o conjunto de classificadores e gerar o classificador final, as previsões são combinadas usando um esquema de votação majoritária. No qual, a maior quantidade de predições feitas, por cada árvore de decisão, é usada para determinar o rótulo de um atributo X, é escolhida como o rótulo para a predição final.

2.4.2 Aprendizado Não-Supervisionado

Na aprendizagem não-supervisionada, do inglês "unsupervised learning", o objetivo é aprender um mapeamento de entrada para uma saída sem um supervisor, diferentemente dos algoritmos com aprendizagem supervisionada que assumem a existência de um supervisor ou de uma medida de adequação para classificação de exemplos de treinamento. Este tipo de aprendizado não possui a vantagem de um ambiente de treinamento com casos para calibração de um modelo de classificação [14]. Ao invés disso, os algoritmos não-supervisionados propõem hipóteses para explicar as observações a partir de informações encontradas nos dados que descrevem os objetos e seus relacionamentos. Logo, o objetivo é que os objetos dentro de um grupo sejam similares (ou relacionados) uns aos outros e diferentes (ou não relacionados a) os objetos em outros grupos [14].

2.4.2.1 Algoritmo k-means

O algoritmo de análise de agrupamento, k-means, é um do tipo não-supervisionado e um dos mais conhecidos e utilizados, por ser um algoritmo simples e que proporciona resultados efetivos em diversas aplicações de aprendizado não-supervisionado. Além disso, ele é muito utilizado por ser o que possui o maior número de variações, tornando-o flexível para ser utilizado em várias aplicações [48].

O k-means se trata de um algoritmo de clusterização ou agrupamento que visa definir um centróide para cada k agrupamentos a serem feitos. E a partir da definição de cada centróide e suas respectivas características, o objetivo é classificar os dados que mais assemelham suas características com um determinado centróide. Dessa forma, é formado um agrupamento ou cluster de dados com um conjunto de características único definidos pelo centróide [49].

A partir da escolha de k centróides iniciais, isto é, o número de *clusters* desejados, cada objeto do conjunto de dados é atribuído ao centróide mais próximo a ele. O centróide de cada *cluster* é atualizado com base nos pontos atribuídos a ele. Após isso, é repetida a atribuição dos objetos e os centróides são atualizados, até que nenhum ponto altere os *clusters*, ou equivalentemente, até que os centróides permaneçam os mesmos [14].

Um exemplo de operação do k-means é dado na Figura 2.10 (adaptada de [14]), no qual é mostrado que, começando a partir de três centróides, os clusters finais são encontrados em quatro etapas de atualização de atribuição. Essas etapas (iterações) exibem o agrupamento k-means e cada etapa mostra como estavam os centróides no início da iteração e a atribuição dos pontos a esses centróides na iteração subsequente. Os centróides são indicados pelo símbolo "+" e todos os pontos pertencentes ao mesmo cluster, têm a mesma forma (círculo, triângulo ou quadrado) e mesma cor.

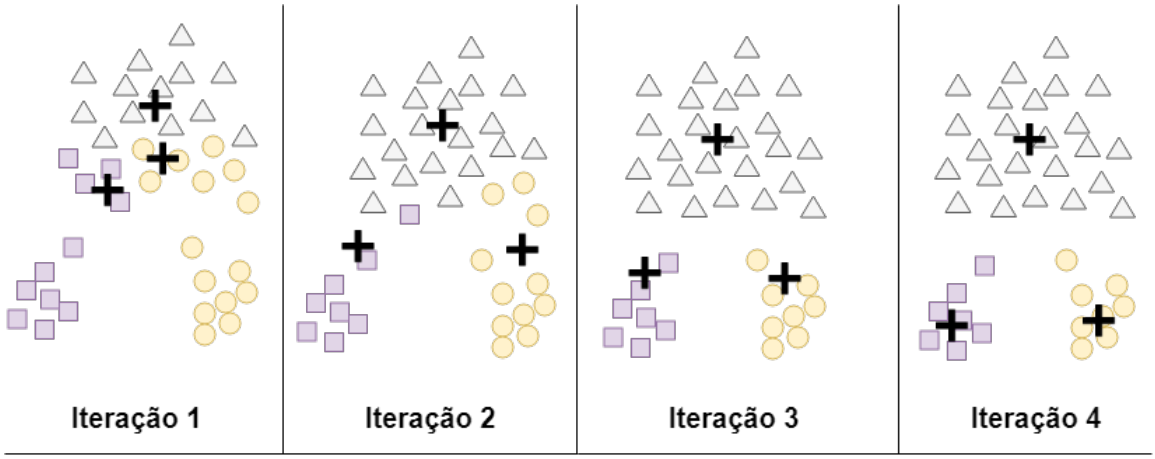


Figura 2.10: Utilizando o k-means para achar três *clusters* em um conjunto de dados de exemplo.

Dessa forma é possível ver que, ainda na Figura 2.10, na primeira iteração, os pontos são atribuídos aos centróides iniciais, que estão todos no maior grupo de pontos. Para este exemplo, é usada a média dos pontos como o centróide. Depois que os pontos são atribuídos a um centróide, o centróide é atualizado de lugar. Na segunda etapa, os pontos são atribuídos aos centróides atualizados e os centróides são atualizados novamente. A partir desta etapa até a iteração final, dois dos centróides se movem para os dois pequenos grupos de pontos na parte inferior das áreas. Quando o algoritmo k-means termina na quarta iteração, porque não ocorrem mais mudanças, os centróides identificaram os agrupamentos naturais de pontos.

Para atribuir um ponto ao centróide mais próximo, é preciso de uma medida de proximidade que quantifique a noção de “mais próximo"para os dados específicos em consideração. A distância euclidiana (L2) é frequentemente usada para pontos de dados no espaço euclidiano. Então, a distância euclidiana entre dois pontos é [50]:

$$d(x, y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}, \tag{2.10}$$

onde n é o número de dimensões do espaço euclideano dos dados e x_k e y_k são, respectivamente, os k-ésimos atributos de x e y. Logo, quanto menor a distância d entre dois pontos x e y, mais próximos ou mais relacionados estão os dados. A partir desta medida de proximidade, é possível calcular a distância entre cada ponto no conjunto de dados e os centróides e, assim, atribuir este ponto a um *cluster* que tem uma menor distância entre um centróide, entre todos os outros centróides.

Além disso, o algoritmo k-means tem uma função objetiva que visa minimizar a soma do erro quadrado, do inglês “sum of the squared error” (SSE). Essa função objetiva mede

a qualidade de um cluster, em outras palavras, é calculado o erro de cada ponto de dados, isto é, a distância euclidiana ao centróide mais próximo de cada ponto e, depois, é calculada a soma total dos erros quadrados. Dados dois conjuntos diferentes de clusters que são produzidos por duas execuções diferentes do k-means, preferimos aquele com o menor erro quadrado, pois isso significa que os centróides dessa clusterização são uma representação melhor dos pontos em seu respectivo cluster [48]. O SSE é definido da forma:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)^2, \tag{2.11}$$

onde $dist$ é distância Euclidiana, K é o número de clusters, c_i é o centróide de um i-ésimo *cluster* C_i e x um objeto do conjunto de dados. E o centróide de um i-ésimo cluster é dado por:

$$C_i = \frac{1}{m_i} \sum_{x \in C_i} x, \tag{2.12}$$

2.4.2.1.1 Exemplo de utilização do k-means

Para ilustrar, dado um centróide de um único cluster contendo três pontos de duas dimensões, (1,1), (2,3) e (6,2), o seu centróide e o SSE são:

$c_1 =((1 + 2 + 6)/3 , (1 +3 + 2)/3) = (3,2)$,

$SSE = [(1-3)^2 + (1-2)^2] + [(2-3)^2 + (3-2)^2] + [(6-3)^2 + (2-2)^2] = 16.$

2.4.3 Aprendizado por reforço

A aprendizagem por reforço, do inglês "reinforcement learning", se preocupa com a forma como os modelos gerados, chamados de agentes autônomos, devem tomar ações em um ambiente, de modo a escolher as melhores ações para atingir seu objetivo. O problema, devido à sua generalidade, é estudado em muitas outras disciplinas, como teoria dos jogos, para o modelo aprender a jogar jogos de tabuleiros, robótica, para o algoritmo aprender a controlar um robô móvel, entre outras disciplinas. Cada vez que o agente executa uma ação em seu ambiente, o algoritmo tenta maximizar alguma noção de recompensa acumulativa para as ações corretamente feitas. Porém, há a penalização para ações feitas erroneamente. Dessa forma, é possível indicar a conveniência do estado resultante para o agente (modelo) [51].

Neste aprendizado, o conceito de maximizar a recompensa através do desempenho é feito por um processo de ajuste dos parâmetros feitos pela interação contínua com o ambiente. Dessa forma, ao invés de haver um supervisor que indique o resultado esperado

a cada estímulo fornecido como entrada, existe um procedimento que atribui uma nota para a resposta da máquina de aprendizado ao estímulo, com o objetivo de alcançar o nível máximo de sucesso no seu funcionamento com base em um índice estabelecido [51].

Logo, na aprendizagem por reforço, o agente aprende com uma série de reforços, que são punições ou recompensas. Um exemplo seria o ponto para uma vitória no final de um jogo de damas, que significa que o agente que executou certas ações que culminaram em um resultado final correto. Porém, a falta de gorjeta no final da viagem, dá ao agente de táxi uma indicação de que ele fez algo de errado. Cabe ao agente decidir quais das ações anteriores ao reforço foram as mais indicadoras para que ele alcance o sucesso [52].

2.5 Métricas de avaliação e performance

Um modelo gerado por um algoritmo de AM deve ajustar bem os dados de entrada e prever corretamente os rótulos de classe de registros que não vistos anteriormente. Portanto, o objetivo primordial dos algoritmos de AM é construir modelos com boa capacidade de generalização, isto é, modelos que preveem com precisão os rótulos de classe de registros anteriormente desconhecidos. Essa precisão ou a avaliação do desempenho de um modelo é baseada nas contagens de registros de teste, correta e incorretamente, previstas pelo modelo. Essas contagens são tabuladas em uma tabela conhecida como matriz de confusão, do inglês "confusion matrix" [53].

A Tabela 2.2 descreve a matriz de confusão para um problema de classificação binária. Cada entrada f_{ij} nesta tabela denota o número de registros da classe i previstos como da classe j. Por exemplo, f_{11} é o número de registros da classe 1 previstos corretamente como a classe 1 e cada registro previsto de forma correta é chamado de verdadeiro positivo, do inglês true positive (TP). E o f_{00} é o número de registros da classe 0 previstos corretamente como a classe 0 e cada registro previsto é chamado de verdadeiro negativos, do inglês "True negative" (TN). Já o f_{10} é o número de registros da classe 1 previstos incorretamente como a classe 0, e cada registro previsto é chamado de falso negativo, do inglês "False negative" (FN). E, por fim, o f_{01} é o número de registros da classe 0 previstos incorretamente como classe 1, onde cada registro previsto é chamado de falso positivo, do inglês "False positive" (FP) [14].

Embora uma matriz de confusão forneça as informações necessárias para determinar o desempenho de um modelo de classificação, resumir essas informações com um único número tornaria mais conveniente comparar o desempenho de diferentes modelos. Isso pode ser feito usando uma métrica de desempenho, como a acurácia, que é definida da seguinte maneira [53]:

Tabela 2.2: Matriz de confusão de exemplo para um problema de duas classes.

		Previsto	
		Classe = 1	Classe = 0
Real	Classe = 1	f_{11} (TP)	f_{10} (FN)
	Classe = 0	f_{01} (FP)	f_{00} (TN)

$$\text{Acurácia} = \frac{\text{Número de predições corretas}}{\text{Número total de predições}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{2.13}$$

Equivalentemente, o desempenho de um modelo pode ser expresso por sua taxa de erro, que é dada pela seguinte equação [53]:

$$\text{Taxa de erro} = \frac{\text{Número de predições erradas}}{\text{Número total de predições}} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{2.14}$$

Porém, a acurácia trata todas as classes como igualmente importantes, e ela não é adequada para analisar conjuntos de dados desequilibrados, em que a classe rara é considerada mais interessante do que a classe majoritária. Para isso, foram desenvolvidas duas novas métricas chamadas de precisão e revocação, do inglês "precision" e "recall", respectivamente. A revocação e a precisão são duas métricas amplamente utilizadas em aplicações em que a determinação bem-sucedida de uma das classes é considerada mais significativa do que a detecção das outras classes [14]. Uma definição formal dessas métricas é fornecida abaixo [53]:

$$\text{Precisão} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{2.15}$$

$$\text{Revocação} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{2.16}$$

A precisão determina a fração de registros que realmente são de classes positivas no grupo em que o classificador declarou como uma classe positiva. Quanto maior a precisão, menor o número de erros falsos positivos cometidos pelo classificador. Revocação mede a fração de exemplos positivos corretamente previstos pelo classificador. Classificadores com grande valor de revocação têm poucos exemplos positivos classificados erroneamente como de classe negativa [14].

Geralmente, é possível construir modelos de linha de base que maximizam uma métrica, mas não a outra. Por exemplo, um modelo que declara que cada registro é da classe positiva terá uma recordação perfeita, mas uma precisão muito baixa. Então, construir

um modelo que maximize a precisão e a revocação é o principal desafio dos algoritmos de classificação. Essas duas métricas podem ser resumidas em outra conhecida como medida de F_1, do inglês "F_1 measure", conforme a seguir [53]:

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}, \tag{2.17}$$

Dessa forma, conseguir um valor alto da medida de F_1 é o objetivo dos algoritmos classificadores, pois essa métrica é a relação entre as duas principais métricas de desempenho dos modelos classificatórios. Logo, a partir do momento em que F_1 tem um valor alto, os valores de precisão e de revocação também são igualmente altos [54].

Capítulo 3

Framework de EDM Proposto

Este capítulo apresenta o *framework* de EDM proposto, que tem como objetivo analisar os dados dos cursos de EAD da Enap, afim de encontrar variáveis que influenciem o desempenho dos participantes, permitindo a melhora na qualidade desses cursos. Além disso, esse *framework* pode ser bastante útil para reduzir as taxas de reprovação. Para isso, foi feito um diagrama de blocos com as etapas necessárias para chegar no resultado final do *framework* proposto.

Conforme pode ser visto na Figura 3.1, a primeira etapa (Bloco A) é a base de dados da Enap contendo todo o conjunto de dados registrados por ações de alunos, tutores, informações de cursos etc, e esta etapa será explicada na seção 3.1. O segundo passo (Bloco B) é o processo de realizar as consultas no banco de dados da Enap para conseguir extrair os dados que serão efetivamente utilizados, todo este processo será explicado na seção 3.2. A terceira etapa (Bloco C) é a etapa do pré-processamento dos dados, que será explicada na seção 3.3. O quarto passo (Bloco D) é a utilização dos algoritmos de AM para minerar os dados, que será explicado na seção 3.4.

Ademais, a seção 3.5 explica como foi feito o processo para viabilizar a análise de desistências nos cursos, a partir do entendimento da base de dados da Enap (seção 3.1) e das consultas para extração dos dados (seção 3.2).

3.1 Base de Dados da Enap

Entre as principais plataformas educacionais, o Moodle, acrônimo de "Modular Object Oriented Dynamic Learning Environment", é um sistema de gerenciamento de aprendizado gratuito e de código aberto, desenvolvido colaborativamente pela comunidade virtual que suporta a expansão de um único ambiente de ensino para professores, participantes e administradores, sendo um sistema seguro, robusto e integrado [55].

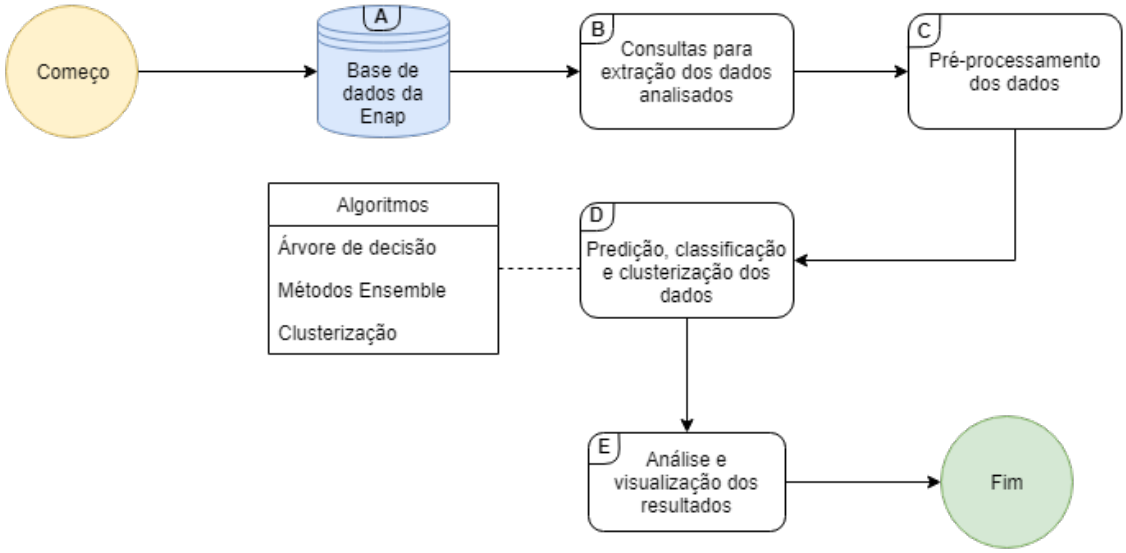


Figura 3.1: Diagrama de blocos do framework de EDM proposto.

Tendo em vista que o AVA utilizado pela Enap é o Moodle, foi necessário entender toda a lógica por trás do modelo entidade relacionamento (MER) dele, pois é este modelo que descreve os dados ou os aspectos do domínio de informação de um negócio de uma maneira abstrata, porém compreensível a ponto de viabilizar consultas e extração dos dados desejados.

