UNIVERSIDADE FEDERAL FLUMINENSE

HUGO ARRAES HENLEY

PRELUDE: Uma Arquitetura para Análise de Evasão no Ensino Superior por meio de Aprendizado de Máquina Relacional

NITERÓI

2018

UNIVERSIDADE FEDERAL FLUMINENSE

HUGO ARRAES HENLEY

PRELUDE: Uma Arquitetura para Análise de Evasão no Ensino Superior por meio de Aprendizado de Máquina Relacional

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Engenharia de Sistemas e Informação

Orientador:
Daniel Cardoso Moraes de Oliveira

Co-orientador:
Aline Marins Paes Carvalho

NITERÓI

2018

Ficha catalográfica automática - SDC/BEE

H514p Henley, Hugo Arraes
PRELUDE: Uma Arquitetura para Analise de Evasao no Ensino Superior por meio de Aprendizado de Maquina Relacional / Hugo Arraes Henley ; Daniel Cardoso Moraes de Oliveira, orientador ; Aline Marins Paes Carvalho, coorientadora. Niterói, 2018.
63 p. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2018.

DOI: http://dx.doi.org/10.22409/PGC.2018.m.12439331783

1. ILP. 2. Aprendizado de Máquina. 3. Aprendizado Relacional. 4. Evasão de Alunos do Ensino Superior. 5. Produção intelectual. I. Título II. Cardoso Moraes de Oliveira,Daniel, orientador. III. Marins Paes Carvalho, Aline, coorientadora. IV. Universidade Federal Fluminense. Escola de Engenharia.

CDD -

Bibliotecária responsável: Fabiana Menezes Santos da Silva - CRB7/5274

HUGO ARRAES HENLEY

PRELUDE: Uma Arquitetura para Análise de Evasão no Ensino Superior por meio de Aprendizado de Máquina Relacional

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Engenharia de Sistemas e Informação

Aprovada em Maio de 2018.

BANCA EXAMINADORA

Prof. Daniel Cardoso Moraes de Oliveira - Orientador, UFF

Profª. Aline Marins Paes Carvalho - Co-orientadora, UFF

Prof. Alexandre Plastino de Carvalho, UFF

Profª. Kate Cerqueira Revoredo, UNIRIO

Niterói

2018

Dedico este trabalho à minha esposa Priscila, por ter me apoiado tanto, sempre.

Agradecimentos

À minha família, por ter me ensinado desde cedo a importância do estudo e por nada ter me deixado faltar.

À minha esposa Priscila, por ter sido sempre tão companheira em todos os momentos. Este Mestrado não seria possível se não fosse todo o seu apoio nesses últimos anos. Obrigado pela sua compreensão em todos os momentos que precisei me dedicar aos estudos.

Aos meus orientadores Daniel de Oliveira e Aline Paes. Vocês foram simplesmente incríveis. Sou muito grato por toda a orientação que me deram no último ano. Tenho certeza que este trabalho não seria possível sem toda a ajuda e ensinamento de vocês. Muito obrigado!

À STI/UFF, por ter cedido os dados que tornaram essa Dissertação possível. Além disso, sou muito grato por todo o aprendizado que esta Superintendência me proporcionou enquanto fui funcionário.

À UFF, por ter me proporcionado ensino superior gratuito e de qualidade, tanto na graduação quanto na pós-graduação.

Aos membros da banca examinadora, por terem aceitado fazer parte da mesma.

Resumo

A evasão de alunos no ensino superior é um problema atualmente enfrentado por diversos países no mundo. No Brasil esse problema também ocorre, e é especialmente crítico quando ocorre em instituições públicas federais e estaduais, visto que estas universidades oferecem ensino gratuito, sendo financiadas pelo governo. Dessa forma, a evasão de alunos pode significar uma falta de retorno do investimento realizado pelo governo, já que as chances destes alunos ingressarem no mercado de trabalho se tornam reduzidas e, por consequência, há dificuldade de suprir a demanda interna necessária pelas empresas, importante para o desenvolvimento do país. Apesar deste tema fazer parte de debates e reflexões na comunidade educacional brasileira, a predição de evasão de alunos não é uma tarefa trivial devido à grande quantidade de dados envolvidos, e suas relações. Além disso, na utilização de algoritmos de aprendizado de máquina tradicionais, todos os dados relacionados aos estudantes devem ser guardados em uma única relação. Isso pode não ser factível e, mesmo quando é, pode reduzir as chances de inferir novo conhecimento. Esse trabalho introduz o PRELUDE (dropout PRediction using machinE Learning for UnDErgraduate courses) como uma abordagem para detecção de evasão de alunos utilizando aprendizado lógico-relacional através de Programação Lógica Indutiva, onde é possível gerar regras que descrevem o comportamento da evasão destes alunos. As regras foram aprendidas através da utilização de três estratégias diferentes: treinamento dos cursos separadamente, em grupos de acordo com as taxas de evasão, e em grupos de acordo com suas áreas científicas. A técnica de validação cruzada foi utilizada como uma das formas de verificação do desempenho do modelo, além do uso de treinamento com dados passados e validação através de dados futuros, e a verificação da capacidade de generalização das regras, quando regras aprendidas por um curso são utilizadas em outro(s). Por fim, as regras geradas são discutidas para avaliar se podem ser compreendidas.

Palavras-chave: aprendizado de máquina relacional, predição de evasão de alunos, ilp, aleph

Abstract

Students' dropout in higher education is a problem currently faced by several countries in the world. In Brazil this problem also occurs, and is especially critical when it occurs in public institutions, since these universities offer free education, being financed by the government. In this way, student dropout can mean a lack of return on the investment made by the government, since the chances of these students entering the job market become reduced and, consequently, it is difficult to supply the internal demand of the companies, important for the development of the country. Although this topic is part of debates and reflections in the Brazilian educational community, effectively detecting students at risk of dropout is not a trivial task due to a large amount of data involved, and their relationships. On top of that, to use traditional machine learning algorithms, all data associated with students in a University should be stored in a single relation. This may not be feasible, and even when it is, it can reduce the chances to infer new knowledge. This work introduces the PRELUDE (dropout PRediction using machinE Learning for UnDErgraduate courses) as an approach for detecting students' dropout using logical-relational learning through Inductive Logic Programming, where it is possible to generate rules that describe the behavior of these dropouts. The rules were learned through the use of three different strategies: training courses separately, in groups according to dropout rates, and in groups according to their scientific areas. The cross-validation technique was used as one of the ways to verify the performance of the model, besides the use of training with past data and validation through future data, and the verification of the generalization capacity of the rules, when rules learned by one or more courses are used in others. Finally, the generated rules are discussed to evaluate its comprehensibility.

Keywords: relational learning, student dropout prediction, ilp, aleph

Lista de Figuras

2.1 Conjunto de trens de Michalski 17

4.1 Arquitetura do PRELUDE 28

6.1 F-Measure por curso e por estratégia quando o modelo é testado com dados de 2011. 52

Lista de Tabelas

1.1 Estatísticas para Universidades públicas no Brasil (fonte: Censo educacional do governo brasileiro) 2

2.1 Fatos lógicos relacionados a dados simplificados de uma Universidade. . . . 14

4.1 Estudantes e seus cursos 29

4.2 Classificação etária 30

5.1 Taxas de Evasão por curso 36

5.2 Número de exemplos por curso, considerando os anos de 2009 e 2010. . . . 37

5.3 Predicados usados como Background Knowledge 38

6.1 Resultados preditivos da validação cruzada para o Grupo I (alta taxa de evasão), com cursos treinados separadamente. 43

6.2 Resultados da validação cruzada para o Grupo II (taxa média de evasão), com cursos treinados separadamente. 44

6.3 Resultados da validação cruzada para o Grupo III (baixa taxa de evasão), com cursos treinados separadamente. 44

6.4 Resultados preditivos da validação cruzada quando os cursos pertencentes ao mesmo Grupo são treinados juntos 45

6.5 Resultado da validação cruzada quando diferentes cursos são combinados . 45

6.6 Resultados preditivos da validação cruzada com cursos agrupados pelas áreas de pesquisa 46

6.7 Desempenho quando regras aprendidas por um ou mais cursos são aplicadas em outros cursos. 47

6.8 Regras aprendidas quando apenas o curso é treinado separadamente, contendo dados de alunos matriculados em 2009 e 2010, aplicadas em alunos ingressantes em 2011 do mesmo curso treinado. 49

6.9 Regras aprendidas pelo Grupo I, quando todos os cursos deste grupo são treinados juntos, testadas em cada curso do grupo, contendo alunos ingressantes em 2011. 49

6.10 Regras aprendidas pelo Grupo II, quando todos os cursos deste grupo são treinados juntos, testadas em cada curso do grupo, contendo alunos ingressantes em 2011. 50

6.11 Regras aprendidas pelo Grupo III, quando todos os cursos deste grupo são treinados juntos, testadas em cada curso do grupo, contendo alunos ingressantes em 2011. 50

6.12 Regras aprendidas através do treinamento de todos os cursos de Ciências Exatas juntos, nos anos de 2009 e 2010, testadas com alunos ingressantes em 2011 de cada um dos cursos do grupo. 51

6.13 Regras aprendidas através do treinamento de todos os cursos de Ciências Biomédicas juntos, nos anos de 2009 e 2010, testadas com alunos ingressantes em 2011 de cada um dos cursos do grupo. 51

6.14 Tempos de execução do treinamento, considerando o somátorio do tempo de cada fold, por curso. 55

6.15 Tempos de execução do treinamento, considerando o somátorio do tempo de cada fold, por grupos separados por taxa de evasão. 55

6.16 Tempos de execução do treinamento, considerando o somátorio do tempo de cada fold, por grupos separados por área de pesquisa. 55

Lista de Abreviaturas e Siglas

UFF	:	Universidade Federal Fluminense;
MDMR	:	Mineração de dados multi-relacional;
ILP	:	Inductive Logic Programming;
PRELUDE	:	Prediction using Machine Learning for Undergraduate courses;
BK	:	Background Knowledge;
LRL	:	Lógica e Aprendizado Relacional;
ETL	:	Extract-Load-Transform;
SVM	:	Support Vector Machine;
ROC	:	Receiver Operating Characteristic;
MLP	:	Multilayer Perceptron;
UFPE	:	Universidade Federal de Pernambuco;
UFCG	:	Universidade Federal de Campina Grande;
STI/UFF	:	Superintendência de Tecnologia da Informação da UFF;

Sumário

1 Introdução **1**

1.1 Apresentação do Problema 1

1.2 Proposta 3

1.3 Motivação 7

1.4 Organização do Trabalho 8

2 Aprendizado de Máquina Relacional **9**

2.1 Prolog 10

2.2 Programação em Lógica Indutiva 12

2.3 Aleph: A Learning Engine for Proposing Hypotheses 15

2.3.1 Aplicação em Aprendizado de Máquina 17

3 Trabalhos Relacionados **22**

4 PRELUDE: um método para predição de evasão de alunos de graduação através de aprendizado relacional **27**

4.1 Arquitetura da Solução 27

4.1.1 Explicação dos componentes 28

5 Configuração Experimental **33**

5.1 Parâmetros utilizados 33

5.2 Validação 34

5.3 Dados 35

5.4 Predicados 37

5.4.1 Base de conhecimento 38

5.4.2 Experimentos 40

6 Resultados **42**

6.1 Treinamento com cursos individualmente 43

6.2 Agrupamento de cursos considerando seus Grupos 45

6.3 Capacidade de generalização das regras 46

6.4 Avaliando a capacidade de predição do modelo 48

6.5 Análise qualitativa das regras 52

6.6 Tempo de execução do modelo 54

7 Conclusão e Trabalhos Futuros **57**

Referências **60**

Capítulo 1

Introdução

1.1 Apresentação do Problema

A redução da taxa de evasão em cursos de graduação é um importante problema em aberto, em nível mundial, segundo Husainy *et al.* [21]. Este problema é especialmente crítico em países subdesenvolvidos, como o Brasil. Isso ocorre, pois esses países apresentam orçamento educacional limitado (o orçamento educacional brasileiro foi reduzido de R$ 7,9 bilhões para R$ 6,7 bilhões em apenas um ano [1]). A evasão de estudantes no Brasil é comumente parte dos debates e reflexões na comunidade educacional brasileira e é uma das principais prioridades nas discussões sobre políticas públicas educacionais. Além disso, a evasão de alunos dessas instituições representa um gasto dos recursos já escassos e, consequentemente, representa a falta de retorno do investimento realizado pelo governo no grupo de alunos que evadem seus cursos, já que as chances desses alunos se incluírem no mercado de trabalho ficam reduzidas e, como consequência, o pagamento de impostos diretos e indiretos é reduzido, além de uma maior dificuldade por parte das empresas em suprir suas demandas internas através da contratação de profissionais qualificados. Além da perda financeira, e o investimento de tempo de todos os envolvidos no processo de formação de um aluno, o abandono dos estudos significa uma perda social e econômica [14].

Apesar do aumento no número de vagas nas instituições públicas, que tem como um de seus objetivos a inclusão de alunos em um sistema de educação de qualidade, de acordo com o Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira [INEP, 2010], em média, apenas 62,4% do total de alunos de uma Universidade pública consegue concluir seus estudos. Ainda, de acordo com o Censo, o número de alunos concluintes não

[1] https://g1.globo.com/educacao/noticia/mec-preve-orcamento-15-universidade-baixa-federal-universidades-em-2017.ghtml

Tabela 1.1: Estatísticas para Universidades públicas no Brasil (fonte: Censo educacional do governo brasileiro)

Ano	Vagas	Estudantes ingressantes	Estudantes matriculados	Estudantes concluintes
2005	127,334	143,731	579,587	86,011
2006	144,445	161,509	589,821	83,686
2007	155,040	172,334	615,542	89,257
2008	169,502	186,043	643,101	84,036
2009	210,236	223,624	752,847	91,576
2010	248,534	269,216	833,934	93,442
2011	270,121	282,007	927,086	98,383
2012	283,445	300,453	985,202	96,270
2013	291,444	299,203	1,045,507	107,792
2014	299,234	311,536	1,083,586	119,988

cresce na mesma proporção das vagas ofertadas ao longo do tempo, como pode ser visto na Tabela 1.1.

