Universidade Federal do Triângulo Mineiro - UFTM

Mestrado Profissional em Matemática em Rede Nacional - PROFMAT

Dissertação de Mestrado

# Classificação de estudantes através de árvores de decisão via Python e RapidMiner

Natalia Gonçalves Caetano

**Uberaba - Minas Gerais**

Julho de 2016

# Classificação de estudantes através de árvores de decisão via Python e RapidMiner

Natalia Gonçalves Caetano

Dissertação de Mestrado apresentada à Comissão Acadêmica Institucional do PROFMAT-UFTM como requisito parcial para obtenção do título de Mestre em Matemática.

**Orientador**: Prof. Dr. Leandro Cruvinel Lemes.

**Uberaba - Minas Gerais**
Julho de 2016

**Catalogação na fonte: Biblioteca da Universidade Federal do Triângulo Mineiro**

C131c

Caetano, Natalia Gonçalves
Classificação de estudantes através de árvores de decisão via Python e RapidMiner / Natalia Gonçalves Caetano. -- 2016.
71 f. : il., fig., graf., tab.

Dissertação (Mestrado Profissional em Matemática em Rede Nacional) -- Universidade Federal do Triângulo Mineiro, Uberaba, MG, 2016
Orientador: Prof. Dr. Leandro Cruvinel Lemes

1. Árvores (Teoria dos grafos). 2. Mineração de dados (Computação). 3. Pesquisa educacional. 4. Estudantes - Avaliação. 5. Python (Linguagem de programação de computador). 6. RapidMiner (Programa de computador). I. Lemes, Leandro Cruvinel. II. Universidade Federal do Triângulo Mineiro. III. Título.

CDU 519.172.1

**Natalia Gonçalves Caetano**

**Classificação de estudantes através de árvores de decisão via Python e RapidMiner**

Dissertação apresentada ao curso de Mestrado Profissional em Matemática em Rede Nacional-PROFMAT, da Universidade Federal do Triângulo Mineiro, como parte das atividades para obtenção do título de Mestre em Matemática.

29 de julho 2016.

**Banca Examinadora**

Prof. Dr. Leandro Cruvinel Lemes
Orientador
Unviversidade Federal do Triângulo Mineiro

Prof. Me. Elder Vicente de Paulo Sobrinho
Universidade Federal do Triângulo Mineiro

Prof. Dr. Edmilson Rodrigues Pinto
Universidade Federal de Uberlândia

*À Deus, porque ele é bom.*

*Ao meu marido, para que o nosso futuro seja próspero.*

# Agradecimentos

Agradeço a Deus, por atender aos meus desejos, por me abençoar e por estar comigo em todos os momentos da minha vida.

Agradeço à Universidade Federal do Triângulo Mineiro pela oportunidade de continuar meus estudos.

Agradeço ao Júnior, meu marido, pelo amor, apoio, compreensão, paciência e dedicação durante esse tempo difícil. Ele, que também é meu colega de profissão, dedicou muito do seu tempo a me ajudar em todas as etapas desse mestrado.

Agradeço à minha mãe pela bondade, dedicação, por fazer da minha vida a razão da dela e por me amar tanto.

Agradeço a todos os meus professores pelos ensinamentos, principalmente ao meu orientador Leandro por destinar seu tempo à minha pesquisa.

Agradeço aos meus colegas de classe pelo tempo de estudo e de descontração, principalmente aos meus companheiros Robson e Max, pelas conversas e risadas compartilhadas durante todo esse tempo.

Agradeço aos meus amigos, especialmente, Larissa e Geanne, pelo apoio e por ouvirem meus desabafos em momentos tão difíceis. Agradeço também e, principalmente, ao Henderson, por empenhar seu tempo em me ajudar com as imagens desta pequisa.

Agradeço à Escola Estadual Professor José Inácio de Souza por compreender este momemto ímpar em minha vida, especialmente à minha amiga Alessandra, não só pelo apoio nos trabalhos compartilhados, mas também pelas palavras de incentivo durante todo esse processo.

Agradeço a todos que, de alguma forma, contribuíram para que este estudo fosse concluído.

*"Mas em todas estas coisas somos*
*mais que vencedores, por meio daquele*
*que nos amou".*
*Romanos 8:37*

# Resumo

Mineração de dados educacionais é uma área de pesquisa que utiliza ferramentas de mineração de dados para interpretar dados nos contextos educacionais. Neste trabalho, utiliza-se a mineração de dados para classificar alunos de acordo com o nível de conhecimento com base em notas e atividades anteriores. No estudo, usa-se dados reais para contrução de modelos atráves do algoritmo de árvore de decisão com o objetivo de avaliar regras de classificação para intervenções didáticas e pedagógicas. A partir dos modelos criados, extraiu-se informações relevantes para previsão de resultados finais e identificação de pontos importantes no desenvolvimento do plano de ensino e aprendizagem dos alunos.
**Palavras-chave:** Mineração de dados, Árvore de decisão, Classificação, Python, Rapidminer.

# Abstract

Educational data mining - EDM is a research field concerned with data mining tools for data analisys over educational datasets. The purpose at this work is to use data mining to classify students according to their knowledge level based in their past grades and activities. We use real data to construct decision tree models with purpose of test classification rules for didactic and pedagogical interventions. Considering the created models, relevant information for student's final result predictions and for important points on teaching and learning process development identifications was obtained.
**Keywords:** Data mining, Decision Tree, Classification, Python, Rapidminer.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# INTRODUÇÃO

Atualmente, grandes volumes de dados são gerados e armazenados diariamente pelos sistemas e transformá-los em conhecimentos úteis é um desafio. Para atender essa necessidade, pode-se utilizar o KDD (*Knowledge Discovery in Databases*). Essa expressão foi formalizada em 1989 em referência ao conceito de procurar conhecimento a partir de bancos de dados. Segundo Fayyad et al. (1996), KDD é um processo de várias etapas, não trivial, para identificação de padrões compreensíveis, válidos, novos e potencialmente úteis a partir de grandes conjuntos de dados. O processo KDD é não trivial, pois a complexidade presente na sua execução não é um processo de computação direta. Já a expressão "padrões válidos" indica que o conhecimento deve ser verdadeiro e adequado ao contexto, e o termo "padrão novo" indica que se deve acrescentar novos conhecimentos aos existentes, trazendo algum benefício, de acordo com Goldschmidt and Passos (2005). Para realizar o processo de KDD, são necessárias algumas fases como seleção, pré-processamento, transformação, mineração de dados e análise, coforme dito por Fayyad et al. (1996).

A fase de seleção de dados é a primeira no processo de descoberta de informação, e é onde dados relevantes são selecionados para a análise. Possui impacto significativo sobre a qualidade do resultado final, uma vez que nessa fase é escolhido o conjunto de dados contendo todas os possíveis atributos.

O pré-processamento é importante no processo de KDD. O intuito dessa etapa é remover dados irrelevantes, recuperar dados incompletos e avaliar possíveis dados diferentes do conjunto.

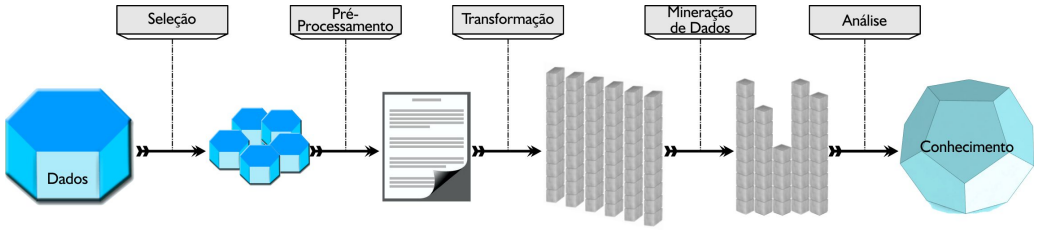
Na fase de transformação, os dados preprocessados serão preparados e transformados para um formato adequado ao algoritmo de mineração.

A mineração de dados é uma etapa onde um método e um algoritmo são selecionados e aplicados com o objetivo de identificar os padrões nos dados transformados.

Na análise, o conhecimento obtido é avaliado para que as decisões sejam tomadas da melhor maneira possível ou documentada para grupos interessados.

Na Figura 1, é possível observar as fases para que os dados se transformem em informação.

Figura 1: Processo de Descoberta do Conhecimento em Bancos de Dados




Embora seja ideal que as etapas sejam executadas na ordem apresentada, o processo pode ser repetido várias vezes, e o resultado de cada uma depende dos resultados das anteriores. Segundo Boente et al. (2008), como os processos para descobrir conhecimento em bancos de dados são elaborados, deve-se planejar o processo desde o início e estabelecer os objetivos de acordo com a aplicação do KDD. O processo de KDD pode ser aplicado em diversas áreas, incluindo marketing, finanças, detecção de fraudes, manufaturas, telecomunicações, educação e outros.

A etapa *data mining*, expressão inglesa que surgiu na década de 90 e também conhecida como mineração de dados, é muito importante no processo de KDD. Ela é responsável pela busca de conhecimento válido dos dados, e sua funcionalidade é organizar dados, buscando padrões ou discrepâncias relevantes. A mineração de dados pode ser uma importante ferramenta que potencializa a inovação e a lucratividade no setor empresarial. Sua utilização pode ser feita em grandes bancos de dados, e as informações podem ser exibidas através de clusterings, nuvem de tags, árvores de decisão e outros. Segundo Witten et al. (1999), o uso de banco de dados em atividades cotidianas traz a mineração de dados para a vanguarda das novas tecnologias empresariais.

Para alguns autores como Fayyad et al. (1996), o KDD e a mineração de dados são processos distintos, mas para outros, como Han et al. (2011), o termo mineração de dados tornou-se mais popular para grandes quantidades de dados armazenados em bancos de dados ou outro tipo de banco de armazenamento. Segundo Berry and Linoff (1997), mineração de dados é a exploração e análise, de forma automática ou semiautomática, de grandes bases de dados com objetivo de descobrir padrões. Para Cabena et al. (1998), mineração de dados é uma área interdisciplinar que mobiliza conhecimentos de análise estatística de dados, aprendizagem de máquina, reconhecimento de padrões e visualização de dados. De acordo com Prass et al. (2012), a mineração de dados traz uma série de ideias e técnicas em diversas áreas. Estatísticos, pesquisadores de Inteligência Artificial e administradores de bancos de dados usam técnicas diferentes para interpretar e avaliar os resultados obtidos com a mineração para, no final, chegar à informação. Atualmente, encontram-se disponíveis várias ferramentas de mineração de dados que ajudam analistas a classificar dados, formular hipóteses, realizar diagnósticos, prever interesses dos clientes

e descobrir perfis de comportamento. Todas essas possibilidades também são utilizadas na área da educação.

A Mineração de Dados Educacionais (do inglês, *Educational Data Mining*, ou EDM) utiliza métodos e ferramentas do campo mais amplo de mineração de dados, segundo Witten et al. (1999). É uma área de pesquisa recente e pouco explorada no Brasil que desenvolve métodos para interpretar dados em contextos educacionais com potencial para melhorar a qualidade do ensino, pois visa entender melhor o discente no seu processo de aprendizagem. Recentemente, muitas instituições de ensino precisam lidar com uma ampliação de cursos, vagas e, consequentemente, com muito dados. Com o auxílio da mineração de dados educacionais é possível descobrir fatores que podem interferir, por exemplo, na evasão ou conclusão dos cursos, o que possibilita identificar falhas e criar medidas para a solução do problema de cada escola ou Universidade. Problemas teóricos e práticos podem ser solucionados com o desenvolvimento e a aplicação de métodos computadorizados, para detectar padrões em grandes conjuntos de dados educacionais difíceis de analisar devido ao grande volume de dados existentes, é o que afirma Romero et al. (2010).

O método de mineração de dados usado nesse trabalho é a árvore de decisão. Segundo Da Silva (2005), árvores de decisão são modelos estatísticos que analisam informações visando a classificação, regressão e previsão de dados. Sua construção é formada por entradas e saídas chamadas classes. Para Gama et al. (2004), o intuito é dividir um problema complexo em subproblemas mais simples e, recursivamente, esta técnica é aplicada a cada subproblema. Mitchell (1997), afirma que as árvores de decisão estão entre os mais populares algoritmos de inferência e têm sido aplicadas em várias áreas como, por exemplo, diagnóstico médico e risco de crédito.

A mineração dos dados deste estudo é feita com o auxílio do Rapidminer, uma plataforma para análise preditiva que dispensa o conhecimento de qualquer linguagem de programação. Assim, o Rapidminer disponibiliza um ambiente completo para análise de dados, seja ela para estudantes, professores, pequenos negócios ou grandes empresas. Qualquer análise produzida pelo Rapidminer, por mais simples que seja, é baseada num conjunto de dados, oferecendo uma interface gráfica fácil de utilizar, que permite a criação rápida de processos para análise de dados e disponibiliza diferentes formas de visualização dos resultados.

Utilizamos, também, Python para classificar estudantes fora da amosta e para criar operadores personalisados. Python é uma linguagem de programação de alto nível, isto é, a linguagem de programação se assemelha muito à linguagem verbal. A escolha do Python deve-se a dois fatores: primeiro por Python ser uma linguagem de programação livre e segundo por ser bem-aceito nas comunidades acadêmicas. Estes dois fatores fizeram com

que o Python tenha uma gama de bibliotecas com métodos interessantes para mineração de dados. O Rapidminer é compatível com Python, ou seja, scripts escritos na linguagem Python podem ser executados pelo Rapidminer. Este fato, faz com que o Rapidminer tenha infinitas opções de operadores personalizados através da linguagem Python.

Este trabalho tem como objetivo classificar alunos, utilizando árvore de decisão, a partir de resultados finais, e, assim, identificar grupos em risco de reprovação para acompanhamento pedagógico e/ou grupos com potenciais para bolsas de estudo e atividades de monitorias. Geralmente, os alunos desempenham inúmeras atividades além das atividades de sala de aula, portanto é difícil acompanhar e reconhecer as necessidades individuais deles. Dessa forma, a adoção de mecanismos semi-automatizados podem viabilizar a detecção precoce de grupos de alunos com risco de desligamento, e tentar minimizar o problema.

O trabalho está organizado da seguinte forma: Inicialmente, temos o Capítulo 1 (Trabalhos Relacionados) com informações de trabalhos desenvolvidos na área de mineração de dados Educacionais, principalmente com o uso da árvore de decisão, o que possibilita melhor compreensão do estudo. Na sequência, temos o Capítulo 2 (Árvores de decisão com Rapidminer) com o recurso tecnológico utilizado no desenvolvimento desse trabalho, assim como instruções de manuseios e um foco maior no operador *Decision tree*, que permite gerar a árvore de decisão. O Capítulo 3 (Aplicações e resultados) consiste na descrição dos processos de construções de dois modelos, que inclui o entendimento dos dados, ferramentas utilizadas, pré-processamento, visualização de dados, construções de modelos de classificação, e, em seguida, as análises dos resultados. Finalmente, no Capítulo 4 (Considerações Finais), concluímos o estudo.

# 1 TRABALHOS RELACIONADOS

Neste capítulo, serão apresentadas duas seções sobre trabalhos desenvolvidos na área de Mineração de Dados Educacionais. Na Seção 1.1, tem-se trabalhos realizados no exterior e na Seção 1.2, tabalhos feitos no Brasil.

## 1.1 Trabalhos Internacionais

Muitos autores realizam análises e propõem ferramentas no contexto de Mineração de Dados Educacionais. Em Al-Radaideh et al. (2011), foi construído um modelo de classificação para prever a melhor rotina de estudos para os alunos da escola. Os dados consistem de 248 casos que foram coletados de seis escolas básicas na Cidade Mafraq, na Jordânia. A árvore de decisão gerada com algoritmo *C4.5*, chegou a um total de 87% de precisão.

Em Baradwaj and Pal (2012), o desempenho dos estudantes foi analisado utilizando o algoritmo de classificação chamado *ID3*, para prever as notas dos estudantes no final do semestre. Este foi aplicado no curso de aplicações informáticas de 2007 a 2010 na Universidade de Purvanchal VBS, Uttar Pradesh. Seu estudo foi destinado a ajudar alunos e professores a encontrarem formas de melhorar seus desempenhos. Os dados foram coletados a partir de 50 alunos, e, em seguida, um conjunto de regras foi extraído para análise.