Como o Moodle disponibiliza o seu código para toda a comunidade de forma gratuita, foi possível retirar informações sobre o seu MER na sua documentação, disponível *online* e no Apêndice A. Logo, ao analisá-lo foi descoberto que haviam 380 tabelas e aproximadamente 1000 relacionamentos entre as tabelas. Isso o configura como um modelo altamente complexo, porém de bastante diversidade de tipos de dados, permitindo diversas atuações da mineração de dados nele. Outra descoberta foi que haviam 14 entre as 380 tabelas que eram imprescindíveis para utilização do software pelos participantes, tutores e responsáveis pelo gerenciamento e oferta dos cursos.

A Tabela 3.1 descreve a função de cada uma delas e nelas estão presentes a maioria das funcionalidades que os participantes e tutores podem fazer durante a ministração de um curso. Nela, onde está escrito tarefas essenciais, significa os exercícios avaliativos durante o curso e de cada módulo, mini-casos de exercícios avaliativos, exercícios de fixação, testes de conhecimentos. Onde tem tarefas complementares, são as atividades práticas, atividades em gincanas, atividades em fóruns. E sobre a nota final do aluno, para ele ser aprovado, a sua nota deve ser maior ou igual a 60. Caso contrário, ele estará reprovado no curso.

Foi disponibilizado para este trabalho, a base de dados do Moodle, utilizado pela

Tabela 3.1: Função de cada tabela no modelo entidade relacionamento do Moodle.

Nome	**Função**
mdl_user	Armazenar todas as informações dos usuários
mdl_assign	Conter informações de tarefas complementares
mdl_assign_grades	Armazenar as notas das tarefas complementares
mdl_chat	Conter a troca de mensagens dos participantes
mdl_course	Armazenar as informações dos cursos
mdl_course_categories	Conter as informações de categorias de um curso
mdl_course_modules	Armazenar as informações dos módulos de um curso
mdl_forum_posts	Conter as informações de postagens em fóruns dos alunos
mdl_forum_read	Armazenar as informações de leitura de fóruns dos alunos
mdl_grade_grades	Conter as notas finais dos alunos em todos os cursos
mdl_quiz	Armazenar as informações de atividades essenciais
mdl_quiz_grades	Conter as notas das tarefas essenciais
mdl_enrol_unenrol	Guardar o cancelamento da inscrição do aluno em um curso
mdl_logstore_standard_log	Conter as informações de *log* dos usuários

Enap, durante a oferta de todos os cursos nos anos de 2015 a 2016. Essa base de dados é bastante extensa com aproximadamente 30 GB e 1 milhão de registros. Isso permite que este trabalho tenha uma fundamentação e embasamento significativo, por se tratar, principalmente, de dados reais e por ter uma grande quantidade de informações.

Essa grande quantidade de dados, também, torna o resultado do *framework* proposto confiável, pois uma grande massa de dados significa que certas características dela, são características gerais e não isoladas, como pode acontecer ao analisar um banco de dados com poucos registros. Porém, a desvantagem é que grandes massas de dados, do inglês *Big Data*, como essa, geram um custo computacional muito alto, o que acaba tornando a aplicação de mineração de dados mais demorada que o usual [56].

3.2 Consultas para extração dos dados

Como os dados do Moodle foram armazenados em um banco de dados relacional, mais precisamente no banco de dados PostgreSQL [57], foi preciso utilizar a linguagem de consulta estruturada, do inglês *Structured Query Language* (SQL), para realizar as consultas e extrair os dados desejados em arquivos no formato de valores separados por vírgula, do inglês *Comma-separated values* (CSV). Com o estudo da estrutura de dados do Moodle na seção 3.1, foi realizada uma consulta para pegar informações dos participantes, tutores, cursos e afins. Parte dela pode ser vista na Figura 3.2, e ela por completo se encontra no Apêndice B.

```sql
SELECT u.id as userid,u.firstname,u.lastname,u.email,u.city,u.country,c.fullname, c.id as course_id, g.finalgrade,
case
    when  g.finalgrade > 60.0 then 1
    else 0
end as passed,
case
    when  (SELECT COUNT(compl.id) AS countrecord FROM mdl_course_modules_completion compl
INNER JOIN mdl_course_modules m ON compl.coursemoduleid = m.id
WHERE compl.userid=u.id AND m.course=c.id AND m.completion > 0 AND compl.completionstate > 0) >
(SELECT COUNT(cm.id) as num_modules
FROM mdl_course_modules cm
INNER JOIN mdl_modules m ON m.id=cm.module
WHERE  cm.course=c.id)/1.1  then 1
    else 0
end as ninety_modules_done,
```

Figura 3.2: Parte da consulta realizada para extrair 19 variáveis do Moodle.

Esta consulta utilizou as 14 tabelas do banco de dados descritas na Tabela 3.1 e retornou os dados de 144.719 notas finais distribuídas em 76.551 participantes de um total de 351 cursos, observe que o mesmo participante pode participar de mais de um curso. Deste total de notas finais, foi descoberto que haviam 113.419 aprovações e 31.570 reprovações. E para os 351 cursos, haviam 856 tutores. Ademais, como pode ser visto na Figura 3.3, foi descoberto que havia participantes de 9 outros países, além do Brasil. Isso mostra a relevância dos cursos de EAD, que, mesmo longe das origens das instituições de ensino, os alunos têm acesso ao ensino e à informação, permitindo que o cenário educacional se torne mais acessível para o mundo inteiro.

Além disso, na consulta realizada na Figura 3.2, foi aplicada uma inspeção subjetiva, que retornou 17 variáveis propostas e 2 variáveis padrões da base de dados do Moodle, totalizando 19 variáveis. A descrição de cada uma delas pode ser vista na Tabela 3.2 e o objetivo de retorná-las foi para a utilização delas nos algoritmos de classificação, predição e clusterização dos dados. Nessa tabela, a variável "ninety_modules_done" é um variável binária, no qual se o aluno tiver feito 90 % dos módulos do curso, a variável é igual a 1 e, caso contrário, é igual a 0. O mesmo ocorre para as variáveis "sixty_modules_done" e "thirty_modules_done", porém com suas respectivas porcentagens. As variáveis "num_modules_done" e "num_modules_have" foram normalizadas para terem valores de 0 a 30, e, assim, terem valores consistentes.

A variável "finalgrade" não está na Tabela 3.2, pois ela é uma variável apenas para indicar a nota final do aluno no curso e, consequentemente, se ele foi aprovado ou reprovado. Logo, ela só será utilizada como métrica para saber se o algoritmo acertou na classificação, predição ou clusterização de um registro. E para o aluno ser aprovado, ele precisa ter uma nota final (finalgrade) maior ou igual a 60. Caso contrário, ele estará reprovado. Um fato importante de ser salientado é que a média nas notas das tarefas (*mean_quiz_grade*, *grade_mean_assign*), não fazem parte da composição da nota final

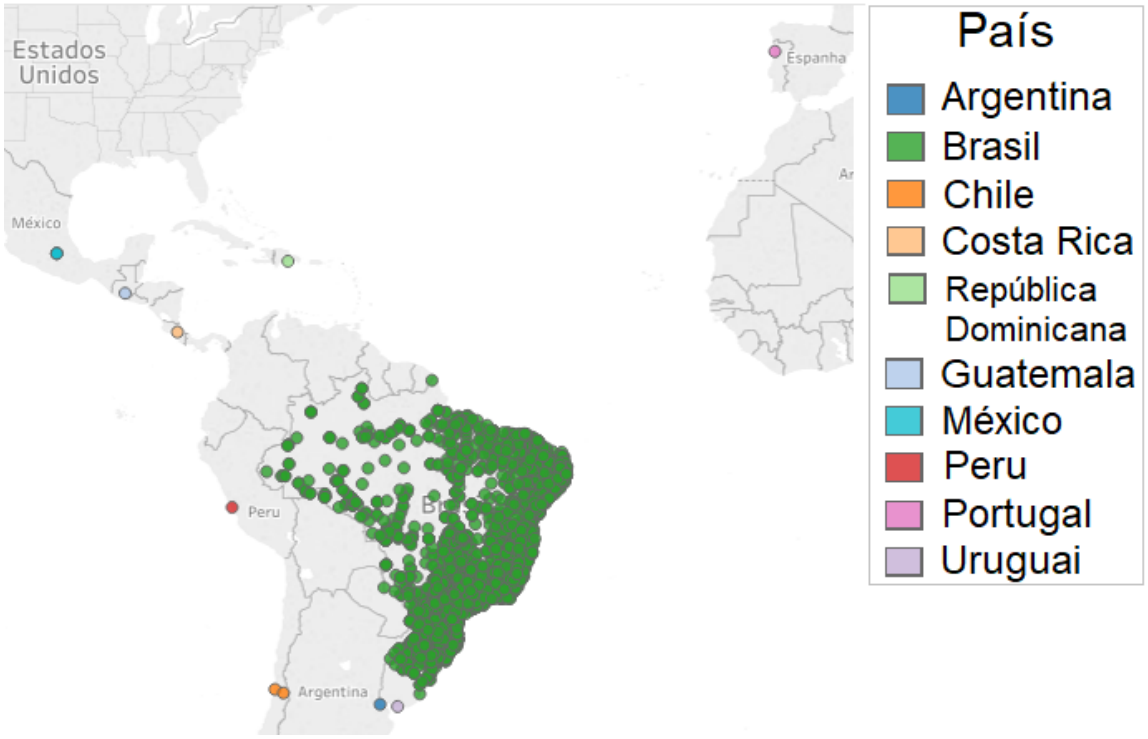


Figura 3.3: Distribuição geográfica dos participantes dos cursos da Enap de 2015 a 2016.

no curso (*finalgrade*). A nota final do aluno é dada pela nota em uma avaliação que ele faz ao final do curso.

No trabalho [8], foi feita, também, a utilização de algumas variáveis para a predição das performances dos alunos de somente um curso, enquanto que neste trabalho foi feita a predição e classificação para todos os cursos. Além disso, neste trabalho anterior foram utilizadas 7 variáveis completamente diferentes das usadas neste trabalho, como pode ser visto na Tabela 3.3. Nesta tabela, é possível visualizar, também, que todas essas 7 variáveis são variáveis padrão do Moodle.

Outra consulta realizada foi para extrair informações sobre os participantes que desistiram dos cursos, como por exemplo, o número de módulos feitos, pelo participante, até o momento da desistência e o período em que ela ocorreu. Além dessas informações, foram recuperados os números de módulos, número de tutores e período de duração dos cursos. Tudo isso foi feito para uma posterior análise geral das desistências nos cursos. A consulta realizada pode ser vista em parte na Figura 3.4 e completa no Âpendice C.

Tabela 3.2: Novas variáveis propostas e padrão do Moodle para utilizar nos algoritmos subsequentes.

Variável	**Descrição**	**Tipo**
mean_quiz_grade	Média das notas das tarefas essenciais	Proposta
count_quiz_made	Número de tarefas essenciais feitas	Proposta
grade_mean_assign	Média das notas das tarefas complementares	Proposta
count_assign_made	Número de tarefas complementares feitas	Proposta
num_act_user	Número de ações de um participante em um curso	Proposta
num_act_tutors	Número de ações de um tutor em um curso	Proposta
ninety_modules_done	Se o aluno fez 90 % dos módulos do curso	Proposta
sixty_modules_done	Se o aluno fez 60 % dos módulos do curso	Proposta
thirty_modules_done	Se o aluno fez 30 % dos módulos do curso	Proposta
num_modules_done	Número de módulos feitos pelo participante no curso	Proposta
num_modules_have	Número de módulos do curso	Padrão
std_quiz_grade	Desvio padrão das notas das tarefas essenciais	Proposta
std_dev_assign_grade	Desvio padrão das notas das tarefas complementares	Proposta
num_forum_posts	Número de publicações feitas em fóruns pelo participante	Proposta
num_forum_discussions	Número de participações em discussões de fóruns	Proposta
num_forum_sub	Número de inscrições em fóruns	Proposta
num_forum_reads	Número de visualizações de fóruns	Proposta
num_chats	Número de conversas feitas pelo participante	Proposta
num_tutors	Número de tutores em um curso	Padrão

Tabela 3.3: Variáveis utilizadas no trabalho anterior.

Variável	**Descrição**	**Tipo**
count_quiz_view	Número de visualizações das tarefas essenciais	Padrão
count_quiz_submmited	Número de submissões enviadas para as tarefas essenciais	Padrão
count_assign_submitted	Número de tentativas feitas nas tarefas complementares	Padrão
count_quest_view	Número de visualizações em questionários	Padrão
count_quest_submitted	Número de submissões enviadas para os questionários	Padrão
count_forum_uploaded	Número de *uploads* em fóruns	Padrão
count_forum_view	Número de visualizações em fóruns	Padrão

```
SELECT u.id as userid, u.firstname, u.lastname, c.fullname, c.id as courseid,

(SELECT COUNT(compl.id) AS countrecord FROM mdl_course_modules_completion compl
INNER JOIN mdl_course_modules m ON compl.coursemoduleid = m.id
WHERE compl.userid=u.id AND m.course=c.id AND m.completion > 0 AND compl.completionstate > 0) as num_modules_done,

(SELECT COUNT(cm.id) as num_modules
FROM mdl_course_modules cm
INNER JOIN mdl_modules m ON m.id=cm.module
WHERE  cm.course=c.id) as num_modules_have

(SELECT COUNT(us.id) as num_tutores FROM mdl_role_assignments rs
INNER JOIN mdl_user us ON us.id=rs.userid
INNER JOIN mdl_context e ON rs.contextid=e.id
INNER JOIN mdl_course cs ON cs.id=e.instanceid
WHERE e.contextlevel=50 AND rs.roleid!=5 AND cs.id=c.id
GROUP BY cs.id) as num_tutors,
```

Figura 3.4: Parte da consulta feita para extrair os dados dos alunos desistentes.

3.3 Pré-processamento dos dados

A etapa de pré-processamento deve ser aplicada para tornar os dados mais adequados para a mineração de dados. O pré-processamento é uma área ampla que consiste em várias estratégias e técnicas diferentes, que estão inter-relacionadas de maneiras complexas. E as etapas envolvidas no pré-processamento de dados incluem, principalmente, a limpeza de dados para remover insconsistências (seção 3.3.1), como observações duplicadas, e a seleção de recursos relevantes para a tarefa de mineração de dados em questão (seção 3.3.2) [14].

3.3.1 Remoção de inconsistências

Ao analisar os dados extraídos na seção 3.2, foi possível notar algumas inconsistências que precisavam ser resolvidas. Para isto, foi feito um diagrama de blocos com as etapas necessárias para a resolução de todas as insconsistências, como pode ser visto na Figura 3.5. A primeira das inconsistências (Bloco A) foram os registros que tinham as notas finais vazias, e isto se deve ao fato da configuração do curso, o campo da nota final ser balizada como não obrigatório, o que leva ao não preenchimento do campo. A solução foi remover os registros com notas vazias e a remoção destes registros foi de 0,00448 % do total. O segundo problema encontrado (Bloco B) foram as notas negativas, que também pode ser explicado por meio do momento da configuração, a nota mínima foi definida como um número negativo. Novamente, para solucionar isso, foi feita a remoção destes registros, que resultou numa remoção de 0,0000898 % do total. Por fim, a última inconsistência encontrada (Bloco C), foram as notas maiores que 100, que também pode ser explicada pela configuração da nota máxima ser maior do que 100. E para solucionar isso, foi feita

a normalização das notas no intervalo de 0 a 100. Ao final, o total de remoção dos dados foi de aproximadamente de 0,004 %.

Portanto, a quantidade de dados removidos é insignificante para degradar o desempenho dos algoritmos EDM, porque, de acordo com [14], a porcentagem de remoção de dados que pode afetar o desempenho dos algoritmos EDM é de aproximadamente 15 % ou mais. No entanto, esta etapa é necessária para transformar os dados brutos de entrada em um formato apropriado para execução e análise de algoritmos EDM subsequentes.

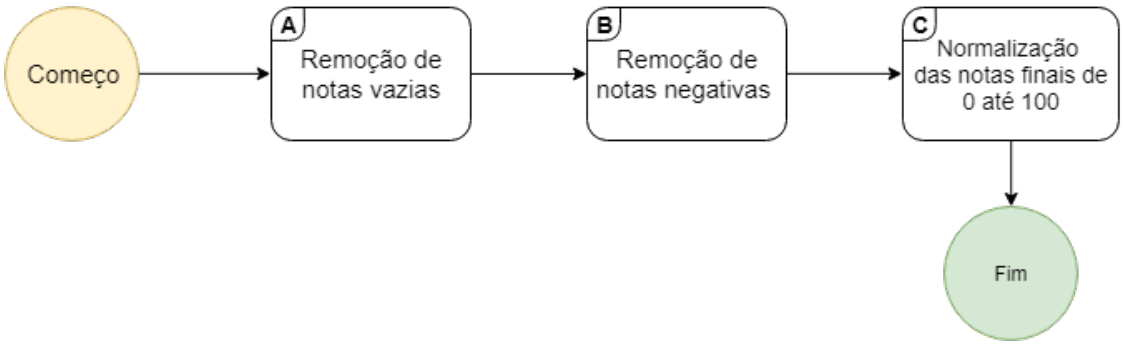


Figura 3.5: Diagrama de blocos da remoção de inconsistência dos dados.

3.3.2 Seleção das variáveis de maiores influências

A seleção de recursos ou variáveis de maiores influências é um benefício importante, pois muitos algoritmos de mineração de dados funcionam melhor se o número de variáveis (dimensionalidade), nos dados for menor. Isso ocorre porque a redução da dimensionalidade pode eliminar recursos irrelevantes e reduzir o ruído, gerado por uma alta dimensionalidade. Outro benefício é que a redução da dimensionalidade, garante um modelo mais compreensível, pois o modelo envolverá menos atributos. Além disso, essa redução pode permitir que os dados sejam mais facilmente visualizados quando comparados com dados com muitos atributos, isto é, com uma alta dimensionalidade [58].