Não há, ainda, uma definição amplamente aceita para o termo evasão [52, 20]. Entretanto, nesse trabalho, é considerada uma evasão quando um dos três casos ocorre:

1. Quando o aluno efetua uma troca de curso dentro da mesma universidade;
2. Quando o aluno continua no curso mas troca de universidade;
3. Quando o aluno interrompe seus estudos.

O último tipo de evasão é o mais crítico, pois significa que o estudante passa a estar fora do sistema educacional brasileiro. Existem diversos motivos que podem estar relacionados a uma alta taxa de evasão, como problemas sócio-econônimos, acadêmicos (cursos inflexíveis e com conteúdo obsoleto) e questões demográficas (violência urbana, distância até a universidade), *etc* [49, 8]. Assim, é uma das principais prioridades das universidades e do governo entender as causas do abandono para planejar contra-medidas destinadas a reduzir as taxas de evasão. No entanto, devido ao enorme número de alunos (64.000 alunos de graduação, no caso da UFF), e os diversos fatores que podem estar relacionados com as evasões, a análise por aluno ou por grupo de alunos está longe de ser trivial. Dito isto, para melhor analisar as causas desse problema e ser capaz de planejar uma ação preventiva, o uso de algoritmos computacionais se faz necessária.

Em virtude dos fatos anteriormente apresentados, algumas pesquisas [40, 12, 5, 7, 6, 16] têm sido realizadas com o objetivo de analisar o comportamento dos alunos que

evadem, conseguindo prever a saída destes antes mesmo que ela ocorra. Uma vez que o número de alunos matriculados nessas instituições e a quantidade de informações associadas a cada aluno pode ser muito volumosa, análises manuais são inviáveis e tendenciosas ao erro, sendo dessa forma necessária a utilização de abordagens computacionais para realizar tais análises complexas. Algoritmos e modelos de aprendizado de máquina podem ser utilizados para predição de evasão e, uma vez que esses casos são identificados, é possível que cada coordenação de curso realize um melhor acompanhamento dos mesmos, entendendo as causas de cada evasão e tomando ações para que elas não aconteçam, como, por exemplo, destinar mais recursos aos alunos que mais necessitam. Esses recursos podem ser variados, desde financeiros até reforço escolar.

1.2 Proposta

Diversas abordagens têm sido desenvolvidas visando detectar as causas de alta taxas de abandono escolar nas universidades [12, 27, 45, 7]. Algumas das pesquisas relacionadas [6, 45] consideram este tema como um problema de classificação definido por dois grupos: alunos que terminam o curso e alunos que evadem.

Devido ao fato de alguns cursos terem um alto índice de evasão nos dois primeiros semestres, alguns trabalhos existentes optaram por investigar as evasões que ocorrem neste período. Atributos como idade, sexo, bairro de residência, coeficiente de rendimento, número de reprovações em disciplinas e curso do aluno foram utilizados para treinamento e validação da base de dados através de algoritmos como Naive Bayes, C5.0, Support Vector Machine (SVM), Multilayer Perceptron (MLP), Logit, entre outros.

Embora tais abordagens apresentem resultados úteis, ainda há espaço para otimizações. A maioria dos modelos de previsão existentes é baseada em algoritmos clássicos de aprendizado supervisionado, onde os dados são amostrados de forma homogênea e idêntica, onde estes são rotulados e representados em uma única relação [37]. Representar todos os dados associados aos alunos de uma universidade em uma única relação pode não ser viável e, mesmo quando for, pode exigir o acréscimo de atributos derivados que, de certa forma, consolidem as informações contidas no banco de dados. Um exemplo de atributo derivado é o número de reprovações, que pode ser extraído através do histórico do aluno. Nesse processo pode haver perda de informação, visto que um novo atributo foi criado para representar um outro conjunto de atributos de forma única.

Por exemplo, vamos considerar que a Universidade X tem 4 tabelas em seu banco

de dados sobre as informações dos alunos: (i) uma tabela que armazena as notas e a progressão acadêmica dos alunos; (ii) uma tabela que armazena informações pessoais sobre os alunos, como endereço, idade, afiliação, *etc.*; (iii) uma tabela que armazena informações sobre a educação anterior do aluno (registros do ensino médio); e (iv) uma tabela que armazena informações sobre os cursos (disciplinas, professores, *etc.)*. Nesse cenário, é difícil gerar uma única relação a ser usada como entrada para um algoritmo tradicional de mineração de dados. Assim, é necessário considerar múltiplas relações no processo de mineração.

Além disso, um problema em utilizar algoritmos de mineração de dados tradicionais, como nos trabalhos anteriormente apresentados, é que estes assumem que os dados são homogêneos, independentes uns dos outros e descritos no formato de atributo-valor. No caso do uso de aprendizado relacional, os atributos são descritos de acordo com as suas relações reais, como as já existentes no banco de dados e é possível extrair conhecimento das informações da forma como ela se apresenta no conjunto de dados de origem.

Para realizar este trabalho, os dados de alunos da Universidade Federal Fluminense (UFF) foram utilizados, uma vez que a Superintendência de Tecnologia da Informação da UFF (STI/UFF), departamento responsável pela administração dos dados que guardam todo o histórico acadêmico de um aluno, além de algumas informações de natureza social e pessoal, cedeu, gentilmente, todo o acesso aos dados necessários para a pesquisa. Como estes dados obtidos a partir da base de dados da STI/UFF são comumente multi-relacionados, estruturados e amostrados a partir de relacionamentos complexos essa análise se torna difícil. Assim, devido a essa suposição, os algoritmos tradicionais como o C5.0 falham em capturar e descrever corretamente e de forma expressiva qual o comportamento do aluno que evadiu, bem como em expressar características dos relacionamentos expressos.

Por outro lado, a área de pesquisa de mineração de dados multi-relacional (Dzeroski e Lacc 2001) (i.e. MDMR) tem como objetivo principal induzir automaticamente hipóteses a partir de dados dispostos em múltiplas tabelas relacionadas. Assim, algoritmos de MDMR se tornam bastante adequados para descobrir padrões e descrever características dos relacionamentos inseridos no cenário apresentado. Nesse contexto, a área de Programação em Lógica Indutiva (i.e. ILP do inglês Inductive Logic Programming) (Raedt 2008) tem obtido bastante destaque em sua aplicação para MDMR, sendo a interseção de aprendizado de máquina e representação do conhecimento. Em ILP, as hipóteses são induzidas no formato de programas lógicos e com isso se beneficiam do poder de expressividade da

lógica de primeira-ordem (Nilsson e Maluszski 1995).

Além dos exemplos, um conhecimento preliminar a respeito do problema também pode ser fornecido como entrada para os algoritmos de ILP, ambos descritos por cláusulas definidas por lógica de primeira-ordem. Esta dissertação tem como objetivo propor um modelo preditivo de boa acurácia para alunos que evadem seus cursos utilizando MDMR.

Para superar os problemas mencionados, neste trabalho, o PRELUDE é proposto (dropout PRediction using machinE Learning for UnDErgraduate courses). O PRELUDE é uma nova abordagem de previsão de possível evasão que gera modelos de previsão baseados em algoritmos relacionais de aprendizado de máquina, mais especificamente em Aprendizagem Lógico-Relacional utilizando Programação em Lógica Indutiva (*i.e.* ILP) [15]. Um algoritmo de ILP é usado uma vez que algoritmos tradicionais de aprendizado de máquina, como o ID3, não são adequados para processar dados heterogêneos, distribuídos de forma não independente e relacionados. Além disso, ao induzir programas lógicos, ILP permite que especialistas em educação interpretem e analisem facilmente os modelos aprendidos. Desta forma, todos os dados coletados são convertidos em uma representação lógica para formar o conjunto de exemplos e usar o sistema de ILP chamado Aleph [48]. Para avaliar o PRELUDE, estudos experimentais foram realizados a partir de dados reais de estudantes da Universidade Federal Fluminense (UFF) [2] em Niterói, Brazil. Os experimentos foram conduzidos com dados de estudantes de graduação de 12 diferentes cursos. A avaliação experimental visa responder às seguintes questões:

1. **Q#1** Como obter regras precisas, aprendidas com um sistema de aprendizado baseado em lógica, referentes a evasão de alunos da universidade?

2. **Q#2** É mais apropriado aprender o modelo de previsão de abandono usando dados isolados de cada curso, ou aprender um modelo de previsão considerando dados de vários cursos juntos, realizando o treinamento em grupos?

3. **Q#3** Como reutilizar um modelo de previsão de abandono aprendido para um curso específico em outros cursos? Essa abordagem poderia ser benéfica para cursos com poucos exemplos positivos de evasão?

4. **Q#4** Como usar as regras aprendidas através do treinamento de anos passados para prever a evasão de alunos no anos seguinte?

[2] www.uff.br

5. **Q#5** As regras inferidas pelo modelo, que descrevem a evasão nos cursos de graduação, são compreensíveis?

1.3 Motivação

Considerando o cenário nacional referente ao alto índice de evasão de alunos das Universidades Federais no Brasil e o impacto relacionado ao alto custo representado pela manutenção dos alunos na Universidade ao longo dos cursos, o presente trabalho se faz necessário tendo em vista a economia de recursos financeiros que podem ser atingidos caso a taxa de evasão seja reduzida. Além disso, as universidades federais têm um importante papel no que diz respeito ao acesso a uma formação de qualidade para a população, elevando o nível sócio-econômico do país.

Devido ao grande número de alunos matriculados nas universidades públicas, um sistema de apoio à tomada de decisão se faz muito útil, podendo auxiliar a gestão das universidades na identificação dos alunos com elevado risco de evasão antes mesmo desta de fato ocorrer, possibilitando uma ação preventiva, muito difícil de ser realizada sem o auxílio da análise automática em grandes conjuntos de dados, possível através da utilização de técnicas de aprendizado de máquina.

Este trabalho tem, então, como objetivo a criação de um sistema baseado em aprendizado de máquina para uso inicialmente na Universidade Federal Fluminense. A expectativa é que a Universidade consiga melhorar sua capacidade de atuação junto aos alunos que apresentam risco de evasão. Hoje essa atenção direcionada não é viável devido ao grande número de alunos matriculados na Universidade e, além disso, os padrões de evasão são, em sua maioria, desconhecidos.

Explicações obtidas a partir do modelo induzido podem levar a decisões sobre carga horária, o professor que deve lecionar determinada disciplina, o melhor turno para alocar um curso, para que grupo de alunos é mais efetivo direcionar determinado tipo de bolsa acadêmica, a necessidade de criação de grupos de estudo, convocação de tutores e outras. As regras geradas pelo modelo podem, inclusive, ser acopladas ao sistema acadêmico da universidade (idUFF) de forma a executar essas inferências de forma automática e integrada.

1.4 Organização do Trabalho

Este trabalho está organizado em 6 capítulos além desta introdução:

- O Capítulo 2 apresenta o conhecimento teórico básico para compreensão do aprendizado de máquina relacional, assim como a Linguagem Prolog, e é finalizado com a apresentação de uma solução de um problema clássico na área de Aprendizado Relacional.
- O Capítulo 3 apresenta os trabalhos relacionados, que também apresentam diferentes abordagens para tratar a predição de alunos com risco de evasão de cursos de ensino superior e médio.
- O Capítulo 4 mostra a arquitetura do PRELUDE, sistema proposto neste trabalho, além de detalhar seus componentes.
- O Capítulo 5 discute a configuração experimental utilizada, isto é, os parâmetros, a disposição dos dados utilizados para fazer parte do treinamento, predicados lógicos definidos e técnicas utilizadas para validação dos resultados.
- O Capítulo 6 mostra e discute os resultados obtidos pelo PRELUDE. Nesse capítulo as questões aqui levantadas são respondidas.
- O Capítulo 7 conclui este trabalho, discutindo sobre a contribuição desta pesquisa para a elevação da qualidade da educação superior brasileira, além de sugerir trabalhos futuros.

Capítulo 2

Aprendizado de Máquina Relacional

Sistemas construídos a partir de técnicas de aprendizado de máquina são capazes de aprender a executar uma tarefa, levando em consideração experiências passadas [30]. Diversas técnicas de aprendizado são amplamente utilizadas há muitos anos para resolver diversos tipos de problemas em diversas áreas, como detecção de doenças através da análise de imagens [4], detecção de fraude em transações financeiras [23], reconhecimento facial [31], entre outras. Porém, muitas dessas técnicas possuem limitações no que diz respeito à representação de domínios que envolvam diferentes entidades e seus relacionamentos. Tendo em vista essa limitação imposta pelas técnicas de aprendizado convencionais (não relacionais), pesquisadores como Ryszard Michalski [1983] e Gordon Plotkin [1970] empregaram esforço na procura de um método que fosse mais rico nesse tipo de representação, chamada de relacional.

O aprendizado relacional pode ser utilizado em casos onde a natureza do dado é relacional e, como consequência, mostra claros benefícios no que diz respeito a representação do conhecimento gerado [42], pois pode ser interpretado pelo usuário.

Esse tipo de aprendizado tem sido empregado em diversos problemas onde o dado tem essa natureza, como por exemplo na classificação de documentos na web [28], categorização de textos [10], análise de ocorrência de crimes [53] e outros.

No caso do presente trabalho, a natureza dos dados é relacional, ou seja, existem diversas entidades, relacionamentos entre elas e deseja-se descobrir se existe uma relação tal que resulte na evasão de um aluno de graduação. Se essa relação existe, ela deve ser expressa de forma interpretável e facilmente aplicável em dados futuros, caracterizando assim o poder de predição. Esses dados são expressos por lógica, e esse tipo de abordagem é estudada pela sub-área de aprendizado de máquina conhecida como Programação em

Lógica Indutiva [34].

2.1 Prolog

O Prolog é uma linguagem de programação utilizada desde cerca de 1970 por programadores para resolver problemas computacionais que envolvem lógica matemática, análise de estruturas bioquímicas e diversas áreas de inteligência artificial. Como diversas implementações de aprendizado de máquina relacional são implementadas em Prolog, essa seção traz uma breve introdução à linguagem e apresenta alguns conceitos importantes da mesma.

A programação em Prolog envolve descrever fatos conhecidos e os relacionamentos de entidades em um determinado problema, além de quais são os relacionamentos verdadeiros dentro do contexto do problema dado. Desta forma, o Prolog é capaz de realizar inferências, sendo uma boa escolha quando o tema é aprendizado de máquina relacional.