Outro estudo que usou técnicas de mineração de dados para melhorar o desempenho dos alunos foi feito por El-Halees (2009). Os dados consistiram de 151 casos de um sistema de gestão de base de dados realizado na Universidade de Gaza. Os dados foram coletados a partir de registros pessoais e acadêmicos dos estudantes. O autor utilizou as técnicas de mineração de dados, a saber: regras de associação, classificação, agrupamento e detecção de outlier. Os resultados revelaram informações úteis a partir de regras de associação e modelos de classificação. Além disso, o estudo tinha agrupado os dados dos alunos para identificar características e detectar possíveis discrepâncias. O conhecimento obtido foi interessante para melhorar o desempenho dos alunos.

O estudo de Nandeshwar and Chaudhari (2009) teve como objetivo prever ins-

crições de estudantes usando dados de admissão. Os pesquisadores usaram dados da Universidade da Virginia que consiste de 112390 casos. Vários modelos de classificação dos alunos foram construídos. Eles compararam os dados de diferentes alunos e identificou-se como melhores resultados *J48* e *Rido*.

Além disso, Kabakchieva (2013), analisou a matrícula e os dados pessoais de alunos da Universidade de Economia Mundial na Bulgária. O objetivo foi o de prever o desempenho dos estudantes com base nas características dos pré-universitários. O pesquisador aplicou vários algoritmos de classificação e os resultados mostraram que árvore de decisão com *J48* teve a maior exatidão global.

Em outra análise, Garcıa-Saiz and Zorrilla (2011) tentaram aplicar diferentes técnicas de classificação para um conjunto de dados educacionais estabelecidos para comparar seus desempenhos e escolher os melhores algoritmos para serem integrados na sua ferramenta (*E-learning Web Miner*). Esta ferramenta é destinada a ajudar os professores a descobrirem o desempenho de seus alunos. Foram utilizados os dados do campo de nome Introdução aos métodos multimídias, oferecido em três anos letivos de 2007-2010 na Universidade de Cantabria. Eles usaram diferentes métodos de classificação e descobriram que o desempenho e a precisão das técnicas dependiam do tipo dos atributos e do tamanho do conjunto de dados. Entre os achados, o *J48* foi dito adequado para conjuntos de dados com mais de 100 instâncias e atributos nominais com dados faltantes.

Outra comparação de diferentes algoritmos de mineração de dados foi realizada por Romero et al. (2008). A pesquisa teve como objetivo classificar os alunos com as mesmas notas finais em diferentes grupos, dependendo das atividades realizadas em um curso web-based. Essas atividades incluem: o número de atribuições feitas, o número de questionários, o tempo total usado em testes e outros. Seu conjunto de dados consiste de 438 estudantes da Universidade de Córdoba. Eles avaliaram o desempenho dos algoritmos com base nos tipos dos atributos: categóricos e numéricos. Eles descobriram que dois algoritmos de árvore de decisão, *CART* e *C4.5*, foram os melhores para dados categóricos.

Kovacic (2010) utilizou a classificação para prever o sucesso dos estudantes com base em variáveis sóciodemográficas (idade, sexo, etnia, educação, situação de trabalho e invalidez) e de ambiente de estudo (programa de curso). O conjunto de dados contém dados dos alunos no curso de Sistemas de Informação na Politécnica da Nova Zelândia. Quatro árvores de classificação, *CHAID*, *CHAID exaustiva*, *QUEST* e *CART* foram utilizadas nesse estudo e *CART* foi a melhor com uma porcentagem de classificação global de 60,5%.

Khan and Choi (2014), utilizaram a mineração de dados, com o método de árvore de decisão. Os dados foram minerados de forma a calcular as chances de obtenção de uma bolsa escolar. Foram utilizados os algoritmos *ID3* e *J48*, sendo o primeiro de melhor

eficiência, mesmo sendo o outro mais rápido de classificar os dados e criar uma árvore menor. De acordo com os autores, o *ID3* fornece uma árvore com mais regras, o que significa maior cruzamento de dados e tomada de decisão mais profunda, e é por isso que o resultado previsto foi mais preciso do que *J48*. Concluiu-se que o sistema desenvolvido pode ser muito útil na previsão da probabilidade de o estudante ganhar bolsa de estudos.

## 1.2 Trabalhos Nacionais

No Brasil, as oportunidades para praticar a mineração de dados educacionais está aumentando e os cursos de Educação à distância criaram oportunidades para as pesquisas na área, segundo Baker et al. (2011). Em um estudo, Gottardo et al. (2012), apresenta uma preliminar do uso de algumas técnicas de mineração de dados em uma base de dados do sistema Moodle, com o objetivo de verificar a adequação do conjunto de atributos propostos, contendo informações de estudantes em um curso realizado a distância. Escolheu-se uma disciplina com um total de 155 estudantes concluintes em quatro turmas diferentes e desenvolveu-se procedimentos para extração dos atributos considerados significativos para o trabalho. No desenvolvimento dos experimentos foram utilizados os algoritmos de classificação *RandomForest* e *MultilayerPerceptron*. O principal aspecto a ser destacado a partir dos testes refere-se aos resultados superiores obtidos de um experimento em relação a outro. Este fato aponta para a viabilidade da utilização de um conjunto amplo de atributos para representação de estudantes, potencialmente generalizáveis a diversos cenários de cursos Educação à Distância.

França and do Amaral (2013), apresentam o uso de técnicas de mineração de dados para a formação de grupos similares de estudantes com dificuldades de aprendizagem no ensino de Programação. Para a realização do trabalho, utilizou-se os dados provenientes de avaliações da aprendizagem da disciplina Programação Orientada a Objetos, ministrada no primeiro semestre de 2010, no curso de Licenciatura em Computação da Universidade de Pernambuco, onde havia 33 estudantes matriculados. Os resultados apresentados confirmam que técnicas de clusterização são bastante úteis para a formação de grupos homogêneos de estudantes. O uso do algoritmo *K-means* permitiu agrupar alunos pelas suas dificuldades de aprendizagem e identificou um grupo de aprendizes com dificuldades em entender Estruturação de Sistemas em Camadas e outro composto por estudantes com problemas em criar Arrays. Ambos os grupos foram reprovados ao final do semestre.

Detoni et al. (2014), mostraram resultados da aplicação de técnicas de aprendizado de máquina, utilizando como atributos unicamente contagens de interações ao longo do tempo, na predição de reprovação de estudantes. Para a realização do trabalho, foram

obtidos os dados anonimizados dos cursos a distância de Licenciatura Educação do Campo e de Licenciatura em Pedagogia. Demonstrou-se que utilizar apenas a quantidade de interações dos alunos é viável para gerar predições razoavelmente precisas, ainda que menos precisas do que trabalhos anteriores que utilizam atributos muito mais específicos.

Santos et al. (2012), relatam um estudo sobre a aplicação de técnicas de mineração de dados que permitem, em estágios anteriores às avaliações somativas, identificar alunos que têm maior risco de reprovação. Os dados que sustentam a abordagem são oriundos de avaliações formativas aplicadas no decorrer da disciplina através do Ambiente Virtual de Aprendizagem Moodle. Resultados preliminares mostram que os modelos criados permitem a identificação da propensão à reprovação com taxa de acerto em torno de 69%. A primeira análise realizada foi a respeito do coeficiente de correlação entre a nota na avaliação somativa e as demais variáveis existentes. Na segunda análise, foi aplicado o algoritmo de clusterização *k-means*, através da utilização da ferramenta *Weka*. Na terceira análise, a fim de identificar a tendência dos estudantes a terem sucesso na avaliação somativa, foram aplicados algoritmos de classificação sobre os dados disponíveis. Dentre os algoritmos utilizados, são apresentados apenas os resultados gerados pelo *REPTree* e pelo *J48* que geraram árvores com poucos nodos e maior capacidade preditiva. Todos os experimentos de classificação realizados utilizaram a técnica de validação cruzada *Leave-One-Out* para medir a capacidade de generalização dos modelos. Os três grupos de experimentos realizados comprovam uma realidade esperada: Quanto maior a dedicação nas atividades presenciais e semipresenciais, melhor o desempenho do aluno nas avaliações somativas.

No trabalho de Moro et al. (2014), apresentou-se resultados parciais de uma pesquisa de mestrado que identificou, por meio de técnicas de Mineração de Dados Educacionais, padrões relevantes de comportamentos de alunos que interagiram com um sistema educacional web. A *DAMICORE*, ferramenta de mineração de dados, foi aplicada aos dados com os algoritmos *Normalized Compression Distance*, *Neighbor Joining* e *FastNewman* e diante dos resultados obtidos, é possível supor, que a análise pode favorecer a tomada de decisão do professor e trazer melhorias para o processo de ensino e aprendizagem.

Manhães et al. (2011) apresentam um estudo que identifica os alunos com risco de evasão, através do uso de técnicas de mineração de dados. O trabalho avaliou a técnica através de três experimentos onde foram aplicados dez algoritmos de classificação sobre uma base de dados acadêmicos de alunos do curso de Engenharia Civil da Universidade Federal do Rio de Janeiro. Os algoritmos utilizados foram: *OneR*, *JRip*, *DecisionTable*, *SimpleCart*, *J48*, *RandomForest*, *SimpleLogistic*, *MultilayerPerceptron*, *NaiveBayes*, *BayesNet* e os métodos de classificação empregados pelos algoritmos foram: aprendizado de regras (*OneR e JRip*), tabela de decisão (*DecisionTable*), árvore de decisão (*Simple-*

*Cart, J48 e RandomForest*), modelos lineares de regressão logística (*SimpleLogistic*), modelo de rede neural artificial (*MultilayerPerceptron*), modelos probabilístico (*BayesNet*), classificador probabilístico simples baseado na aplicação do teorema de Bayes (*Naive-Bayes*). Os experimentos retornaram dados com acurácia média variando entre 75 a 80%. Concluiu-se que os desempenhos obtidos pelos algoritmos de mineração de dados dos mais simples aos mais sofisticados foram semelhantes e que, a acurácia dos classificadores e a taxa de erro são fortemente influenciadas pelos bancos de dados.

# 2 ÁRVORES DE DECISÃO COM RAPIDMINER

Neste capítulo é apresentada uma visão geral da ferramenta computacional usada nesse trabalho, o Rapidminer, com foco no operador *Decision tree*. Na Seção 2.1, apresentaremos um estudo, incluindo, o passo a passo de uma possível utilização desta ferramenta. O operador *Decision Tree* é estudado na Seção 2.2. As informações desse capítulo foram retiradas da documentação oficial do Rapidminer. Como a documentação está toda em inglês e o público-alvo são professores do ensino básico, optamos por fazer uma releitura em português de todas as funções do Rapidminer que serão utilizadas nesse trabalho. Assim, espera-se que as próximas seções contribuam para um melhor entendimento do Capítulo 3, mesmo para aqueles professores que possuem dificuldades com a língua inglesa.

## 2.1 Entendendo o Rapidminer

Esta seção está subdividida em outras nove nas quais são apresentadas a terminologia do Rapidminer (Subseção 2.1.1), a importação dos dados (Subseção 2.1.2), a criação de processos (Subseção 2.1.3), a transformação dos dados (Subseção 2.1.4), a criação de modelos (Subseção 2.1.5), a validação de modelos (Subseção 2.1.6), a visualização dos resultados (Subseção 2.1.7), a aplicação de modelos (Subseção 2.1.8) e a criação de operadores personalizados (Subseção 2.1.9).

### 2.1.1 Terminologia do Rapidminer

Alguns termos técnicos e seus significados são importantes para o entendimento do programa Rapidminer. Apresentaremos abaixo uma lista destes, bem como a tradução para o inglês, como são representados no programa.

- **Operadores** (em inglês *operators*) são blocos de construção utilizados para realizar ações com conjuntos de dados (em inglês *dataset*).

- **Repositório** (em inglês *repository*) é o local onde encontram armazenados os conjuntos de dados e os processos do programa.

- **Portas** (em inglês *port*) são os pontos através dos quais os operadores se conectam. As portas são representadas por semicírculos marcados nas laterais dos operadores. Existem vários tipos de portas. A Tabela 2.1 resume os tipos e funções de algumas portas que serão utilizadas nesse trabalho:

Tabela 2.1: Portas no Rapidminer

| Nome da porta | Abreviação no Rapidminer | Tradução para o inglês | Função |
|---|---|---|---|
| Resultado | res | Result | Mostra o resultado do um processo |
| Exemplo | exa | Example set | Retorna um exemplo de conjunto |
| Treino | tra | Training | Retorna um conjunto de treinamento |
| Modelo | mod | Model | Retorna o modelo criado por um processo. |
| Saída | out | Output | Retorna o resultado de um processo |
| Não Rotulado | unl | Unlabeled | Aplica dados não rotulados |
| Entrada | inp | Input | Fonte de entrada |

- **Processo** (em inglês *process*) é um conjunto de operadores conectados através de suas portas de forma que suas ações, na ordem em que estão conectadas, conduzem a um objetivo.

### 2.1.2 Adicionando conjuntos de dados ao repositório

Para adicionar conjuntos de dados ao repositório do Rapidminer, clique no botão *Add Data* na janela *Repository* do programa. Logo em seguida escolha o arquivo que deseja analisar. O Rapidminer é compatível com arquivos em formato Access, BibTeX, CSV, Excel, Sparse, STATA, dentre outros. Neste trabalho utilizamos apenas arquivos CSV.

Arquivos CSV são arquivos comuns separados por vírgula. Estes podem ser criados e editados em qualquer editor de texto. O Exemplo 2.1.1 mostra o formato de um arquivo CSV da maneira como ele é apresentado num editor de texto qualquer.

**Exemplo 2.1.1.** *Nesse exemplo, com dados fictícios, ilustramos como é redigido um arquivo CSV num editor de texto qualquer. As vírgulas são usadas para separar as colunas como se fosse uma tabela:*

```
nome,idade,sexo,profissao
ana,23,f,empresario
carlos,29,m,advogado
pedro,19,m,estudante
leticia,21,f,estudante
roberto,25,m,empresario
```

Os próximos passos serão apresentados considerando o Exemplo 2.1.1. Após clicar em *Add Data* e escolher o arquivo que deseja estudar basta clicar no botão *Next*. A próxima etapa consiste em especificar o formato dos dados. Como nossos dados estão na forma correta, avançaremos clicando uma segunda vez no botão *Next*. Na próxima etapa, podemos configurar os atributos. Nesta etapa você pode excluir ou renomear colunas, definir o formato dos atributos e, também, mudar a regra que rege cada um deles. Os tipos que cada atributo possui já são sugeridos pelo Rapidminer. Mudaremos apenas suas regras. Como a primeira coluna é de identificação, clique no ícone de engrenagem do lado da palavra "nome", escolha a opção *Change Role* (que significa mudar regra em português) e, no campo que foi aberto, digite *id* (abreviação para identificador). Pressione o botão *OK*. Note que a coluna "nome" é agora de cor diferente indicando que há uma regra sobre ela. Mudaremos também a coluna "profissao" seguindo os passos anteriores, porém digitando *label* no lugar de *id*. A regra *label* indica que a coluna agora está marcada. Esta marcação é necessária caso haja necessidade de nos referirmos a alguma coluna durante algum processo. Clicando novamente no botão *next*, basta dar um nome para o conjunto de dados. Trabalharemos com os dados do Exemplo 2.1.1 um pouco mais, dessa forma, se o leitor quiser dar o nome de "users" para este conjunto de dados, ficará mais fácil acompanhar as etapas que se seguem. Se tudo estiver correto, uma tabela mostrando os dados importados aparecerá na tela. No caso do exemplo dado nesta seção, a tabela que aparecerá é ilustrada pela Figura 2.1.

Figura 2.1: Adição de “users” no repositório

ExampleSet (5 examples, 2 special attributes, 2 regular attributes)

| Row No. | nome | sexo | idade | profissao |
|---|---|---|---|---|
| 1 | ana | f | 23 | empresario |
| 2 | carlos | m | 29 | advogado |
| 3 | pedro | m | 19 | estudante |
| 4 | leticia | f | 21 | estudante |
| 5 | roberto | m | 25 | empresario |

### 2.1.3 Construindo processos

Processos podem ser feitos de diversas formas conectando os vários tipos de operadores existentes no Rapidminer. Logo, para criar um processo você precisa saber quais operadores se conectam e como estes realizam ações de forma a conduzir o processo para o resultado esperado. Nessa subseção, criaremos o primeiro processo com o Rapidminer.