Embora possa parecer que tal abordagem perderia informações, esse não é o caso se recursos redundantes e irrelevantes estiverem presentes. Os recursos redundantes duplicam muitas, ou até mesmo, todas as informações contidas em um ou mais atributos. E os recursos irrelevantes quase não contêm informações úteis para a tarefa de mineração de dados em questão. Por exemplo, os números de identificação dos alunos são irrelevantes para a tarefa de prever as médias de notas dos alunos. Dessa forma, os recursos redundantes e irrelevantes podem reduzir a precisão da classificação e a qualidade dos clusters encontrados [58].

Enquanto alguns atributos irrelevantes e redundantes podem ser eliminados imediatamente usando o conhecimento do domínio dos dados, selecionar o melhor subconjunto de recursos de maiores influências requer uma abordagem sistemática. A principal abordagem é a chamada de abordagem incorporada. Nela, a seleção de recursos ocorre naturalmente como parte do algoritmo de mineração de dados. Especificamente, durante a operação do algoritmo de mineração de dados, o próprio algoritmo decide quais atributos usar e quais ignorar. Algoritmos para a construção de classificadores de árvore de decisão, geralmente operam dessa maneira utilizando a importância de Gini [14], abordada na seção 2.4.1.1.1.

Logo, pelo fato da consulta realizada na seção 3.2 ter retornado muitas variáveis, foi necessário realizar a seleção das variáveis de maiores influências, isto é, as variáveis que influenciam significativamente no desempenho dos algoritmos de AM, para garantir um resultado mais preciso, confiável e menos complexo. Para isso, foi feita uma árvore de decisão com todas as 26 variáveis e, a partir disso, calcular a importância de Gini de cada uma. No gráfico da Figura 3.6 é possível ver as importâncias de Gini calculadas para cada variável, em ordem crescente de importância. Note que o valor da importância de Gini máximo é igual a 1 e o minímo é 0, porém a escala está de 0 até 0.3358 para uma melhor visualização das barras.

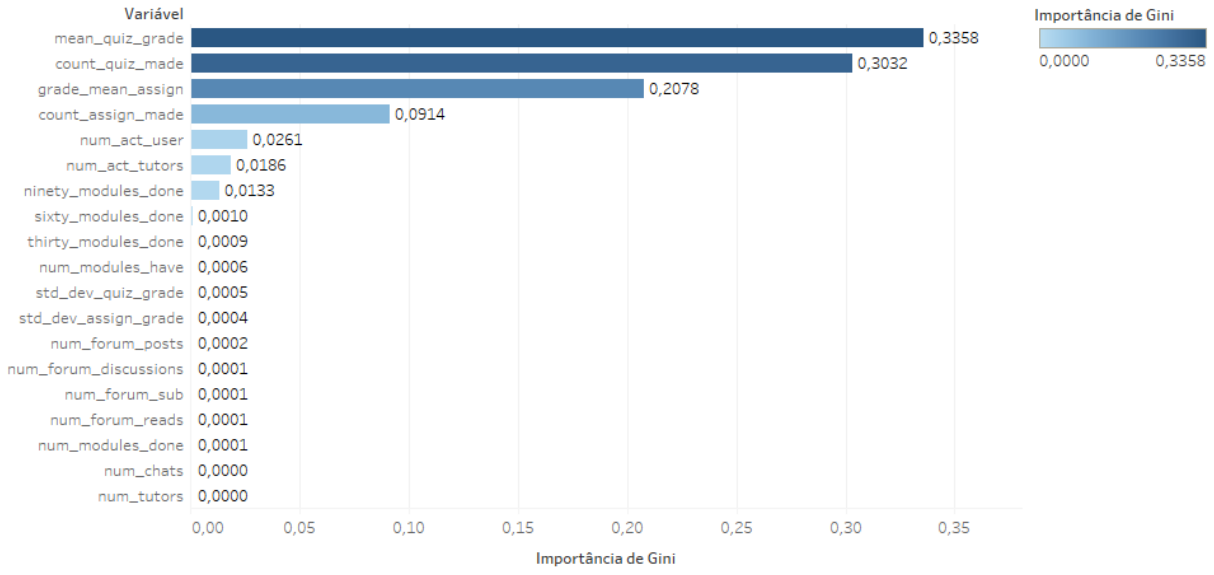


Figura 3.6: Gráfico da importância de Gini calculada para cada variável.

Nesta Figura 3.6, pode-se perceber que a variável "mean_quiz_grade" têm a maior importância de Gini com valor de 0.3358 e, portanto, é a variável que mais tem influência na performance dos algoritmos de AM a serem utilizados neste trabalho. Em seguida vêm a variável "count_quiz_made" com a segunda maior importância, "grade_mean_assign"

com a terceira maior, “count_assign_made” com a quarta e com a quinta, sexta e sétima maiores importâncias são as variáveis, “num_act_user”, “num_act_tutors”, “ninety_modules_done”, respectivamente. Abaixo da sétima variável de maior importância, os valores são muito pequenos e, consequentemente, ao utilizá-las a performance permanece inalterada. Portanto, essas 12 variáveis com menor importância de Gini do que a variável “ninety_modules_done”, podem ser desconsideradas, sobrando assim, 7 variáveis para serem utilizadas para a aplicação dos algoritmos subsequentes.

3.4 Classificação, predição e clusterização dos dados

Os algoritmos de AM utilizados para a mineração dos dados, extraídos na seção 3.2, foram: o algoritmo CART para árvore de decisão, o algoritmo de *Adaboost* baseado em árvore de decisão, o *random forest* e o algoritmo de clusterização *k-means*, no qual todos eles foram explicados na seção 2.4. O principal objetivo foi classificar e predizer se o participante foi aprovado ou reprovado, a partir das 7 variáveis selecionadas na seção 3.3.2. Dessa forma, foi possível analisar cada característica dos dados a partir das variáveis para, posteriormente, entender como se comporta e quais são as ações necessárias para um aluno ser aprovado ou reprovado. Com a clusterização, é possível analisar as variáveis de forma mais isoladas, pois é utilizada no máximo 3 variáveis para tentar classificar os participantes em aprovados ou reprovados por este método. Com isso, é possível entender quais são as características que mais se relacionam com a aprovação ou reprovação do aluno.

Primeiramente, para construir esse algoritmos foi utilizada a linguagem de programação Python, pois é uma linguagem muito conhecida por suportar vários paradigmas de programação, incluindo orientação a objetos, imperativo, funcional e procedural, e, principalmente, por ser uma linguagem muito utilizada para análise e mineração de dados. A consequência disso é o Python ter uma biblioteca padrão ampla e bastante abrangente para a mineração de dados [59].

A primeira das bibliotecas de Python utilizadas é chamada de “Pandas”. Essa biblioteca foi feita para manipulação e análise de dados. Em particular, ela oferece estruturas de dados e operações para manipular tabelas de conjuntos de dados, para filtrar objeto de dados com indexação integrada, além de permitir o alinhamento de dados e manipulação integrada de dados ausentes. Com essas características, ela se torna bastante importante para o processo de manipulação dos dados e, uma posterior, aplicação deles nos algoritmos de AM.

A segunda biblioteca e, talvez a mais importante, é a “scikit-learn”. Essa biblioteca consiste em uma ferramenta simples e eficiente para mineração de dados e análise de

dados. Ela foi construída com diversas outras bibliotecas de matemática, estatística e de visualização de dados. A sua característica principal é ter implementado diversos algoritmos de AM, tornando a sua utilização bem prática por permitir todos os tipos de adaptação de qualquer algoritmo para obter o resultado desejado.

3.4.1 Classificação e predição dos dados

Como mencionado anteriomente, para realizar uma primeira classificação e predição dos dados, foi utilizado como base o algoritmo CART para ser implementado no Python. Esse código implementado encontra-se no Âpendice D. Primeiramente, os dados foram importados para o Python, com a biblioteca *Pandas*, através do arquivo em CSV gerado pelas consultas na seção 3.2. Com o conjunto de dados importados, foi realizada a seleção das variáveis a serem utilizadas definidas na seção 3.3.2.

A partir disso, o conjunto de dados foi divido para 70 % dele ser utilizado como conjunto de treinamento e 30 % como conjunto de testes. Após esse passo, foi utilizada a biblioteca *sckit-learn* para auxiliar na construção do classificador com o algoritmo de CART, de forma a recursivamente descobrir cada atributo de teste que gerasse aos nós filhos, o menor grau de impureza de Gini. Durante a construção desse classificador, foi percebido que a árvore só precisava ter no máximo, um nível de profundidade igual a 5, pois a partir desse nível a árvore ficava muito complexa de entender e de visualizar, além da performance não ser alterada significativamente. Após a modelagem do classificador, foi possível gerar a imagem da árvore de decisão e prever se os alunos do conjunto de testes foram aprovados ou reprovados com bases nas suas características. O resultado e a visualização da árvore são apresentados no Capítulo 4.

Uma segunda classificação e predição dos dados foi feita utilizando o algoritmo de Adaboost para tentar melhorar a performance do classificador simples da árvore de decisão. O código para este método se encontra no Âpendice E. Os mesmos dados importados para a árvore de decisão, foram importados para este método e, novamente, eles foram divididos em 70 % e 30 %, respectivamente, para o conjunto de treinamento e para o conjunto de testes. Após essa etapa, foi criado o classificador baseado na árvore de decisão, com o número de classificadores bases igual a 10. Este número foi utilizado, pois percebeu-se que a partir dessa quantidade de classificadores, a performance não aumentava significativamente e o custo computacional se elevava bastante. Durante a etapa de construção dos classificadores, foi possível calcular a importância de cada um deles (Equação 2.7) a partir das suas respectivas taxas de erro (Equação 2.6), como é mostrado na Tabela 3.4.

Nessa Tabela 3.4, é possível ver que a taxa de erro dos classificadores vai diminuindo e, consequentemente, aumentando a sua importância, conforme o número do classificador base aumenta. Isso se dá pelo fato do Adaboost focar nos conjuntos de dados de dificíl

Tabela 3.4: Importância e erro dos classificadores base gerados pelo Adaboost.

Classificador	**Taxa de erro** ($\in_i$)	**Importância** (α_i)
1	0.3637	0.2796
2	0.3741	0.2574
3	0.3058	0.4098
4	0.3043	0.4135
5	0.2986	0.4270
6	0.2773	0.4791
7	0.2867	0.4557
8	0.2036	0.6820
9	0.1549	0.8485
10	0.0282	1.7698

classificação, logo, à medida que eles vão aprendendo a classificar esses dados difíceis, a sua taxa de erro diminui. Ao final, com a combinação desses classificadores, obtém-se um classificador forte com o desempenho melhor do que o simples classificador da árvore de decisão.

A terceira e última classificação e predição dos dados foi feita utilizado o algoritmo de random forest, com o mesmo objetivo de melhorar a performance do classificador simples da árvore de decisão. O código para este método, também, se encontra no Âpendice E. Foram importados para este método, os mesmos dados do Adaboost e, novamente, eles foram divididos em 70 % para o conjunto de treinamento e 30 % para o conjunto de testes. Pelo mesmo motivo do Adaboost, foi determinado um conjunto de 10 classificadores bases. Durante a etapa de construção deles, foi possível calcular a margem de cada um deles a partir da Equação 2.9, como pode ser vista na Tabela 3.5. Nela, é possível notar que quanto maior a probabilidade do classificador ter previsto corretamente um dado ($Y_\theta = Y$) e menor a máxima probabilidade do classificador prever incorretamente esse mesmo dado ($Y_\theta = Z$), maior é a probabilidade de que o classificador preveja corretamente esse registro. E a média das probabilidades de acertos desses classificadores deu 0.8512, o que significa uma alta probabilidade de acerto na média.

Após a construção dos classificadores do Adaboost e do random forest, foi possível gerar a imagem da árvore de decisão de cada um deles e prever se os alunos do conjunto de testes foram aprovados ou reprovados com bases nos seus atributos (características). O resultado de cada predição e a visualização das árvores são mostrados no Capítulo 4.

3.4.2 Clusterização dos dados

A clusterização dos dados foi feita utilizando o algoritmo k-means, com o objetivo de analisar, de uma forma mais isolada, as variáveis de maior influência na aprovação dos alu-

Tabela 3.5: Margem dos classificadores base gerados pelo random forest.

Classificador	$P(Y_\theta = Y)$	$\max_{Z \neq Y} P(Y_\theta = Z)$	**Margem**
1	0.9502	0.0498	0.9004
2	0.9428	0.0572	0.8856
3	0.9807	0.0193	0.9613
4	0.7740	0.2260	0.5480
5	0.9807	0.0193	0.9613
6	0.9730	0.0270	0.9459
7	0.8885	0.1115	0.7770
8	0.8731	0.1269	0.7461
9	0.9733	0.0267	0.9465
10	0.9198	0.0802	0.8396

nos. O código para a clusterização dos dados encontra-se no Âpendice F. Primeiramente, os dados utilizados nos algoritmos de predição e clusterização, também foram importados para este método. Porém, como o k-means é um algoritmo não-supervisionado de agrupamento, não é necessário separar os dados em conjunto de treinamento e de testes, basta agregar os dados a partir das características semelhantes entre eles.

Antes de fazer o agrupamento, foi necessário escolher o número k de *clusters*. Como haviam duas classes bem definidas, aprovado e reprovado, para a clusterização, o número de k foi escolhido como igual a 2 para todos os agrupamentos. Após isso, foi escolhida a variável "mean_quiz_grade" e a "finalgrade" para fazer a primeira clusterização dos participantes em aprovados ou reprovados. Então, em seguida, foi feita a clusterização dos dados afim de encontrar dois *clusters*. Com base na Equação 2.11, foi possível calcular, a soma dos erros quadrados desse cluster. Por ter milhares de pontos na clusterização, o valor do SSE foi alto e resultou no valor de 59746.

O segundo agrupamento gerado foi com as variáveis "count_quiz_made" e a "finalgrade". Após o agrupamento dos dados, foi calculado o SSE e resultou 224645. Dessa forma, foi possível notar que quanto menor a importância dos atributos confome a importância de Gini, maior era o SSE. E por este motivo, só foram feitos dois agrupamentos com duas variáveis.

Logo, afim de gerar uma clusterização mais fidedigna, foram utilizadas, dessa vez, três variáveis para a terceira tarefa de agrupamento. Foram elas, a variável "mean_quiz_grade", "count_quiz_made" e a "finalgrade". Após a geração dos dois clusters, o resultado do SSE diminuiu significativamente em relação ao primeiro cluster gerado, com o resultado de 19225.

Com o objetivo de ter outra perspectiva de clusterização com três variáveis, foi feito um quarto agrupamento com as variáveis "ninety_modules_done", "mean_quiz_grade" e a "finalgrade". Com o agrupamento gerado, o SSE gerado foi de 22932. Note que esse valor

foi maior que o SSE do terceiro agrupamento pelo fato da variável “ninety_modules_done” ter uma importância de Gini menor que a “count_quiz_made”, utilizada na terceira clusterização. A visualização de todos os 4 agrupamentos gerados, bem como a visualização dos seus centróides, serão apresentados no Capítulo 4.

3.5 Análise de desistências nos cursos

Com o objetivo de analisar as desistências nos cursos oferecidos pela Enap, foi utilizado novamente o Python e a biblioteca *pandas*, porém não foi preciso utilizar a biblioteca *sckit-learn*, pois não foi aplicado técnicas de AM para tal. Porém, para poder gerar gráficos visualmente agradáveis, foi utilizado o *Tableau*, que é um *software* exclusivo para visualização de dados. A partir do arquivo CSV gerado pela segunda consulta realizada na seção 3.2, os dados foram importados para o Python. Nestes dados, foram encontrados 19170 participantes desistentes e para cada um deles existe o tipo de justificativa que cada participante deu para desistir do curso, porém o tipo é representado por números de 0 a 4, isto é, são 5 tipos de justificativa.

Logo, foi preciso relacionar cada número de acordo com a sua categoria de justificativa e essa informação foi encontrada na tabela “mdl_enrol_unenrol_options”. Com essa informação, o número 4 foi relacionado à categoria “Dificuldades técnicas”, que significa quando o participante tem muitas dificuldades técnicas para acessar a plataforma Moodle. O número 3 foi atribuído à categoria “Tutor não acompanhando”, que é o cenário em que o tutor não ajuda suficientemente o participante e ele não consegue completar o curso sem essa ajuda. O número 2 foi relacionado à categoria “Curso complexo”, que é quando o participante acha o curso muito complexo para ele. O número 1 foi atribuído à categoria “Falta de Tempo”, que significa que o participante não tem tempo suficiente para completar o curso. Por fim, o número 0 foi relacionado à categoria “Outra”, que quer dizer que o participante não achou que sua justificativa se encaixava em nenhumm dos outros tipos de justificativas.

Após essa etapa, foi pego o número total de cada justificativa em relação a soma total das justificativas, para verificar a porcentagem de cada tipo de justificativa em relação ao total. Dessa forma, é possível realizar a análise para descobrir qual é são as maiores e menores causas de desistências. Depois desse tipo de análise, foi dividido o período dos cursos em três etapas: até 1/3 do curso, entre 1/3 e 2/3 do curso, e depois de 2/3 do curso. Isso foi feito com o objetivo de construir uma análise para saber a taxa de desistência nesses três períodos dos cursos. Para isso, foi pego o número total de desistência em cada um desses três período em relação a soma total das desistências.

Um último tipo de análise feita foi para descobrir até quantos módulos feitos, os participantes desistiam dos cursos. Para isso, foi feito a divisão do número total de módulos de um curso em três partes: até 1/3 dos módulos, entre 1/3 e 2/3 dos módulos, depois de 2/3 dos módulos. Logo, foi possível relacionar todas as desistências com a parte da divisão em que elas se encaixavam e, consequentemente, foi possível descobrir qual a maior e menor taxa de módulos feitos até o momento das desistências dos participantes.