Podemos expressar, através do Prolog, a posse de um determinado objeto por uma pessoa, por exemplo. Dizendo ao Prolog que Carlos é dono de um carro, podemos perguntar se o Carlos é realmente o dono de determinado carro e esperar "sim"como resposta. Podemos também realizar a pergunta no sentido inverso, que seria questionar se determinado carro é dono de Carlos, onde esperaríamos a resposta "não", caso o Prolog não consiga provar este caso.

Um programa em Prolog consiste de um conjunto de *cláusulas*, que podem ser *fatos* ou *regras* na seguinte forma:

```
head :- body.
```

O termo *head* é a cabeça da cláusula, enquanto o *body* é o corpo da cláusula. O corpo consiste de um ou mais componentes, separados por vírgulas, que são lidas como o conector **e** da lógica. Quando não há corpo, a cláusula é chamada de fato (pois o corpo é considerado sempre verdadeiro), como nos exemplos abaixo:

```
gosta(alice, bob).
cachorro(joca).
```

Quando há corpo, a cláusula é chamada de *regra* e tem a seguinte forma:

```
cachorro_grande(X):-cachorro(X),grande(X).
```

Variáveis são definidas por termos que começam com letra maiúscula, enquanto constantes (*atoms*) são definidas como termos que começam com letra minúscula. O fato representado acima por $gosta(alice, bob)$ consiste de dois objetos e mostra o relacionamento entre eles, expressando que *alice* gosta de *bob*, mas nada diz com relação ao sentimento de *bob* por *alice*. A ordem dos objetos é arbitrária, mas uma vez escolhida deve ser utilizada de maneira consistente ao longo do programa para que não se tenha problemas com relação à interpretação dos fatos e seus relacionamentos com as entidades.

Dessa forma, a programação em Prolog consiste basicamente de:

- Especificar alguns fatos sobre objetos e seus relacionamentos.
- Definir regras sobre os objetos e seus relacionamentos.
- Realizar perguntas sobre os objetos e seus relacionamentos.

Um exemplo com cláusulas e regras é usado abaixo para ilustrar os conceitos acima explicados:

```
pai(angelo,miguel).
pai(miguel,hugo).

avo(X,Y) :- pai(X,Z), pai(Z,Y).
```

Nesse caso, os fatos são dados por $pai(angelo, miguel)$. e $pai(miguel, hugo)$., onde *pai* é chamado de nome do *predicado*, enquanto os termos entre parênteses são chamados de argumentos. O número de argumentos define a *aridade* do predicado. Dessa forma, pode-se representar o predicado *pai* como $pai/2$, o que significa que este predicado tem aridade igual a 2. Ainda no exemplo acima, *angelo*, *miguel* e *hugo* são *atoms*.

Por último, a regra relacionada a *avo* é definida e deve ser lida como:

X é avô de Y **SE** X é pai de Z **E** Z é pai de Y.

Após definidos os fatos e as regras, pode-se efetuar perguntas ao Prolog, como:

```
?- pai(angelo,X).
X = miguel;

?- avo(angelo,X).
X = hugo;
```

Ao perguntar, através do uso da variável X, de quem *angelo* é pai, Prolog retorna X = *miguel*. A segunda pergunta é relacionada à regra definida como *avo* e, nesse caso, o Prolog retorna corretamente que *angelo* é avô de *hugo*.

Muitos problemas clássicos da literatura podem ser resolvidos com Prolog, como por exemplo o problema de coloração de grafos, que consiste em atribuir cores a certos elementos do grafo sujeito a determinadas condições. A coloração de grafos pode ser aplicada em diversos problemas reais, como o agendamento de provas na universidade ou separação de produtos explosivos. Para o primeiro, imagine que queiramos agendar provas na universidade de modo que duas disciplinas com estudantes em comum não tenham provas agendadas para o mesmo horário. Para o segundo, imagine que um conjunto de produtos químicos são necessários para um processo de produção e existem produtos que quando guardados juntos têm chance de explosão. Neste caso, deseja-se saber qual o menor número de comportimentos necessários para, dado um número de produtos P e as possíveis combinações que resultam em explosão, qual o menor número de compartimentos C para guardar os produtos em segurança.

2.2 Programação em Lógica Indutiva

Tradicionalmente, os métodos de aprendizado de máquina assumem que cada exemplo é apresentado em um formato proposicional, contendo atributos e seus valores [30]. Nesse cenário, cada exemplo é assumido como independente um do outro, com tamanho fixo (ou seja, utilizando os mesmos atributos), e distribuídos homogeneamente. No entanto, alguns domínios do mundo real, como os dispostos em bancos de dados de universidades, são organizados em um formato relacional. Embora seja possível transformar os dados em formato relacional em um conjunto de exemplos em formato proposicional, distribuídos de forma independente, esse processo pode introduzir dois problemas adicionais aos conjuntos de dados gerados: uma quantidade muito grande de atributos pode ser necessária e a dificuldade de criar atributos que representem conceitos espalhados entre os indivíduos e suas relações.

O aprendizado de máquina relacional se concentra em induzir modelos quando os exemplos têm uma estrutura complexa, composta de vários componentes relacionados, ou quando os exemplos são relacionados de tal forma que seus relacionamentos não podem ser ignorados durante a fase de aprendizado [50]. Os métodos que utilizam Aprendizado Lógico-Relacional (LRL) [15] manipulam essas relações internas e externas usando

representações baseadas em lógica, desenvolvidas no campo da Representação do Conhecimento [9], e combinadas com algoritmos de aprendizado de máquina. A maioria dos métodos desenvolvidos no campo LRL são exemplos de Programação Lógica Indutiva (ILP, do inglês *Inductive Logic Programming*) [32, 41].

A Programação em Lógica Indutiva é uma área de pesquisa de interseção entre o aprendizado de máquina e a programação lógica, tendo como objetivo aprender programas lógicos a partir de exemplos e de um conhecimento preliminar, conhecido como Base de conhecimento (BK, do inglês *Background Knowledge*) [44]. Esse tipo de método de aprendizado difere de outros métodos de aprendizado de máquina principalmente pela representação dos dados e modelo aprendido de simples compreensão por humanos e pela capacidade de aprendizado em domínios relacionais.

Um método ILP tem como objetivo induzir um programa lógico (uma teoria), constituído de cláusulas formadas por lógica de primeira ordem, a partir de um conjunto de exemplos e Base de conhecimento, ambos também expressos em lógica de primeira ordem. Mais especificamente, a principal tarefa a ser observada por um método ILP é a seguinte:

Dado:

- *Um conjunto E^+ de exemplos positivos e um conjunto E^- de exemplos negativos e,*
- *Base de conhecimento (BK),*

ambos expressos como cláusulas de primeira ordem.

Encontrar:

- *Uma hipótese $\mathcal{H}$ composta de cláusulas de primeira ordem tal que $BK \wedge \mathcal{H} \vDash E^+$ ($\mathcal{H}$ é completa) e $BK \wedge \mathcal{H} \nvDash E^-$ ($\mathcal{H}$ é consistente), isto é, $\mathcal{H}$ é correta.*

Considere, por exemplo, um domínio simplificado relacionado a um site de uma universidade típica [11, 15], contendo entidades pertencentes a estudantes, faculdades, funcionários, departamento, cursos, projetos e publicações. A tabela 2.1 exibe alguns fatos relacionados às entidades e seus relacionamentos pertencentes a esse domínio. Aqui, segue-se a convenção da Programação Lógica, onde letras minúsculas iniciam os nomes de constantes e predicados, enquanto letras maiúsculas iniciam o nome de uma variável. Por exemplo, o fato $departamento(url1, computacao)$ representa a informação que o departamento de *computacao* tem a URL $url1$, onde $departamento/2$ é um predicado com dois termos e $url1$ e *computacao* são termos constantes; $professorAluno(maria, joao)$

indica que existe uma relação de professor-aluno entre as entidades *maria* e *joao*. Em linguagem natural, podemos dizer a mesma informação que *Maria é professora de João*.

Tabela 2.1: Fatos lógicos relacionados a dados simplificados de uma Universidade.

departamento(url1,computacao).	siteAluno(url6,jorge).
curso(url3,estruturas_de_dados).	professorDisciplina(joao,estruturas_de_dados).
departamento(url2,engenharia).	alunoCurso(maria,ciencia_computacao).
siteAluno(url4,laura).	matriculaDisciplina(jorge,estruturas_de_dados).
professorAluno(maria,joao).	cidadeCurso(computacao,niteroi).

A informação pode ser enriquecida usando regras em seu *Background*. Por exemplo, a regra

$$professorAluno(Professor, Aluno) \leftarrow professorDisciplina(Professor, Disciplina), \\ matriculaDisciplina(Aluno, Disciplina).$$

afirma que *Aluno frequenta um Curso ministrado por um Professor SE o Professor ensina um Curso e o Aluno está matriculado nesse Curso.* Mesmo que não se tenha a relação $professorAluno$ explicitamente no banco de dados, ainda é possível gerar a informação que *Jorge é aluno do curso lecionado por João*, usando a regra acima e os fatos descritos na Tabela 2.1.

Agora, suponha que se deseja aprender um modelo que generalize o relacionamento de orientação professor-aluno. Nesse caso, os fatos originados do predicado $orientador/2$ seriam colocados no conjunto de exemplos. Assim, usando um sistema ILP, a seguinte regra pode ser descoberta automaticamente, a partir dos exemplos, dos outros fatos e das regras no BK:

$$orientador(Professor, Aluno) \leftarrow siteAluno(Url, Aluno), \\ contem(Url, orientando), contem(Url, Professor).$$

A regra afirma que *Professor é orientador do Aluno se o Aluno tem uma página com Url que contém a palavra orientando e contém o nome do Professor.*

O aprendizado padrão em ILP é conhecido como aprendizado a partir de implicação lógica, do inglês *learning from entailment* [18], e é seguido pela maioria dos sistemas [48, 35, 39]. Os algoritmos de ILP procuram no espaço de busca de cláusulas (que é definido como um *lattice*) usando ou uma abordagem *top-down*, ou *bottom-up* ou híbrida. Na

abordagem *top-down*, a busca por uma única cláusula começa com a mais geral, que prova todos os exemplos positivos e negativos, e o sistema a especializa a cada iteração para tornar os exemplos negativos improváveis, preservando ao máximo a prova dos exemplos positivos. A busca *bottom-up* faz o caminho inverso: parte da cláusula mais específica, criada a partir de um exemplo positivo e generaliza essa cláusula, visando provar outros exemplos positivos, mas tentando evitar provas de exemplos negativos.

2.3 Aleph: A Learning Engine for Proposing Hypotheses

Neste trabalho, o sistema Aleph [48] é utilizado, pois ele implementa algoritmos de ILP. O Aleph utiliza como padrão o aprendizado a partir de implicação lógica, assumindo que o BK é um conjunto de cláusulas *Horn* definidas [37] e considera os exemplos como fatos lógicos. Ele segue uma estratégia *hbrida*, baseando-se no conceito de uma cláusula mais específica, conhecida como *Bottom Clause* [33].

A *Bottom Clause* $\perp e$ criada com relação a um exemplo positivo e, e a Base de conhecimento (BK) é a cláusula mais específica dentro do espaço de hipóteses que cobre o exemplo e, ou seja,

$$BK \cup \perp e \vDash e \tag{2.1}$$

Qualquer cláusula que cubra o exemplo e em relação a BK deve ser mais geral que $\perp e$. Qualquer cláusula que não seja mais geral que $\perp e$ não pode cobrir e e pode ser desconsiderada com segurança. Assim, a *bottom-clause* limita a busca por uma cláusula que cubra o exemplo e, já que captura todas as informações relevantes para e e BK.

Em geral, $\perp$ poderia ter uma cardinalidade infinita. Assim, os sistemas *Mode Directed Inverse Entailment* [33], como Aleph, consideram um conjunto de *Mode declarations* (declarações de modo), definido pelo usuário juntamente com outras configurações para restringir a busca por uma boa hipótese. Uma declaração de modo [33] tem o formato $modeh(recall, atom)$ ou $modeb(recall, atom)$, em que *recall* é um número inteiro maior que 1 ou $'*'$ e *atom* é chamado de um átomo básico. Declarações do tipo *modeh* indicam predicados que devem aparecer na cabeça das cláusulas e *modeb*, predicados que devem aparecer no corpo das cláusulas. *Recall* é o número máximo de diferentes instanciações de *atom* permitidas a aparecer em uma cláusula (onde '*' significa um número indefinido de vezes). Os termos em atom têm seus tipos definidos, podendo ser variáveis de entrada ($+$), saída ($-$), ou constantes ($\#$). O significado das varíaveis é o seguinte:

- Entrada (+) - uma variável de entrada do tipo T em um literal do corpo B_i aparece como uma variável de saída do tipo T em um literal do corpo que aparece antes de B_i ou aparece como uma variável de entrada do tipo T na cabeça da cláusula.
- Saída(−) - uma variável de saída do tipo T na cabeça da cláusula deve aparecer como uma variável de saída do tipo T em qualquer literal do corpo da cláusula.
- Constante(#) - um argumento denotado por $\#T$ deve ser uma constante no literal.

Para gerar cada cláusula candidata a fazer parte da hipótese final $\mathcal{H}$, o Aleph começa a partir de um exemplo positivo não coberto e, constrói $\perp(e)$ e especializa uma cláusula $\mathcal{C}$, incluindo os literais da *Bottom Clause* no corpo de $\mathcal{C}$. Os literais são selecionados em $\perp e$ de acordo com a pontuação computada por alguma função de avaliação. O Aleph insere literais em $\mathcal{C}$ desde que haja alguma melhoria na pontuação, mas restringe a cobertura de exemplos negativos a um número máximo, definido pelo parâmetro *noise*. Portanto, se o parâmetro *noise* for 0, nenhum exemplo negativo poderá ser coberto pela cláusula.

Depois de chegar ao final desse processo de construção, a cláusula produzida é incluída na hipótese $\mathcal{H}$ e o processo é reiniciado. A configuração padrão do Aleph remove do conjunto de exemplos todos os positivos cobertos pela hipótese atual, mas continua considerando todos os exemplos negativos durante todo o processo. Isso acontece porque o objetivo é cobrir o máximo possível de exemplos positivos, cobrindo o mínimo de exemplos negativos possíveis. Assim, não é um problema produzir cláusulas cobrindo exemplos positivos já cobertos anteriormente, mas ele tenta evitar a prova de qualquer exemplo negativo.