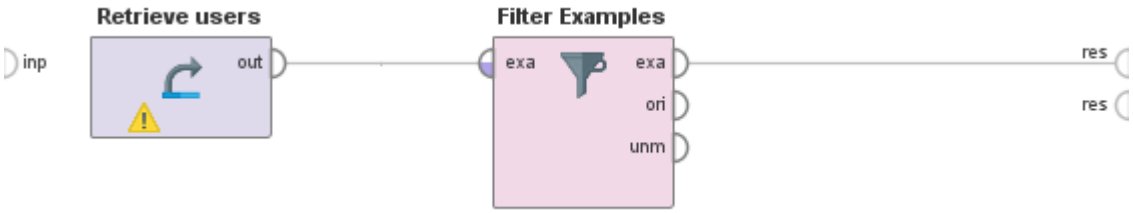
O processo mais simples é formado por um só operador. No repositório do Rapidminer, escolha um conjunto de dados e arraste-o para a janela denominada *Process*. Observe que um operador denominado *retrieve*, termo em inglês que significa “receber”, é inserido na janela de processos. A ação deste operador é simplesmente receber o conjunto de dados do repositório. Conecte a porta *out* do operador *retrieve* na porta *res* da janela de processos, esta última se encontra no canto superior direito da janela de processos. Nesse momento, o processo deve estar como na Figura 2.2. Pronto! Você criou seu primeiro processo, basta apertar a tecla F11 para ver o resultado. Caso o conjunto de dados importado seja o mesmo da Seção 2.1.2, o resultado é o mesmo da Figura 2.1.

Figura 2.2: Processo de visualização de “users”

### 2.1.4 Transformando o conjunto de dados

Na maioria das vezes, o conjunto de dados precisa ser transformado. Um exemplo de operador com essa função é o operador *Filter Examples* (que em português significa filtro de exemplos). A função do operador *Filter Examples* é simplesmente filtrar os elementos do conjunto de dados. Para encontrar um operador na janela *Operators*, basta digitar seu nome no campo *Seach for Operators*. Supondo que, após adicionar o arquivo em formato CSV do Exemplo 2.1.1 ao repositório do Rapidminer com o nome de *users*, desejássemos filtrar os usuários selecionando apenas os do sexo masculino. Deve-se realizar algumas etapas para que o processo tenha a aparência da Figura 2.3.

Figura 2.3: Processo de filtragem em "users"




- Arraste o conjunto de dados *users* do repositório do Rapidminer para a janela de processos;
- Arraste o operador *Filter Examples* da janela *Operators* para a janela de processos;
- Clique no operador *Filter Examples*;
- Na janela *Parameters*, clique no botão *Add Filters*;
- No primeiro campo digite "sexo" sem aspas, no segundo selecione a opção *equals* (em português, igual) e no terceiro digite "m" sem as aspas. Isso faz com que o operador filtre o conjunto de dados, selecionando apenas aquelas amostras cujo sexo é igual a "m";
- Voltando a janela de processos do Rapidminer, conecte a porta *out* do operador *Retrieve users*, na porta *exa* do lado esquerdo do operador *Filter Examples*;
- Conecte a porta *exa* do lado direito do operador *Filter Examples*, na porta *res* no canto superior direito da janela de processos do Rapidminer;
- Aperte a tecla F11 para executar o processo.

O resultado do processo acima é apresentado pela Figura 2.4. Observe que só as pessoas do sexo masculino aparecem agora na tabela.

Figura 2.4: Resultado do processo de filtragem em “users”

ExampleSet (3 examples, 2 special attributes, 2 regular attributes)

| Row No. | nome | sexo | idade | profissao |
|---|---|---|---|---|
| 1 | carlos | m | 29 | advogado |
| 2 | pedro | m | 19 | estudante |
| 3 | roberto | m | 25 | empresario |

### 2.1.5 Processo de modelagem

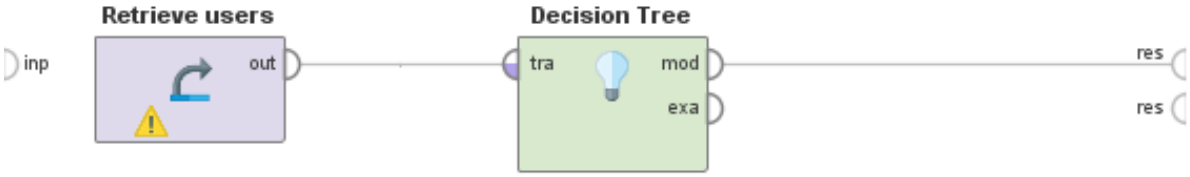
Até este ponto, apresentamos noções básicas sobre como adicionar dados ao repositório e ao processo do Rapidminer. Também explicamos como transformar o conjunto de dados. Nos próximos parágrafos, daremos exemplo de processos de modelagem.

Em poucas palavras, modelagem é a representação de características, padrões ou relações entre os elementos de um conjunto de dados. Esta representação pode ser através de diagramas, tabelas, gráficos, grafos, etc.

Utilizaremos, neste trabalho, a modelagem através do operador *Decision Tree* (árvore de decisão em português). Nesta seção, apresentaremos apenas como construir o processo utilizando este operador. Detalhes específicos do processo serão dados na Seção 2.2.

Novamente, para ilustrar como funciona a construção de um processo de modelagem com o operador *Decision Tree* no programa Rapidminer, consideraremos o conjunto de dados do Exemplo 2.1.1 adicionado ao repositório com nome de *users*. Para que o operador *Decision Tree* funcione, é necessária a marcação de um atributo como *label* explicado na Subseção 2.1.2. A Figura 2.5 representa o processo em questão, e as etapas para a sua construção são:

Figura 2.5: *Decision Tree* aplicado ao conjunto “users”



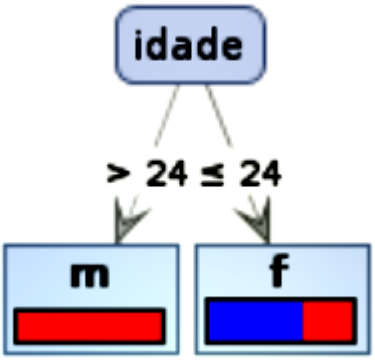

- Arraste o conjunto de dados *users* do repositório do Rapidminer para a janela de

processos;

- Arraste o operador *Decision Tree* da janela *Operators* para a janela de processos;
- Conecte a porta *out* do operador *Retrieve users* na porta *tra* do operador *Decision Tree*;
- Conecte a porta *mod* do operador *Decision Tree* na porta *res* no canto superior direito da janela de processos do Rapidminer;
- Aperte a tecla F11 para rodar o processo;

O resultado do processo é apresentado na Figura 2.6 em forma de grafo.

Figura 2.6: Grafo resultante da aplicação de *Decision Tree* em "users"




O grafo da Figura 2.6 diz que se a pessoa, no conjunto de dados *users*, tiver uma idade maior que 24 anos, então esta é do sexo masculino, caso contrário é do sexo feminino. Note que não é verdade, pois Pedro tem 19 anos, portanto uma idade menor ou igual a 24 anos, e é do sexo masculino. No entanto, note que este modelo fornece uma regra para decidir se a pessoa é do sexo masculino utilizando apenas a idade e, mesmo assim, é uma regra que conseguiu classificar corretamente 4 das 5 pessoas. De fato, poderíamos fazer melhor apenas observando a tabela, no entanto, quando a quantidade de elementos do conjunto de dados é muito grande (o que ocorre na maioria dos casos reais), criar modelos manualmente para classificar os dados do conjunto é inviável, e obter uma classificação com 80% de acerto através de regras de decisão se torna uma tarefa extremamente difícil. Na Seção 2.1.6 aprenderemos processos de validação de modelos.

### 2.1.6 Validando modelos

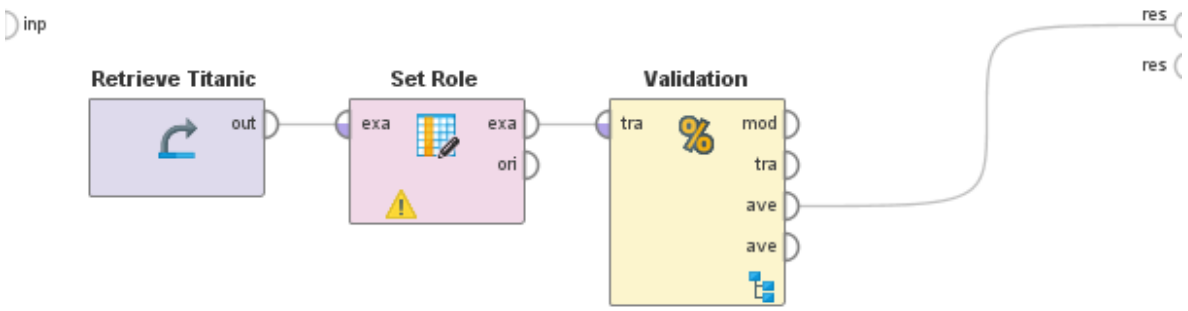
Neste ponto, espera-se que o leitor saiba como adicionar determinados conjuntos de dados e operadores à janela de processos. Dessa forma conduziremos o texto de

forma sucinta utilizando de figuras para representar as estruturas dos processos a serem trabalhados. Nesta subseção, apresentaremos o processo de validação.

O Rapidminer já disponibiliza vários conjuntos de dados para que você possa praticar a construção de processos. Dessa vez utilizaremos um desses conjuntos de dados. No repositório do Rapidminer, dentro da pasta *Samples* (que significa exemplos em português), você encontra vários exemplos de conjuntos de dados e de processos prontos para estudo. Dentro da pasta *data* (que significa dados em português), você encontra alguns conjuntos de dados prontos para serem trabalhados. Arraste aquele com o nome de *Titanic* para a janela de processos.

Como o conjunto de dados "Titanic" já estava no repositório, não tivemos a oportunidade de mudar as regras que regem os atributos deste conjunto durante sua importação. Desta forma, utilizaremos o operador *Set Role* para dizer que a coluna "name" é de identificação e a coluna "survived" será marcada para criação do modelo de árvore de decisão. Para a construção de um processo como o da Figura 2.7, siga os seguintes passos:

Figura 2.7: Validação de modelo de classificação de tripulantes do Titanic




- Adicione o operador *Set Role* à janela de processos;
- Conecte a porta *out* do operador *Retrieve Titanic* na porta *exa* do operador *Set Role*;
- Clique no operador *Set Role*;
- Na janela de parâmetros, clique no botão *Edit List*;
- Clique no botão *Add Entry* para adicionar entradas até que tenham um total de duas;
- No campo *atribute name* da primeira entrada, selecione *Name*;
- No campo *atribute name* da segunda entrada, selecione *Survived*;
- No campo *target role* da primeira entrada, selecione *id*;

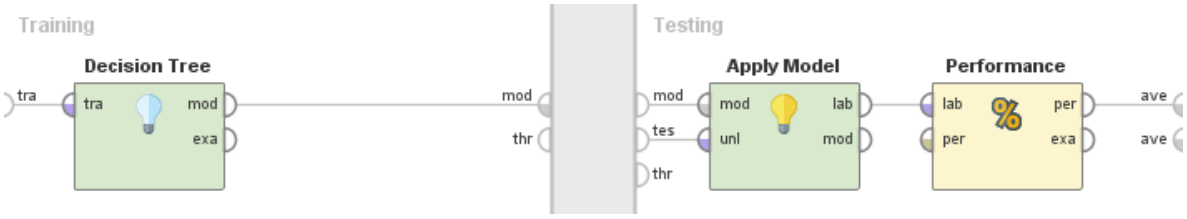
- No campo *target role* da segunda entrada; selecione *label*;
- Clique no botão *Apply* para aplicar as regras.

Para validar o modelo de árvore de decisão obtida através do operador *Decision Tree*, neste caso usada para classificar as pessoas que sobreviverão ou não do famoso naufrágio do "Titanic", utilizaremos o operador *X-Validation*. Siga os seguintes passos:

- Arraste o operador *X-Validation* para janela de processos;
- Conecte a porta *exa* do operador *Set Role* na porta *tra* do operador *X-Validation*;
- Clique duas vezes sobre o operador *X-Validation*;

Nesse momento, pode-se ver na Figura 2.8 dois subprocessos do operador *X-Validation*, um para treinamento (em inglês *training*) e um para teste (em inglês *testing*).

Figura 2.8: Processos internos do operador *X-Validation*




- Arraste o operador *Decision Tree* para janela de treinamento;
- Conecte a porta *tra* da janela de treinamento na porta *tra* do operador *Decision Tree*;
- Conecte a porta *mod* do operador *Decision Tree* na porta *mod* da janela de treinamento;
- Arraste o operador *Apply Model* para a janela de teste;
- Conecte a porta *mod* da janela de teste na porta *mod* do operador *Apply Model*;
- Conecte a porta *tes* da janela de teste na porta *unl* do operador *Apply Model*;
- Arraste o operador *Performance (Classification)* para a janela de teste;
- Conecte a porta *lab* do operador *Apply Model* na porta *lab* do operador *Performance (Classification)*;
- Conecte a porta *per* do operador *Performance (Classification)* na porta *ave* da janela de teste;

- Clique em *Process* para voltar para a janela de processos;
- Conecte a primeira porta *ave* do operador *X-Validation* na porta *res* da janela de processos;
- Aperte F11 para rodar o processo.

Se tudo estiver correto, uma janela com a tabela apresentada na Figura 2.9 será apresentada na tela.

Figura 2.9: Resultado da aplicação de *Decision Tree* no conjunto "Titanic"

**accuracy: 78.69% +/- 4.45% (mikro: 78.69%)**

| | true Yes | true No | class precision |
|---|---|---|---|
| pred. Yes | 331 | 110 | 75.06% |
| pred. No | 169 | 699 | 80.53% |
| class recall | 66.20% | 86.40% | |

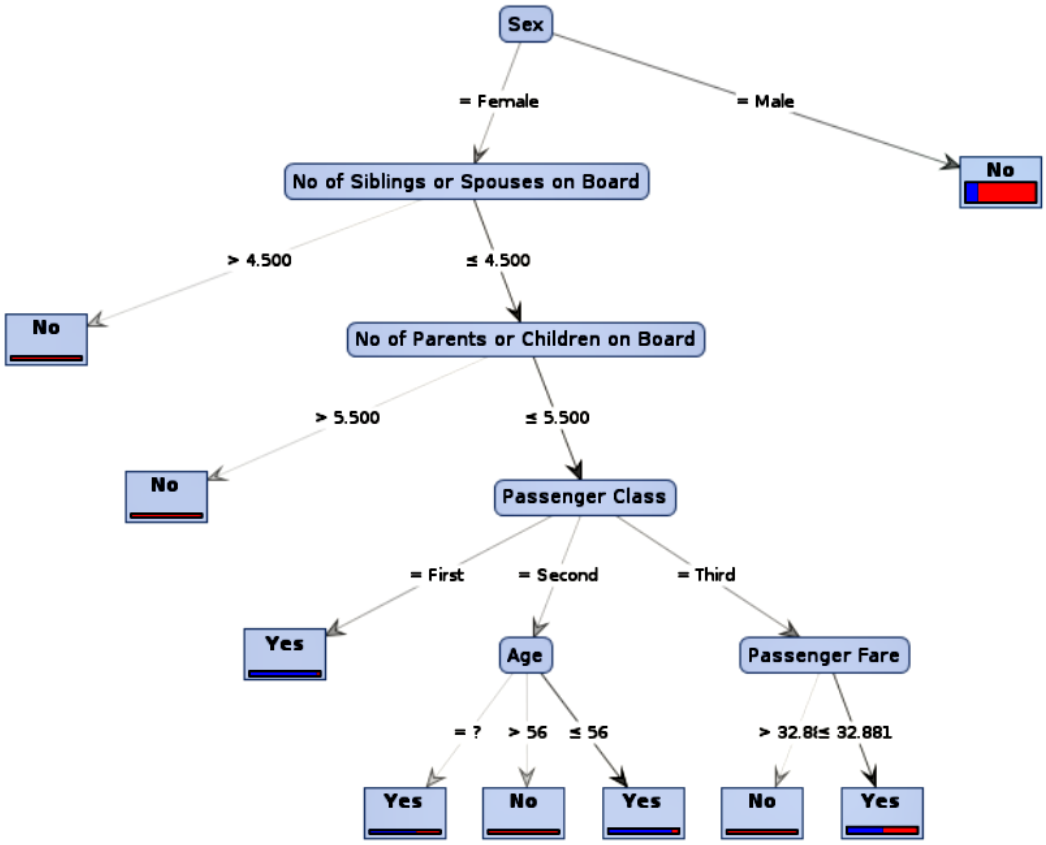
### 2.1.7 Visualizando resultados

Ao executar um processo, os resultados deste são apresentados pelo Rapidminer. Nesta subseção trataremos de entender o que significa alguns dos resultados que utilizaremos neste trabalho.