Todo o código feito em Python para a construção dessas análises encontra-se no Âpendice G. E o resultado delas e suas respectivas visualizações, serão apresentadas no Capítulo 4 a seguir.

Capítulo 4

Resultados

Este capítulo visa apresentar a validação dos resultados obtidos pelo *framework* de EDM proposto. Além disso, é feita a visualização desses resultados, pois a visualização dos dados é uma das etapas primordiais da mineração de dados.

4.1 Avaliação da Performance dos Algoritmos

As performances dos algoritmos CART (árvore de decisão), Adaboost, random forest e k-means foram avaliadas por meio das métricas de precisão, revocação e medida de F_1, Equações 2.15 a 2.17, respectivamente. Isso foi feito a partir da matriz de confusão gerada por cada algoritmo, após a aplicação de conjunto de testes nos classificadores gerados por eles, exceto pelo k-means. Como explicado anteriomente na seção 3.4.2, o k-means agrupou todo o conjunto de dados, e após esse agrupamento, verificou-se o número de instâncias TP, TN, FN, FP classificadas, para então gerar a matriz de confusão deste algoritmo.

O resultado da performance desses 4 algoritmos e do algoritmo realizado em [8], podem ser vistos na Tabela 4.1. De acordo com essa tabela, o Adaboost tem a melhor performance de 0.88 de medida de F_1, que pode ser explicado pelo fato de ter o seu conceito baseado em treinar os classificadores base, nos dados que são difíceis de serem classificados corretamente. O resultado do random forest foi bastante similar ao do Adaboost, porém a razão dele ter uma performance um pouco inferior é que ele utiliza conjunto de dados aleatórios, ao contrário do Adaboost que escolhe os dados laboriosos de serem classificados.

O algoritmo CART da árvore de decisão se mostrou um bom classificador, porém pelo fato de ser um ser simples classificador, não teve tanto êxito quantos os métodos ensemble (Adaboost e random forest). O resultado do k-means apresentado nesta Tabela 4.1, foi o gerado com os 3 recursos que apresentaram o melhor desempenho para o algoritmo. A

Tabela 4.1: Performance dos algoritmos utilizados.

Algoritmo	**Precisão**	**Revocação**	**F1**
Adaboost	0.89	0.88	0.88
Random forest	0.87	0.85	0.86
CART	0.83	0.84	0.83
Árvore de decisão em [8]	0.81	0.82	0.82
k-means	0.79	0.81	0.80

razão dele ter tido uma performance mais baixa do que dos outros algoritmos é que ele é feito para agrupar objetos em clusters com padrões semelhantes. Portanto, como não há um supervisionamento da classificação correta para ele, o k-means tenta o reconhecimento de padrões para agrupar os dados. Porém os dados de participantes de cursos podem ser muito variados por se tratar de pessoas, o que torna o agrupamento uma tarefa árdua de ser realizada. Mas ainda assim, apesar desses fatores negativos, o k-means se mostrou uma boa técnica de clusterização de dados educacionais.

O algoritmo do trabalho [8] também se mostrou um bom classificador pela sua performance apresentada. Porém pelo fato de ter utilizado o algoritmo J48, a sua performance foi um pouco inferior à do algoritmo CART, que é a versão mais recente de algoritmos de árvore de decisão.

4.2 Visualização das classificações e clusterizações geradas

A visualização de dados é a exibição de informações em um formato gráfico ou tabular. Para ela ser bem-sucedida, é necessário que as informações sejam convertidas em um formato visual para que as características dos dados e as relações entre eles possam ser analisadas ou relatadas. Portanto, para que facilite o entendimento das visualizações, ela foi divida em duas partes: a visualização das árvores geradas pelos algoritmos CART, Adaboost e random forest, e a visualização das clusterizações geradas pelo k-means.

4.2.1 Visualização das árvores

Primeiramente, foi gerada a visualização da árvore do algoritmo CART, que foi divida em duas figuras para uma melhor visualização. A parte esquerda da árvore encontra-se na Figura 4.1 e a parte direita na Figura 4.2. Como é possível ver na legenda das figuras, existem 6 tipos de cor de nó, que indicam a probabilidade de aprovação de um participante caso chegue neste nó. A primeira cor é o azul escuro que corresponde a 78 % de chance de aprovação. O azul intermediário corresponde a uma probabilidade de aprovação entre 65

% e 78 %, enquanto o azul claro corresponde a uma chance de aprovação entre 58 % e 65 %. A cor bege significa que o participante tem entre 45 % e 58 % de chance de aprovação e o nó castanho claro significa que o participante tem entre 32 % e 45 %. E finalmente, o nó castanho escuro significa que o participante tem menos de 32 % de chance de aprovação.

Também é possível ver que em cada nó tem o número de registros que chegaram até o nó (samples), e nos números entre colchetes significa que o número, antes da vírgula, corresponde ao total de participantes reprovados e depois da vírgula, o número de participantes aprovados. Os dois somados resultam no total de registros deste nó. Além disso, em cada nó é mostrado a impureza de Gini dele.

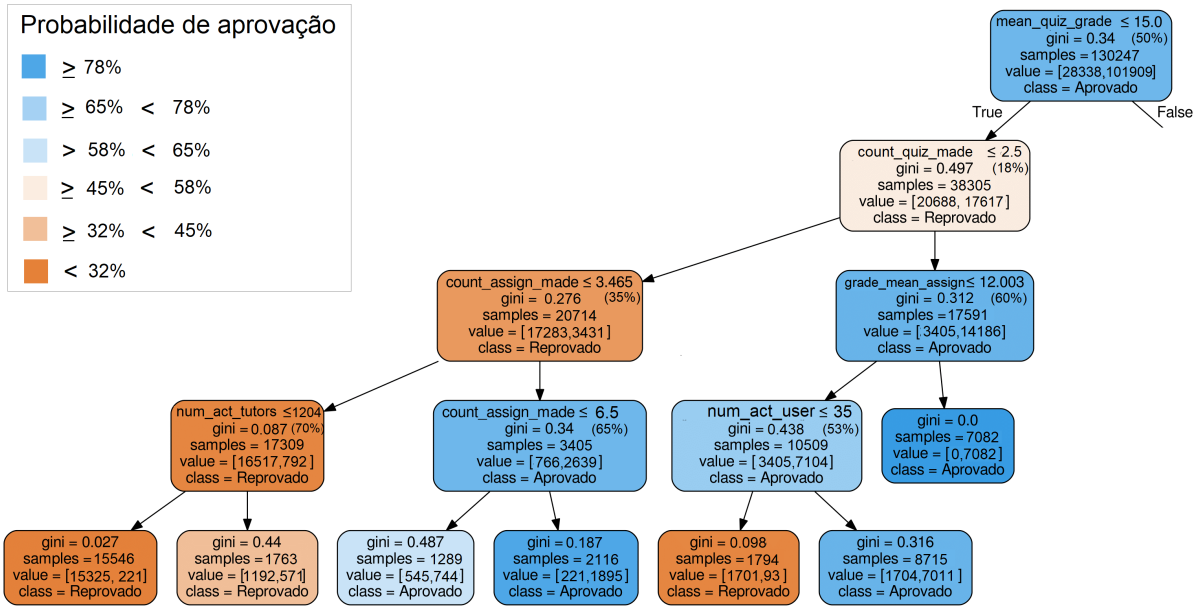


Figura 4.1: Parte esquerda da árvore gerada pelo algoritmo CART.

Analisando a árvore gerada, pode-se notar que a média das notas nas tarefas essenciais (mean_quiz_grade) é um fator determinante para a aprovação do participante, pois essa variável é utilizada como nó raiz e se a nota for maior que 15 (50 % da média máxima), o participante vai para a parte da direita da árvore. E a parte direita da árvore contém muitos nós com chances muito altas de aprovação, o que não acontece na parte esquerda da árvore.

Outro ponto interessante de se notar é que se o participante tiver essa média menor que 15, porém o número de atividades essenciais feitas (count_quiz_made) for maior que duas atividades, ele ainda tem chances maiores de aprovação. A não ser que o participante tenha um número de ações menor que 35, que corresponde a 53 % do número total de ações a ser feita. Um ponto a ser notado é que essa árvore não está totalmente balançeada,

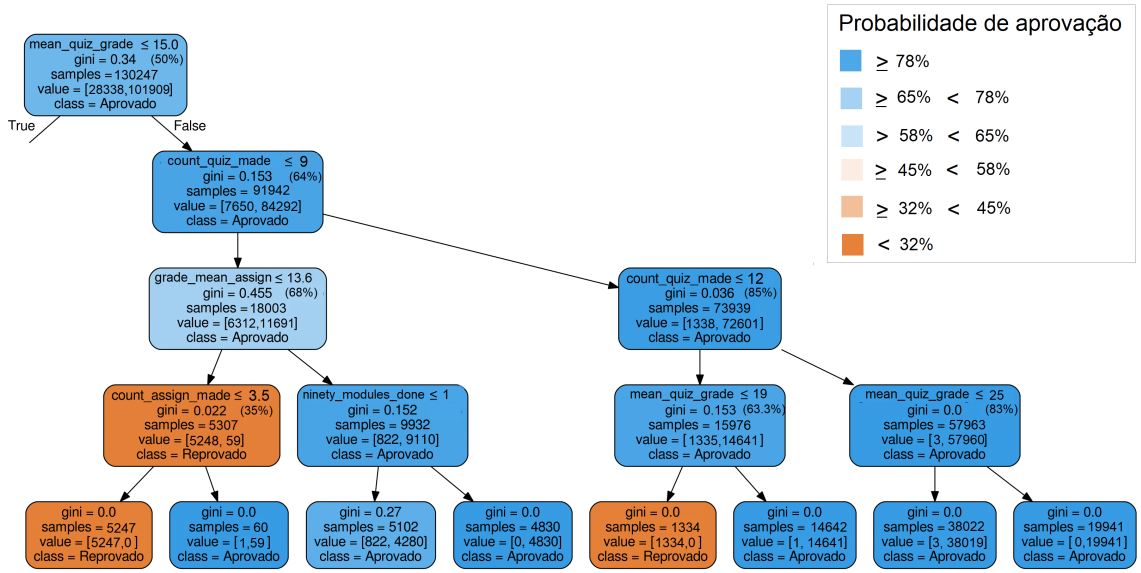


Figura 4.2: Parte direita da árvore gerada pelo algoritmo CART.

isto é, o número de nós folhas da parte esquerda (6) não é o mesmo que o do nós folhas da parte direita (8), o que pode prejudicar na performance do algoritmo.

A segunda árvore gerada foi com o algoritmo Adaboost e pode ser vista nas Figuras 4.3 a 4.4. Da mesma forma que a árvore gerada pelo CART, existe 6 tipos de cores dos nós para indicar a probabilidade de aprovação do aluno, porém algumas probabilidades mudaram. Um fator importante a ser notado nesta árvore na parte esquerda é que, quando o participante apresenta menos de 5 tarefas essenciais feitas (count_quiz_made), isto é, 35 % do total de tarefas, e tem a nota média nas tarefas complementares (grade_mean_assign) inferior a 3 (30 % da nota máxima), o participante pode ser aprovado dependendo do número de ações realizadas pelo tutores. Outro ponto importante é que, ao contrário da árvore anterior, essa está totalmente baleançada, com o número de nós folhas da parte direita e esquerda sendo iguais.

A terceira e última árvore gerada, foi através do algoritmo random forest, que pode ser visualizada nas Figuras 4.5 a 4.6. Assim como as outras duas árvores geradas, há 6 tipos de cores dos nós, indicando a probabilidade de aprovação. Essa árvore se assemelha bastante à árvore do Adaboost, porém um ponto interessante de se notar é que, na parte direi da árvore, se o participante apresentar mais que 10 tarefas essenciais feitas (count_quiz_made), isto é, 71 % do total de tarefas, e tiver a nota média nas tarefas essenciais (mean_quiz_grade) inferior a 20 (66 % da nota máxima), ele ainda pode ser reprovado dependendo da sua média nas tarefas complementares feitas durante o curso. Da mesma forma que a árvore do Adaboost, essa árvore também está balanceada.

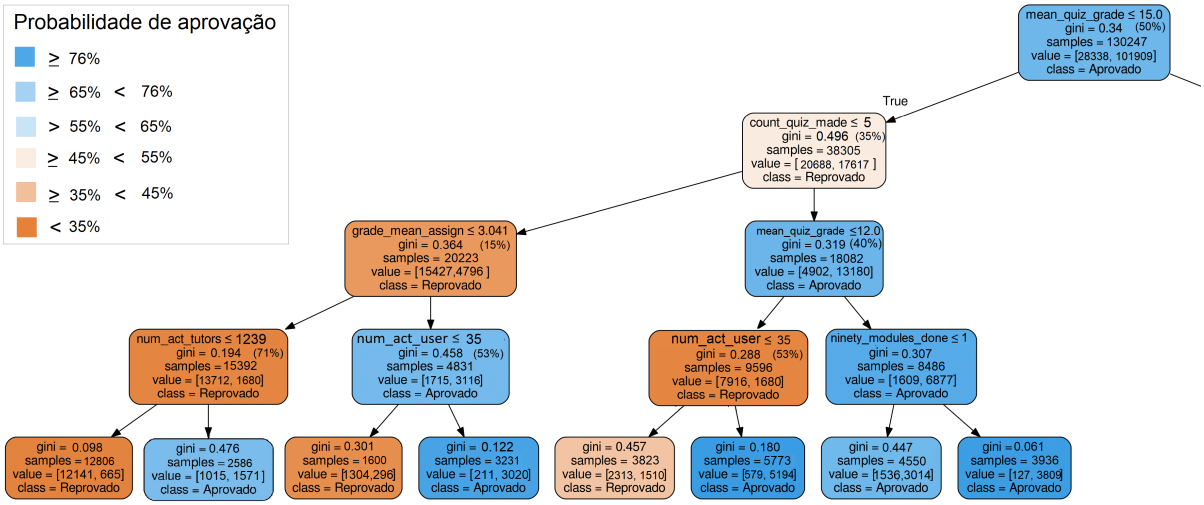


Figura 4.3: Parte esquerda da árvore gerada pelo algoritmo Adaboost.

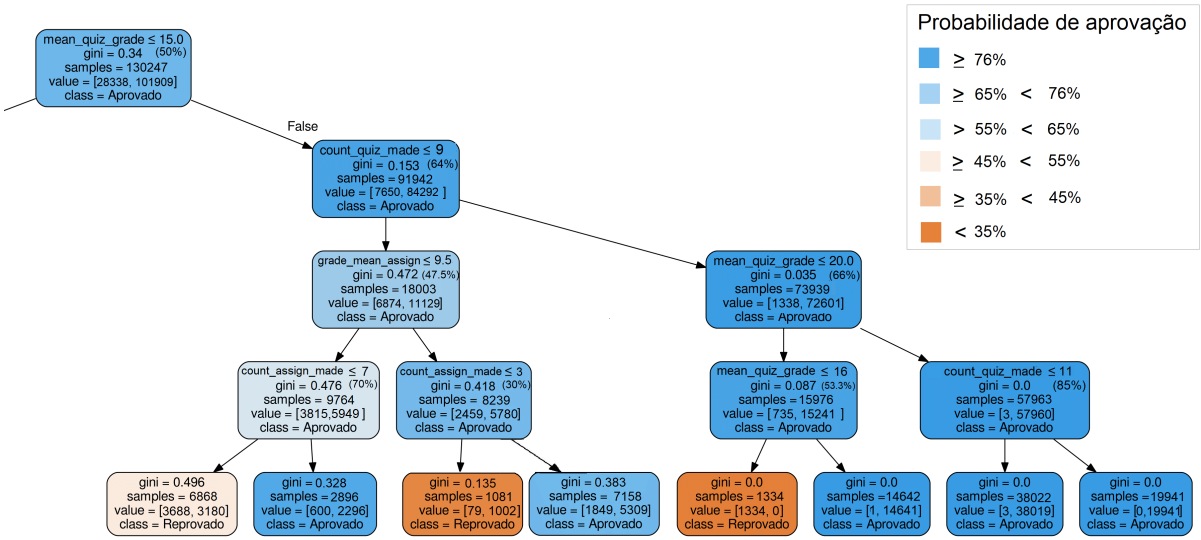


Figura 4.4: Parte direita da árvore gerada pelo algoritmo Adaboost.

4.2.2 Visualização das clusterizações

A primeira clusterização gerada, conforme mencionado na seção 3.4.2, foi utilizando as variáveis "mean_quiz_grade" e "finalgrade" para fazer a clusterização dos participantes em aprovados ou reprovados. A Figura 4.7 mostra a clusterização gerado com seus respectivos centróides em azul, no qual os pontos vermelhos são os participantes do cluster reprovado e os pontos verdes são os aprovados. Como explicado na seção 3.2, a variável "finalgrade" corresponde à nota final do participante no curso e para ele ser aprovado, essa nota deve ser maior ou igual à 60, caso contrário ele estará reprovado no curso. Após

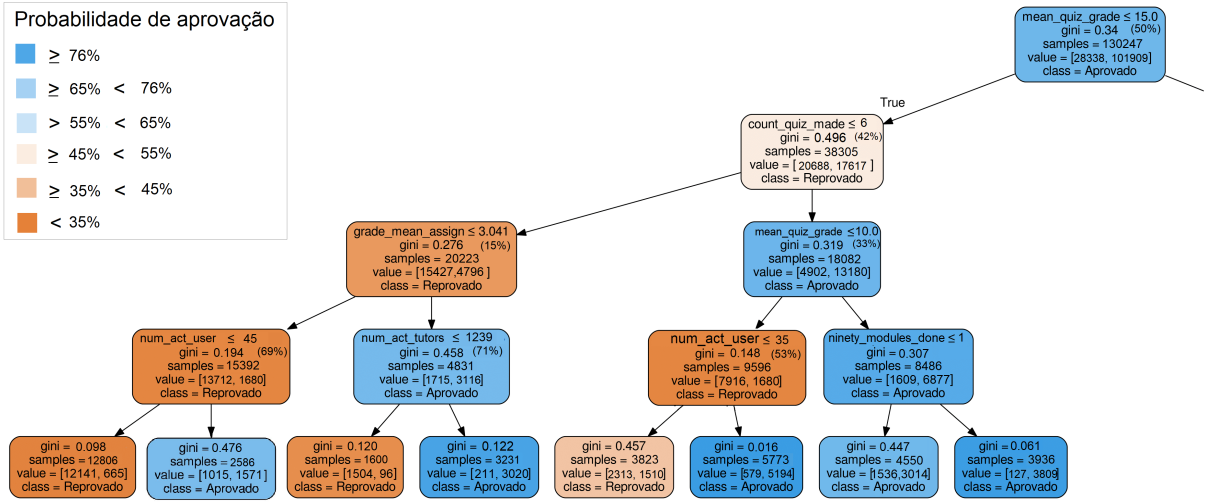


Figura 4.5: Parte esquerda da árvore gerada pelo algoritmo random forest.