A configuração do Aleph necessita de dois arquivos:

- *arquivo.b*, contendo a base de conhecimento
- *arquivo.f*, contendo os exemplos positivos

Um terceiro arquivo também pode ser fornecido, contendo os exemplos negativos. Ele não é obrigatório, pois o Aleph é capaz de aprender através dos exemplos positivos apenas. Esse arquivo é o *arquivo.n*. Idealmente, os arquivos devem conter o mesmo o nome, pois dessa forma são automaticamente reconhecidos pelo Aleph, não exigindo maiores configurações.

Na Base de conhecimento (*arquivo.b*), além das declarações de modo e dos tipos explicados acima, também é necessário declarar os *determinations*. Os *determinations*

declaram o predicado que pode ser usado para construir a hipótese [46] e tem a seguinte forma:

```
determination(Target/Aridade,Background/Aridade).
```

O primeiro argumento é o nome e a aridade do predicado definido como *target*. O segundo argumento é o nome e aridade do predicado que pode aparecer no corpo das cláusulas. No caso do presente trabalho, um exemplo de determination é:

```
determination(evasao/1,curso_aluno/2).
```

Neste caso, *evasao* é o *target* e tem aridade 1, enquanto *curso_aluno* é o predicado que pode aparecer no corpo das cláusulas e tem aridade 2.

2.3.1 Aplicação em Aprendizado de Máquina

O problema dos trens de *Michalski* pode ser utilizado para verificar como o Aleph pode ser utilizado para resolver problemas de aprendizado de máquina.

Esse problema consiste em um conjunto de dez trens: cinco deles viajam para oeste, os outros cinco para leste. Cada trem é formado por uma locomotiva que puxa vagões e, cada trem viaja para leste ou oeste dependendo de suas propriedades. O problema é ilustrado na figura 2.1.

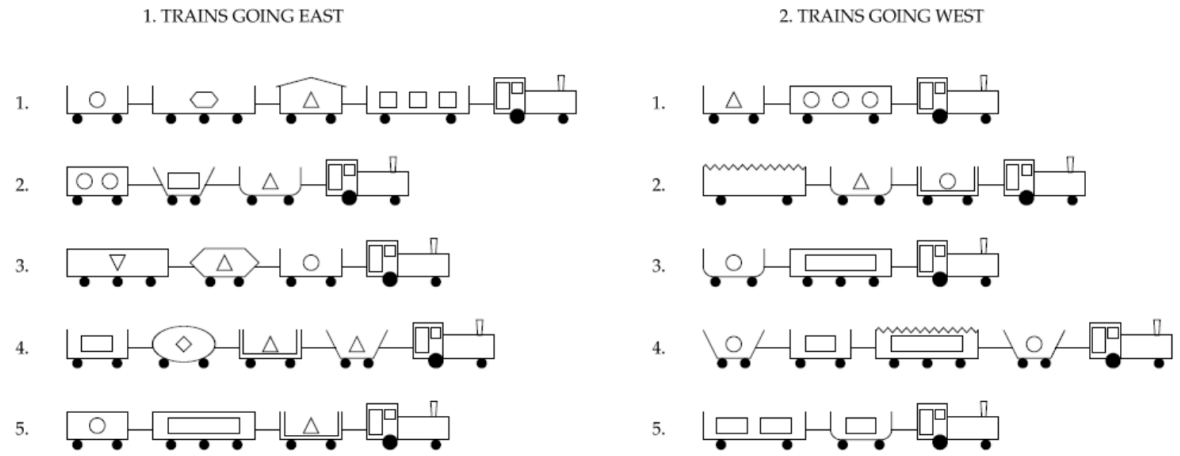

Figura 2.1: Conjunto de trens de Michalski

O que se deseja com esse problema é verificar se o modelo é capaz de, através de uma determinada configuração de trem e todas os outros exemplos conhecidos, descobrir se esse trem viaja para leste ou oeste. Como todas as possíveis configurações de trem são conhecidas pelo algoritmo, trata-se de aprendizado supervisionado.

O arquivo de Base de conhecimento é formado basicamente por um conjunto de parâmetros opcionais, a cláusula que é utilizada como *target (modeh)*, o corpo *modeb* e o *determination*, que determina qual predicado pode ser utilizado para construir a hipótese e os predicados em si.

No caso do problema acima, define-se:

```
:- modeh(1,eastbound(+train)).
:- modeb(1,short(+car)).
:- modeb(1,closed(+car)).
:- modeb(1,long(+car)).
:- modeb(1,open_car(+car)).
:- modeb(1,double(+car)).
:- modeb(1,jagged(+car)).
:- modeb(1,shape(+car,#shape)).
:- modeb(1,load(+car,#shape,#int)).
:- modeb(1,wheels(+car,#int)).
:- modeb(*,has_car(+train,-car)).
```

Onde se expressa os *modes*, os predicados usados e o tipo de seus respectivos argumentos (variáveis de entrada, saída ou constantes) e também a relação do predicado (um número inteiro ou '*').

```
:- determination(eastbound/1,short/1).
:- determination(eastbound/1,closed/1).
:- determination(eastbound/1,long/1).
:- determination(eastbound/1,open_car/1).
:- determination(eastbound/1,double/1).
:- determination(eastbound/1,jagged/1).
:- determination(eastbound/1,shape/2).
:- determination(eastbound/1,wheels/2).
:- determination(eastbound/1,has_car/2).
:- determination(eastbound/1,load/3).
```

Os fatos também são incluídos no arquivo de Base de conhecimento. Nele, predicados que indicam os carros, suas formas e os trens existentes são inseridos.

```
car(car_11). car(car_12). car(car_13). car(car_14).
car(car_21). car(car_22). car(car_23).
```

```
car(car_31). car(car_32). car(car_33).

shape(elipse). shape(hexagon). shape(rectangle). shape(u_shaped).
shape(triangle). shape(circle). shape(nil).

train(east1). train(east2). train(east3). train(east4). train(east5).
train(west6). train(west7). train(west8). train(west9). train(west10).
```

Após, exemplos específicos para cada tipo de trem ilustrado na Figura 2.1 são construídos em linguagem lógica. O primeiro trem que vai para leste, pode ser definido como:

```
long(car_11).
short(car_12).
long(car_13).
short(car_14).

open_car(car_11).
closed(car_12).
open_car(car_13).
open_car(car_14).

shape(car_11,rectangle).
shape(car_12,rectangle).
shape(car_13,rectangle).
shape(car_14,rectangle).

load(car_11,rectangle,3).
load(car_12,triangle,1).
load(car_13,hexagon,1).
load(car_14,circle,1).

wheels(car_11,2).
wheels(car_12,2).
wheels(car_13,3).
wheels(car_14,2).

has_car(east1,car_11).
has_car(east1,car_12).
```

```
has_car(east1,car_13).
has_car(east1,car_14).
```

Essa mesma lógica é aplicada a todos os outros 9 trens: todas as características dos trens são escritas em linguagem lógica de forma a ser possível extrair as informações de acordo com o identificador comum *car*. No exemplo acima, pode-se interpretar, por exemplo, que: o carro identificado como *car*_12 é fechado, pequeno, tem formato retangular, carrega um triângulo em seu vagão, possui duas rodas e vai para leste.

Além desses fatos, os arquivos que contém os exemplos positivos (obrigatórios) e/ou negativos também deve ser informado. Dessa forma, o arquivo *train.f*, que contém os trens que vão para leste (exemplos positivos), terá a forma:

```
eastbound(east1).
eastbound(east2).
eastbound(east3).
eastbound(east4).
eastbound(east5).
```

O arquivo com exemplos negativos, *train.n*, contém os predicados relacionados aos trens que vão para oeste.

```
eastbound(west6).
eastbound(west7).
eastbound(west8).
eastbound(west9).
eastbound(west10).
```

Após a entrada de todos os dados, pede-se ao Aleph que leia os arquivos através do comando *read_ all(train).*, responsável por carregar os exemplos e a Base de conhecimento, e então se executa o comando *induce.*, responsável por gerar a hipótese $\mathcal{H}$.

Como resultado, o Aleph mostrará, além de outras informações, as regras aprendidas e, o resultado do treinamento.

```
[theory]

[Rule 1] [Pos cover = 5 Neg cover = 0]
eastbound(A) :-
    has_car(A,B), short(B), closed(B).
```

```
[Training set performance]
          Actual
        +        −
     + 5        0        5
Pred
     − 0        5        5

       5        5       10

Accuracy = 1
```

Através do resultado pode-se perceber que o modelo foi capaz de aprender uma regra, que é expressa de forma lógica, sendo interpretada como: o trem vai para leste se contém um carro pequeno e fechado. De fato, após uma análise mais detalhada na Figura 2.1, pode-se verificar que absolutamente todos os trens com carros pequenos e fechados vão para leste. Para o exemplo dos trens, que contém apenas 10 exemplos, é possível identificar o padrão existente sem o uso de aprendizado de máquina. Porém, para problemas que possuem relações extremamente complexas e que englobam dezenas de milhares de dados, tal possibilidade se torna muito improvável.

Além da regra, uma Matriz de Confusão é gerada, mostrando que todos os exemplos que eram positivos foram preditos como positivos e todos os negativos foram preditos como negativos, resultando em uma acurácia de 100%. Como o Aleph sempre gera uma Matriz de de Confusão como esta, é possível calcular outras medidas de desempenho como *Precision*, *Recall*, e, como consequência, a *F-Measure*.

Capítulo 3

Trabalhos Relacionados

O número de evasões de alunos no ensino superior tem despertado a atenção de muitas pessoas, incluindo pesquisadores em educação e mineração de dados [40] devido as grandes consequências sócio-econômicas para o país. A possibilidade de melhorar a educação de um grupo maior de pessoas, e ao mesmo tempo reduzir custos financeiros ao prever os alunos com o risco de abandono, têm servido de motivação para a alocação de recursos de pesquisa com objetivo de minimizar esse problema.

Os estudos publicados até o momento em que esta dissertação foi escrita e que utilizaram algoritmos de aprendizado de máquina tradicionais têm maiores desafios em representar objetos heterogêneos e suas relações. Este trabalho, no entanto, se concentra em usar métodos que podem extrair informações diretamente dos bancos de dados relacionais, com intervenção mínima, e produzir um conjunto de dados lógico. Os trabalhos aqui citados foram desenvolvidos utilizando regras de aprendizado de máquina não relacionais dentro do mesmo contexto de predição de evasão de alunos do Ensino Superior e Médio. Os estudos foram realizados por universidades localizadas não só no Brasil, mas também no exterior.

Um estudo realizado pela Universidade Federal de Campina Grande (UFCG) [5], motiva o trabalho afirmando que apenas 62,4% dos alunos ingressantes no ensino superior concluem seus cursos e que, no caso da UFCG, muitas evasões ocorrem nos primeiros anos dos alunos nos cursos. Com base nessa motivação, o trabalho utiliza diversos algoritmos de aprendizado supervisionado com objetivo de comparar o desempenho destes. A base de dados utilizada continha dados de alunos de 130 cursos entre os anos de 2002 e 2014. Quatro diferentes algoritmos foram utilizados, são eles: *Naive Bayes* [26], C5.0, máquina de vetores de suporte (SVM), regressão logística e perceptron de múltiplas camadas. O trabalho mostra resultados para os cursos de Engenharia Elétrica, Enfermagem e Ciên-

cia da Computação separadamente e também agrupando todos os cursos em apenas um treinamento. O trabalho apresentou acurácias entre 89% e 95%, mas F-Measures entre 66% e 80%. De acordo com os autores e também com os resultados demonstrados, o desempenho se mostrou consistente considerando os diferentes classificadores testados.

O método apresentado em [7] realiza uma outra abordagem, já que aplica classificação com opção de rejeição após a inferência realizada por uma Rede Neural Artificial, para decidir sobre a classe dos alunos mal classificados. Essa decisão consiste em evitar a classificação de uma instância não vista caso a decisão não seja suficientemente confiável. Também neste trabalho foram consideradas apenas as evasões ocorridas no primeiro ano de curso do aluno. A estratégia de rejeição se mostrou benéfica para o modelo, dado que a curva Acurácia X Taxa de rejeição mostrou o aumento da Acurácia conforme a taxa de rejeição também aumentava. Os alunos rejeitados pelo algoritmo foram os alunos que apresentavam difícil predição, visto que seguiam no curso mas seu histórico acadêmico demonstrava um baixo desempenho, similar a de alunos com maior possibilidade de evadir. O trabalho, no entando, usou como grupo de teste apenas 32 alunos do curso de Ciência da Computação, não levando em consideração os outros cursos da Universidade. Dessa forma, não há como saber se o modelo criado é capaz de generalizar para outros cursos.

Remis et al [6] levaram em consideração 11.495 alunos de graduação da Universidade Católica de Brasília, uma instituição privada de ensino superior. Para realização dos experimentos, quatro classificadores foram utilizados: *Average One-Dependency Estimator*, J48 (uma implementação da árvore de decisão C4.5), redes neurais perceptron de múltiplas camadas e máquinas de comitê. Como forma de validação, 2/3 dos dados foram utilizados para treinamento e 1/3 para teste. Acurácia e curvas ROC (*receiver operating characteristic*) [19] foram utilizadas para demonstrar os resultados. O desempenho se mostrou consistente em relação aos diversos métodos testados, com acurácias entre 76,2% e 80,6%.

A *BGC Trust University Bangladesh*, propôs um trabalho com objetivos mais gerais, onde o objetivo é a predição de evasão para Universidades, institutos e colégios. Para isso, SVM, *Naive Bayes* e probabilidade condicional foram utilizados. Para a realização dos experimentos, 1.200 alunos foram considerados dentro de um universo de 30.000. Atributos como condições financeiras, idade e gênero foram utilizados, mostrando resultados percentuais que indicam a probabilidade de um aluno evadir dependendo desses atributos.