A Figura 2.9 apresenta o resultado do processo de validação do modelo de árvore de decisão obtido pelo operador *Decision Tree* aplicado ao conjunto de dados "Titanic". O objetivo deste processo é classificar quem sobreviveu ou não ao acidente. A matriz apresentada na Figura 2.9 é denominada *Confusion Matrix* (que em português significa Matriz de Confusão). Quando somado os números inteiros da coluna *true Yes*, obtém-se a quantidade de tripulantes do "Titanic" que, de fato, sobreviveram, considerando o conjunto de dados estudado. Somando os inteiros da coluna *true No* obtém-se o número de tripulantes que não sobreviveram ao naufrágio. Em laranja mais escuro encontramos o número de classificações corretas realizadas pelo modelo gerado pelo operador *Decision Tree*. A porcentagem 66,20% representa os sobreviventes classificados corretamente dentre todos que sobreviveram. Já 86,40% é a porcentagem de não sobreviventes classificados corretamente dentre todos que não sobreviveram. A porcentagem 75,06% é obtida fazendo o quociente do número de tripulantes classificados como sobreviventes dentre os que de fato sobreviveram, pelo número classificações de sobreviventes (corretas ou não) geradas pelo modelo. Por fim 80,53% é a porcentagem obtida fazendo o quociente do número de tripulantes classificados como não sobreviventes dentre os que de fato não sobreviveram, pelo número total de classificações de não sobreviventes (corretas ou não) geradas pelo modelo de árvore de decisão.

O termo *accuracy* (que em português significa acurácia) é a porcentagem de que, escolhida aleatoriamente uma amostra no conjunto de dados, o modelo classifique corretamente a amostra. Na Figura 2.9, a *accuracy* é de 78,68%. O valor 4,45% que aparece junto à *accuracy* é o desvio padrão. Para calcular a *accuracy*, basta dividir o número de amostras classificadas corretamente pelo total de amostras do conjunto de dados. O termo *mikro* denota a *accuracy* média de todas *confusion matrix* agregadas ao conjunto de dados estudado. Como só temos uma, *accuracy* e *mikro* serão os mesmos.

Figura 2.10: Grafo da árvore de decisão relacionada ao conjunto "Titanic"




Outro resultado importante para o nosso trabalho é o grafo de árvore de decisão. Considerando o processo de validação da Subseção 2.1.6, o operador *X-Validation* tem mais três portas que não utilizamos. A primeira porta *ave* do operador mostra como resultado uma *confusion matrix* já explicada nos parágrafos anteriores. Ao conectar a porta *mod* deste operador à porta *res* da janela de processos e, novamente executarmos o processo através da tecla F11, obteremos o grafo da árvore de decisão (ver Figura 2.10). Os atributos que aparecem no grafo dentro de blocos retangulares com pontas

arredondadas são chamados de nós (*node*). Os atributos que aparecem no grafo dentro de blocos retangulares não arredondados são chamados de folhas. Nas folhas estão os atributos regidos pela regra *label*. As setas ligando os nós e folhas são chamadas de arestas (*edges*). O nó que não recebe seta alguma é chamado de raiz (*root*).

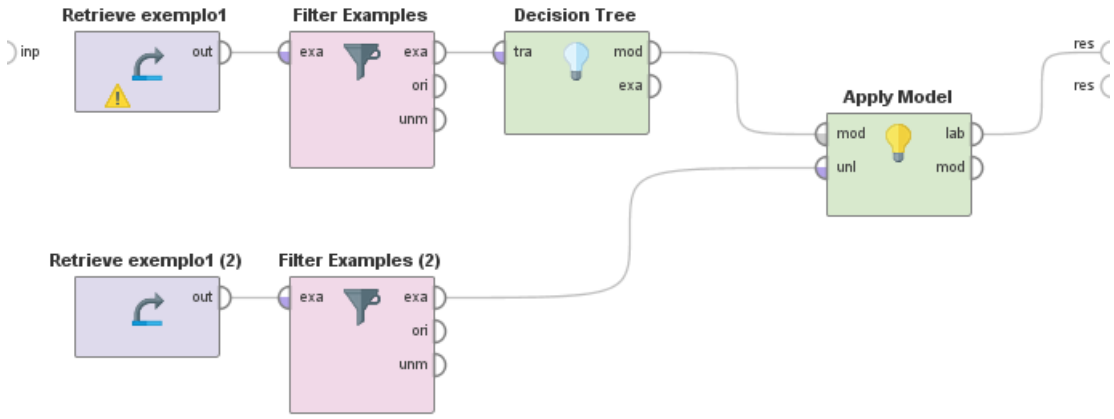
Na aba *Results*, considerando ainda o processo de validação da Subseção 2.1.6, temos outra aba chamada *description*. Nesta aba encontramos uma estrutura denominada *Tree* (árvore em português). Uma *Tree* apresenta o número exato de classificações que cada regra que um modelo de árvore de decisão tenha gerado. A seguir, apresentamos a *tree* gerada pelo processo de classificação de sobreviventes do Titanic:

```
Sex = Female
|   No of Siblings or Spouses on Board > 4.500: No {Yes=0, No=6}
|   No of Siblings or Spouses on Board <= 4.500
|   |   No of Parents or Children on Board > 5.500: No {Yes=0, No=2}
|   |   No of Parents or Children on Board <= 5.500
|   |   |   Passenger Class = First: Yes {Yes=139, No=5}
|   |   |   Passenger Class = Second
|   |   |   |   Age = ?: Yes {Yes=2, No=1}
|   |   |   |   Age > 56: No {Yes=0, No=2}
|   |   |   |   Age <= 56: Yes {Yes=92, No=9}
|   |   |   Passenger Class = Third
|   |   |   |   Passenger Fare > 32.881: No {Yes=0, No=5}
|   |   |   |   Passenger Fare <= 32.881: Yes {Yes=106, No=97}
Sex = Male: No {Yes=161, No=682}
```

### 2.1.8 Aplicando o modelo

Nesta subseção, mostraremos um processo que aplica um modelo e fornece a previsão para algum atributo selecionado. Para a melhor compreensão do leitor, apresentaremos a imagem do processo e na sequência, o passo a passo de sua construção. O processo deve ter a aparência da Figura 2.11.

Figura 2.11: Processo de aplicação de *Decision Tree* ao conjunto "Exemplo1"




Considere o arquivo em formato CSV representado abaixo:

```
Prova 1,Prova 2,Prova 3,Prova 4,Bolsista,Resultado
9,3,8,1,Sim,Aprovado
6,1,3,5,Sim,Reprovado
4,7,1,1,Sim,Reprovado
9,9,2,10,Sim,
5,2,0,9,Sim,Reprovado
10,9,9,5,Não,Aprovado
4,3,10,3,Sim,Aprovado
9,10,5,0,Sim,Aprovado
1,6,6,7,Sim,
3,5,4,7,Sim,
10,4,8,0,Sim,Aprovado
6,0,7,4,Sim,Reprovado
8,3,7,3,Sim,Aprovado
8,0,4,1,Não,Reprovado
9,6,9,10,Não,Aprovado
5,8,5,2,Sim,Aprovado
1,1,8,1,Sim,Reprovado
4,8,5,6,Sim,
10,10,4,10,Não,Aprovado
2,1,10,8,Sim,Aprovado
```

Observe que alguns dos alunos não têm valor algum no atributo "Resultado". Faremos a previsão para tal atributo. Supondo que o arquivo acima esteja no repositório do Rapidminer com o nome de "Exemplo1", siga os passos a seguir:

- Arraste o conjunto de dados “Exemplo1” para a janela de processos do Rapidminer;
- Repita o passo anterior colocando um segundo conjunto de dados “Exemplo1” na janela de processos;
- Arraste o operador *Filter Examples* para a janela de processos;
- Repita o passo anterior ficando com dois operadores *Filter Examples* e *Filter Examples (2)* na janela de processos;
- Conecte a porta *out* do operador *Retrieve Exemplo1* a porta *exa* do lado esquerdo do operador *Filter Examples*;
- Conecte a porta *out* do operador *Retrieve Exemplo1 (2)* a porta *exa* do lado esquerdo do operador *Filter Examples (2)*;
- Clique no operador *Filter Examples*;
- Clique no botão *Add Filters*;
- Na janela *Create Filters: filters*, no primeiro campo escolha Resultado e no segundo campo escolha *is not missing* (“não está perdido” em português);
- Clique no botão *OK*;
- Clique no operador *Filter Examples (2)*;
- Clique no botão *Add Filters*;
- Na janela *Create Filters: filters*, no primeiro campo escolha Resultado e no segundo campo escolha *is missing* (“está perdido” em português);
- Clique no botão *OK*;
- Arraste o operador *Decision Tree* para a janela de processos;
- Conecte a porta *exa* do lado direito do operador *Filter Examples* a porta *tra* do operador *Decision Tree*;
- Arraste o operador *Apply Model* para a janela de processos;
- Conecte a porta *mod* do operador *Decision Tree* a porta *mod* do operador *Apply Model*;
- Conecte a porta *exa* do operador *Filter Examples (2)* a porta *unl* do operador *Apply Model*;

- Conecte a porta *lab* do operador *Apply Model* a porta *res* na janela de processos do Rapidminer;
- Aperte F11 para executar o processo.

Após a execução do processo, os resultados podem ser analisados através de uma tabela fornecida pelo Rapidminer. Para melhor compreensão dos dados, interpretaremos o exemplo representado na Figura 2.12.

Figura 2.12: Resultado da aplicação de *Decision Tree* no conjunto "Exemplo1"

| Row No. | Resultado | prediction(Resultado) | confidence(Aprovado) | confidence(Reprovado) | Prova 1 | Prova 2 | Prova 3 | Prova 4 | Bolsista |
|---|---|---|---|---|---|---|---|---|---|
| 1 | ? | Reprovado | 0 | 1 | 9 | 9 | 2 | 10 | Sim |
| 2 | ? | Aprovado | 0.909 | 0.091 | 1 | 6 | 6 | 7 | Sim |
| 3 | ? | Aprovado | 0.909 | 0.091 | 3 | 5 | 4 | 7 | Sim |
| 4 | ? | Aprovado | 0.909 | 0.091 | 4 | 8 | 5 | 6 | Sim |

Na Figura 2.12, é possível notar que o atributo "Resultado" não recebe valor algum (por isso os sinais de interrogação) e que há uma coluna, a saber *Prediction (Resultado)*, que exibe uma previsão sobre os valores que o atributo "Resultado" poderia receber. A Figura 2.12 também apresenta a confiança da predição do Rapidminer nas colunas *confidence(Aprovado)* e *confidence(Reprovado)*. Na linha 1, por exemplo, a confiança de que o valor do atributo "Resultado" é "Reprovado" é de 100%. Já na linha 2, a confiança de que o valor do atributo "Resultado" é "Reprovado" é de $9,1\%$ e de que o valor é "Aprovado" é de $90,9\%$.

### 2.1.9 Criando um operador personalizado

Nesta seção, apresentaremos como criar operadores personalizados utilizando o Python. Antes de mais nada devemos instalar a extensão que permite a criação de *scripts* em Python dentro de um operador. Siga os seguintes passos:

- No *menu* do Rapidminer, clique em *Extensions* e depois em *MarketPlace*;
- No campo *Search* digite Python. Você encontrará a extensão chamada *Python Scripting*;
- Clique na extensão e depois marque a caixa *Select for installation*;
- Clique no botão *Install 1 packages*;
- Marque a caixa *I accept the terms of all license agreements*;

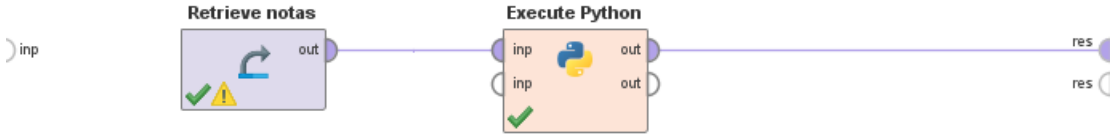
- Clique novamente no botão em *Install 1 packages*;
- O programa te perguntará se você deseja reiniciar o Rapidminer para concluir a instalação, clique no botão *Yes*;
- Caso você tenha algum processo aberto que não esteja salvo, o programa perguntará se você deseja salvar seu processo. Caso não tenha processos para serem salvos, clique no botão *No*. Caso tenha, clique no botão *Yes*;
- O Rapidminer se reiniciará e a extensão estará pronta para utilização.

Existem infinitas possibilidades de criação de operadores de preprocesso, modelagem, validação, aplicação de modelos e visualização de resultados. Acreditamos que a melhor maneira de aprender a construção de processos é na prática, portanto apresentaremos nos parágrafos que se seguem, como criar um operador personalizado de preprocessamento dos dados. Considere o seguinte arquivo no formato CSV:

```
afonso,1,9,8
ana,4,5,7
bruno,1,9,7
lucas,1,3,4
mariana,4,7,9
pedro,1,2,4
tatiane,6,3,6
zenaide,7,2,8
```

O arquivo acima é um arquivo fictício. Cada linha do arquivo contém um nome de um aluno e a nota que o aluno tirou em três provas. Acrescentaremos uma última coluna no conjunto de dados contendo a média das notas de alguns alunos. Para a construção do processo representado na Figura 2.13, siga os seguintes passos:

Figura 2.13: Processo com o operador *Execute Python*




- Adicione o conjunto de dados com o nome de “notas” ao repositório do Rapidminer;
- Arraste o conjunto de dados “notas” para a janela de processos;
- Arraste o operador *Execute Python* para a janela de processos;

- Conecte a porta *out* do operador *Retrieve notas* a porta *inp* do operador *Execute Python*;
- Conecte a porta *out* do operador *Execute Python* a porta *res* da janela de resultados;
- Clique no operador *Execute Python* e depois no botão *Edit Text* na janela de parâmetros;
- Apague todo o *script*;
- Copie e cole, no lugar do *script* que foi apagado, o código em Python abaixo:

```
import pandas

def rm_main(data):

    #transformando dados em dicionario
    dicionario = data.to_dict()

    # adicionando a chave "media" ao conjunto de dados
    dicionario["media"] = {}

    # calculando e acrescentando cada media de cada aluno
    for i in range(8):
        dicionario["media"][i] = ( dicionario["att2"][i] +
                                   dicionario["att3"][i] +
                                   dicionario["att4"][i] ) / 3.0
    # transformando o dicionario de volta em dataframe
    resultado = pandas.DataFrame(dicionario)

    # retornando o conjunto de dados, agora com a media
    return resultado
```

- Clique no botão *Apply*;
- Pressione a tecla F11 para executar o processo;

Figura 2.14: Resultado do processo com o operador *Execute Python*

| Row No. | att1 | att2 | att3 | att4 | media |
|---|---|---|---|---|---|
| 1 | afonso | 1 | 9 | 8 | 6 |
| 2 | ana | 4 | 5 | 7 | 5.333 |
| 3 | bruno | 1 | 9 | 7 | 5.667 |
| 4 | lucas | 1 | 3 | 4 | 2.667 |
| 5 | mariana | 4 | 7 | 9 | 6.667 |
| 6 | pedro | 1 | 2 | 4 | 2.333 |
| 7 | tatiane | 6 | 3 | 6 | 5 |
| 8 | zenaide | 7 | 2 | 8 | 5.667 |

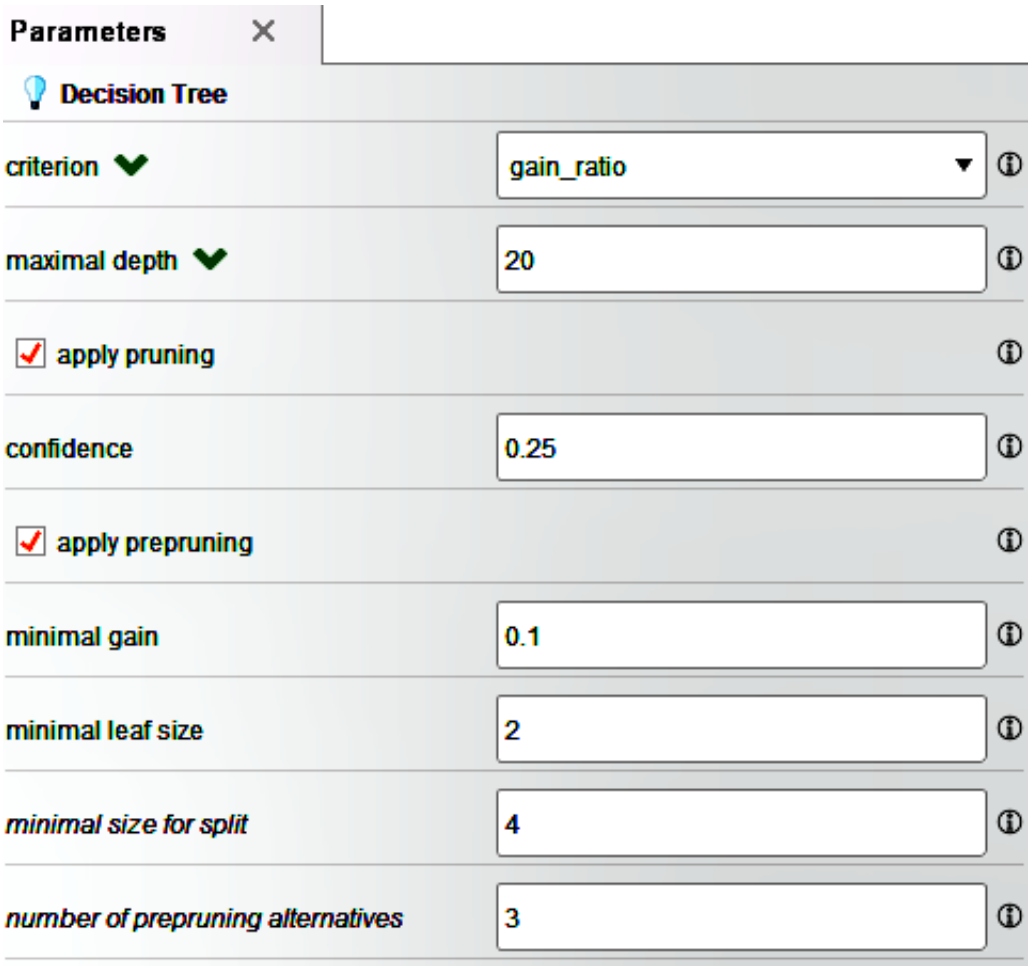
O resultado do processo pode ser observado na Figura 2.14. Este é um simples exemplo de processo utilizando operadores construídos com *scripts* na linguagem Python. Operadores muito mais elaborados podem ser criados, no entanto a intenção deste trabalho é apenas dar uma ideia de como acontece a relação entre Rapidminer e Python.