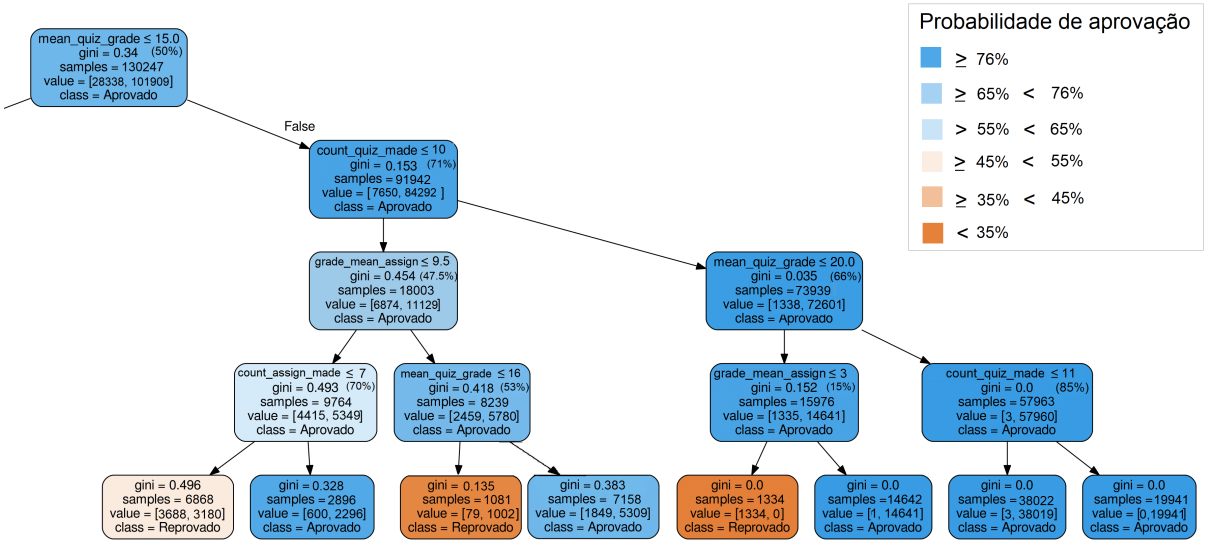


Figura 4.6: Parte direita da árvore gerada pelo algoritmo random forest.

a clusterização e com base na equação Equação 2.12, foi possível calcular o ponto do centróide dos dois agrupamentos. O centróide do cluster 1, dos participantes reprovados, foi de (23.5, 10.4), isto é, com o valor de 23.5 de nota final (finalgrade) e 10.4 de média das notas nas tarefas essenciais. Isso significa que os participantes que tiveram a nota média das tarefas essenciais próximo a 10.4 (34,6 % da nota total) foram classificados como reprovados. O centróide do cluster 2, dos aprovados, foi de (85.20, 26.61), o que significa que os participantes que tiveram a nota média nas tarefas essenciais mais próxima de 26.61 (88 % da nota total) do que de 10.4 (centróide do cluster 1), foram classificados como aprovados.

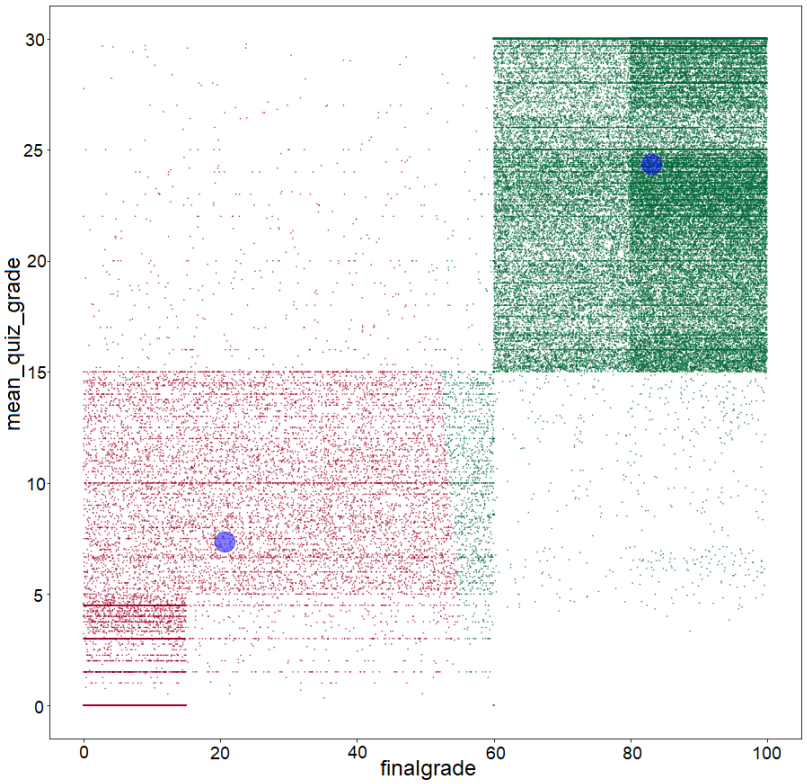


Figura 4.7: Clusterização com as variáveis “mean_quiz_grade” e “finalgrade”.

O segundo agrupamento gerado foi com as variáveis “count_quiz_made” e a “finalgrade”. Na Figura 4.8 é mostrada a clusterização gerada com essas novas duas variáveis. Após o agrupamento dos dados, foi calculado o centróide do cluster 1 (reprovados) e o seu resultado foi de (20.17, 2.35). Dessa forma, os participantes que fizeram aproximadamente duas tarefas essenciais (14 % do total) foram classificados como reprovados. Por outro lado, o centróide do cluster 2 (aprovados) deu (82.9, 10.9), o que significa que os participantes que fizeram em torno de 11 ou mais tarefas essenciais (78 % do total) foram rotulados como aprovados.

Na terceira clusterização, foram utilizadas as variáveis “mean_quiz_grade”, “count_quiz_made” e “finalgrade”. A Figura 4.9 mostra o resultado dessa clusterização. Após a geração dos dois cluters, o resultado do centróide do cluster 1 (reprovados) foi de (21, 3.2, 7.4). O que significa que os alunos que fizeram cerca de 3 tarefas essenciais (21 % do total) e tiveram uma média aproximada de 7.4 nessas tarefas (25 % da média maior), foram classificados como reprovados. Já o centróide do cluster 2 (aprovados) resultou em (81, 11, 21.3). Isso significa que os alunos que fizeram cerca de 11 tarefas essenciais (78,5 % do total) e tiveram uma média aproximada de 21.3 nessas tarefas (70 % da média máxima), foram rotulados como aprovados. Este tipo de clusterização foi o que teve a melhor perfor-

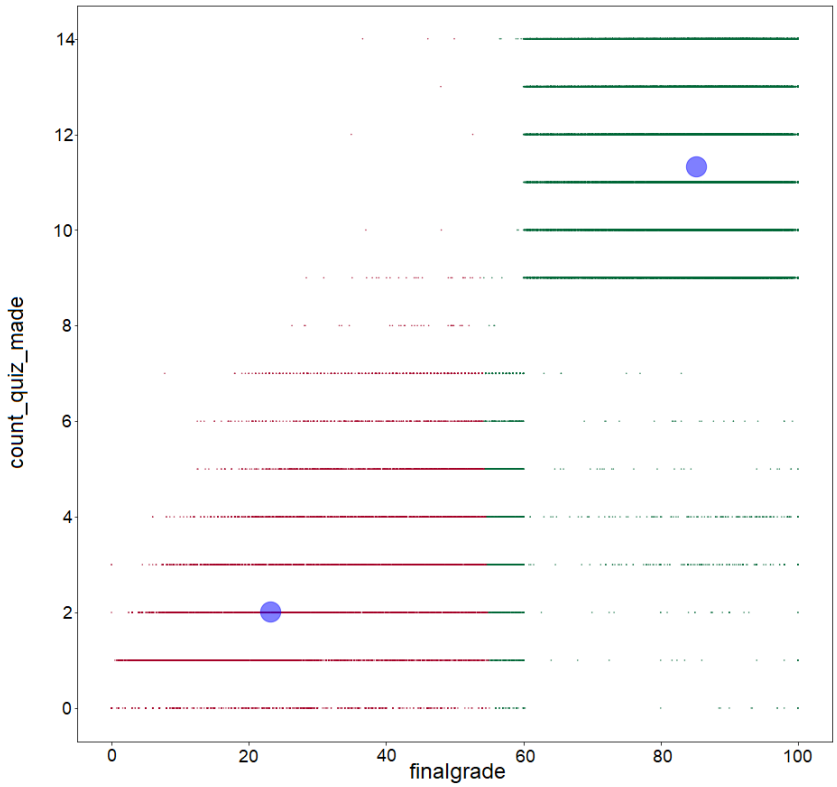


Figura 4.8: Clusterização com as variáveis “count_quiz_made” e “finalgrade”.

mance entre todos os tipos e que foi citado na seção 4.1. O fator determinante para essa clusterização ser a melhor é que são utilizados as duas variáveis (“mean_quiz_grade” e “count_quiz_made”) de maiores valores da importância de Gini.

No quarto agrupamento, foram utilizadas as variáveis “ninety_modules_done”, “mean_quiz_grade” e “finalgrade”. Na Figura 4.10 é possível ver o resultado da clusterização com essas três variáveis. Com o agrupamento gerado, o centróide do cluster 1 (reprovados) deu (22.3, 9,4, 0). Note que a variável “ninety_modules_done” é binária, portanto ela só assume o valor 0, se o aluno não tiver feito 90 % dos módulos de um curso. Caso contrário, ela assume o valor 1. Com o valor do centróide do cluster 1, pode-se afirmar que os participantes com média nas atividades próxima de 9.4 (31 % da média máxima) e não fizeram 90 % dos módulos (ninety_modules_done = 0), foram taxados como reprovados. Já o centróide do cluster 2 (aprovados) foi de (82.3, 26.2, 1), o que significa que o participante que teve média nas tarefas essencias próxima de 26.2 (87 % da média máxima) e concluiu 90 % dos módulos do curso, foi rotulado como aprovado.

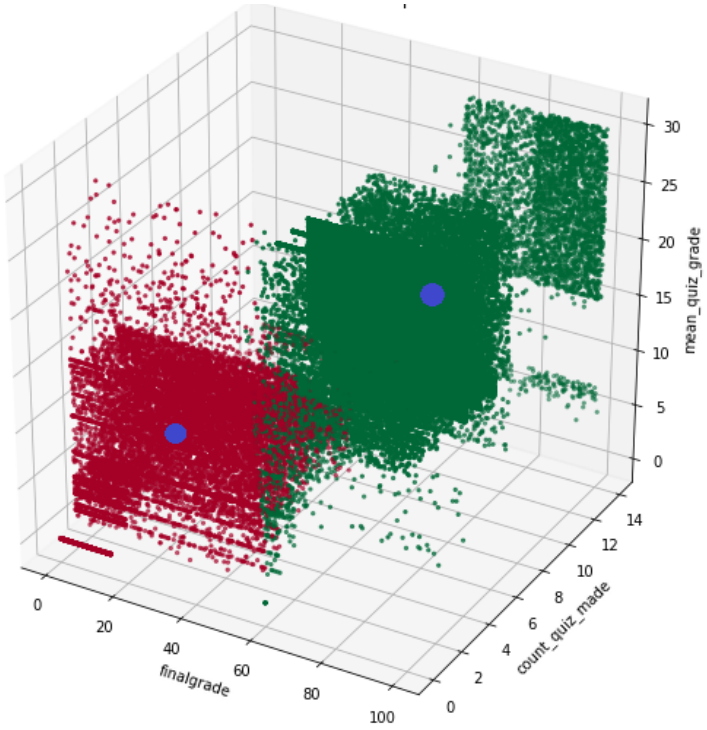


Figura 4.9: Clusterização com as variáveis “mean_quiz_grade”, “count_quiz_made” e “finalgrade”.

4.3 Visualização da análise das desistências

O objetivo da visualização de dados é a interpretação da informação visualizada e a formação de um modelo mental da informação. Dessa forma, a visualização é imprescindível para analisar as desistências dos participantes, o período em que elas ocorrem e quais são os principais motivos para a evasão dos cursos. Portanto, primeiramente, foi gerado um gráfico com as razões de desistência dos cursos alegadas pelos participantes, como pode ser visto na Figura 4.11. No total, foram 19170 desistentes, o que resulta em aproximadamente de 25 % de desistentes do total de 76551 alunos na Enap.

Como mostrado ainda na Figura 4.11, a falta de tempo é a principal razão dos participantes (61,79 %), seguida da categoria “Outros” (20,23 %), que é a segunda razão principal. Na ordem decrescente de porcentagem vem a categoria “Dificuldades técnicas” com 9,67 %, seguido dos participantes que acham o curso complexo (8,23 %), e, com a menor e desprezível porcentagem (0,08 %), são os participantes que declararam que os tutores não deram a ajuda necessária durante o curso.

O período em que essas desistências ocorreram, como mostrado no gráfico da Figura 4.12, é em sua maioria em até 1/3 da ministração dos cursos (94,40 %). Em seguida, vem o período de entre 1/3 e 2/3 dos cursos com 2,86 %. E o terceiro período, depois

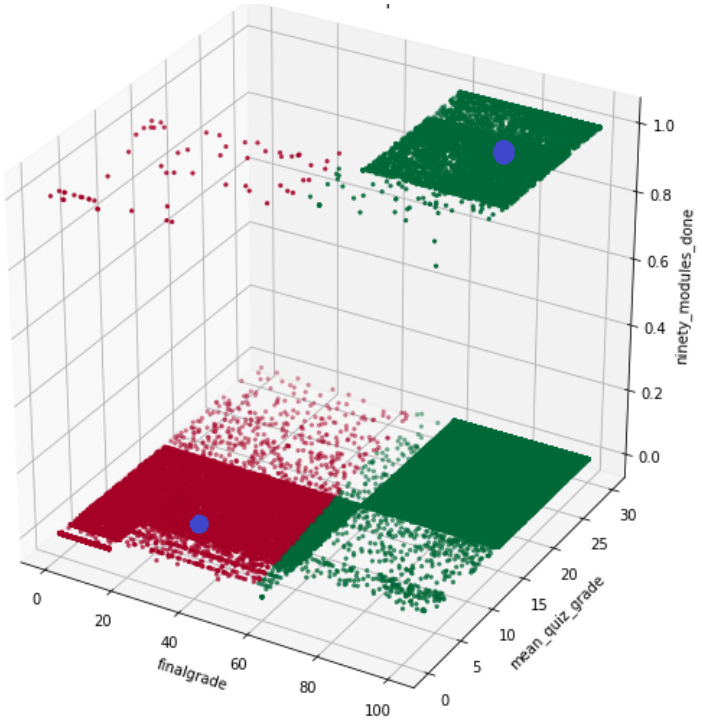


Figura 4.10: Clusterização com as variáveis “ninety_modules_done”, “mean_quiz_grade” e “finalgrade”.

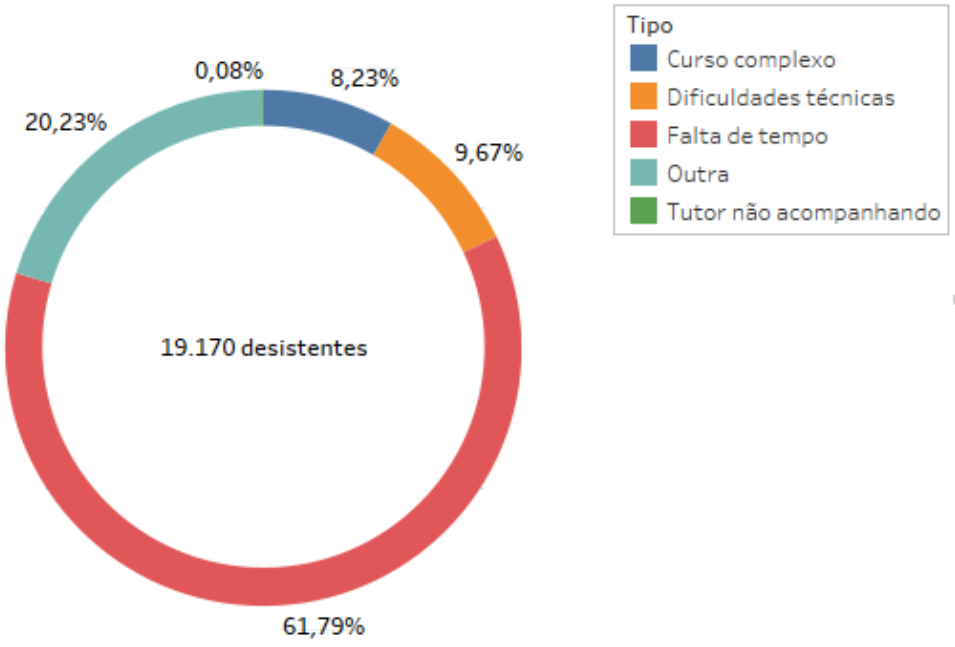


Figura 4.11: Gráfico com as razões de desistências alegadas pelos participantes.

de 2/3 da ministração dos cursos, aparece com 2,74 %. Os dois últimos períodos tiveram porcentagens bastante parecidas, porém o segundo período teve uma maior porcentagem. Logo, a lógica constatada é se o participante não desistir logo no começo, que é a maioria dos casos, ele provavelmente desistirá em meados da metade do curso, poucos desistem no final do curso.