Dekker [16], por sua vez, realizou a pesquisa considerando alunos da *Eindhoven University of Technology*, na Holanda. O curso escolhido foi Engenharia Elétrica, motivado

pela sua taxa de evasão de cerca de 40%. Dados entre os anos de 2000 e 2009 foram coletados e 648 alunos foram escolhidos para fazer parte da base de dados com base no tipo de escola que estudaram antes de ingressar na universidade. Diversos algoritmos tradicionais foram utilizados, como: CART (SimpleCart), C4.5, BayesNet, SimpleLogistic, JRip e RandomForest. Foi obtida uma acurácia entre 71% e 79%. Considerando a F-Measure, o melhor resultado foi de 82%, obtido através da utilização do J48, parte do *CostSensitiveClassifier* [1] do Weka [2].

Outro trabalho também se concentrou em comparar o desempenho dos classificadores tradicionais de aprendizado de máquina, como o perceptron de múltiplas camadas, árvores de decisão, e *Naive Bayes*, na detecção precoce de evasão no ensino superior público ou privado [16, 17, 36, 6, 27]. A maioria deles também se concentra em estudantes calouros e induz variáveis agregadas de suas notas, como as que mencionamos até agora. Em [36], além do banco de dados da universidade, também é utilizada a informação dos dados oficiais do censo e dados demográficos. Neste presente trabalho, apenas o banco de dados oficial da Universidade Federal Fluminense é utilizado, visto que os dados a informações como o censo dos alunos não foi disponibilizado.

Em uma linha diferente, outros trabalhos se concentram em prever o abandono do ensino médio e o mau desempenho [43], seja em relação ao curso inteiro ou a uma ou mais disciplinas.

Por exemplo, em [25], em primeiro lugar, foram concebidas várias métricas da perspectiva do educador, com foco em alunos com risco de abandono ou reprovação. Em seguida, essas métricas foram usadas para verificar o comportamento dos algoritmos tradicionais de aprendizado de máquina em fazer previsões oportunas nesses casos, usando técnicas de importância de recursos. Embora o presente trabalho compartilhe os mesmos objetivos, como o uso de dados históricos, a obtenção de um modelo interpretável e a previsão de alunos em risco o mais cedo possível, não se pretende estudar especificamente os erros cometidos pelo classificador. Em vez disso, este trabalho se concentra em aprender um programa lógico que poderia ser usado para realizar a inferência a qualquer momento da vida acadêmica do aluno e apontar os que têm o potencial de abandonar seus cursos.

Em [51], o foco também foi nos alunos das escolas, com o objetivo de prever o risco de fraco desempenho em disciplinas como Ciências e Matemática. Eles usaram recursos de comportamento e demografia em algoritmos tradicionais de aprendizado de máquina, como a regressão logística, *Naive Bayes* e árvore de decisão. O trabalho apresentado em [29] também se concentrou nos alunos das escolas, mas eles prestaram atenção especial

ao manusear dados de alta dimensão e a um conjunto de dados desequilibrado. Eles combinaram programação genética e algoritmos de indução de regras propositivas para lidar com essas questões.

Embora o problema da evasão nos cursos de ensino médio e de graduação esteja relacionado, as causas, explicações, custos decorrentes do abandono e formas de evasão podem ser bem diferentes. Por exemplo, em várias escolas secundárias brasileiras, se um aluno é reprovado em uma disciplina, ele precisa participar de todas as aulas daquele ano novamente. Este não é o caso dos cursos de graduação, o que pode implicar em diferentes custos e medidas tomadas para aliviar o problema.

Após analisar os trabalhos anteriormente citados, pode-se constatar que, embora muitos destes tenham objetivos semelhantes ao trabalho aqui proposto, onde o interesse é prever a evasão de um aluno através de dados históricos, o método utilizado pela maioria deles difere em diversas formas.

Em primeiro lugar, o presente trabalho não se restringe aos dados de alunos em um determinado semestre ou ano. Em vez disso, o objetivo é prever um abandono em qualquer momento do curso. Embora a previsão no início do curso seja benéfica, no sentido em que há mais tempo para se aplicar estratégias educacionais ou até mesmo de apoio financeiro para resolver o problema, os alunos no meio ou até mesmo no final do curso significam um gasto acumulado para as finanças públicas muito maior. Desta forma, a capacidade de prever a evasão de um aluno em qualquer momento de seu curso pode ter um maior impacto nas finanças públicas, além de viabilizar a intervenção em qualquer momento da vida acadêmica do aluno.

Em segundo lugar, diversos trabalhos anteriores são obrigados a induzir novas variáveis em diversas ocasiões, ao agregar os resultados de diferentes disciplinas, ou a extrair novas características de uma informação não contida no banco de dados, como as disciplinas mais difíceis. Tais agregações podem ser necessárias nestes trabalhos porque cursos diferentes têm disciplinas diferentes, ou seja, os dados são distribuídos heterogeneamente. Dessa forma, não é possível criar um atributo para cada aluno, pois diferentes exemplos teriam um número diferente de atributos, gerando dados de tamanho variável. O aprendizado relacional empregado neste presente trabalho não tem esse problema, pois pressupõe que os dados não são distribuídos de forma homogênea, possibilitando o uso de diferentes fatos lógicos para as informações relacionadas a cada aluno.

Em terceiro lugar, quando um novo curso é adicionado no grupo de treinamento, é necessário pensar novamente sobre a nova informação adicionada, como por exemplo na

nova agregação que deve ser realizada. Com a técnica utilizada no presente trabalho (aprendizado relacional), é possível criar regras gerais na Base de conhecimento (BK) que serão consideradas como parte dos dados apenas quando o processo de aprendizado as julgar úteis. Dessa forma, não há necessidade de pensar em possíveis agregações para cada curso que se deseja adicionar ao treinamento.

Capítulo 4

PRELUDE: um método para predição de evasão de alunos de graduação através de aprendizado relacional

Foi proposto neste trabalho um sistema de predição usando aprendizado de máquina para estudantes de graduação (PRELUDE, do inglês *dropout PRediction using machinE Learning for UnDErgraduate courses*) composto por métodos de extração de dados de bancos relacionais e transformação em predicados Prolog, que são utilizados pelo Aleph como a base de conhecimento (BK) e exemplos. A arquitetura desse sistema, bem como a explicação mais detalhada de seus componentes é discutida neste capítulo.

4.1 Arquitetura da Solução

Para se obter de forma automática as regras que descrevem o comportamento dos alunos, é necessário um processo para extrair informações relevantes sobre eles do banco de dados, e, formatá-los de uma forma que o compilador Prolog possa interpretá-los e o sistema Aleph possa treinar os conjuntos de dados. O PRELUDE é um sistema composto por 5 principais componentes: (1) Banco de dados relacional, (2) *Extract Transform Load* (ETL) (3) Base de conhecimento (*Knowledge Base*), (4) Gerador de base de conhecimento (BK Generator) e (5) o Aleph. A Figura 4.1 mostra a arquitetura conceitual do processo e suas etapas.

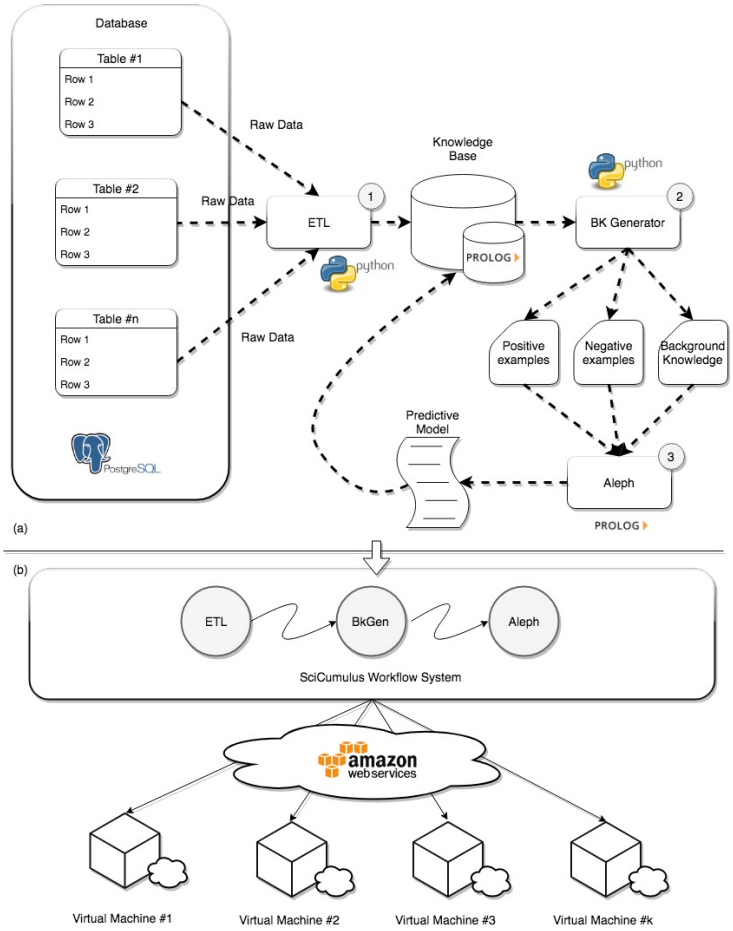

Figura 4.1: Arquitetura do PRELUDE

4.1.1 Explicação dos componentes

Como os dados dos alunos e professores estão gravados em um banco de dados relacional, é necessário extrair as informações do mesmo. Esse banco de dados contém tabelas que armazenam informações sobre estudantes, notas, professores, turmas, etc. O banco de dados contém diversas relações $R = \{r_1, r_2, ..., r_k\}$. Cada relação r_k tem um esquema $\Re$, com seus atributos tipados associados. Consideremos como um exemplo a relação r_1 que associa um aluno com seu curso como apresentado na Tabela 4.1. Dessa forma, o esquema para r_1 pode ser declarado como $\Re_1$= (Matricula: Integer, Curso: Integer), o qual pode ser especificado como $r_1(\Re_1)$.

O componente ETL segue o procedimento *Extract-Transform-Load* comumente utilizado em Data Warehouses [22]: (i) A etapa de *extração* é responsável por recuperar os dados de origem; (ii) A etapa de *Transformação* é responsável por converter os dados para um formato adequado; e (iii) A etapa de Carregamento é responsável por carregar os dados processados no banco de dados de destino (ou base de conhecimento no contexto

Tabela 4.1: Estudantes e seus cursos

Matricula	Curso
731083	31
831094	31
041085	41

deste trabalho). Dessa forma, esse componente ETL extrai dados relevantes do banco de dados de origem para gerar fatos em formato Prolog que serão usados para gerar os modelos preditivos.

O primeiro passo para se realizar o ETL é identificar quais são as informações relevantes a serem extraídas do banco de dados. No caso deste presente trabalho, essa identificação leva em consideração quais são as informações que podem estar associadas ao abandono dos alunos. Durante esta primeira etapa, uma série de consultas são executadas no banco de dados. Considerando que no PRELUDE tem-se um banco de dados relacional, as relações podem ser manipuladas por um conjunto de operadores: união ($\cup$), interseção ($\cap$) e diferença (-), desde que seu esquema $\Re_i$ e $\Re_j$ sejam compatíveis. Por exemplo, considerando $R(\Re)$ e $S(\Re)$ então $(R \cup S)$, $(R \cap S)$ e $(R - S)$ são todas expressões válidas. Quando R e S apresentam esquemas diferentes, a expressão $(R \bowtie S)$ também é válida.

Dessa forma, a componente ETL pode consultar o banco de dados usando linguagens de consulta, como o SQL tradicional. Vale a pena notar que o usuário deve conhecer o esquema do banco de dados, isto é, o usuário deve saber quais consultas executar para extrair as informações importantes. Como alguns dos dados recuperados precisam de limpeza, o componente ETL também remove dados ruidosos, incompletos e inválidos e os transforma em informações mais úteis. Um exemplo de transformação de dados é a conversão da data de nascimento para a idade dos alunos e professores e classificá-las dentro de faixas etárias, como os exibidos na Tabela 4.2. Além desta transformação, o nome de usuário de redes sociais dos alunos também foram convertidos em binários, indicando apenas se o aluno se diz pertencer a uma determinada rede social ou não.

Quando todos os dados são processados, a componente de ETL precisa colocar os dados no formato que o sistema Aleph pode consumir para gerar o modelo preditivo. Dessa forma, a componente de ETL converte dados brutos recuperados do banco de dados em fatos do Prolog. Um exemplo de dados consultados é apresentado na Tabela 4.1. As 3 tuplas apresentadas na Tabela 4.1 são transformadas nos fatos do Prolog apresentados

Tabela 4.2: Classificação etária

Classe	Faixa etária (anos)
A	15-19
B	20-24
C	25-29
D	30-34
E	35-39
F	40-44
G	45-49
H	50-54
I	55-59
J	60-64
K	65-69
L	70-74
M	75-79
N	80-84

a seguir como *aluno_curso*/2:

```
aluno_curso(731083,31).
aluno_curso(831094,31).
aluno_curso(041085,41).
```

No caso dos fatos acima mencionados, *aluno_curso* é o nome do predicado, 731083 é matrícula do estudante e 31 é o seu ID do curso (curso de Ciência da Computação).

Outro exemplo de fato em Prolog é aquele que associa um aluno específico ao seu CR (Coeficiente de Rendimento). Tais informações podem ser vinculadas, pois compartilham um identificador comum (matrícula), similarmente às chaves primárias e estrangeiras em bancos de dados relacionais. Dessa forma, as informações sobre a pontuação geral de um aluno podem ser vinculadas ao seu curso por meio da matrícula aluno, como os seguintes fatos:

```
cr_aluno(731083,7.96).
cr_aluno(831094,5.34).
cr_aluno(041085,8.79).
```

Usando as informações apresentadas em fatos relacionados a *curso_aluno* e *cr_aluno*, o Prolog poderia inferir que o aluno identificado por 731083 tem um CR igual a 7.96 e pertence ao curso com ID 31. A conversão dos dados brutos extraídos do banco de dados para os fatos do Prolog são executados para todos os dados recuperados pelas consultas.

A componente de ETL, em seguida, armazena todos os fatos gerados em uma base de conhecimento.