## 2.2 Árvores de decisão

Nesta seção apresentaremos detalhes do operador *Decision Tree*. Para trabalhar com o operador *Decision Tree*, o conjunto de dados pode ser numérico ou categórico (atributos que possuem valores textuais em vez de números), no entanto, a coluna que desejamos classificar deve ser categórica. Mais ainda, esta mesma coluna deve ser regida pela regra *label*. Na seção anterior, vimos como fazer um atributo ser regido pela regra *label* tanto na importação dos dados para o repositório do Rapidminer, quanto no processo, neste último caso usamos o operador *Set Role*.

Parâmetros de um operador são simplesmente características que este possui. As possibilidades para diversificação dos parâmetros de um operador podem ser encontradas, no programa Rapidminer, clicando no operador que está na janela de processo e olhando para a janela *Parameters*. Como exemplo, a Figura 2.15 apresenta os parâmetros do operador *Decision Tree*.

Figura 2.15: Parâmetros do operador *Decision Tree*




O parâmetro *criterion* determina como particionar o conjunto de dados de acordo com uma determinada regra de decisão. Os critérios disponíveis no Rapidminer são: *Information gain*, *Gain ratio*, *Gini index* e *Accuracy*. Para detalhes sobre cada um destes critérios recomendamos o leitor a leitura do livro Data Mining With Decision Trees dos autores Rokach and Maimon (2014).

Outro parâmetro que pode ser usado para modificar o modelo é o *maximal depth* que estabelece a profundidade máxima de uma árvore, que pode variar dependendo do tamanho do conjunto de dados. Esse parâmetro é utilizado para restringir o tamanho da árvore de decisão e criação de novas regras para o modelo da árvore é interrompido quando a sua profundidade máxima é atingida. Para que a profundidade seja ilimitada, deve-se atribuir o valor de $-1$ para este parâmetro.

A aplicação de poda (*apply pruning*) é uma técnica em que folhas são removidas com o objetivo de simplificar o grafo da árvore de decisão perdendo o mínimo possível de informação relevante. Isto é feito a fim de aumentar a capacidade de classificação em conjuntos de dados gigantes. Existem várias técnicas de poda, tais como *Cost Compexity Pruning*, *Reduced Error Pruning*, *Minimum Error Pruning* (MEP), dentre outras apre-

sentadas por Rokach and Maimon (2014). Como nesse trabalho não alteraremos esses parâmentros, deixaremos as caixas *apply prunning* e *apply preprunning* marcadas.

# 3 APLICAÇÕES E RESULTADOS

Neste capítulo são estudados dois conjundo de dados para classificação de estudantes a partir de resultados anteriores. Inicialmente, mostra-se um estudo com dados coletados no ambiente escolar de duas escolas portuguesas e na sequência, é apresentado outro estudo, com dados coletados em uma Universidade do Canadá. Os dados podem ser encontrados, respectivamente no site da *Machine Learning Repository* com o nome de *Student Performance Data Set* e no site da *OpenMV.net Datasets* com o nome de *Class grades.*

## 3.1 Atributos irrelevantes e alta *accuracy*

Nesse estudo, usa-se dados coletados no ambiente escolar de duas escolas portuguesas, por meio de relatórios e questionários. O conjunto de dados usado nesse trabalho é o da disciplina de Matemática, e ele inclui atributos como notas e características sociais, somando 33 no total. A seguir informações sobre cada um dos atributos. São eles:

- *school* – Escola que o aluno estuda (*GP* para alunos que estudam na escola Gabriel Pereira ou *MS* para alunos que estudam na escola Mousinho da Silveira).
- *sex* - Sexo do aluno (*F* para aluno do sexo feminino ou *M* para o sexo masculino).
- *age* - Idade do aluno (De 15 a 22 anos).
- *address* - Endereço residencial do aluno (*U* para alunos residentes na zona urbana ou R para zona rural).
- *famsize* - Quantidade de pessoas na família (*LE3* para quantidades menores ou igual a 3 ou *GT3* para superiores a 3).
- *Pstatus* - Relacionamento dos pais (*T* para os pais que moram juntos ou *A* para os que moram separados).
- *Medu* - Escolaridade da mãe (0 - nenhuma, 1 - ensino fundamental (4ª série), 2 – ensino fundamental (5ª à 9ª série), 3 - ensino médio ou 4 - ensino superior).

- *Fedu* - Escolaridade do pai (0 - nenhuma, 1 - ensino fundamental (4ª série), 2 – ensino fundamental (5ª à 9ª série), 3 - ensino médio ou 4 - ensino superior).
- *Mjob* - Profissão da mãe (*teacher* para professora, *health* para profissões relacionadas a saúde, *services* para serviços públicos, *at home* para profissionais do lar ou *other* para outras profissões).
- *Fjob* - Profissão do pai (*teacher* para professor, *health* para profissões relacionadas a saúde, *services* para serviços públicos, *at home* para profissionais do lar ou *other* para outras profissões).
- *reason* - Motivação de escolha da escola (*home* para alunos que escolheram a escola devido à proximidade de casa, *reputation* para alunos que escolheram a escola pela boa reputação, *course* para alunos que escolheram a escola pela preferência de curso ou *other* para outros motivos de escolha).
- *guardian* - Responsável pelo aluno (*mother* se for a mãe, *father* se for o pai ou *other* para qualquer outro responsável).
- *traveltime* - Tempo gasto no deslocamento de casa até a escola (1 – menos de 15 min, 2 – de 15 a 30 minutos, 3 - de 30 minutos a 1 hora, ou 4 – mais de 1 hora).
- *studytime* - Tempo de estudo semanal (1 – menos de 2 horas, 2 – de 2 a 5 horas, 3 - de 5 a 10 horas, ou 4 – mais de 10 horas).
- *failures* – Quantidade de reprovação (1 – uma reprovação, 2 – duas reprovações, 3 - três reprovações, ou 4 – quatro ou mais reprovações).
- *schoolsup* - Apoio escolar extra (*yes* para os alunos que tiveram o apoio ou *no* para os que não tiveram).
- *famsup* - Apoio educacional da família (*yes* para os alunos que tiveram o apoio ou *no* para os que não tiveram).
- *paid* - Aulas extras de matemática (*yes* para os alunos que tiveram aulas extras ou *no* para os que não tiveram).
- *activities* - Atividades extracurriculares (*yes* para os alunos que fizeram as atividades ou *no* para os que não fizeram).
- *nursery* - Frequentou a escola maternal (*yes* para os alunos que frequentaram a escola maternal ou *no* para os que não frequentaram).

- *higher* – Intenção de cursar o ensino superior (*yes* para os alunos que pretendem cursar o Ensino Superior ou *no* para os que não pretendem).
- *internet* – Acesso à Internet em casa (*yes* para os alunos que possuem o acesso ou *no* para os que não possuem).
- *romantic* – Relacionamento amoroso (*yes* para os alunos que possuem o relacionamento ou *no* para os que não possuem).
- *famrel* - Qualidade das relações familiares (intervalo inteiro de 1 a 5, onde 1 significa relações muito ruins e 5 muito boas).
- *freetime* - Tempo livre após a escola (intervalo inteiro de 1 a 5, onde 1 significa pouco tempo livre e 5 muito tempo livre).
- *goout* – Tempo para sair com os amigos (intervalo inteiro de 1 a 5, onde 1 significa pouco tempo livre e 5 muito tempo livre).
- *Dalc* - Consumo de álcool durante a semana (intervalo inteiro de 1 a 5, onde 1 significa nenhum consumo de álcool e 5 consumo excessivo).
- *Walc* - Consumo de álcool no fim de semana. (intervalo inteiro de 1 a 5, onde 1 significa nenhum consumo de álcool e 5 consumo excessivo)
- *health* - Estado de saúde atual (intervalo inteiro de 1 a 5, onde 1 significa estado de saúde muito ruim e 5 muito bom).
- *absences* - Número de faltas escolares (de 0 a 93).
- *G1* – Nota total obtida no primeiro período do curso (de 0 a 20).
- *G2* – Nota total obtida no segundo período do curso (de 0 a 20).
- *G3* – Nota total obtida no último período do curso (de 0 a 20).

O atributo escolhido para fazer a previsão, foi o *G3* que representa a nota final obtida no último período do curso. O objetivo do estudo em classificar os estudantes de acordo com o desempenho nesse atributo, se deve ao fato da forte correlação com os atributos *G2* e *G1*. Isso ocorre porque *G3* é resultado do último ano, enquanto *G1* e *G2* correspondem às séries anteriores. Como a nota representada pelo atributo *G3* varia entre 0 e 20, os alunos foram classificados da seguinte maneira:

- A - Para alunos com notas maiores que 16

- B - Para alunos com notas acima de 12 e menores ou iguais a 16
- C - Para alunos com notas acima de 8 e menores ou iguais a 12
- D - Para alunos com notas acima de 4 e menores ou iguais a 8
- E - Para alunos com notas menores ou iguais a 4

Neste estudo são usados os resultados de 350 alunos para prever o resultado dos outros 45 e para aplicar a mineração dos dados utilizamos o *Decision Tree* para gerar a árvore de decisão.

Inicialmente, a árvore foi gerada sem alteração nos parâmetros, ou seja, com valores definidos como padrão pelo Rapidminer. Como o conjunto de dados usado nessa análise possui vários atributos e muitos alunos, a árvore obtida possui muitas regras, o que ocasiona a complexidade da sua análise. Para melhor entendimento, o parâmetro *criterion* foi alterado. Para cada uma das quatro opções disponíveis do parâmetro, uma árvore diferente foi gerada. Analisando os modelos, é possível perceber que o nó raiz foi o atributo *G2* em todos os modelos. Isso nos permite concluir que as notas obtidas no segundo ano do curso são muito importantes para classificar os alunos.

Outro fator importante a ser considerado, é que dos 32 atributos utilizados para gerar o grafo da árvore, 12 não apareceram, pois o algoritmo não considerou esses atributos relevantes para a classificação. São eles: *school*, *address*, *famsize*, *Pstatus*, *Fjob*, *schoolsup*, *paid*, *activities*, *nursery*, *internet*, *romantic*, *Dalc*.

Com o intuito de encontrar o modelo com a maior *accuracy*, as quatro opções para o *criterion* (*Gain ratio*, *Information gain*, *Gini index* e *Accuracy*) foram testadas, e em todas, a análise foi realizada com o *maximal depth* (na Tabela 3.1 representado por *DE*) variando do valor mínimo ao máximo. Observando a Tabela 3.1, pode-se extrair algumas informações.

Tabela 3.1: Comparativo entre parâmetros no conjunto de dados *Student Performance Data Set*

| Gain ratio | Information gain | Gini index | Accuracy | DE |
|---|---|---|---|---|
| 76.57% +/- 6.23% | 77.43% +/- 5.34% | 83.43% +/- 3.79% | 82.57% +/- 5.78% | 15 |
| 76.86% +/- 6.69% | 77.43% +/- 5.34% | 83.43% +/- 3.79% | 82.57% +/- 5.78% | 14 |
| 76.86% +/- 6.69% | 77.43% +/- 5.34% | 83.43% +/- 3.79% | 82.57% +/- 5.78% | 13 |
| 75.43% +/- 7.36% | 77.43% +/- 5.34% | 83.43% +/- 3.79% | 82.57% +/- 5.78% | 12 |
| 76.86% +/- 6.57% | 77.43% +/- 5.34% | 83.43% +/- 3.79% | 82.57% +/- 5.78% | 11 |
| 76.29% +/- 6.77% | 77.43% +/- 5.34% | 83.43% +/- 3.79% | 82.57% +/- 5.78% | 10 |
| 78.00% +/- 6.65% | 77.14% +/- 5.11% | 83.43% +/- 3.79% | 82.57% +/- 5.78% | 9 |
| 78.00% +/- 6.27% | 78.00% +/- 5.58% | 83.43% +/- 3.79% | 82.57% +/- 5.78% | 8 |
| 72.29% +/- 6.79% | 79.14% +/- 5.72% | 83.43% +/- 3.79% | 82.57% +/- 5.78% | 7 |
| 77.14% +/- 7.00% | 78.57% +/- 6.67% | 83.43% +/- 4.00% | 83.14% +/- 4.51% | 6 |
| 75.14% +/- 8.19% | 81.14% +/- 6.15% | 82.86% +/- 4.04% | 82.86% +/- 4.61% | 5 |
| 71.14% +/- 5.92% | 77.71% +/- 6.36% | 82.57% +/- 5.34% | 82.57% +/- 4.86% | 4 |
| 62.29% +/- 3.93% | 71.14% +/- 6.93% | 77.43% +/- 2.98% | 77.43% +/- 2.98% | 3 |
| 45.71% +/- 6.56% | 61.71% +/- 3.66% | 62.86% +/- 2.86% | 62.86% +/- 2.86% | 2 |
| 40.29% +/- 0.86% | 40.29% +/- 0.86% | 40.29% +/- 0.86% | 40.29% +/- 0.86% | 1 |

- Em todas as opções do parâmetro *criterion*, com a profundidade inferior ou igual a 5, todas as *accuracy* decresceram, ou seja, com a profundidade baixa, a precisão diminuiu.

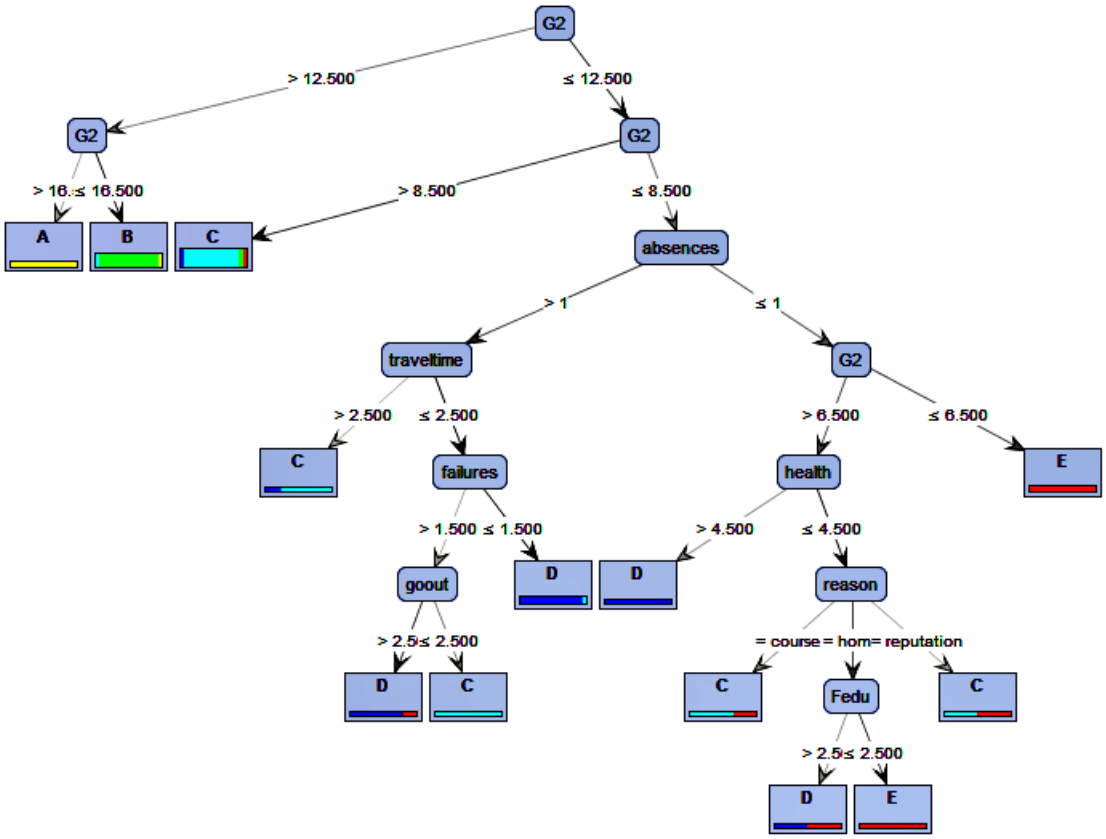
- Para profundidade mínima, todas as *accuracy* tiveram o mesmo valor (40.29%). Essa porcentagem foi a menor *accuracy* obtida na análise.