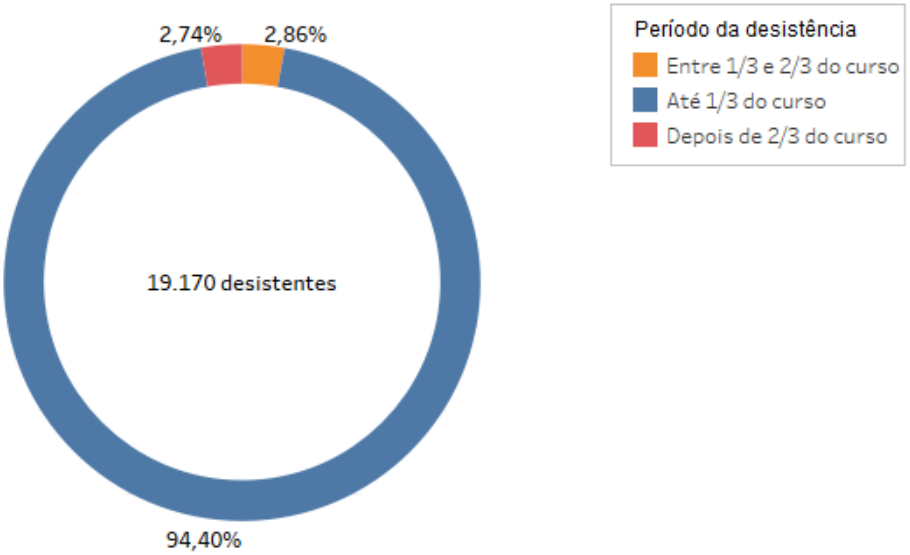


Figura 4.12: Gráfico da relação entre as desistências e os períodos em que elas ocorrem .

No gráfico da Figura 4.13, pode ser visto que a maioria dessas desistências ocorrem quando o participante só fez até 1/3 de todos os módulos disponíveis dos cursos, que corresponde à porcentagem de 84,43 %. Logo após, com a porcentagem de 9,98 %, são os participantes que fizeram entre 1/3 e 2/3 dos módulos e desistiram dos cursos. Por último, vem os participantes que desistiram com mais de 2/3 dos módulos feitos, com a porcentagem de 5,59 %. A lógica segue a mesma linha da Figura 4.12, no qual se o participante não desistir com 1/3 dos módulos feitos do total de módulos, a tendência é ele desistir aproximadamente entre 1/3 e 2/3 dos módulos feitos e, dificilmente, desistir depois de ter feito mais do que 2/3 dos módulos do curso.

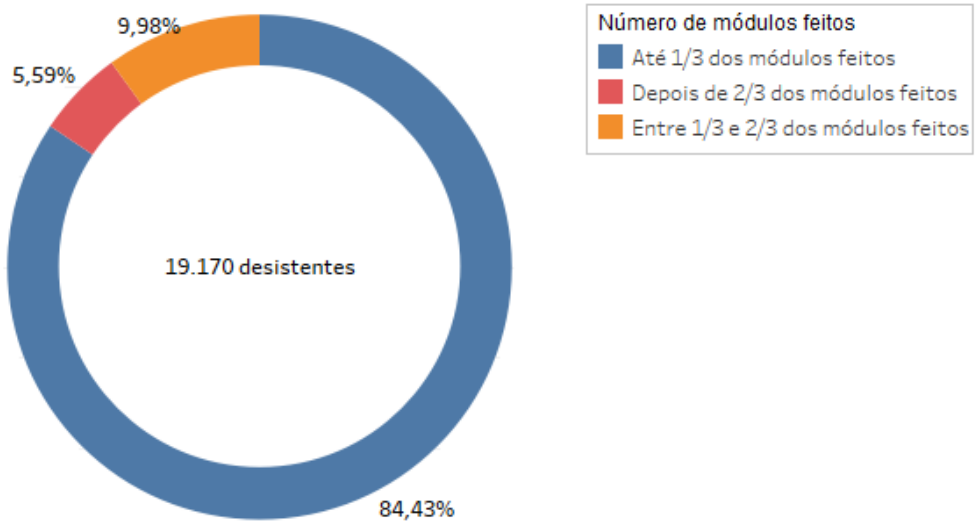


Figura 4.13: Gráfico da relação das desistências com número de módulos feitos nos cursos.

Capítulo 5

Conclusão e Trabalhos Futuros

Esse trabalho de conclusão de curso consistiu na construção de um *framework* de EDM através da modelagem das variáveis do Moodle advindas da base de dados da Enap durante os anos de 2015 a 2016. Para entender melhor a influência das variáveis padrão do Moodle e das variáveis propostas, nas taxas de desempenho e abandono dos participantes, foram utilizados diversos algoritmos de aprendizado de máquina.

Entre esse algoritmos, a performance do Adaboost baseado em árvore de decisão superou os desempenhos das outras abordagens utilizadas e, principalmente, do trabalho [8] realizado na mesma base de dados desse trabalho. Com esse algoritmo, foi obtido uma precisão de 89 %, uma revocação e medida de F_1 de 88 % cada, em termos de avaliação do desempenho dos participantes. Essa alta performance leva a conclusão de que o resultado da classificação e predição foi significativamente boa.

A partir das árvores de decisão geradas, foi possível entender quais são as características dos alunos aprovados e reprovados. Isso tem um grande significado, pois ao mapear essas ações, é possível entender como o desempenho e participação de um participante em cada atividade (essencial ou complementar), a interação com os tutores e com os módulos dos cursos etc, podem influenciar diretamente no resultado da sua nota final. Logo, com esse novo entendimento, este trabalho atinge uma das principais motivações da EDM, que é gerar um conhecimento, até então, desconhecido e que quando descoberto, pode ser bastante útil para o ambiente educacional.

A análise de agrupamento ou clusterização mostrou que pode contribuir para a análise dos comportamentos que, quando observados separadamente, possibilitam o planejamento de ações pedagógicas específicas com maior efetividade. Esse resultado também tem bastante importância, pois um dos motivos da EDM ter crescido continuamente, é pelo fato dela possibilitar aos educadores, uma melhor compreensão dos processos de ensino, de aprendizagem e de motivação dos alunos tanto em ambientes individuais quanto em ambientes colaborativos de ensino.

Ademais, a análise das desistências nos cursos mostrou que boa parte dos participantes, cerca de 94 %, desiste no começo dos cursos e com menos de 1/3 dos módulos feitos, correspondendo à aproximadamente 84 % do total. E o principal fator dessas desistências é a falta de tempo, com cerca de 61 % do total dos casos.

Portanto, compreende-se que este trabalho contribui ao meio científico, de modo a facilitar e direcionar novas pesquisas na área de mineração de dados educacionais, tanto em cursos ministrados à distância quanto cursos presenciais.

5.1 Trabalhos Futuros

Como perspectiva de trabalho futuro é sugerida a construção de um sistema de comunicação efetiva e direta com os participantes da Enap, de modo que, utilizando os modelos de classificação deste trabalho, o sistema alertasse aos participantes que seguem certas características que resultam em uma reprovação. Essa abordagem poderia diminuir o número de reprovações nos cursos, pois muitas vezes os alunos se descuidam ou não sabem como devem seguir na direção correta para serem aprovados nos cursos.

Outra perspectiva de trabalho futuro é a construção e modelagem de um depósito de dados, do inglês *Data Warehouse*, com o objetivo de unificar as bases de dados educacionais do Moodle, não só da Enap. Isso favorece a análise de grandes volumes de dados e a obtenção de informações estratégicas que podem facilitar a tomada de decisão no ambiente educacional em geral.

Referências

[1] Gerhardt, L.A.: *The future of distance learning: the process and the product.* IEEE International Conference on Information Technology Based Higher Education and Training, 2005. 1, 2

[2] Parack, S., Z. Zahid e F. Merchant: *Application of data mining in educational databases for predicting academic trends and patterns.* IEEE International Conference on Technology Enhanced Education, 2012. 1

[3] Sachin, R. B. e M. S. Vijay: *A survey and future vision of data mining in educational field.* IEEE Second International Conference on Advanced Computing and Communication Technologies, 2012. 2

[4] Baker, R.S.J.D.: *Data Mining for Education*, volume 3. Oxford, UK: Elsevier, 2010. 2

[5] Baker, R. S. J., S. Isotani e A. M. J. B. Carvalho: *Mineração de dados educacionais: Oportunidades para o brasil.* Revista Brasileira de Informática na Educação, 19(2), 2011. 2, 3, 11

[6] Romero, C. e S. Ventura: *Educational data mining: a review of the state of the art.* IEEE Transactions on Systems, Man, and Cybernetics, 40, 2010. 2

[7] Almeida, L. R. de, J. P. C. L. da Costa, R. T. de Sousa, E. P. de Freitas, E. D. Canedo, J. B. Prettz, E. Zacarias e G. D. Galdo: *Motivating attendee's participation in distance learning via an automatic messaging plugin for the moodle platform.* IEEE Frontiers in Education Conference (FIE), 2016. 4

[8] Coelho, V. C. G., J. P. C. L. da Costa, D. A. da Silva, R. T. de S. Júnior, F. L. L. de Mendonça e D. G. Silva: *Mineração de dados educacionais no ensino à distância governamental.* Conferências Ibero-Americanas WWW/Internet e Computação Aplicada, 2016. 4, 35, 46, 47, 58

[9] Taylor, K.D., J.W. Honchell e W.E. DeWitt: *Distance learning in courses with a laboratory.* IEEE Technology-Based Re-Engineering Engineering Education Proceedings of Frontiers in Education FIE'96 26th Annual Conference, 1996. 6

[10] Ramos, T., A. Gomes, M. Lucena, I. Nunes, R. Valentim e G. Nóbrega: *Use of educational data mining to identify distance learning students' profiles and patterns of participation.* IEEE Iberian Conference on Information Systems and Technologies (CISTI), 2017. 6

[11] Shangping, D. e Z. Ping: *A data mining algorithm in distance learning.* IEEE International Conference on Computer Supported Cooperative Work in Design, 2008. 7

[12] Banu, R. K. e R. Ravanan: *Analysis of e-learning in data mining — a dreamed vision for empowering rural students in india.* IEEE International Conference on Recent Trends in Information Technology (ICRTIT), 2011. 7

[13] Yubing, A. e Z. Jianping: *The application of data mining technology in distance learning evaluation.* IEEE International Forum on Information Technology and Applications, 2010. 7

[14] Tan, P. N., M. Steinbach e V. Kumar: *Introduction to data mining*, volume 1. Addison-Wesley Longman Publishing Co. Inc, Boston, MA, 2005. 7, 9, 12, 15, 16, 18, 20, 22, 24, 25, 28, 29, 37, 38, 39

[15] Agarwal, S.: *Data mining - data mining concepts and techniques.* IEEE International Conference on Machine Intelligence and Research Advancement, 2013. 7, 8

[16] Lakshmi, B. N. e G.H. Raghunandhan: *A conceptual overview of data mining.* IEEE National Conference on Innovations in Emerging Technology, 2011. 8

[17] Wu, X., X. Zhu, G. Wu e W. Ding: *Data mining with big data.* IEEE Transactions on Knowledge and Data Engineering, 26, 2013. 9

[18] Patel, F. N.: *Large high dimensional data handling using data reduction.* IEEE International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016. 9

[19] Kuleshov, A., A. Bernstein e Y. Yanovich: *High-dimensional density estimation for data mining tasks.* IEEE International Conference on Data Mining Workshops (ICDMW), 2017. 9

[20] Kalra, M. e N. Lal: *Data mining of heterogeneous data with research challenges.* IEEE Symposium on Colossal Data Analysis and Networking (CDAN), 2016. 9

[21] Klusch, M., S. Lodi e M. Gianluca: *The role of agents in distributed data mining: issues and benefits.* IEEE/WIC International Conference on Intelligent Agent Technology, 2003. 9

[22] Romero, C. e S. Ventura: *Educational data mining - a survey from 1995 to 2005.* Expert Systems with Applications, 33(1):135–146, 2007. 10

[23] Baker, R. e K. Yacef: *The state of educational data mining in 2009: A review and future visions.* J. Educ. Data Mining, 1(1):3–17, 2009. 10, 11

[24] Romero, C. e S. Ventura: *Data mining in education.* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 3(1):12–27, 2003. 10

[25] Castro, F., A. Vellido, A. Nebot e F. Mugica: *Applying data mining techniques to e-learning problems.* Evolution of Teaching and Learning Paradigms in Intelligent Environment (Studies in Computational Intelligence), 62:183–221, 2007. 11

[26] Brusilovsky, P. e C. Peylo: *Adaptive and intelligent web-based educational systems.* International Journal of Artificial Intelligence in Education, 13:156–169, 2003. 11

[27] Khan, M. A., W. Gharibi e S. K. Pradhan: *Data mining techniques for business intelligence in educational system: A case mining.* IEEE World Congress on Computer Applications and Information Systems (WCCAIS), 2014. 11

[28] Costa, E., R. S. Baker, L. Magalhães e T. Marinho: *Mineração de dados educacionais: Conceitos, técnicas, ferramentas e aplicações.* Jornada de Atualização em Informática na Educação, 1(1):1–29, 2013. 12

[29] Ahmed, S., R. Paul e A. S. M. L. Hoque: *Knowledge discovery from academic data using association rule mining.* IEEE International Conference on Computer and Information Technology (ICCIT), 2014. 12

[30] Ramos, J. L. C., R. E. D. e Silva, J. C. S. S., R. L. Rodrigues e A. S. Gomes: *A comparative study between clustering methods in educational data mining.* IEEE Latin America Transactions, 14(8):3775–3761, 2016. 13

[31] Reddy, L. S., V. B. Velpula, S. Sailaja B. R. e K. B. Madhavi: *Finding peculiar students from student database using outlier analysis - data mining approach.* IEEE International Conference on MOOC, Innovation and Technology in Education (MITE), 2014. 13

[32] Angra, S. e S. Ahuja: *Machine learning and its applications - a review.* IEEE International Conference on Big Data Analytics and Computational Intelligence (ICBDAC), 2017. 13

[33] Alpaydın, E.: *Introduction to machine learning.* The MIT Press, 3, 2014. 14, 15, 16

[34] Yuntian, W.: *Based on machine learning of data mining to further explore.* IEEE International Conference on Computer Science and Information Processing (CSIP), 2012. 14

[35] Antony, P. J., P. Manujesh e N. A. Jnanesh: *Data mining and machine learning approaches on engineering materials — a review.* IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT), 2016. 14

[36] Singh, A., N. Thakur e A. Sharma: *A review of supervised machine learning algorithms.* IEEE International Conference on Computing for Sustainable Global Development (INDIACom), 2016. 14

[37] Li, X. e N. Ye: *A supervised clustering and classification algorithm for mining data with mixed variables.* IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 36(2):396–406, 2006. 16

[38] Kesavaraj, G. e S. Sukumaran: *A study on classification techniques in data mining.* IEEE Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), 2013. 16

[39] Li, L. e X. Zhang: *Study of data mining algorithm based on decision tree.* IEEE International Conference On Computer Design and Applications, 2010. 16

[40] Bhargava, N., S. Dayma, A. Kumar e P. Singh: *An approach for classification using simple cart algorithm in weka.* IEEE International Conference on Intelligent Systems and Control (ISCO), 2017. 17

[41] Breiman, L., J. Friedman, R. Olshen e C. Stone: *Classification and Regression Trees*, volume 1. Wadsworth and Belmont and CA, 1984. 17, 19

[42] Jaworski, M., P. Duda e L. Rutkowski: *New splitting criteria for decision trees in stationary data streams.* IEEE International Conference on Intelligent Systems and Control (ISCO), 29(6), 2018. 18

[43] Sofeikov, K. I., I. Y. Tyukin, A. N. Gorban, E. M. Mirkes, D. V. Prokhorov e I. V. Romanenko: *Learning optimization for decision tree classification of non-categorical data with information gain impurity criterion.* IEEE International Joint Conference on Neural Networks (IJCNN), 2014. 18

[44] Kumari, P., P. K. Jain e R. Pamula: *An efficient use of ensemble methods to predict students academic performance.* IEEE International Conference on Recent Advances in Information Technology (RAIT), 2018. 21

[45] Seni, G. e J. Elder: *Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions*, volume 1. Wadsworth, Belmont, CA, 2010. 21, 22

[46] Yoav, F. e R. E. Schapire: *A decision-theoretic generalization of online learning and an application to boosting.* Journal of Computer and System Sciences, 1997. 23

[47] Ho, Tin Kam: *Random decision forests.* IEEE Proceedings of 3rd International Conference on Document Analysis and Recognition, 2002. 24

[48] Qiao, J. e Y. Zhang: *Study on k-means method based on data-mining.* IEEE Chinese Automation Congress (CAC), 2015. 25, 27

[49] Arthur, D. e S. Vassilvitskii: *k-means++: The advantages of careful seeding.* Proc. of the ACM-SIAM symposium on Discrete algorithms, 2006. 25

[50] Larson, R.: *Elementary Linear Algebra*, volume 7. Wadsworth, Belmont, CA, 1994. 26

[51] Mitchell, T.: *Machine Learning*, volume 1. McGraw-Hill Science, 1997. 27, 28

[52] Norvig, P. e S. J. Russell: *Artificial Intelligence: A Modern Approach*, volume 3. Prentice Hall, 1994. 28

[53] Powers, D. M. W.: *Evaluation: From precision, recall and f-factor to roc, informedness, markedness correlation.* Journal of Machine Learning Technologies, 2(1):37–63, 2011. 28, 29, 30