Quando esse processo é concluído, o componente *BK Generator* é chamado, onde os fatos gerados são divididos em 3 arquivos a serem consumidos pelo Aleph: arquivo de exemplos positivos, arquivo de exemplos negativos e base de conhecimento. Neste trabalho, exemplos positivos são fatos referentes a alunos que abandonaram seus cursos, enquanto exemplos negativos são alunos que não evadiram. O BK contém todos os outros fatos produzidos pelo componente ETL e também pode conter regras definidas pelos especialistas do domínio. Esses arquivos são então passados como parâmetro para o sistema Aleph que gera o modelo preditivo. Como Aleph retorna um conjunto de cláusulas Prolog como modelo preditivo, ele também pode ser armazenado na base de conhecimento. A fim de promover a reprodutibilidade, um *workflow* foi modelado para automatizar todo o processo do PRELUDE. Este *workflow* pode ser invocado informando todos os cursos que serão treinados juntos como um parâmetro e todas as etapas descritas na Figura 4.1 são executadas automaticamente. A forma de invocação do *workflow*, ilustrado na etapa (b) do processo, é independente da etapa (a), onde os arquivos finais usados para treinamento são gerados. Dessa forma, qualquer sistema que seja capaz de processar dados de forma sistemática pode ser utilizado para invocar o *script* responsável pela geração dos arquivos de treinamento e testes. Um exemplo de sistema que pode ser utilizado é o SciCumulus [13], que torna simples a exploração de parâmetros em paralelo na nuvem de forma otimizada.

No caso deste trabalho, cada etapa do processo é executada através do uso de *scripts* programados nas linguagens Ruby, Python e Shell Script. A extração, transformação e carregamento dos dados (ETL) é feita através da execução de consultas ao banco de dados por um *script* Ruby, que é responsável por executar estas consultas e as retornar diretamente em formato de lógica de primeira ordem. A transformação é executada nos predicados gerados (como a idade do aluno e do professor, exemplificados anteriormente). A geração do BK também é realizada por este mesmo *script* em outra etapa. Após a geração dos exemplos, a separação destes em *folds* para utilização de validação cruzada é invocada por um *script* em Python. Por último, um *script* shell invoca o Aleph k vezes, uma para cada fold (onde o valor de k é igual a 10 no caso deste trabalho, mas pode ser alterado por se tratar de uma configuração), e retorna os resultados para cada *fold* separadamente, para que os resultados possam ser agregados e assim as métricas calculadas: *Accuracy*, *Precision*, *Recall* e *F-Measure*, todas as medidas com suas médias e desvio padrões.

O script responsável pela etapa ETL é invocado através do seguinte comando:

```
ruby run.rb 8,9,17,18,44,48
```

Os parâmetros passados ao *script* são os IDs dos cursos em que se deseja agregar para o treinamento. É importante notar que as consultas a serem executadas no banco de dados em questão são codificadas em arquivos de configuração. No caso da necessidade da extração de novos atributos, novas entradas devem ser adicionadas a classe responsável pela definição das consultas no programa escrito em Ruby.

Os *scripts*, por sua, vez, são executados dentro de *containers Docker* [3], tornando possível a fácil distribuição dos mesmos e execução por diferentes usuários. Além disso, todas as configurações necessárias para execução estão contidas no *container*. Dessa forma, pacotes necessários para conexão com o Oracle (banco de dados usado neste trabalho), rodam dentro do container, diminuindo possíveis problemas de compatibilidade de ambiente. Os resultados são armazenados no computador que atua como *host* dos *containers*, para que possam então ser analisados.

Todo esse fluxo de trabalho a ser executado pode ser modelado em um sistema de Workflow (Figura 4.1 (b)) permitindo que os usuários explorem o conjunto de dados de muitas maneiras diferentes, considerando várias experimentações em relação aos cursos que poderiam ser agrupados. O fluxo de trabalho permite agrupar cursos considerando o campo do conhecimento (Ciências Humanas, Matemática e Ciências, Ciência da Computação, Biologia, Química, etc) e verificar se o comportamento de abandono é o mesmo em diferentes níveis de granularidade. No próximo capítulo, mostra-se a abordagem proposta para a utilização dos dados da Universidade Federal Fluminense.

Capítulo 5

Configuração Experimental

Neste capítulo, os experimentos realizados no trabalho são apresentados e discutidos, de forma a responder as questões propostas no Capítulo 1. A configuração experimental é descrita, incluindo todos os parâmetros que foram modificados no Aleph, as estratégias para validação dos dados são discutidas, assim como os dados que foram escolhidos para serem utilizados como conjunto de testes e validação da solução. Os atributos que foram utilizados também são mostrados, sendo estes representados em forma de predicados. Os experimentos que foram realizados são descritos neste capítulo. Os resultados, por sua vez, fazem parte do Capítulo 6.

5.1 Parâmetros utilizados

O aprendizado baseado em lógica geralmente gera grandes espaços de busca. Assim, para viabilizar o processo de aprendizagem, os sistemas ILP, como Aleph, contam com *bias* de linguagem, para restringir o espaço de busca.

No Aleph, o *bias* da linguagem inclui as declarações de *modes* [47], que devem ser definidas de acordo com o domínio em questão. O *bias* do espaço de busca inclui vários parâmetros para restringir o espaço de pesquisa, definir o algoritmo de aprendizado e definir os parâmetros de otimização. Pode-se usar também os valores padrões para todos esses parâmetros, caso não seja necessário alterar nenhum deles. No entanto, para explorar melhor o espaço de busca, pode ser necessário alterar os valores padrões de alguns parâmetros, para que seja possível encontrar cláusulas de maior qualidade.

Enquanto o *bias* da linguagem é definido de acordo com o domínio, o *bias* do espaço de busca pode ser definido seguindo métodos empíricos. Neste trabalho, grande parte dos

parâmetros Aleph foram utilizados com seus valores padrões, porém quatro deles foram alterados empiricamente, da seguinte maneira:

- o parâmetro *nodes* foi aumentado de seu valor padrão de 5000 para 10000. Este parâmetro é responsável por restringir o número de cláusulas candidatas produzidas em cada iteração do algoritmo de geração de cláusulas; o aumento no valor de *nodes* permite explorar mais cláusulas, o que pode justificar o uso de ambientes de computação em núvem, por contarem com a oferta de grande poder computacional que pode ser utilizado em um curto período de tempo.
- o parâmetro *clauselength* restringe o número máximo de literais em uma cláusula. O valor foi aumentado do seu valor padrão de 4 para 10;
- o parâmetro *noise* define quantos exemplos negativos são aceitáveis para serem cobertos por uma cláusula. Permitir que mais exemplos negativos sejam cobertos tem o efeito de permitir cláusulas menos específicas, o que também pode aumentar o número de exemplos positivos cobertos. Isto é particularmente importante para conjuntos de dados com algum ruído. O valor desse parâmetro foi alterado de seu valor padrão de 0 para 10;
- o parâmetro *minpos* especifica o número mínimo requerido de exemplos positivos cobertos por uma cláusula. Para permitir uma melhor generalização, o valor foi aumentado para 2, quando o valor padrão é 1. Desta forma, não se permite ter uma cláusula que cubra apenas um exemplo positivo, pois tal cláusula poderia ser o próprio exemplo.

O aumento dos valores usados como padrão pelo Aleph também aumenta o espaço de busca. Dessa forma, essa decisão tem como consequência a possibilidade de obter teorias finais que melhor representam o domínio. Por outro lado, o processo como um todo se torna mais dispendioso, podendo ser necessária a utilização de grande poder computacional.

5.2 Validação

Para melhor avaliar o modelo preditivo, a técnica de validação cruzada *k-fold* [24] foi usada para separar o conjunto de treinamento e validação em 10 conjuntos de dados para cada conjunto de dados. Esse procedimento permite que todos os exemplos sejam

considerados como um exemplo de treinamento ou um exemplo de teste em diferentes iterações. Para tanto, o conjunto de predicados é dividido aleatoriamente em k conjuntos disjuntos, chamados *folds* (conjunto de dados), com aproximadamente o mesmo tamanho. Então, na primeira iteração, o *fold* $\#1$ é definido como teste, enquanto os *folds* $\#2 \ldots 10$ são unidos e usados como conjunto de treinamento. Na segunda iteração, o *fold* $\#2$ é usado como conjunto de testes, enquanto os conjuntos $\#1, 3 \ldots 10$ são unidos e definidos como conjunto de treinamento e assim por diante. Este procedimento é repetido k vezes, para que cada um dos *folds* seja usado como teste em uma das iterações. Para cada iteração, medidas como *Precision*, *Recall*, *Accuracy* e *F-Measure* foram computados sobre o conjunto de testes, usando a teoria aprendida no conjunto de treinamento correspondente para avaliar o desempenho do modelo. Após todas as iterações, o valor médio e o desvio padrão foram calculados para cada métrica de desempenho.

5.3 Dados

A universidade oferece mais de 100 cursos de graduação, o que inviabiliza a produção e análise de resultados de cada um deles devido ao custo financeiro envolvido relativo à grande quantidade de recursos computacionais necessários para processamento. Assim, alguns cursos foram escolhidos para fazer parte do grupo de treinamento com base na sua taxa de evasão. Esta decisão faz sentido para o propósito deste trabalho, uma vez que o objetivo é reduzir a taxa de desistência, levando a uma economia financeira por parte do governo. Além disso, reduzir a taxa de evasão significa que uma maior quantidade de mão de obra qualificada estará apta a atuar no mercado de trabalho após concluírem seus cursos. Como consequência, essas pessoas de alguma forma devolverão o investimento que foi feito nelas, pelo governo, através da geração de tecnologia, passagem de conhecimento, pesquisa em diversas áreas, pagamento de impostos diretos e indiretos, através do aumento da capacidade de consumo, etc. No entanto, para analisar a sensibilidade do modelo em relação à taxa de evasão, alguns cursos com média e baixa taxa de evasão também foram escolhidos.

Assim, pode-se ver os cursos experimentados aqui como pertencentes a três tipos de cursos de graduação. O primeiro grupo contém os cursos com algumas das maiores taxas de evasão na Universidade (representando mais recursos governamentais gastos possivelmente sem retorno). Eles também são úteis para o processo de treinamento devido ao número de alunos que de fato desistiram. Estes cursos são: Física, Matemática, Química, Estatística, Ciência da Computação e Engenharia de Telecomunicações. O segundo grupo

contém cursos com taxas médias de abandono, de 30% a 60%. Estes cursos são Farmácia, Engenharia de Produção, Engenharia Química e Engenharia Civil. O terceiro grupo contém dois cursos com baixas taxas de abandono, abaixo de 30%. São eles: Nutrição e Medicina. A taxa de evasão desses cursos é listada na Tabela 5.1.

Tabela 5.1: Taxas de Evasão por curso

Cursos	Taxa de Evasão
Física	87,81%
Matemática	87,61%
Química	82,55%
Estatística	71,42%
Ciência da Computação	62,38%
Engenharia de Telecomunicações	60,12%
Farmácia	43,50%
Engenharia de Produção	39,40%
Engenharia Química	39,38%
Engenharia Civil	32,62%
Nutrição	21,91%
Medicina	7,23%

Para cada curso, apenas os alunos que ingressaram na Universidade nos anos de 2009 e 2010 foram considerados. Esses anos foram escolhidos principalmente porque, depois de 2009, a universidade começou a inserir maiores informações no banco de dados que foram consideradas relevantes no que diz respeito à vida acadêmica dos alunos. Além disso, no momento de conclusão deste trabalho (2018), a maioria dos alunos ingressantes em 2009 e 2010 já teriam concluído seus cursos. No caso do prazo máximo de conclusão dos cursos, há uma regra na UFF que determina que os alunos têm o direito de concluir seus estudos em até 50% mais tempo do que o tempo padrão do curso sem terem suas matrículas automaticamente canceladas. Dessa forma, se um curso tem 10 semestres, o aluno tem até 15 semestres para terminar os estudos sem ser desligado automaticamente do programa de graduação. Como o tempo esperado varia entre 8 e 12 semestres para os cursos listados na tabela 5.1, considerar os alunos matriculados em 2009 e 2010 resulta em uma boa chance de ter a maioria deles marcados como concluintes ou não concluintes. Os alunos que abandonaram seus cursos são marcados como alunos que evadiram (exemplos positivos), enquanto todos os outros estão marcados como alunos que não evadiram.

A Tabela 5.2 mostra o número de exemplos relacionados a cada curso, considerando tanto os alunos positivos (estudantes evadindo seus cursos), quanto os negativos (alunos que terminaram seus cursos no tempo).

Tabela 5.2: Número de exemplos por curso, considerando os anos de 2009 e 2010.

Cursos	Exemplos Positivos	Exemplos Negativos	Total
Física	209	29	238
Matemática	396	56	452
Química	123	26	149
Estatística	85	34	119
Ciência da Computação	141	85	226
Engenharia de Telecomunicações	95	63	158
Farmácia	77	100	177
Engenharia de Produção	80	123	203
Engenharia Química	89	137	226
Engenharia Civil	61	126	187
Nutrição	32	114	146
Medicina	24	308	332

5.4 Predicados

A Tabela 5.3 mostra os predicados que foram escolhidos para serem usados como *Background Knowledge*:

Com base nos predicados mostrados na Tabela 5.3, é possível gerar fatos que representam o histórico acadêmico de um aluno, como os seguintes:

```
ano_semestre_ingresso(909044,2009,1).
cidade_aluno(909044,'Nova Friburgo').
classe_idade_aluno(909044, 'D').
cr_aluno(909044,4.78).
curso_aluno(909044,9).
estado_civil(909044,'Solteiro(a)').
genero_aluno(909044,'M').
numero_de_trancamentos(909044,3).
```

Tabela 5.3: Predicados usados como Background Knowledge

Predicado	Argumentos
ano_semestre_ingresso	*matricula, ano, semestre*
cidade_aluno	*matricula, cidade*
classe_idade_aluno	*matricula, classe_idade*
cr_aluno	*matricula, cr*
curso_aluno	*matricula, curso*
estado_civil_aluno	*matricula, estado_civil*
genero	*matricula, genero*
quantidade_trancamentos_aluno	*matricula, qtd*
resultado_aluno_turma	*matricula, resultado, disciplina, turma, ano, semestre*
professor_turma	*matricula, turma, disciplina, anosemestre*
tem_facebook	*matricula, boolean*
tem_twitter	*matricula, boolean*

```
resultado_aluno_turma(909044,aprovado,1111,'K1',2009,2).
resultado_aluno_turma(909044,aprovado,1093,'K2',2010,1).
resultado_aluno_turma(909044,reprovado,1102,'K4',2010,1).
professor_turma(303649,'K1',1111,20092).
professor_turma(303675,'K2',1093,20101).
professor_turma(303683,'K4',1102,20101).
tem_facebook(909044,1).
tem_twitter(909044,0).
```

Os predicados acima conseguem expressar com riqueza diversos detalhes da vida acadêmica do aluno de matrícula 909044. Pode-se verificar, por exemplo, que o aluno é Solteiro, de sexo Masculino, pertence ao curso de código 9 (Nutrição), trancou sua matrícula 3 vezes, foi aprovado em duas disciplinas, ministradas pelo professor de matrícula 303649 e 303675, respectivamente, e foi reprovado na disciplina ministrada pelo professor de matrícula 303683. Além disso, o aluno informou fazer parte da rede social *Facebook*, ao mesmo tempo em que não informa se possui *twitter*.