- A maior *accuracy* conseguida foi 83.43%, usando a opção *gini index* com a profundidade máxima, que nesse caso é 8. Essa *accuracy* pode ser considerada alta, já que, segundo Hämäläinen and Vinni (2010), as taxas médias em trabalhos relacionados ao assunto, são de 72%.

Vale destacar que o *criterion* mais usado no Rapidminer é o *gain ratio*, utlizado por 76% dos usuários e, nesta análise, essa opção com a profundidade máxima selecionada, forneceu a menor *accuracy* dentre todos os *criterion*.

A árvore escolhida para uma análise mais detalhada, foi a com maior *accuracy*. O grafo dela pode ser observado através da Figura 3.1

Figura 3.1: Grafo com maior *accuracy* (*Student Performance Data Set*)




O grafo da árvore confirma que o nó raiz é o atributo *G2*. Nesse caso, acreditamos que aulas extras, monitorias e atividades complementares nesse ano do curso pode interferir nos resultados do ano final do curso.

Além disso, é possível perceber que apenas oito atributos apareceram no grafo. São eles: *G2*, *absences*, *traveltime*, *failures*, *goout*, *health*, *reason*, *fedu*. Dessa forma, para melhorar os resultados, sugerimos que um trabalho de conscientização com os alunos seja feito. O intuito seria interferir no atributo *absences*, mostrando para o aluno a importância de comparecer à escola todos os dias.

Outra forma que poderia modificar positivamente os resultados, é interferir no atributo *failures*, oferecendo uma atividade de reforço aos alunos que já são repetentes. Na sequência é possível observar a *Tree*, ou seja, a árvore que gerou o grafo da Figura 3.1.

```
Tree
G2 > 12.500
|   G2 > 16.500: A {D=0, C=0, B=0, A=19, E=0}
|   G2 <= 16.500: B {D=0, C=7, B=86, A=4, E=0}
G2 <= 12.500
```

```
|   G2 > 8.500: C {D=11, C=123, B=11, A=0, E=7}
|   G2 <= 8.500
|   |   absences > 1
|   |   |   traveltime > 2.500: C {D=1, C=3, B=0, A=0, E=0}
|   |   |   traveltime <= 2.500
|   |   |   |   failures > 1.500
|   |   |   |   |   goout > 2.500: D {D=4, C=0, B=0, A=0, E=1}
|   |   |   |   |   goout <= 2.500: C {D=0, C=3, B=0, A=0, E=0}
|   |   |   |   failures <= 1.500: D {D=33, C=2, B=0, A=0, E=0}
|   |   absences <= 1
|   |   |   G2 > 6.500
|   |   |   |   health > 4.500: D {D=4, C=0, B=0, A=0, E=0}
|   |   |   |   health <= 4.500
|   |   |   |   |   reason = course: C {D=0, C=2, B=0, A=0, E=1}
|   |   |   |   |   reason = home
|   |   |   |   |   |   Fedu > 2.500: D {D=1, C=0, B=0, A=0, E=1}
|   |   |   |   |   |   Fedu <= 2.500: E {D=0, C=0, B=0, A=0, E=5}
|   |   |   |   |   reason = reputation: C {D=0, C=1, B=0, A=0, E=1}
|   |   |   G2 <= 6.500: E {D=0, C=0, B=0, A=0, E=19}
```

Para uma análise mais detalhada, o modelo obtido foi testado com o auxílio do Rapidminer em um conjunto teste com 45 alunos aleatórios. O resultado desse teste está no Apêndice A. A seguir, tem-se a Tabela 3.2 com a previsão dada pela plataforma para cada aluno. Os dados dessa tabela foram retirados das Tabelas A.1 e A.2.

Tabela 3.2: Resultado da aplicação do *Decision Tree* ao conjunto *Student Performance Data Set*

| Row no. | Final | prediction (Res. Final) | conf.(D) | conf.(C) | conf.(B) | conf.(A) | conf.(E) |
|---|---|---|---|---|---|---|---|
| 15 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 16 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 17 | ? | B | 0,000 | 0,072 | 0,887 | 0,041 | 0,000 |
| 18 | ? | E | 0,000 | 0,000 | 0,000 | 0,000 | 1,000 |
| 19 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 20 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 21 | ? | C | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 |
| 22 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 23 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 24 | ? | D | 0,943 | 0,057 | 0,000 | 0,000 | 0,000 |
| 25 | ? | A | 0,000 | 0,000 | 0,000 | 1,000 | 0,000 |

Nota-se que algumas previsões são dadas com 100% de precisão, como nas linhas 18, 21 e 25, por exemplo. Na linha 18, o programa fornece 100% de confiança para que o aluno obtenha conceito E, na Tabela 3.2 conf.(E). Já na linha 25, a confiança de 100% é para que o aluno obtenha conceito A, na Tabela 3.2 conf.(A). Nesse caso, sugere-se que o professor proponha uma parceria entre os dois alunos, de forma que o aluno com previsão de conceito A, auxilie o aluno com previsão de conceito E, com o intuito de melhorar seu desempenho na disciplina.

Após a execução do processo de validação do modelo, obtivemos a matriz de confusão, que pode ser observada através da Tabela 3.3.

Tabela 3.3: Resultado da validação do modelo aplicado ao conjunto de dados *Student Performance Data Set*

| accuracy:83.43% +/- 3.79% (mikro:83.43%) | | | | | | |
|---|---|---|---|---|---|---|
| | *True.* D | *True.* C | *True.* B | *True.* A | *True.* E | *Class Precision* |
| *Pred.* D | 38 | 6 | 0 | 0 | 1 | 84.44% |
| *Pred.* C | 13 | 124 | 12 | 0 | 8 | 78.98% |
| *Pred.* B | 0 | 7 | 85 | 4 | 0 | 88.54% |
| *Pred.* A | 0 | 0 | 0 | 19 | 0 | 100% |
| *Pred.* E | 3 | 4 | 0 | 0 | 26 | 78.79% |
| *Class Recall* | 70.37% | 87.94% | 87.63% | 82.61% | 74.29% | |

Nota-se que a *accuracy* foi de 83.43% e o desvio padrão de +/-3.79%. É possível perceber que na coluna *True*. D, tem-se 54 alunos com conceito D, dos quais, o programa previu 38 corretamente. Já na coluna *True*. C, tem-se 141 alunos com esse conceito, dos quais, o programa previu 124 corretamente. Analogamente, na coluna *True*. B, há 85 alunos previstos corretamente, em um total de 97. Na coluna *True*. A, foram previstos 19 corretamente, de um total de 23 alunos já na coluna *True* E, de um total de 35 alunos, 26 foram previstos corretamente.

Uma informação importante a ser considerada, é a previsão do conceito A, que foi de 100%. Nesse caso, podemos usar tal fato para selecionar alunos que possam ser monitores da disciplina de Matemática.

## 3.2 Informações úteis e baixa *accuracy*

Neste estudo, o objetivo é fornecer ao leitor informações úteis, mesmo quando o modelo tiver uma baixa *accuracy*. O conjunto de dados utilizados foi coletado em um curso de Engenharia Química da Universidade McMaster disponível em Datasets (2011). Ele possui seis atributos e dados de 99 alunos. A seguir, tem-se os atributos e seus significados.

- *Prefix* - Esse atributo é um prefixo que indica o ano em que o aluno fez a primeira matrícula na Universidade.
- *Assignment* - Esse atributo possui valores que representam a média das notas de todas as tarefas realizadas pelos alunos.
- *Tutorial* - Esse atributo possui valores que representam a média das notas de participação em todos os tutoriais realizados.
- *Midterm* – Esse atributo fornece a nota obtida pelos alunos em uma avaliação realizada aproximadamente no meio do curso.
- *TakeHome* – Esse atributo fornece as notas da avaliação realizada em casa.
- *Final* – Esse atributo representa a média de todas as perguntas da avaliação final, uma prova escrita.

Esse curso permite que os alunos trabalhem em grupos para tarefas, tutoriais e no exame realizado em casa. Para isso, os grupos foram selecionados e variados durante o semestre.

O objetivo da análise é descobrir o quanto os atributos são bons indicadores do desempenho do aluno na avaliação final, atributo escolhido para fazer a previsão. Como a

nota da avaliação final varia entre 28.06 e 108.9, os alunos foram classificados da seguinte maneira:

- A - Para alunos com notas maiores que 100
- B - Para alunos com notas acima de 80 e menores ou iguais a 100
- C - Para alunos com notas acima de 60 e menores ou iguais a 80
- D - Para alunos com notas acima de 40 e menores ou iguais a 60
- E - Para alunos com notas menores ou iguais a 40

Nessa análise são usados os resultados de 79 alunos para prever o resultado dos outros 20, e para aplicar a mineração dos dados utilizamos o *Decision Tree* para gerar a árvore de decisão.

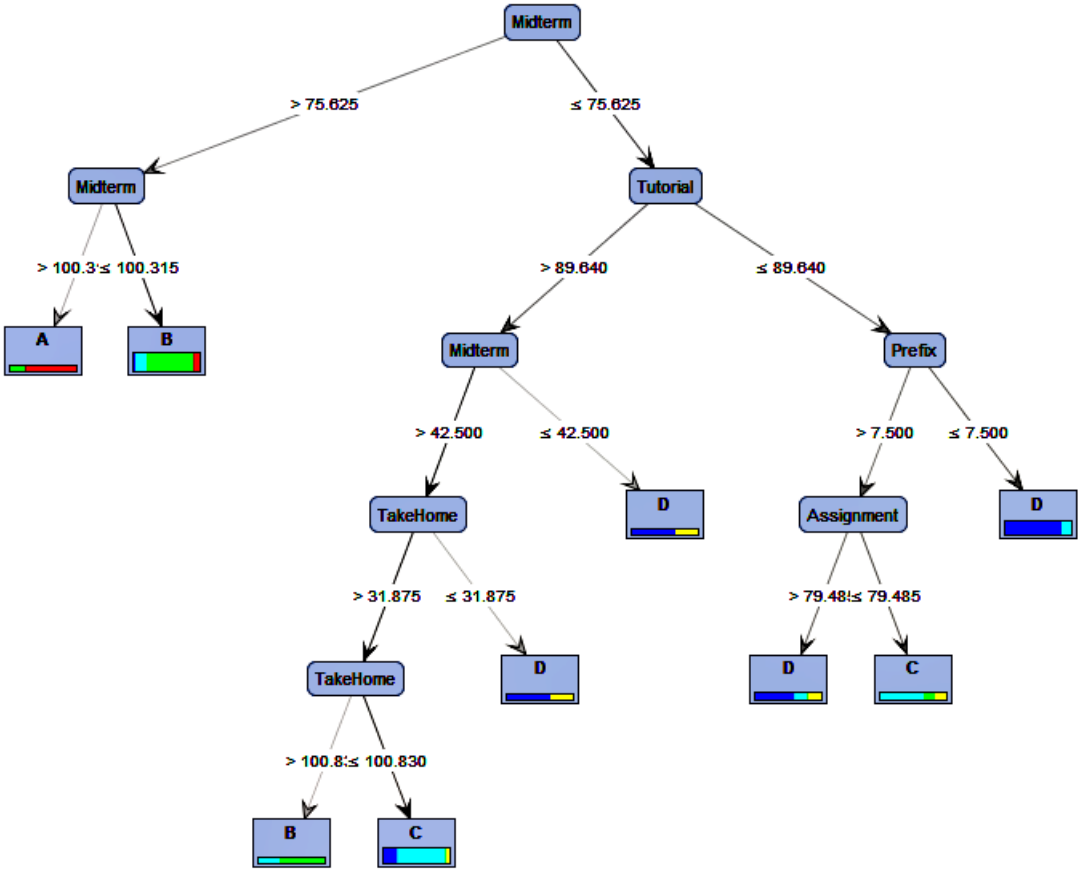
Na Tabela 3.4, tem-se os resultados das validações dos modelos gerados, usando as quatro opções do parâmetro *criterion* (*Gain ratio*, *Information gain*, *Gini index* e *accuracy*) e variando o parâmetro *maximal depth* (representado na Tabela 3.4 por DE) da profundidade máxima à mínima.

Tabela 3.4: Comparativo entre parâmetros no conjunto de dados *Class grades*

| Gain ratio | Information gain | Gini index | Accuracy | DE |
|---|---|---|---|---|
| 31.61% +/- 21.02% | 39.29% +/- 11.85% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 19 |
| 32.86% +/- 20.97% | 39.29% +/- 11.85% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 18 |
| 32.86% +/- 20.97% | 39.29% +/- 11.85% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 17 |
| 31.61% +/- 21.75% | 39.29% +/- 11.85% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 16 |
| 32.96% +/- 20.97% | 39.29% +/- 11.85% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 15 |
| 32.86% +/- 22.41% | 39.29% +/- 11.85% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 14 |
| 34.29% +/- 21.11% | 39.29% +/- 11.85% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 13 |
| 32.86% +/- 22.41% | 39.29% +/- 11.85% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 12 |
| 29.11% +/- 20.97% | 39.29% +/- 11.85% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 11 |
| 30.36% +/- 20.25% | 39.29% +/- 11.85% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 10 |
| 29.11% +/- 20.92% | 40.54% +/- 12.24% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 9 |
| 33.04% +/- 23.35% | 39.29% +/- 11.85% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 8 |
| 31.61% +/- 23.81% | 39.29% +/- 11.85% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 7 |
| 39.29% +/- 21.28% | 39.29% +/- 11.85% | 39.46% +/- 14.91% | 50.71% +/- 16.91% | 6 |
| 40.54% +/- 18.37% | 40.71% +/- 18.88% | 39.46% +/- 17.78% | 46.96% +/- 16.17% | 5 |
| 44.29% +/- 15.93% | 41.96% +/- 17.32% | 43.21% +/- 13.33% | 46.96% +/- 12.95% | 4 |
| 46.79% +/- 15.65% | 38.04% +/- 16.85% | 40.54% +/- 12.24% | 45.71% +/- 13.20% | 3 |
| 36.61% +/- 15.03% | 46.96% +/- 10.26% | 49.46% +/- 7.19% | 48.21% +/- 8.03% | 2 |
| 25.36% +/- 1.07% | 25.36% +/- 1.07% | 25.36% +/- 1.07% | 25.36% +/- 1.07% | 1 |

O modelo com a maior *accuracy* foi conseguido com a opção *accuracy* do parâmetro *criterion*, com o *maximal depth* na profundidade máxima, que no caso desse modelo é 6, o que pode ser observado no grafo da árvore na Figura 3.2 .