[54] Marom, N. D., L. Rokach e A. Shmilovici: *Using the confusion matrix for improving ensemble classifiers*. IEEE Convention of Electrical and Electronics Engineers in Israel, 2010. 30

[55] *Moodle. community driven, globally supported*. Disponível em: https://moodle.org/. 31

[56] Sowmya, R. e K. R. Suneetha: *Data mining with big data*. IEEE International Conference on Intelligent Systems and Control (ISCO), 2017. 33

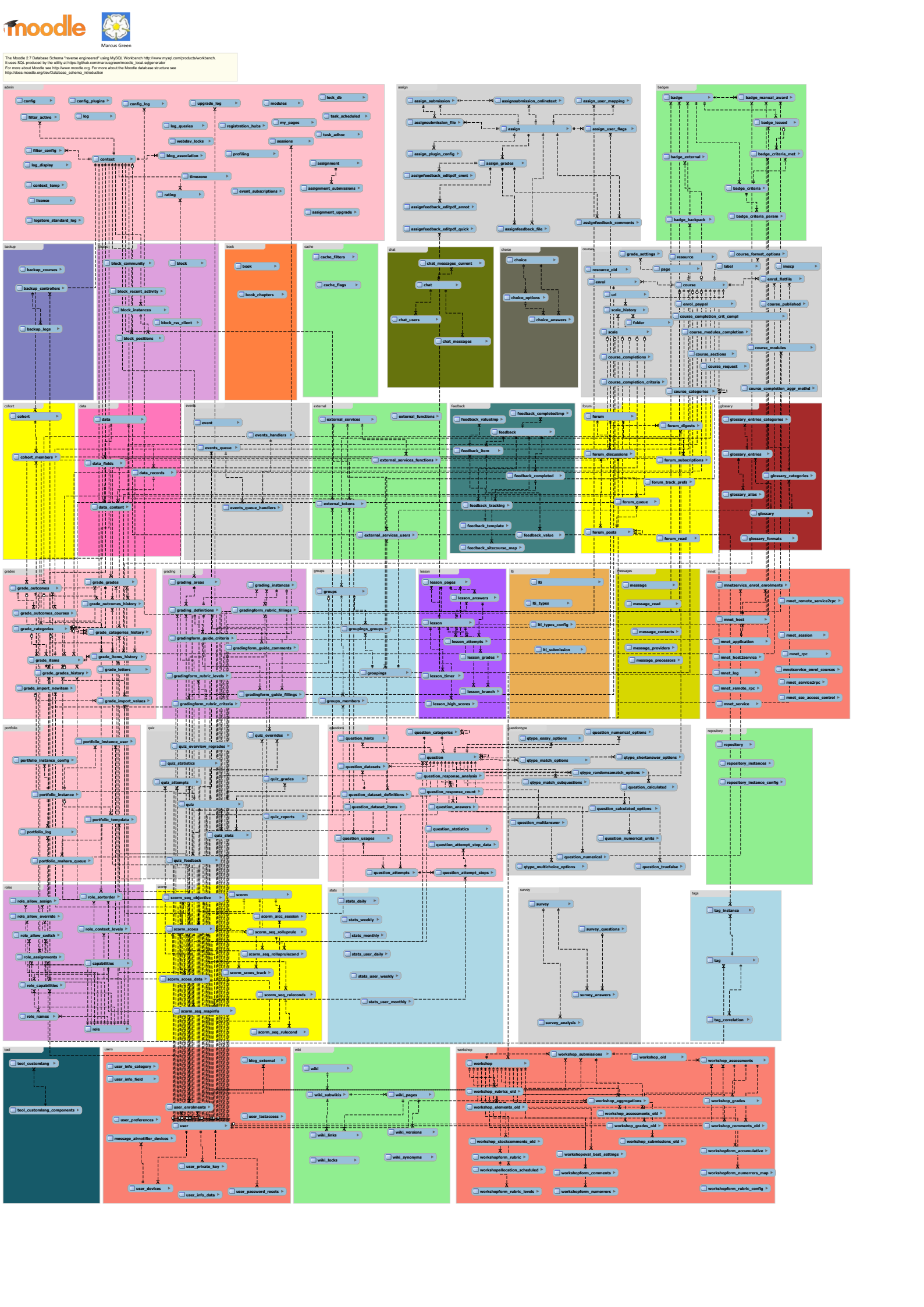
[57] *Postgresql: The world's most advanced open source relational database*. Disponível em: https://www.postgresql.org/. 33

[58] Saleem, A., K. H. Asif, A. Ali, S. M. Awan e M. A. Alghamdi: *Pre-processing methods of data mining*. IEEE/ACM International Conference on Utility and Cloud Computing, 2014. 38

[59] Mitrpanont, J., W. Sawangphol, T. Vithantirawat e S. Paengkaew: *A study on using python vs weka on dialysis data analysis*. IEEE International Conference on Information Technology (INCIT), 2017. 40

Apêndice A

Modelo Entidade Relacionamento do Moodle


Marcus Green
The Moodle 2.7 Database Schema "reverse engineered" using MySQL Workbench http://www.mysql.com/products/workbench.
It uses SQL produced by the utility at https://github.com/marcusgreen/moodle_local-sqlgenerator
For more about Moodle see http://www.moodle.org. For more about the Moodle database structure see
http://docs.moodle.org/dev/Database_schema_introduction
admin
config
config_plugins
config_log
upgrade_log
modules
lock_db
filter_active
log
log_queries
registration_hubs
my_pages
task_scheduled
webdav_locks
sessions
task_adhoc
filter_config
context
blog_association
profiling
log_display
assignment
timezone
context_temp
rating
event_subscriptions
assignment_submissions
license
assignment_upgrade
logstore_standard_log
assign
assign_submission
assignsubmission_onlinetext
assign_user_mapping
assignsubmission_file
assign
assign_user_flags
assign_plugin_config
assign_grades
assignfeedback_editpdf_cmnt
assignfeedback_editpdf_annot
assignfeedback_editpdf_quick
assignfeedback_file
assignfeedback_comments
badges
badge
badge_manual_award
badge_issued
badge_external
badge_criteria_met
badge_criteria
badge_backpack
badge_criteria_param
backup
backup_courses
backup_controllers
backup_logs
blocks
block_community
block
block_recent_activity
block_instances
block_rss_client
block_positions
book
book
book_chapters
cache
cache_filters
cache_flags
chat
chat_messages_current
chat
chat_users
chat_messages
choice
choice
choice_options
choice_answers
courses
grade_settings
resource
course_format_options
resource_old
page
label
imscp
enrol
course
enrol_flatfile
url
enrol_paypal
course_published
scale_history
course_completion_crit_compl
folder
scale
course_modules_completion
course_modules
course_completions
course_sections
course_request
course_completion_criteria
course_categories
course_completion_aggr_methd
cohort
cohort
cohort_members
data
data
data_fields
data_records
data_content
events
event
events_handlers
events_queue
events_queue_handlers
external
external_services
external_functions
external_services_functions
external_tokens
external_services_users
feedback
feedback_valuetmp
feedback_completedtmp
feedback
feedback_item
feedback_completed
feedback_tracking
feedback_template
feedback_value
feedback_sitecourse_map
forum
forum
forum_digests
forum_discussions
forum_subscriptions
forum_track_prefs
forum_queue
forum_posts
forum_read
glossary
glossary_entries_categories
glossary_entries
glossary_categories
glossary_alias
glossary
glossary_formats
grades
grade_outcomes
grade_grades
grade_outcomes_history
grade_outcomes_courses
grade_categories
grade_categories_history
grade_items
grade_items_history
grade_letters
grade_grades_history
grade_import_newitem
grade_import_values
grading
grading_areas
grading_instances
grading_definitions
gradingform_rubric_fillings
gradingform_guide_criteria
gradingform_guide_comments
gradingform_rubric_levels
gradingform_guide_fillings
gradingform_rubric_criteria
groups
groups
groupings_groups
groupings
groups_members
lesson
lesson_pages
lesson_answers
lesson
lesson_attempts
lesson_grades
lesson_timer
lesson_branch
lesson_high_scores
lti
lti
lti_types
lti_types_config
lti_submission
messages
message
message_read
message_contacts
message_providers
message_processors
mnet
mnetservice_enrol_enrolments
mnet_remote_service2rpc
mnet_host
mnet_session
mnet_application
mnet_rpc
mnet_host2service
mnetservice_enrol_courses
mnet_log
mnet_service2rpc
mnet_remote_rpc
mnet_sso_access_control
mnet_service
portfolio
portfolio_instance_user
portfolio_instance_config
portfolio_instance
portfolio_tempdata
portfolio_log
portfolio_mahara_queue
quiz
quiz_overrides
quiz_overview_regrades
quiz_statistics
quiz_attempts
quiz_grades
quiz
quiz_reports
quiz_slots
quiz_feedback
question
question_hints
question_categories
question
question_datasets
question_response_analysis
question_response_count
question_dataset_definitions
question_dataset_items
question_answers
question_statistics
question_attempt_step_data
question_usages
question_attempts
question_attempt_steps
questiontype
qtype_essay_options
question_numerical_options
qtype_match_options
qtype_shortanswer_options
qtype_randomsamatch_options
qtype_match_subquestions
question_calculated
question_calculated_options
question_multianswer
question_numerical_units
question_numerical
qtype_multichoice_options
question_truefalse
repository
repository
repository_instances
repository_instance_config
roles
role_allow_assign
role_sortorder
role_allow_override
role_context_levels
role_allow_switch
role_assignments
capabilities
role_capabilities
role_names
role
scorm
scorm_seq_objective
scorm
scorm_aicc_session
scorm_scoes
scorm_seq_rolluprule
scorm_seq_rolluprulecond
scorm_scoes_track
scorm_scoes_data
scorm_seq_ruleconds
scorm_seq_mapinfo
scorm_seq_rulecond
stats
stats_daily
stats_weekly
stats_monthly
stats_user_daily
stats_user_weekly
stats_user_monthly
survey
survey
survey_questions
survey_answers
survey_analysis
tags
tag_instance
tag
tag_correlation
tool
tool_customlang
tool_customlang_components
users
user_info_category
user_info_field
user_enrolments
user
user_preferences
message_airnotifier_devices
user_private_key
user_devices
user_info_data
blog_external
user_lastaccess
user_password_resets
wiki
wiki
wiki_subwikis
wiki_pages
wiki_links
wiki_versions
wiki_locks
wiki_synonyms
workshop
workshop_submissions
workshop_old
workshop
workshop_assessments
workshop_rubrics_old
workshop_aggregations
workshop_grades
workshop_elements_old
workshop_assessments_old
workshop_grades_old
workshop_comments_old
workshop_stockcomments_old
workshop_submissions_old
workshopform_accumulative
workshopform_rubric
workshopeval_best_settings
workshopallocation_scheduled
workshopform_comments
workshopform_numerrors_map
workshopform_rubric_levels
workshopform_numerrors
workshopform_rubric_config

Apêndice B

Código da consulta completa para retornar as 26 variáveis utilizadas nos algoritmos

In [ ]:

```
SELECT u.id as userid,u.firstname,u.lastname,u.email,u.city,u.country,c.fullname, c.id as course_id
, g.finalgrade,
case when  g.finalgrade > 60.0 then 1 else 0 end as passed,
case when  (SELECT COUNT(compl.id) AS countrecord FROM mdl_course_modules_completion compl
INNER JOIN mdl_course_modules m ON compl.coursemoduleid = m.id
WHERE compl.userid=u.id AND m.course=c.id AND m.completion > 0 AND compl.completionstate > 0) >
(SELECT COUNT(cm.id) as num_modules FROM mdl_course_modules cm
INNER JOIN mdl_modules m ON m.id=cm.module
WHERE  cm.course=c.id)/1.1 then 1 else 0
end as ninety_modules_done, case when  (SELECT COUNT(compl.id) AS countrecord
                                        FROM mdl_course_modules_completion compl
INNER JOIN mdl_course_modules m ON compl.coursemoduleid = m.id
WHERE compl.userid=u.id AND m.course=c.id AND m.completion > 0 AND compl.completionstate > 0) >
(SELECT COUNT(cm.id) as num_modules
FROM mdl_course_modules cm
INNER JOIN mdl_modules m ON m.id=cm.module
WHERE  cm.course=c.id)/1.65 then 1 else 0 end as sixty_modules_done,
case when  (SELECT COUNT(compl.id) AS countrecord FROM mdl_course_modules_completion compl
INNER JOIN mdl_course_modules m ON compl.coursemoduleid = m.id
WHERE compl.userid=u.id AND m.course=c.id AND m.completion > 0 AND compl.completionstate > 0) >
(SELECT COUNT(cm.id) as num_modules
FROM mdl_course_modules cm
INNER JOIN mdl_modules m ON m.id=cm.module
WHERE  cm.course=c.id)/3.3 then 1  else 0  end as thirty_modules_done,
(SELECT COUNT(compl.id) AS countrecord FROM mdl_course_modules_completion compl
INNER JOIN mdl_course_modules m ON compl.coursemoduleid = m.id
WHERE compl.userid=u.id AND m.course=c.id AND m.completion > 0 AND compl.completionstate > 0) as
num_modules_done,
(SELECT COUNT(cm.id) as num_modules
FROM mdl_course_modules cm
INNER JOIN mdl_modules m ON m.id=cm.module
WHERE  cm.course=c.id) as num_modules_have,
CASE WHEN (SELECT (SUM(qg.grade) / COUNT(qg.id)) as grade_mean_quiz FROM mdl_quiz_grades qg
INNER JOIN mdl_quiz q ON qg.quiz=q.id
INNER JOIN mdl_course cs ON q.course=cs.id
INNER JOIN mdl_user us ON us.id=qg.userid
WHERE qg.grade > 0.00 AND us.id=u.id AND cs.id=c.id
) IS NULL THEN 0.0
ELSE (SELECT (SUM(qg.grade) / COUNT(qg.id)) as grade_mean_quiz FROM mdl_quiz_grades qg
INNER JOIN mdl_quiz q ON qg.quiz=q.id
INNER JOIN mdl_course cs ON q.course=cs.id
INNER JOIN mdl_user us ON us.id=qg.userid
WHERE qg.grade > 0.00 AND us.id=u.id AND cs.id=c.id) END AS mean_quiz_grade,
(SELECT (stddev(qg.grade) ) as dev_mean_quiz FROM mdl_quiz_grades qg
INNER JOIN mdl_quiz q ON qg.quiz=q.id
INNER JOIN mdl_course cs ON q.course=cs.id
INNER JOIN mdl_user us ON us.id=qg.userid
WHERE qg.grade > 0.00 AND us.id=u.id AND cs.id=c.id) AS std_dev_quiz_grade,
(SELECT COUNT(qg.id) as count_quiz_made FROM mdl_quiz_grades qg
INNER JOIN mdl_quiz q ON qg.quiz=q.id
INNER JOIN mdl_course cs ON q.course=cs.id
INNER JOIN mdl_user us ON us.id=qg.userid
WHERE qg.grade > 0.00 AND us.id=u.id AND cs.id=c.id) AS count_quiz_made,
CASE WHEN (SELECT (SUM(ag.grade) / COUNT(ag.id)) as mean_assign FROM mdl_assign_grades ag
INNER JOIN mdl_assign a ON ag.assignment=a.id
INNER JOIN mdl_course cs ON a.course=cs.id
INNER JOIN mdl_user us ON us.id=ag.userid
where us.id=u.id and cs.id=c.id) IS NULL THEN 0.0
ELSE (SELECT (SUM(ag.grade) / COUNT(ag.id)) as mean_assign FROM mdl_assign_grades ag
INNER JOIN mdl_assign a ON ag.assignment=a.id
INNER JOIN mdl_course cs ON a.course=cs.id
INNER JOIN mdl_user us ON us.id=ag.userid
where us.id=u.id and cs.id=c.id)
END AS mean_assign_grade,
(SELECT (stddev(ag.grade)) as dev_assign FROM mdl_assign_grades ag
INNER JOIN mdl_assign a ON ag.assignment=a.id
INNER JOIN mdl_course cs ON a.course=cs.id
INNER JOIN mdl_user us ON us.id=ag.userid
where us.id=u.id and cs.id=c.id) AS std_dev_assign_grade,
(SELECT (COUNT(ag.id)) as grade_mean_assign FROM mdl_assign_grades ag
INNER JOIN mdl_assign a ON ag.assignment=a.id
INNER JOIN mdl_course cs ON a.course=cs.id
```

```
INNER JOIN mdl_user us ON us.id=ag.userid
where us.id=u.id and cs.id=c.id) as count_assign_made,
(SELECT  COUNT(p.id) as num_posts
FROM mdl_forum_posts p
INNER JOIN mdl_forum_discussions s ON p.discussion=s.id
INNER JOIN mdl_forum f ON f.id=s.forum
INNER JOIN mdl_course ce ON ce.id=f.course
INNER JOIN mdl_user us ON us.id=s.userid
where ce.id=c.id and us.id=u.id) as num_forum_posts,
(SELECT  count(s.id) as num_reads
FROM mdl_forum_read s
INNER JOIN mdl_forum f ON f.id=s.forumid
INNER JOIN mdl_course ce ON ce.id=f.course
INNER JOIN mdl_user us ON us.id=s.userid
where ce.id=c.id and us.id=u.id) as num_forum_reads,
(SELECT COUNT(s.id)
FROM mdl_forum_discussions s
INNER JOIN mdl_forum f ON f.id=s.forum
INNER JOIN mdl_course ce ON ce.id=f.course
INNER JOIN mdl_user us ON us.id=s.userid
where ce.id=c.id and us.id=u.id) as num_forum_discussions,
(SELECT  count(s.id)
FROM mdl_forum_subscriptions s
INNER JOIN mdl_forum f ON f.id=s.forum
INNER JOIN mdl_course ce ON ce.id=f.course
INNER JOIN mdl_user us ON us.id=s.userid
where ce.id=c.id and us.id=u.id
) as num_forum_sub,
(SELECT COUNT(cm.id) FROM mdl_chat_messages cm
INNER JOIN mdl_chat ca ON ca.id=cm.chatid
INNER JOIN mdl_course cs ON cs.id=ca.course
INNER JOIN mdl_user us ON us.id=cm.userid
where us.id=u.id AND cs.id=c.id) as num_chats,
CASE WHEN
(SELECT COUNT(us.id) as num_tutores FROM mdl_role_assignments rs
INNER JOIN mdl_user us ON us.id=rs.userid
INNER JOIN mdl_context e ON rs.contextid=e.id
INNER JOIN mdl_course cs ON cs.id=e.instanceid
WHERE e.contextlevel=50 AND rs.roleid!=5 AND cs.id=c.id
GROUP BY cs.id) IS NULL THEN 0 ELSE
(SELECT COUNT(us.id) as num_tutores FROM mdl_role_assignments rs
INNER JOIN mdl_user us ON us.id=rs.userid
INNER JOIN mdl_context e ON rs.contextid=e.id
INNER JOIN mdl_course cs ON cs.id=e.instanceid
WHERE e.contextlevel=50 AND rs.roleid!=5 AND cs.id=c.id
GROUP BY cs.id) END as num_tutors,
(SELECT COUNT(l.id) FROM mdl_role_assignments rs
INNER JOIN mdl_user u ON u.id=rs.userid
INNER JOIN mdl_context e ON rs.contextid=e.id
INNER JOIN mdl_course cs ON cs.id=e.instanceid
INNER JOIN mdl_logstore_standard_log l ON l.courseid=cs.id AND rs.userid=l.userid
WHERE e.contextlevel=50 AND rs.roleid!=5 AND cs.id=c.id
GROUP BY cs.id) as num_act_tutors,
(SELECT count(l.id) FROM mdl_role_assignments rs
INNER JOIN mdl_user us ON us.id=rs.userid
INNER JOIN mdl_context e ON rs.contextid=e.id
INNER JOIN mdl_course cs ON cs.id=e.instanceid
INNER JOIN mdl_logstore_standard_log l ON l.courseid=cs.id AND rs.userid=l.userid
WHERE e.contextlevel=50 AND rs.roleid=5 AND cs.id=c.id AND us.id=u.id) as num_act_user,
to_timestamp(g.timemodified)::timestamp AS timemodified FROM mdl_grade_items i
INNER JOIN mdl_course c ON c.id=i.courseid
INNER JOIN mdl_grade_grades g ON i.id=g.itemid
INNER JOIN mdl_user u ON u.id=g.userid
WHERE i.itemtype = 'course' AND u.deleted=0 AND u.confirmed=1 ORDER by u.firstname,u.lastname
```