5.4.1 Base de conhecimento

Os predicados descritos anteriormente foram utilizados no arquivo de *Background Knowledge*, assim como as declarações de modo e os *determinations*.

As configurações utilizadas neste trabalho são mostradas a seguir:

```
:- modeh(1,evasao(+matricula)).

:- modeb(1,curso_aluno(+matricula,-curso)).
:- modeb(1,curso_aluno(+matricula,#curso)).
:- modeb(1,cr_aluno(+matricula,-cr)).
:- modeb(1,cor_aluno(+matricula,-cor)).
:- modeb(1,cor_aluno(+matricula,#cor)).
:- modeb(1,genero_aluno(+matricula,-genero)).
:- modeb(1,genero_aluno(+matricula,#genero)).
:- modeb(1,estado_civil(+matricula,-estadocivil)).
:- modeb(1,estado_civil(+matricula,#estadocivil)).
:- modeb(1,ano_semestre_ingresso(+matricula,-ano,-semestre)).
:- modeb(1,professor_turma(-siape,+codturma,+iddisciplina ,
-anosemestre)).
:- modeb(*,resultado_aluno_turma(+matricula,#resultado ,
-iddisciplina,-codturma,-ano, -semestre)).
:- modeb(1,qtd_semestres_trancados(+matricula,-qtdsemtrancados)).
:- modeb(1,qtd_semestres_trancados(+matricula,#qtdsemtrancados)).
:- modeb(1,tem_facebook(+matricula,#facebook)).
:- modeb(1,tem_twitter(+matricula,#twitter)).
:- modeb(1,cidade(+matricula,#cidade)).
:- modeb(1,classe_idade_aluno(+matricula,-classe)).
:- modeb(1,classe_idade_aluno(+matricula,#classe)).
:- modeb(1,classe_idade_professor(-siape,-classe)).
:- modeb(1,classe_idade_professor(-siape,#classe)).

:- determination(evasao/1,curso_aluno/2).
:- determination(evasao/1,cr_aluno/2).
:- determination(evasao/1,cor_aluno/2).
:- determination(evasao/1,genero_aluno/2).
:- determination(evasao/1,estado_civil/2).
:- determination(evasao/1,ano_semestre_ingresso/3).
:- determination(evasao/1,professor_turma/4).
:- determination(evasao/1,resultado_aluno_turma/6).
```

```
:- determination(evasao/1,qtd_semestres_trancados/2).
:- determination(evasao/1,tem_facebook/2).
:- determination(evasao/1,tem_twitter/2).
:- determination(evasao/1,cidade/2).
:- determination(evasao/1,classe_idade_aluno/2).
:- determination(evasao/1,classe_idade_professor/2).
```

Cada predicado foi salvo em um arquivo diferente por questões de organização. Sendo assim, o carregamento deles é feito da seguinte forma:

```
:- [cursoAluno,crAluno,corAluno,generoAluno,...].
```

Nesse caso, *cursoAluno*, *crAluno*, *corAluno* e *generoAluno* são arquivos prolog (extensão *.pl*).

5.4.2 Experimentos

Os experimentos realizados tem como objetivo responder as questões colocadas no Capítulo 1. Para a **Q#1**, onde deseja-se saber se é possível obter regras aprendidas através do aprendizado relacional para prever evasões, dois conjuntos foram criados: (1) seguindo um procedimento de validação cruzada para cada curso separadamente, para calcular as métricas de desempenho e (2) combinando diferentes cursos no mesmo grupo de treinamento, também usando validação cruzada com 10 *folds*.

Para responder a questão **Q#2**, se seria melhor treinar o modelo contemplando todos os cursos, ainda mantendo a relação entre o aluno e seu curso, os cursos foram treinados em três diferentes grupos. O Grupo I contém os cursos com a maior taxa de evasão, o Grupo II contém os cursos com taxa de evasão média, enquanto o Grupo III contém os cursos com a menor taxa de evasão.

Para responder a questão **Q#3**, as regras aprendidas por um curso foram utilizadas em cursos que não participaram do treinamento.

Com relação à questão **Q#4**, as regras aprendidas através do treinamento dos cursos considerando alunos ingressantes em 2009 e 2010 foram aplicadas em alunos ingressantes em 2011. Nenhum dos alunos ingresssantes em 2011 fez parte do treinamento. Dessa forma é possível validar o modelo em um caso bem próximo do real, onde as regras aprendidas pelo passado seriam utilizadas para classificar novos estudantes.

A questão **Q#5**, que verifica se o modelo é compreensível, é respondida através da

análise qualitativa das regras geradas pelo Aleph.

As questões acima serão respondidas e analisadas no próximo capítulo.

Capítulo 6

Resultados

Esse capítulo tem como objetivo mostrar todos os resultados gerados pelo trabalho através da utilização do PRELUDE para predição de evasão de alunos. Diversas abordagens foram adotadas para que seja possível verificar o desempenho do modelo diante de diversos cenários e grupos de treinamento diferentes.

Primeiro, os cursos foram treinados separadamente e os testes foram realizados através da validação cruzada com 10 *folds*, independente dos grupos aos quais eles pertencem. Essa abordagem permite verificar como o desempenho varia de acordo com a exposição dos próprios exemplos positivos e negativos de cada curso.

Em segundo lugar, os cursos pertencentes a cada grupo são de fato agrupados no mesmo treinamento. Como o curso do aluno faz parte dos predicados, é possível agrupar todos os alunos juntos em um grande conjunto de dados sem que seja necessário se preocupar com agrupamentos de diferentes atributos (ex: diferentes cursos têm diferentes disciplinas, o que pode levar a uma quantidade enorme de atributos) e o modelo pode aprender regras específicas a determinados cursos, se beneficiando do maior conjunto de exemplos. Essa estratégia permite também verificar se os cursos que possuem média e baixa taxa de evasão podem se beneficiar quando o número de exemplos positivos e negativos aumentam.

Após, alguns cursos são combinados entre si de uma forma mais aleatória para verificar se existem determinados cursos que, mesmo com diferentes taxas de evasão, quando agrupados podem resultar em melhores resultados. É possível também verificar se o resultado é benéfico para todos os cursos, apenas alguns deles ou para nenhum.

Os cursos também são agrupados dentro de sua área de pesquisa, visto que com os cursos escolhidos podem se formar duas grandes áreas: Ciências Exatas e Ciências

Biomédicas.

Também é importante verificar a capacidade de generalização das regras, pois deseja-se saber se regras aprendidas por um curso podem ser utilizadas para prever evasão de outros cursos que não participaram do treinamento. Caso positivo, um curso poderia se beneficiar do modelo, mesmo que seu número de exemplos positivos não seja muito grande.

Por último, a capacidade de predição é testada através da exposição das regras geradas pelos treinamentos baseados nos anos de 2009 e 2010 aos dados dos alunos ingressantes em 2011. Para verificar qual das abordagens acima alcança melhor desempenho, esse teste é realizado com os cursos separadamente, em grupos divididos por taxa de evasão e por áreas de pesquisa. Dessa forma, é possível simular um exemplo de uso bastante real, onde se usa um treinamento do passado para prever alunos que ingressaram após, não fazendo parte do conjunto de treinamento.

Os resultados para todas essas abordagens são dispostos a seguir na forma de Tabelas e gráficos para realizar as comparações.

6.1 Treinamento com cursos individualmente

Os resultados considerando os cursos treinados separadamente são exibidos nas tabelas 6.1, 6.2, e 6.3, de acordo com os grupos em que eles se adequam, levando em consideração suas taxas de evasão.

Tabela 6.1: Resultados preditivos da validação cruzada para o Grupo I (alta taxa de evasão), com cursos treinados separadamente.

Cursos	Precision	Recall	Accuracy	F-Measure
Matemática	88,66% ± 3,46%	96,77% ± 3,13%	86,59% ± 5,02%	92,50% ± 2,85%
Física	90,36% ± 6,53%	94,16% ± 3,81%	85,33% ± 6,25%	92,01% ± 3,77%
Química	83,84% ± 7,07%	96,83% ± 3,91%	81,79% ± 8,18%	89,72% ± 5,28%
Estatística	74,15% ± 10,42%	83,46% ± 5,60%	76,13% ± 10,13%	78,13% ± 6,24%
Eng. Telecomunicações	70,78% ± 8,75%	83,06% ± 12,74%	70,71% ± 10,08%	75,96% ± 8,77%
Ciência da Computação	68,68% ± 7,74%	73,76% ± 9,04%	62,43% ± 9,72%	70,99% ± 7,58%

Tabela 6.2: Resultados da validação cruzada para o Grupo II (taxa média de evasão), com cursos treinados separadamente.

Cursos	Precision	Recall	Accuracy	F-Measure
Farmácia	63,99% ± 17,83%	66,43% ± 17,51%	68,43% ± 12,90%	64,12% ± 15,25%
Eng. Produção	56,45% ± 14,81%	33,75% ± 9,76%	62,96% ± 7,77%	41,72% ± 10,88%
Eng. Civil	47,88% ± 19,91%	35,37% ± 16,76%	61,86% ± 7,09%	39,90% ± 16,94%
Eng. Química	28,33% ± 19,79%	16,19% ± 14,44%	61,40% ± 5,04%	19,74% ± 15,10%

Tabela 6.3: Resultados da validação cruzada para o Grupo III (baixa taxa de evasão), com cursos treinados separadamente.

Cursos	Precision	Recall	Accuracy	F-Measure
Nutrição	15,00% ± 32,02%	6,67% ± 13,33%	75,07% ± 7,22%	9,00% ± 18,14%
Medicina	0,00% ± 0,00%	0,00% ± 0,00%	92,40% ± 3,15%	0,00% ± 0,00%

Como pode-se ver dos resultados exibidos nas tabelas 6.1, 6.2 e 6.3, o desempenho segue a taxa de evasão, o que significa que este melhora conforme o número de exemplos na classe dominante aumenta. Apesar dos resultados não apresentarem resultados satisfatórios para todos os cursos, estes satisfazem o objetivo principal deste trabalho: predizer evasão de alunos em cursos com alta taxa de evasão, que como consequência representam maior desperdício de recurso investido na universidade.

Especificamente para o curso de Medicina, as métricas *Precision* e *Recall* receberam o valor de 0% pois não houveram verdadeiros positivos. Para esse caso, nenhuma regra foi aplicável aos estudantes do curso.

Assim, a partir dos resultados acima, pode-se responder a **Q#1**, afirmando que na presença de um conjunto de dados de boa qualidade, os modelos aprendidos têm mais de 70% de *Accuracy* e *F-Measure*. Assim, o poder preditivo do modelo aprendido segue a qualidade e a assimetria do conjunto de dados apresentado na fase de treinamento.

6.2 Agrupamento de cursos considerando seus Grupos

Para observar se há um aumento no poder preditivo das regras aprendidas ao adicionar mais dados, uma combinação de diferentes cursos também foi utilizada. Em outras palavras, deseja-se verificar se o treinamento de vários cursos juntos leva a um melhor desempenho quando comparado ao treinamento de cada curso separadamente. Assim, como os cursos foram separados em diferentes grupos baseados em suas taxas de evasão, o treinamento conjunto dos cursos que pertencem a um mesmo grupo também faz sentido. A tabela 6.4 mostra os resultados da validação cruzada por grupo de taxa de evasão (Grupos I, II e III). Os grupos também foram treinados de acordo com a sua área de pesquisa. Os resultados para essa abordagem são exibidos na Tabela 6.5.

Tabela 6.4: Resultados preditivos da validação cruzada quando os cursos pertencentes ao mesmo Grupo são treinados juntos

Cursos	Precision	Recall	Accuracy	F-Measure
Grupo I	82,38% ± 1,95%	88,69% ± 1,51%	76,40% ± 2,32%	85,41% ± 1,35%
Grupo II	55,11% ± 6,11%	44,19% ± 12,58%	64,76% ± 3,81%	48,22% ± 8,81%
Grupo III	3,33% ± 10,00%	2,00% ± 6,00%	85,52% ± 3,68%	2,50% ± 7,50%

Tabela 6.5: Resultado da validação cruzada quando diferentes cursos são combinados

Cursos	Precision	Recall	Accuracy	F-Measure
Eng. Telecom e Ciência da Computação	68,79% ± 3,27%	72,98% ± 7,25%	62,65% ± 4,50%	70,69% ± 4,50%
Civil, Telecom., Química e Eng. Produção	60,92% ± 8,93%	50,63% ± 10,07%	65,49% ± 6,15%	54,86% ± 8,81%
Medicina, Nutrição e Farmácia	52,50% ± 14,79%	32,64% ± 12,37%	79,86% ± 3,91%	38,98% ± 11,88%

Com relação à abordagem de agrupamento de taxa de abandono na Tabela 6.4, podemos verificar que a *F-Measure* é realmente tão melhor quanto o número de exemplos positivos do grupo. Como o grupo de testes não é exatamente igual quando comparado

ao treinamento individual, ainda não é possível chegar a muitas conclusões, mas leva a acreditar que alguns cursos podem ser beneficiados quando treinados em conjunto, até mesmo para os cursos onde o abandono não é um grande problema. Já com relação a tabela 6.5, os resultados indicam que possivelmente agrupar cursos com alta taxa de evasão pode não ser muito benéfico, mas essa questão será respondida posteriormente, quando o teste será formado pelo mesmo grupo de alunos em todos os casos.