Figura 3.2: Grafo com maior *accuracy* (*Class grades*)




Nota-se que o nó raiz é o atributo *Midterm* e que todos os atributos foram utilizados para a construção da árvore. Acreditamos que uma interferência positiva poderia ocorrer adotando uma medida que melhorasse os resultados da avaliação realizada no meio do curso. Sugerimos ao professor que medidas como listas complementares, trabalhos em grupos e reforço extraclasse, sejam tomadas para a preparação dessa prova.

```
Tree
Midterm > 75.625
|   Midterm > 100.315: A {D=0, C=0, B=1, E=0, A=3}
|   Midterm <= 100.315: B {D=1, C=4, B=16, E=0, A=2}
Midterm <= 75.625
|   Tutorial > 89.640
|   |   Midterm > 42.500
|   |   |   TakeHome > 31.875
|   |   |   |   TakeHome > 100.830: B {D=0, C=1, B=2, E=0, A=0}
|   |   |   |   TakeHome <= 100.830: C {D=4, C=13, B=0, E=1, A=0}
```

```
|   |   |   TakeHome <= 31.875: D {D=2, C=0, B=0, E=1, A=0}
|   |   Midterm <= 42.500: D {D=2, C=0, B=0, E=1, A=0}
|   Tutorial <= 89.640
|   |   Prefix > 7.500
|   |   |   Assignment > 79.485: D {D=3, C=1, B=0, E=1, A=0}
|   |   |   Assignment <= 79.485: C {D=0, C=4, B=1, E=1, A=0}
|   |   Prefix <= 7.500: D {D=12, C=2, B=0, E=0, A=0}
```

O modelo representado pela *Tree* acima é obtido através do operador *Decision Tree* aplicado ao conjunto de dados estudado. O professor pode utilizar este modelo para classificar novos estudantes através de operadores específicos do Rapidminer (ver Subseção 2.1.8) e, também, com programas em Python (ver Apêndice B).

Ainda com o auxílio do Rapidminer, fez-se uma aplicação do modelo obtido para testar a confiança dos resultados. Os 99 alunos analisados foram separados em duas partes: 79 alunos com os resultados finais obtidos e 20 com uma coluna vazia para que o programa forneça a previsão do *label* em cada linha. É possível perceber, na Tabela 3.5, os 20 alunos que não possuem o resultado. A Tabela completa encontra-se no Apêndice A dividida em duas partes (Tabela A.3 e Tabela A.4).

Tabela 3.5: Resultado da aplicação do modelo com maior *accuracy* no conjunto *Class grades*

| Row no. | Final | Previsão(Res. Final) | conf.(D) | conf.(C) | conf.(B) | conf.(E) | conf.(A) |
|---|---|---|---|---|---|---|---|
| 1 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 2 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 3 | ? | D | 0,500 | 0,000 | 0,000 | 0,500 | 0,000 |
| 4 | ? | C | 0,000 | 0,500 | 0,500 | 0,000 | 0,000 |
| 5 | ? | B | 0,000 | 0,000 | 1,000 | 0,000 | 0,000 |
| 6 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 7 | ? | D | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 8 | ? | C | 0,000 | 0,500 | 0,500 | 0,000 | 0,000 |
| 9 | ? | C | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 |
| 10 | ? | B | 0,000 | 0,000 | 1,000 | 0,000 | 0,000 |
| 11 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 12 | ? | B | 0,000 | 0,000 | 0,500 | 0,000 | 0,500 |
| 13 | ? | C | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 |
| 14 | ? | D | 0,500 | 0,000 | 0,500 | 0,000 | 0,000 |
| 15 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 16 | ? | C | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 |
| 17 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 18 | ? | C | 0,000 | 0,500 | 0,500 | 0,000 | 0,000 |
| 19 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 20 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |

Nota-se que o atributo *Final* ainda não tem nenhum valor. Mas há uma nova coluna, *Previsão (Res. Final)*, que exibe uma previsão sobre a classificação do aluno, de A até E, com base nas regras da árvore de decisão. Além da previsão, os resultados também relatam a confiança da predição do Rapidminer. Na linha 3, por exemplo, tendo em conta os outros atributos, com base nas regras do modelo, há uma chance de 50% que o aluno obtenha conceito D, na Tabela 3.5 conf.(D), e 50% de chance de que ele tenha conceito E, na Tabela 3.5 conf.(D). Seja D ou E, o resultado desse aluno pode ser melhorado com uma intervenção do professor, como por exemplo, aulas de reforço. Já na linha 10, o modelo fornece 100% de confiança no resultado do aluno ser conceito B, na Tabela 3.5 conf.(B). Nesse caso, o aluno pode ser selecionado para trabalhar como monitor e auxiliar o professor a ajudar os alunos com mais dificuldade.

Na Tabela 3.6, temos o resultado da validação do modelo selecionado para análise. A *accuracy* foi de 50.71% e o desvio padrão de +/- 16.91%. Percebe-se que na coluna *True.* D (Tabela 3.6), tem-se 24 alunos com conceito D, dos quais, o programa previu 11 corretamente. Já na coluna *True.* C, tem-se 25 alunos com esse conceito, dos quais, o programa previu 13 corretamente. Analogamente, na coluna *True.* B, há 13 alunos previstos corretamente, em um total de 20. Na coluna *True.* E, há um total de 5 alunos, mas nenhum foi previsto corretamente. Na coluna *True.* A, de um total de 5 alunos, 3 foram previstos corretamente.

Tabela 3.6: Resultado da validação do modelo aplicado ao conjunto de dados *Class grades*

| accuracy:50.71% +/- 16.91% (mikro:50.63%) | | | | | | |
|---|---|---|---|---|---|---|
| | *True.* D | *True.* C | *True.* B | *True.* E | *True.* A | *Class Precision* |
| *Pred.* D | 11 | 8 | 0 | 3 | 0 | 50.00% |
| *Pred.* C | 11 | 13 | 3 | 2 | 0 | 44.83% |
| *Pred.* B | 1 | 4 | 13 | 0 | 2 | 65.00% |
| *Pred.* E | 1 | 0 | 0 | 0 | 0 | 0.00% |
| *Pred.* A | 0 | 0 | 4 | 0 | 3 | 42.86% |
| *Class Recall* | 45.83% | 52.00% | 65.00% | 0.00% | 60.00% | |

Um fato importante a considerar, é que a melhor previsão apareceu no conceito B. Tal fato, pode ser utilizado para classificação de alunos com possibilidades de receber bolsas de estudo. Esses alunos também poderiam tornarem-se tutores dos grupos de estudos propostos anteriormente, para melhorar o resultado da avaliação realizada no meio do curso.

O modelo ter precisão baixa é um fato que pode acontecer com qualquer professor e para melhorar a *accuracy* sugere-se que a quantidade de dados da amostra seja aumentado, assim como, aumentar a quantidade de atributos também pode ajudar a melhorar a situação Gottardo et al. (2012).

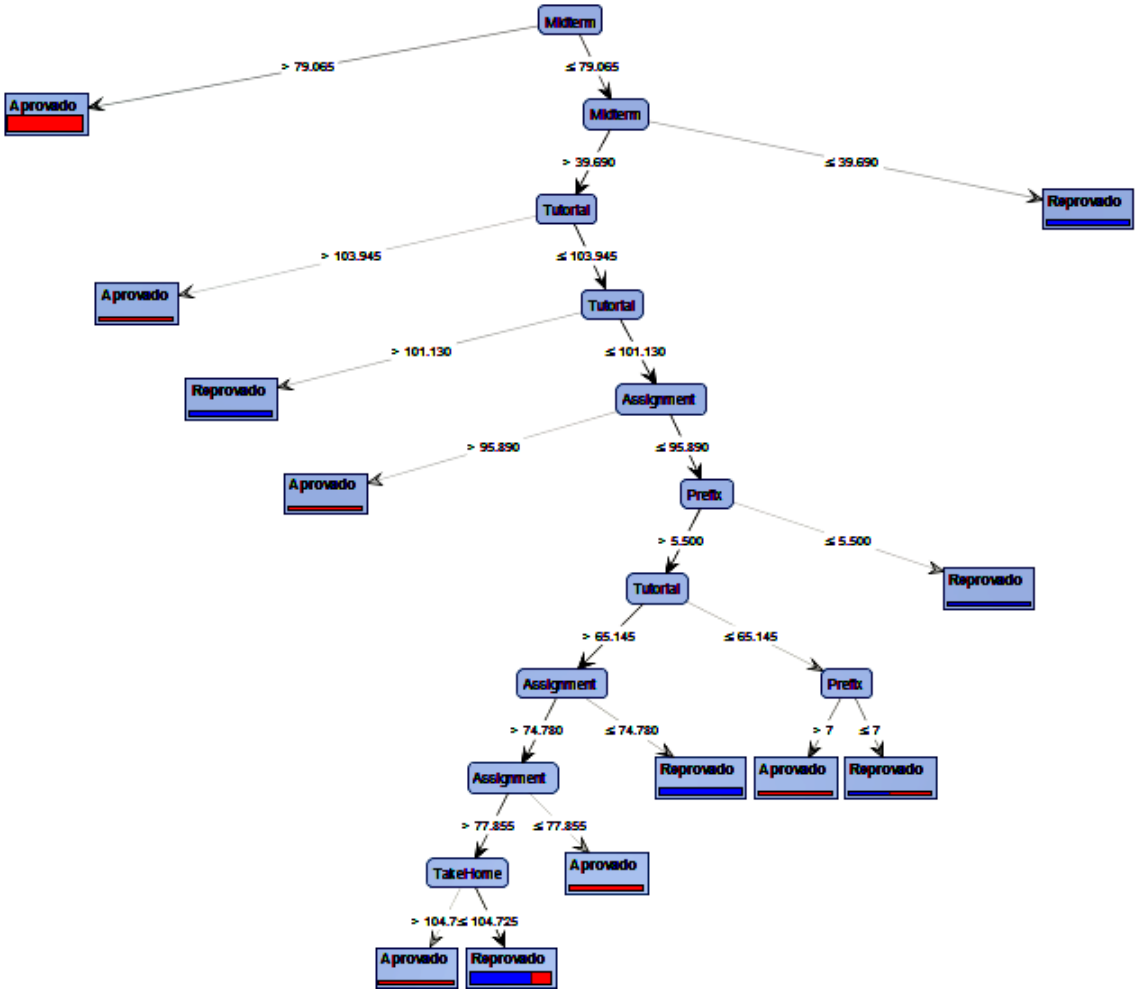
Com o intuito de aumentar a *accuracy*, classificamos os alunos de uma forma diferente. O conjunto de dados, assim como, todos os atributos foram mantidos, porém a nova classificação foi feita da seguinte maneira:

- Aprovado - Para alunos com notas superiores ou iguais a 60% do total.
- Reprovado - Para alunos com notas inferiores a 60% do total.

Nessa nova análise, como na anterior, foram usados os resultados de 79 alunos para prever o resultado dos outros 20, e utilizamos novamente o *Decision Tree* para gerar

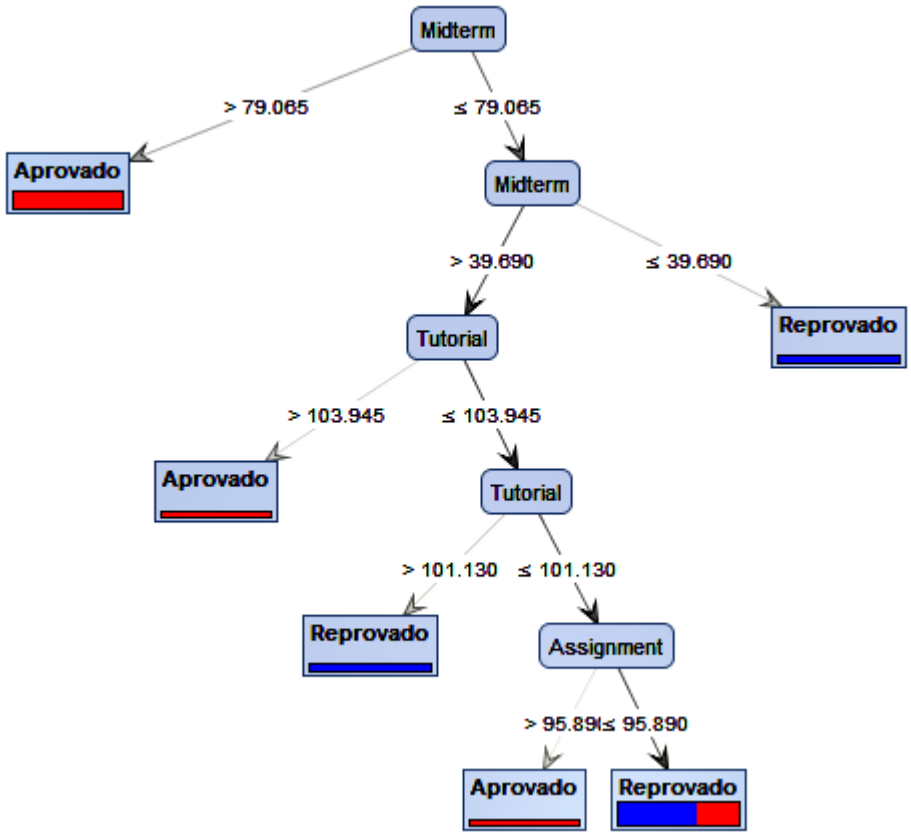
a árvore. Os parâmetros *criterion* e *maximal depth* foram alterados com a intenção de encontrar o modelo com a maior *accuracy*, que foi obtido com o *criterion* na opção *Gain ratio* e o *maximal depth* na profundidade máxima. O grafo da árvore encontrada pode ser observado na Figura 3.3.

Figura 3.3: Grafo com maior *accuracy* (*Class grades* com reclassificação)




Assim como no grafo da Figura 3.2, nesse, o nó-raiz foi o atributo *Midterm* e todos os outros atributos foram utilizados para a construção da árvore. Caso seja de interesse do leitor analisar uma árvore menor, o parâmetro *maximal depth* pode ser alterado, e para o conjunto de dados em questão, diminuir esse parâmetro não influenciou a *accuracy*. Na Figura 3.4, apresentamos o grafo da árvore com profundidade 6, de tal maneira que os atributos *Take Home* e *Prefix* foram omitidos, e o nó-raiz *Midterm* foi mantido.

Figura 3.4: Maior *accuracy* (*Class grades* com reclassificação e profundidade menor)




Na tentativa de facilitar a interpretação do modelo, mostramos que é possível reduzir o tamanho da árvore, sem alterar a *accuracy*, o que pode ser uma ferramenta útil para o professor. O resultado da validação dos modelos pode ser observado na Tabela 3.7.

Tabela 3.7: Resultado da validação do modelo aplicado ao conjunto *Class grades* com uma reclassificação

| accuracy:76.07% +/- 10.20% (mikro:75.95%) | | | |
|---|---|---|---|
| | *True.* Reprovado | *True.* Aprovado | *Class Precision* |
| *Pred.* Reprovado | 36 | 18 | 66.67% |
| *Pred.* Aprovado | 1 | 24 | 96.00% |
| *Class Recall* | 97.30% | 57.14% | |

Nota-se que a *accuracy* foi de 76% e o desvio padrão de +/- 10.20%. Comparando com a validação do modelo anterior, exibida na Tabela 3.6, podemos perceber que houve uma melhora significativa. Assim, o objetivo de aumentar a *accuracy*, alterando a maneira de classificar os alunos, foi atingido. Sugerimos então, que para melhorar modelos com baixa *accuracy*, o leitor modifique a maneira de classificar os alunos, isso pode al-

terar a *accuracy* de maneira positiva. Analisando um pouco mais a tabela, nota-se que dos 37 alunos reprovados, o programa previu 36 corretamente, o que pode favorecer às intervenções para os alunos em risco de reprovação. Como possibilidade de intervenção, sugerimos que o professor utilize medidas já citadas anteriormente, como: monitorias, listas complementares, acompanhamento individual e outros.

# 4 CONSIDERAÇÕES FINAIS

Neste trabalho, apresentamos dois exemplos para classificação de estudantes a partir de resultados anteriores. Neles usamos o desempenho dos estudantes para detectar precocemente as falhas e os potenciais dos alunos, e a partir desses resultados, determinar uma ação que possa impactar no aprendizado do aluno. Utilizou-se árvores de decisão para classificação dos alunos de acordo com o resultado final, com base em atributos disponíveis em bancos de dados reais. Para tal, o Rapidminer, programa usado para criação das árvores, apresentou simples manuseio, pois dispensa conhecimento de qualquer linguagem de programação, e fácil acessibilidade, pois é um *software* livre. Devido ao número de informações relevantes retiradas dos modelos criados, em modelos com alta e baixa *accuracy*, conclui-se que as árvores de decisão são importantes ferramentas para auxílio didático e pedagógico do professor, tanto de ensino superior, como de ensino médio. Sugere-se a criação de projetos de apoio e intervenção didática e pedagógica nos grupos classificados como grupos de risco como opção para amenizar as dificuldades dos estudantes e melhorar a qualidade do ensino.

Para um trabalho futuro, pode-se estender a experiência usando-se outras técnicas de mineração de dados para classificação de alunos com altos índices de reprovação.

Avaliar intervenções didáticas e pedagógicas em grupos classificados como de risco, é muito importante para solidificar o uso de ferramentas de mineração de dados nas escolas e universidades brasileiras.