Apêndice C

Código da consulta para extrair os dados dos participantes desistentes

In [ ]:

```
SELECT u.id as userid, u.firstname, u.lastname, c.fullname, c.id as courseid,

(SELECT COUNT(compl.id) AS countrecord FROM mdl_course_modules_completion compl
INNER JOIN mdl_course_modules m ON compl.coursemoduleid = m.id
WHERE compl.userid=u.id AND m.course=c.id AND m.completion > 0 AND compl.completionstate > 0) as
num_modules_done,

(SELECT COUNT(cm.id) as num_modules
FROM mdl_course_modules cm
INNER JOIN mdl_modules m ON m.id=cm.module
WHERE  cm.course=c.id) as num_modules_have

(SELECT COUNT(us.id) as num_tutores FROM mdl_role_assignments rs
INNER JOIN mdl_user us ON us.id=rs.userid
INNER JOIN mdl_context e ON rs.contextid=e.id
INNER JOIN mdl_course cs ON cs.id=e.instanceid
WHERE e.contextlevel=50 AND rs.roleid!=5 AND cs.id=c.id
GROUP BY cs.id) as num_tutors,

case
    when en.justification='Falta de tempo para realizar o curso' then 1
    when en.justification='Curso é mais complexo do que eu imaginei' then 2
    when en.justification='O tutor não está acompanhando meu desempenho' then 3
    when en.justification='Estou com dificuldades técnicas de acesso ao curso' then 4
    else 0
end as type_justification,

en.justification,
en.comments,
to_timestamp(en.unenroldate)::timestamp as unenrol_date,
to_timestamp(c.timecreated)::timestamp as course_time_created,
to_timestamp(c.startdate)::timestamp as course_start_date,
to_timestamp(c.enddate)::timestamp as course_end_date,
(EXTRACT( DAY FROM (to_timestamp(c.enddate)::timestamp - to_timestamp(c.timecreated)::timestamp)))
as course_duration_days,
(EXTRACT( DAY FROM (to_timestamp(en.unenroldate)::timestamp - to_timestamp(c.startdate)::timestamp
))) as time_spent_days

FROM mdl_enrol_unenrol en
INNER JOIN mdl_course c ON c.id=en.courseid
INNER JOIN mdl_user u ON en.userid=u.id
order by u.firstname, u.lastname
```

Apêndice D

Código em Python do algoritmo CART para árvore de decisão

In [ ]:

```
import csv
import sklearn
import pandas as pd
import numpy as np
import graphviz
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Lendo os dados do arquivo csv
df = pd.read_csv('resultado_arrumado.csv',  low_memory=False)

# Substituindo o valor de NA em número de ações de tutores para zero
df['num_act_tutors'].fillna(0, inplace=True)

# Substituindo o valor de NA nos desvios padrões
df['std_dev_quiz_grade'].fillna(0, inplace=True)
df['std_dev_assign_grade'].fillna(0, inplace=True)

# Removendo coluna a mais chamda de "Unnamed"
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]

# Mostrando as possibilidades da variável se o aluno passou
print("Valores das possibilidades se um aluno passou", df["passed"].unique())
```

In [ ]:

```
# Reduzindo o dataframe para as colunas que vão ser utilizadas na árvore de decisão
df_tree = df[df.columns[9:-1]]

#separando o dataframe em atributos e classes
df_attributes = df_tree.drop('passed', axis=1)
df_class = df_tree['passed']

#deixando só os atributos que são relevantes
df_attributes = df_attributes[['mean_quiz_grade', 'count_quiz_made', 'grade_mean_assign',
                               'count_assign_made',  'num_act_tutors', 'num_act_user' ]]

# Divisão dos dados em conjunto de treinamento (90%) e de teste (10%)
attr_train, attr_test, class_train, class_test = train_test_split(df_attributes, df_class, test_siz
e=0.10)

# Treinando o algoritmo de classificação com o conjunto de treinamento
classifier = DecisionTreeClassifier(max_depth=4, class_weight={0:1,1:1})
classifier.fit(attr_train, class_train)
print(classifier.feature_importances_)
```

In [ ]:

```
# Processo de imprimir a árvore de decisão
dot_data = tree.export_graphviz(classifier, out_file=None, feature_names=list_attr,
                                class_names=['Reprovado', 'Aprovado'],
                                filled=True,
                                rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("arvore_gini")
```

In [ ]:

```
# Utilizando o algoritmo treinado para prever os resultados do conjunto de teste
class_prediction = classifier.predict(attr_test)

# Avaliando os resultados da predição feita com a matriz de confusão
print(confusion_matrix(class_test, class_prediction))
print(classification_report(class_test, class_prediction))
```

Apêndice E

Código em Python dos algoritmos: Adaboost e random forest

In [ ]:

```
import sklearn
import pandas as pd
import numpy as np
import graphviz
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import tree

# Lendo os dados do arquivo csv
df = pd.read_csv('resultado_arrumado.csv',  low_memory=False)

# Reduzindo o dataframe para as colunas que vão ser utilizadas na árvore de decisão
df_tree = df[df.columns[9:-1]]

#separando o dataframe em atributos e classes
df_attributes = df_tree.drop('passed', axis=1)
df_class = df_tree['passed']

#deixando só os atributos que são relevantes
df_attributes =
df_attributes[['mean_quiz_grade','count_quiz_made','grade_mean_assign','count_assign_made',
                               'num_act_tutors','num_act_user', 'ninety_modules_done' ]]

# Divisão dos dados em conjunto de treinamento (90%) e de teste (10%)
attr_train, attr_test, class_train, class_test = train_test_split(df_attributes, df_class, test_siz
e=0.10)

# Treinando o algoritmo de AdaBoost com o conjunto de treinamento
classifier = AdaBoostClassifier(DecisionTreeClassifier(max_depth=4, class_weight={0:1,1:1}),
                         n_estimators=10)
classifier.fit(attr_train, class_train)

# Erro dos classificadores
print classifier.estimator_errors_

# Utilizando o algoritmo treinado para prever os resultados do conjunto de teste
class_prediction = classifier.predict(attr_test)

# Avaliando os resultados da predição feita
print(confusion_matrix(class_test, class_prediction))

# Treinando o algoritmo de RandomForest com o conjunto de treinamento
classifier = RandomForestClassifier(max_depth=4, n_estimators=10)
classifier.fit(attr_train, class_train)

# Utilizando o algoritmo treinado para prever os resultados do conjunto de teste
class_prediction = classifier.predict_proba(attr_test)
predict = classifier.predict(attr_test)

# Margem de cada classificador
print(class_prediction[:10,1], class_prediction[:10,0])

# Avaliando os resultados da predição feita com a matriz de confusão
print(confusion_matrix(class_test, predict))

# Processo de imprimir as duas árvores de decisão
for tree in range(0, 3):
    for tree_in_forest in classifier.estimators_:
        dot_data = tree.export_graphviz(tree_in_forest,out_file=None,feature_names=list_attr,
                                        rounded=True, special_characters=True, filled=True,
                                        class_names=['Reprovado', 'Aprovado'])
        graph = graphviz.Source(dot_data)
        graph.render("arvore_"+tree)
```

Apêndice F

Código em Python do algoritmo k-means para clusterização

In [ ]:

```
import sklearn
import pandas as pd
import numpy as np
import graphviz
import matplotlib.pyplot as plt
import pylab
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D

# Lendo os dados do arquivo csv
df = pd.read_csv('resultado_arrumado.csv',  low_memory=False)

# Reduzindo o dataframe para as colunas que vão ser utilizadas na árvore de decisão
df_tree = df[df.columns[8:-1]]

#separando o dataframe em atributos e classes
df_attributes = df_tree.drop('passed', axis=1)
df_class = df_tree['passed']

#deixando só os atributos que serão utilizados nesta clusterização
df_attributes = df_attributes[['finalgrade','mean_quiz_grade']]

# Sem divisão do conjunto de dados e treinamento (test_size = 0)
attr_train, attr_test, class_train, class_test = train_test_split(df_attributes, df_class, test_siz
e=0)

# Transformando um dataframe num array
X = attr_train.iloc[:, :].values

# Clusterização com o K-Means
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

# valores dos centroides e da soma das distâncias quadradas das amostras para o centróide mais pró
ximo
centroids = kmeans.cluster_centers_
print(centroids)
print(kmeans.inertia_)

#salvar a clusterização gerada em uma figura
plt.figure(figsize=(15,15))
plt.scatter(X[:,0],X[:,1], c=labels, cmap='RdYlGn', s=0.2)
plt.scatter(kmeans.cluster_centers_[:,0] ,kmeans.cluster_centers_[:,1], color='blue', s=500, alpha=
0.5)
plt.xlabel('finalgrade')
plt.ylabel('mean_quiz_grade')
plt.title('K-means com ' + str(round(score*100, 4)) + '% de precisao')
plt.savefig('kmeans-mean_quiz_grade.png')

#Fazendo outra clusterização com 2 outros  atributos
df_attributes = df.drop('passed', axis=1)
df_attributes = df_attributes[['finalgrade','count_quiz_made']]

# Sem divisão do conjunto de dados e treinamento (test_size = 0)
attr_train, attr_test, class_train, class_test = train_test_split(df_attributes, df_class, test_siz
e=0)

# Transformando um dataframe num array
X = attr_train.iloc[:, :].values

kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

# valores dos centroides e da soma das distâncias quadradas das amostras para o centróide mais pró
ximo
centroids = kmeans.cluster_centers_
print(centroids)
print(kmeans.inertia_)

#salvar a clusterização gerada em uma figura
```

```
plt.figure(figsize=(15,15))
plt.scatter(X[:,0],X[:,1], c=labels, cmap='RdYlGn', s=0.2)
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], color='blue', s=500, alpha=
0.5)
plt.xlabel('finalgrade')
plt.ylabel('count_quiz_made')
plt.title('K-means com ' + str(round(score*100, 4)) + '% de precisao')
plt.savefig('kmeans-count_quiz_made.png')

#Fazendo clusterização com três atributos
df_attributes = df_tree.drop('passed', axis=1)
df_attributes = df_attributes[['finalgrade','count_quiz_made', 'mean_quiz_grade']]

attr_train, attr_test, class_train, class_test = train_test_split(df_attributes, df_class, test_siz
e=0)

# Transformando um dataframe num array
X = attr_train.iloc[:, :].values

kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

# valores dos centroides
centroids = kmeans.cluster_centers_
print(centroids)

#soma das distâncias quadradas das amostras para o centróide mais próximo
print(kmeans.inertia_)

# média da soma
print(kmeans.inertia_ / len(kmeans.labels_))

#salvar a clusterização gerada em uma figura
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(X[:,0],X[:,1], X[:,2], c=labels, cmap='RdYlGn', s=5)
ax.set_xlabel('finalgrade')
ax.set_ylabel('count_quiz_made')
ax.set_zlabel('mean_quiz_grade')
ax.set_title('K-means com ' + str(round(score*100, 4)) + '% de precisao')
plt.savefig('kmeans-3d-count-quiz.png')

#Fazendo clusterização com outros três atributos
df_attributes = df.drop('passed', axis=1)
df_attributes = df_attributes[['finalgrade','mean_quiz_grade', 'ninety_modules_done']]

attr_train, attr_test, class_train, class_test = train_test_split(df_attributes, df_class, test_siz
e=0.30)

# Transformando um dataframe num array
X = attr_train.iloc[:, :].values

kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

# valores dos centroides e da soma das distâncias quadradas das amostras para o centróide mais pró
ximo
centroids = kmeans.cluster_centers_
print(centroids)
print(kmeans.inertia_)

#salvar a clusterização gerada em uma figura
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:,0],X[:,1], X[:,2], c=labels, cmap='RdYlGn', s=5)
ax.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], color='blue', s=500, alpha=0
.5)
ax.set_xlabel('finalgrade')
ax.set_ylabel('mean_quiz_grade')
ax.set_zlabel('ninety_modules_done')
ax.set_title('K-means com ' + str(round(score*100, 4)) + '% de precisao')
plt.savefig('kmeans-3d-ninety.png')
plt.show()
```

Apêndice G

Código em Python do algoritmo para a análise dos desistentes

In [ ]:

```
import sklearn
import pandas as pd
import numpy as np
import graphviz
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import tree
from sklearn import svm

# Lendo os dados do arquivo csv
df = pd.read_csv('resultado_desistentes.csv',  low_memory=False)

# Mostrando as possibilidades da variável de porquê o aluno desistiu
print("Valores das possibilidades de por que o aluno desistiu", df["type_justification"].unique())

# Analisando os desistentes por categoria
labels = 'Falta de tempo', 'Outra', 'Curso complexo', 'Tutor nao acompanhando', 'Dificuldades tecni
cas'
sizes= [len(df.loc[df['type_justification'] == 1]), len(df.loc[df['type_justification'] == 0]),
        len(df.loc[df['type_justification'] == 2]),len(df.loc[df['type_justification'] == 3]),
        len(df.loc[df['type_justification'] == 4])]

explode = (0.1, 0, 0, 0, 0)
fig0, ax0= plt.subplots(figsize=(6,5))
ax0.pie(sizes, explode=explode, autopct='%1.1f%%', shadow=True, startangle=90)
ax0.axis('equal')
total0 = sum(sizes)

# Analisando os desistentes por período
labels = 'Ate 1/3 do curso', 'Entre 1/3 e 2/3 do curso', 'Depois de 2/3 do curso'
sizes= [len(df.loc[ (df['time_spent_days'] <= df['course_duration_days']/3)]),
        len(df.loc[ (df['time_spent_days'] > df['course_duration_days']/3) & (df['time_spent_days']
<= df['course_duration_days']/1.5)] ),
        len(df.loc[ (df['time_spent_days'] > df['course_duration_days']/1.5)])]

explode = (0.1, 0, 0)
fig1, ax1 = plt.subplots(figsize=(10,10))
ax1.pie(sizes, explode=explode, autopct='%1.1f%%', shadow=True, startangle=90)
ax1.axis('equal')
total1 = sum(sizes)

# Analisando os desistentes por módulos feitos
labels = 'Ate 1/3 dos modulos feitos', 'Entre 1/3 e 2/3 dos modulos feitos', 'Depois de 2/3 dos mo
dulos feitos'
sizes= [len(df.loc[ (df['num_modules_done'] <= df['num_modules_have']/3)]),
        len(df.loc[ (df['num_modules_done'] > df['num_modules_have']/3) & (df['num_modules_done'] <
= df['num_modules_have']/1.5)]),
        len(df.loc[ (df['num_modules_done'] > df['num_modules_have']/1.5)])]

explode = (0.1, 0, 0)
fig2, ax2 = plt.subplots(figsize=(6,5))
ax2.pie(sizes, explode=explode, autopct='%1.1f%%',shadow=True, startangle=90)
ax2.axis('equal')
total2 = sum(sizes)
for graph in range(0,2):
    plt.title('Total de '+ str(len(df)) +' desistentes')
    plt.legend(
        loc='upper right',
        labels=['%s, %1.1f%%' % (
            l, (float(s) / total) * 100) for l, s in zip(labels, sizes)],
        prop={'size': 11},
        bbox_to_anchor=(0.0, 1),
        bbox_transform=fig1.transFigure
    )
    plt.figsize=(9, 11)
    plt.savefig(graph+'.png',bbox_inches = 'tight')
```