# Referências Bibliográficas


Qasem A Al-Radaideh, AA Ananbeh, and Emad M Al-Shawakfa. A classification model for predicting the suitable study track for school students. *Int. J. Res. Rev. Appl. Sci*, 8(2):247–252, 2011.

Ryan Baker, Seiji Isotani, and Adriana Carvalho. Mineraçao de dados educacionais: Oportunidades para o brasil. *Revista Brasileira de Informática na Educação*, 19(02): 03, 2011.

Brijesh Kumar Baradwaj and Saurabh Pal. Mining educational data to analyze students' performance. *arXiv preprint arXiv:1201.3417*, 2012.

Michael J Berry and Gordon Linoff. *Data mining techniques: for marketing, sales, and customer support*. John Wiley & Sons, Inc., 1997.

Alfredo Nazareno P Boente, Ronaldo R Goldschmidt, Vânia Vieira Estrela, and Estadual da Zona Oeste. Uma metodologia de suporte ao processo de descoberta de conhecimento em bases de dados. 2008.

Peter Cabena, Pablo Hadjinian, Rolf Stadler, Jaap Verhees, and Alessandro Zanasi. *Discovering data mining: from concept to implementation*. Prentice-Hall, Inc., 1998.

Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and data Engineering*, 8(6):866–883, 1996.

Luiza Maria Oliveira Da Silva. *Uma aplicação de árvores de decisão, redes neurais e KNN para a identificação de modelos ARMA não-sazonais e sazonais*. PhD thesis, PUC-Rio, 2005.

OpenMV.net Datasets. Class grades, 2011. URL `http://openmv.net/info/class-grades`. Acessado em: 10/08/2016.

Douglas Detoni, Ricardo Matsumura Araujo, and Cristian Cechinel. Predição de reprovação de alunos de educação a distância utilizando contagem de interações. In *Anais do Simpósio Brasileiro de Informática na Educação*, volume 25, page 896, 2014.

Alaa El-Halees. Mining students data to analyze e-learning behavior: A case study. *Department of Computer Science, Islamic University of Gaza PO Box*, 108, 2009.

Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

Rozelma Soares França and Haroldo José Costa do Amaral. Mineração de dados na identificação de grupos de estudantes com dificuldades de aprendizagemno ensino de programação. *RENOTE*, 11(1), 2013.

João Gama, Pedro Medas, Pedro Rodrigues, and FEP LIACC. Concept drift in decision-tree learning for data streams. In *Proceedings of the Fourth European Symposium on Intelligent Technologies and their implementation on Smart Adaptive Systems, Aachen, Germany, Verlag Mainz*, pages 218–225, 2004.

Diego Garcıa-Saiz and Marta Zorrilla. Comparing classification methods for predicting distance students' performance. In *Proceedings of the International Workshop on Applications of Pattern Analysis (to appear)*, 2011.

Ronaldo Goldschmidt and Emmanuel Passos. *Data Mining: um guia prático*. Gulf Professional Publishing, 2005.

Ernani Gottardo, Celso Kaestner, and Robinson Vida Noronha. Avaliação de desempenho de estudantes em cursos de educação a distância utilizando mineração de dados. In *Anais do Workshop de Desafios da Computação Aplicada à Educação*, pages 30–39, 2012.

Wilhelmiina Hämäläinen and Mikko Vinni. Classifiers for educational data mining. *Handbook of Educational Data Mining, Chapman & Hall/CRC Data Mining and Knowledge Discovery Series*, pages 57–71, 2010.

Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

Dorina Kabakchieva. Predicting student performance by using data mining methods for classification. *Cybernetics and information technologies*, 13(1):61–72, 2013.

Irfan Ajmal Khan and Jin Tak Choi. An application of educational data mining (edm) technique for scholarship prediction. *International Journal of Software Engineering and Its Applications*, 8(12):31–42, 2014.

Zlatko Kovacic. Early prediction of student success: Mining students' enrolment data. 2010.

Laci Mary Barbosa Manhães, Sérgio Manuel Serra da Cruz, Raimundo J Macário Costa, Jorge Zavaleta, and Geraldo Zimbrão. Previsão de estudantes com risco de evasão utilizando técnicas de mineração de dados. In *Anais do Simpósio Brasileiro de Informática na Educação*, volume 1, 2011.

Tom M Mitchell. Machine learning., chapter evaluating hypotheses. *WCB/McGraw-Hill*, pages 128–153, 1997.

Luis F de S Moro, Carla L Rodrigues, Fernando RH Andrade, Alexandre CB Delbem, and Seiji Isotani. Caracterização de alunos em ambientes de ensino online: Estendendo o uso da damicore para minerar dados educacionais. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 3, page 631, 2014.

Ashutosh Nandeshwar and Subodh Chaudhari. Enrollment prediction models using data mining. *Retrieved January*, 10:2010, 2009.

Taiane S Prass, Sílvia RC Lopes, José G Dórea, Rejane C Marques, and Katiane G Brandao. Amazon forest fires between 2001 and 2006 and birth weight in porto velho. *Bulletin of environmental contamination and toxicology*, 89(1):1–7, 2012.

Rapidminer, 2006. URL `https://rapidminer.com`. Acessado em: 20/04/2016.

Machine Learning Repository. Student performance data set, 2014. URL `https://archive.ics.uci.edu/ml/datasets/Student+Performance#`. Acessado em: 03/08/2016.

Lior Rokach and Oded Maimon. *Data mining with decision trees - theory and applications*, volume 81. World Scientific, 2 edition, 2014.

Cristóbal Romero, Sebastián Ventura, Pedro G Espejo, and César Hervás. Data mining algorithms to classify students. In *Educational Data Mining 2008*, 2008.

Cristobal Romero, Sebastian Ventura, Mykola Pechenizkiy, and Ryan SJD Baker. *Handbook of educational data mining*. CRC Press, 2010.

Henrique Santos, Fabiane Camargo, and Sandro Camargo. Minerando dados de ambientes virtuais de aprendizagem para predição de desempenho de estudantes. *Conferencias LACLO*, 3(1), 2012.

Cybele T. M. Vinagre, Renata R. Del Vecchio, and João B. Carvalho. introdução à teoria espectral de grafos e aplicações. 2004.

Ian H Witten, Eibe Frank, Leonard E Trigg, Mark A Hall, Geoffrey Holmes, and Sally Jo Cunningham. Weka: Practical machine learning tools and techniques with java implementations. 1999.

# A Tabelas de aplicação

A seguir tem-se a aplicação do modelo obtido para testar a confiança dos resultados de 45 alunos do conjunto de dados *Student Performance Data Set*. Nessa aplicação, o Rapidminer fornece uma tabela com 40 colunas. São elas:

- Uma coluna com a numeração das linhas
- Uma coluna vazia que representa o resultado final do aluno
- Uma coluna com a previsão do resultado final do aluno
- Cinco colunas com a confiança da previsão do Resultado Final
- Trinta e duas colunas com os valores dos atributos

A fim de visualizar apenas os resultados fornecidos pelo programa, as colunas com os atributos listados na Seção 3.1 do Capítulo 3 foram retiradas da tabela original. Para uma melhor visualização a tabela foi dividida em duas partes: Tabela A.1 e Tabela A.2.

Tabela A.1: Resultado da aplicação do *Decision Tree* ao conjunto *Student Performance Data Set* - parte 1

| Row no. | Final | prediction(Res. Final) | conf.(D) | conf.(C) | conf.(B) | conf.(A) | conf.(E) |
|---|---|---|---|---|---|---|---|
| 1 | ? | C | 0,250 | 0,750 | 0,000 | 0,000 | 0,000 |
| 2 | ? | B | 0,000 | 0,072 | 0,887 | 0,041 | 0,000 |
| 3 | ? | D | 0,943 | 0,057 | 0,000 | 0,000 | 0,000 |
| 4 | ? | C | 0,250 | 0,750 | 0,000 | 0,000 | 0,000 |
| 5 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 6 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 7 | ? | B | 0,000 | 0,072 | 0,887 | 0,041 | 0,000 |
| 8 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 9 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 10 | ? | B | 0,000 | 0,072 | 0,887 | 0,041 | 0,000 |
| 11 | ? | B | 0,000 | 0,072 | 0,887 | 0,041 | 0,000 |
| 12 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 13 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 14 | ? | B | 0,000 | 0,072 | 0,887 | 0,041 | 0,000 |
| 15 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 16 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 17 | ? | B | 0,000 | 0,072 | 0,887 | 0,041 | 0,000 |
| 18 | ? | E | 0,000 | 0,000 | 0,000 | 0,000 | 1,000 |
| 19 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 20 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |

Tabela A.2: Resultado da aplicação do *Decision Tree* ao conjunto *Student Performance Data Set* - parte 2

| Row no. | Final | prediction(Res. Final) | conf.(D) | conf.(C) | conf.(B) | conf.(A) | conf.(E) |
|---|---|---|---|---|---|---|---|
| 21 | ? | C | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 |
| 22 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 23 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 24 | ? | D | 0,943 | 0,057 | 0,000 | 0,000 | 0,000 |
| 25 | ? | A | 0,000 | 0,000 | 0,000 | 1,000 | 0,000 |
| 26 | ? | C | 0,250 | 0,750 | 0,000 | 0,000 | 0,000 |
| 27 | ? | B | 0,000 | 0,072 | 0,887 | 0,041 | 0,000 |
| 28 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 29 | ? | B | 0,000 | 0,072 | 0,887 | 0,041 | 0,000 |
| 30 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 31 | ? | B | 0,000 | 0,072 | 0,887 | 0,041 | 0,000 |
| 32 | ? | D | 0,943 | 0,057 | 0,000 | 0,000 | 0,000 |
| 33 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 34 | ? | E | 0,000 | 0,000 | 0,000 | 0,000 | 1,000 |
| 35 | ? | D | 0,943 | 0,057 | 0,000 | 0,000 | 0,000 |
| 36 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 37 | ? | C | 0,250 | 0,750 | 0,000 | 0,000 | 0,000 |
| 38 | ? | E | 0,000 | 0,000 | 0,000 | 0,000 | 1,000 |
| 39 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 40 | ? | E | 0,000 | 0,000 | 0,000 | 0,000 | 1,000 |
| 41 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 42 | ? | B | 0,000 | 0,072 | 0,887 | 0,041 | 0,000 |
| 43 | ? | D | 0,800 | 0,000 | 0,000 | 0,000 | 0,200 |
| 44 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |
| 45 | ? | C | 0,072 | 0,809 | 0,072 | 0,000 | 0,046 |

Na sequência tem-se o resultado da aplicação do modelo obtido para testar a confiança dos dados de 20 alunos do conjunto de dados *Class grades*, usado na Seção 3.2 do Capítulo 3. Para melhor visualização, os dados foram divididos em duas partes: Tabela A.3 e Tabela A.4.

Tabela A.3: Resultado da aplicação do *Decision Tree* no conjunto de dados *Class grades* - parte 1

| Row no. | Final | Previsão(Res. Final) | conf.(D) | conf.(C) | conf.(B) | conf.(E) | conf.(A) |
|---|---|---|---|---|---|---|---|
| 1 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 2 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 3 | ? | D | 0,500 | 0,000 | 0,000 | 0,500 | 0,000 |
| 4 | ? | C | 0,000 | 0,500 | 0,500 | 0,000 | 0,000 |
| 5 | ? | B | 0,000 | 0,000 | 1,000 | 0,000 | 0,000 |
| 6 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 7 | ? | D | 1,000 | 0,000 | 0,000 | 0,000 | 0,000 |
| 8 | ? | C | 0,000 | 0,500 | 0,500 | 0,000 | 0,000 |
| 9 | ? | C | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 |
| 10 | ? | B | 0,000 | 0,000 | 1,000 | 0,000 | 0,000 |
| 11 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 12 | ? | B | 0,000 | 0,000 | 0,500 | 0,000 | 0,500 |
| 13 | ? | C | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 |
| 14 | ? | D | 0,500 | 0,000 | 0,500 | 0,000 | 0,000 |
| 15 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 16 | ? | C | 0,000 | 1,000 | 0,000 | 0,000 | 0,000 |
| 17 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 18 | ? | C | 0,000 | 0,500 | 0,500 | 0,000 | 0,000 |
| 19 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |
| 20 | ? | C | 0,417 | 0,472 | 0,056 | 0,056 | 0,000 |

Tabela A.4: Resultado da aplicação do *Decision Tree* no conjunto de dados *Class grades* - parte 2

| Row no. | Prefix | Assignment | Tutorial | Midterm | TakeHome |
|---|---|---|---|---|---|
| 1 | 8,00 | 63,40 | 86,21 | 63,12 | 72,78 |
| 2 | 6,00 | 90,74 | 89,64 | 61,25 | 90,00 |
| 3 | 8,00 | 71,79 | 102,87 | 41,88 | 24,77 |
| 4 | 8,00 | 97,33 | 106,74 | 76,88 | 108,89 |
| 5 | 4,00 | 86,86 | 62,64 | 92,50 | 85,19 |
| 6 | 6,00 | 95,60 | 61,40 | 64,38 | 99,81 |
| 7 | 4,00 | 87,93 | 99,47 | 53,12 | 87,96 |
| 8 | 6,00 | 98,49 | 95,43 | 42,50 | 24,77 |
| 9 | 7,00 | 74,35 | 92,93 | 86,25 | 78,70 |
| 10 | 7,00 | 86,29 | 88,81 | 83,12 | 77,96 |
| 11 | 8,00 | 97,00 | 100,52 | 64,38 | 90,74 |
| 12 | 8,00 | 97,33 | 106,74 | 81,25 | 108,89 |
| 13 | 8,00 | 96,41 | 103,71 | 56,25 | 95,93 |
| 14 | 7,00 | 95,60 | 82,28 | 76,88 | 108,33 |
| 15 | 8,00 | 87,52 | 91,58 | 56,25 | 71,85 |
| 16 | 8,00 | 96,73 | 103,71 | 45,00 | 93,52 |
| 17 | 7,00 | 85,34 | 80,54 | 41,25 | 93,70 |
| 18 | 8,00 | 89,94 | 102,77 | 87,50 | 90,74 |
| 19 | 7,00 | 95,60 | 76,13 | 66,25 | 99,81 |
| 20 | 8,00 | 63,40 | 97,37 | 73,12 | 72,78 |

# B Classificação de novos estudantes via Python

Considere a seguinte Tree:

```
Tree
Midterm > 75.625
|   Midterm > 100.315: A {D=0, C=0, B=1, E=0, A=3}
|   Midterm <= 100.315: B {D=1, C=4, B=16, E=0, A=2}
Midterm <= 75.625
|   Tutorial > 89.640
|   |   Midterm > 42.500
|   |   |   TakeHome > 31.875
|   |   |   |   TakeHome > 100.830: B {D=0, C=1, B=2, E=0, A=0}
|   |   |   |   TakeHome <= 100.830: C {D=4, C=13, B=0, E=1, A=0}
|   |   |   TakeHome <= 31.875: D {D=2, C=0, B=0, E=1, A=0}
|   |   Midterm <= 42.500: D {D=2, C=0, B=0, E=1, A=0}
|   Tutorial <= 89.640
|   |   Prefix > 7.500
|   |   |   Assignment > 79.485: D {D=3, C=1, B=0, E=1, A=0}
|   |   |   Assignment <= 79.485: C {D=0, C=4, B=1, E=1, A=0}
|   |   Prefix <= 7.500: D {D=12, C=2, B=0, E=0, A=0}
```

Neste apêndice, apresentaremos o *script* em Python para classificação de novos estudantes. O leitor notará como a linguagem se adequa a Tree de maneira intuitiva devido ao seu alto nível.

```
def classificarEstudante( Midterm,Tutorial,TakeHome,Prefix,Assignment):
    if Midterm > 75.625:
        if Midterm > 100.315: return "A"
        if Midterm <= 100.315: return "B"
    if Midterm <= 75.625:
```

```
    if Tutorial > 89.640:
        if Midterm > 42.500:
            if TakeHome > 31.875:
                if TakeHome > 100.830: return "B"
                if TakeHome <= 100.830: return "C"
            if TakeHome <= 31.875: return "D"
        if Midterm <= 42.500: return "D"
    if Tutorial <= 89.640:
        if Prefix > 7.500:
            if Assignment > 79.485: return "D"
            if Assignment <= 79.485: return "C"
        if Prefix <= 7.500: return "D"
```

Para classificar um novo estudante basta atribuir valores para as variáveis *Midterm*, *Tutorial*, *TakeHome*, *Prefix* e *Assignment*. Por exemplo:

```
print(classificarEstudante( Midterm = 70.0, Tutorial=85.0, TakeHome=40.0,
Prefix=8.0, Assignment=70.0 ) )
```

Os códigos acima podem ser executados nas versões 2.7 e 3.5 do Python.