Assim, pode-se responder parcialmente a questão **Q#2** afirmando que quando o conjunto de dados já possui uma boa proporção de ambas as classes de exemplos, não é sempre necessário combinar diferentes cursos. Por outro lado, isso é essencial quando o conjunto de dados está desbalanceado. Isto é particularmente importante para não direcionar recursos para os cursos que não precisam, com mais confiança.

Para continuar respondendo a questão **Q#2**, dois outros grupos foram formados considerando os cursos com a mesma raiz acadêmica. Os cursos de Engenharia, Matemática, Física, Química e Estatística foram incluídos em um grupo chamado Ciências Exatas, enquanto os outros cursos foram agrupados como Ciências Biomédicas. Os resultados para esses grupos são exibidos na Tabela 6.6.

O grupo de Ciências Exatas demonstrou um melhor desempenho quando comparado ao grupo de Ciências Biomédicas, o que já era esperado, devido a grande diferença no número de exemplos positivos nesses dois grupos.

Tabela 6.6: Resultados preditivos da validação cruzada com cursos agrupados pelas áreas de pesquisa

Grupos	Precision	Recall	Accuracy	F-Measure
Ciências Exatas	78,73% ± 2,80%	79,95% ± 3,09%	72,72% ± 2,49%	79,27% ± 1,86%
Ciências Biomédicas	54,78% ± 13,24%	43,44% ± 23,34%	80,32% ± 4,20%	45,28% ± 16,13%

6.3 Capacidade de generalização das regras

Para que fosse possível verificar ainda mais o poder de generalização da abordagem em relação ao cursos não utilizados durante a fase de treinamento, as regras pelo Aleph foram testadas em cursos que não fizeram parte do treinamento. Considerando que Aleph retorna uma lista de regras aprendidas na fase de treinamento, é possível realizar inferências em um conjunto de exemplos não vistos anteriormente pelo modelo para verificar o quão

genéricas estas regras são. Os resultados para essa estratégia estão listados na Tabela 6.7.

Tabela 6.7: Desempenho quando regras aprendidas por um ou mais cursos são aplicadas em outros cursos.

Curso Treinado	Curso Testado	Precision	Recall	Accuracy	F-Measure
Eng. Telecom	Ciência da Computação	64,71%	85,82%	61,95%	73,78%
Ciência da Computação	Eng. Telecom	64,41%	80,00%	61,39%	71,36%
Eng. Telecom	Farmácia	63,64%	54,55%	66,67%	58,74%
Eng. Telecom	Civil, Química, Eng. Produção	40,27%	64,78%	50,97%	49,67%
Farmácia	Nutrição	37,04%	62,50%	68,49%	46,51%
Eng. Telecom	Nutrição	33,33%	50,00%	72,73%	40,00%
Nutrição	Farmácia	72,41%	27,27%	63,84%	39,62%
Civil, Química, Eng. Produção	Eng. Telecom	70,83%	17,89%	46,20%	28,57%
Civil, Telecom, Química, Eng. Produção	Ciência da Computação	63,16%	17,02%	42,04%	26,82%
Eng. Telecom	Eng. Elétrica	64,29%	12,86%	55,70%	21,43%
Farmácia e Nutrição	Medicina	15,15%	20,83%	85,84%	17,54%
Eng. Telecom	Medicina	8,26%	37,50%	65,36%	13,53%
Medicina e Farmácia	Nutrição	0,00%	0,00%	77,93%	0,00%
Medicina e Nutrição	Farmácia	0,00%	0,00%	56,50%	0,00%

De acordo com os resultados da Tabela 6.7, podemos concluir que a maioria dos cursos na mesma área têm comportamentos de abandono muito relacionados. No entanto, alguns cursos que têm raízes semelhantes, como Engenharia de Telecomunicações e Engenharia Elétrica, não se beneficiaram dessa abordagem, tornando muito difícil extrair um padrão que explica o abandono de ambos. Cursos com baixíssimos índices de abandono, como

Nutrição e Medicina, foram beneficiados ao utilizar regras aprendidas de outros cursos com maior taxa de abandono. Isso é esperado de alguma forma, já que os cursos com maiores taxas de abandono são mais propensos a gerar regras, enquanto os cursos com regras de abandono muito baixas na maioria das vezes geram um conjunto muito pequeno de regras.

Assim, a questão **Q#3** proposta não pode ser respondida positivamente para todos os cursos testados na Tabela 6.7, pois em alguns casos não foi possível extrair um padrão. Isso é bastante compreensível, pois as razões para o abandono de um curso podem, de fato, variar de curso para curso, particularmente quando toda a vida acadêmica relacionada ao aluno é considerada.

6.4 Avaliando a capacidade de predição do modelo

Para verificar o poder preditivo das regras aprendidas quando relacionadas a dados futuros, alunos ingressantes em 2011 foram considerados como conjunto de teste. Dessa forma, regras aprendidas através do treinamento de alunos ingressantes nos anos de 2009 e 2010 foram aplicadas aos alunos de 2011. Os resultados estão nas tabelas 6.8, 6.9, 6.10, 6.11, e podem ser usados para responder a questão **Q#4**, com relação a capacidade de predição do modelo com relação a alunos ingressantes nos anos seguintes ao treinamento.

Tabela 6.8: Regras aprendidas quando apenas o curso é treinado separadamente, contendo dados de alunos matriculados em 2009 e 2010, aplicadas em alunos ingressantes em 2011 do mesmo curso treinado.

Cursos	Precision	Recall	Accuracy	F-Measure
Física	89,51%	99,22%	88,89%	94,12%
Matemática	91,16%	91,78%	84,28%	91,47%
Ciência da Computação	94,37%	84,81%	81,82%	89,33%
Química	79,10%	94,64%	76,39%	86,18%
Eng. Telecom	76,92%	71,43%	65,57%	74,07%
Estatística	90,91%	62,50%	63,64%	74,07%
Farmácia	65,96%	54,39%	54,35%	59,62%
Eng. Civil	73,08%	43,18%	58,97%	54,29%
Nutrição	62,50%	35,71%	69,62%	45,45%
Eng. Produção	58,82%	34,48%	63,89%	43,48%
Eng. Química	52,63%	27,03%	66,36%	35,71%
Medicina	0,00%	0,00%	86,86%	0,00%

Tabela 6.9: Regras aprendidas pelo Grupo I, quando todos os cursos deste grupo são treinados juntos, testadas em cada curso do grupo, contendo alunos ingressantes em 2011.

Cursos	Precision	Recall	Accuracy	F-Measure
Física	90,44%	95,35%	86,81%	92,83%
Matemática	91,78%	91,78%	84,91%	91,78%
Química	78,87%	100,00%	79,17%	88,19%
Estatística	86,67%	81,25%	74,03%	83,87%
Ciência da Computação	96,67%	73,33%	73,86%	83,45%
Eng. Teleco-municações	80,00%	76,19%	70,49%	78,45%

Tabela 6.10: Regras aprendidas pelo Grupo II, quando todos os cursos deste grupo são treinados juntos, testadas em cada curso do grupo, contendo alunos ingressantes em 2011.

Cursos	Precision	Recall	Accuracy	F-Measure
Farmácia	62,00%	54,39%	51,09%	57,94%
Eng. Produção	52,00%	44,83%	61,11%	48,15%
Eng. Civil	57,69%	34,09%	48,72%	42,86%
Eng. Química	39,02%	43,24%	57,01%	41,03%

Tabela 6.11: Regras aprendidas pelo Grupo III, quando todos os cursos deste grupo são treinados juntos, testadas em cada curso do grupo, contendo alunos ingressantes em 2011.

Cursos	Precision	Recall	Accuracy	F-Measure
Nutrição	46,67%	25,00%	63,29%	32,56%
Medicina	0,00%	0,00%	86,86%	0,00%

Através de uma análise das Tabelas 6.8, 6.9, 6.10, 6.11, pode-se afirmar que:

1. **Grupo I**: Física e Ciência da Computação obtiveram um desempenho pior quando comparado aos seus treinamentos quando feitos separadamente. Os outros 4 cursos: Matemática, Química, Estatística e Engenharia de Telecomunicações foram beneficiados, tendo um aumento na *F-Measure*.

2. **Grupo II**: Engenharia de Produção e Química foram beneficiados, enquanto Farmácia e Engenharia Civil tiveram desempenhos piores.

3. **Grupo III**: Esse grupo não se beneficionou dessa estratégia. Nutrição, que tinha uma *F-Measure* de 45,45% teve uma redução para 32,56%. Medicina também não se beneficiou, mantendo sua F-Measure em 0%. Para este caso, apenas uma regra bastante específica (relacionada ao curso de Nutrição) foi gerada durante a fase de treinamento, sendo então impossível a sua adequação a qualquer aluno de M edicina.

Mesmo que o agrupamento possa parecer benéfico na Tabela 6.6, quando os cursos foram agrupados dentro de suas respectivas áreas de pesquisa, a melhor forma de avaliar

se essa abordagem pode ser realmente benéfica quando comparada ao agrupamento por taxa de desistência é aplicar as regras aprendidas aos cursos, considerando apenas os alunos matriculados em 2011, como foi feito anteriormente. Os resultados são mostrados em Tabelas 6.12, 6.13, respectivamente.

Tabela 6.12: Regras aprendidas através do treinamento de todos os cursos de Ciências Exatas juntos, nos anos de 2009 e 2010, testadas com alunos ingressantes em 2011 de cada um dos cursos do grupo.

Cursos	Precision	Recall	Accuracy	F-Measure
Matemática	92,67%	95,21%	88,68%	93,92%
Física	90,58%	96,90%	88,19%	93,63%
Ciência da Computação	94,44%	86,08%	82,95%	90,07%
Química	78,26%	96,43%	76,39%	86,40%
Eng. Telecom.	81,82%	85,71%	77,05%	83,72%
Estatística	84,13%	82,81%	72,73%	83,46%
Eng. Química	66,67%	64,86%	76,64%	65,75%
Eng. Civil	72,22%	59,09%	64,10%	65,00%
Eng. Produção	52,00%	44,83%	61,11%	48,15%

Tabela 6.13: Regras aprendidas através do treinamento de todos os cursos de Ciências Biomédicas juntos, nos anos de 2009 e 2010, testadas com alunos ingressantes em 2011 de cada um dos cursos do grupo.

Cursos	Precision	Recall	Accuracy	F-Measure
Nutrição	50,00%	32,14%	64,56%	39,13%
Farmácia	62,26%	57,89%	52,17%	60,00%
Medicina	0,00%	0,00%	86,86%	0,00%

Os resultados extraídos das Tabelas 6.12 e 6.13 foram condensados e combinados com as outras estratégias apresentadas anteriormente, para facilitar a análise. A Figura 6.1 mostra esses resultados agrupados e, com base na análise feita sobre ela, pode-se afirmar que:

1. Agrupar por área de pesquisa é uma abordagem melhor para a maioria dos Cursos. 8 dos 12 cursos obtiveram um melhor desempenho ou mantiveram o mesmo resultado quando comparado a outras estratégias. Os cursos de Engenharia Química e Engenharia Civil obtiveram uma grande melhoria.

2. Química e Estatística foram os únicos cursos que obtiveram melhor desempenho quando treinados utilizando os Grupos baseados na taxa de evasão (Grupos I, II, III).

3. Física e Nutrição foram os únicos cursos que obtiveram melhor desempenho quando treinados separadamente.

4. Nenhuma das estratégias resultou em um melhor desempenho para Medicina.

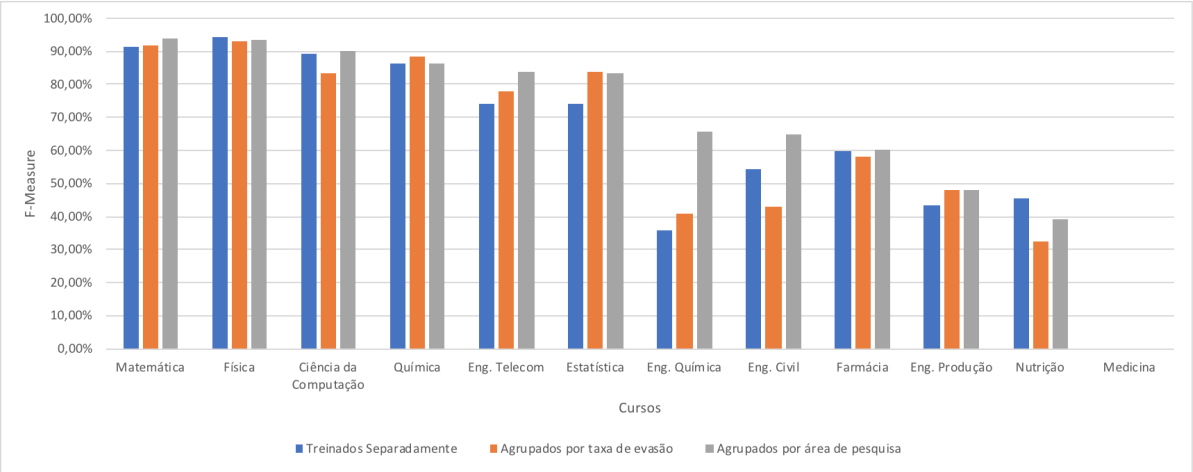

Figura 6.1: F-Measure por curso e por estratégia quando o modelo é testado com dados de 2011.

Quanto à Medicina, a abordagem por área de pesquisa não beneficiou o curso, pois 8 regras foram geradas pelo Aleph e todas específicas para Farmácia ou Nutrição, tendo o identificador único do curso sendo considerado em literais no corpo da regra. Desse modo, Medicina não teve verdadeiros positivos nem falsos positivos, pois seria impossível ter uma regra visando algum estudante de Medicina.

6.5 Análise qualitativa das regras

Finalmente, para responder a **Q#5** e observar o aspecto semântico das teorias aprendidas, apresenta-se nesta seção uma amostra de regras obtidas a partir do treinamento. Algumas delas expressaram riqueza em termos de detalhes, enquanto outras são regras bastante simples.

Por exemplo, a regra abaixo foi gerada a partir dos dados do curso de Engenharia de Telecomunicações.

```
evasao(A) :-
    ano_semestre_ingresso(A,B,C),
    resultado_aluno_turma(A,reprovado,D,E,B,C),
    classe_idade_aluno(A,'A').
```

Essa regra pode ser lida como: Um estudante evade SE ele/ela iniciou seu curso no ano de B, no semestre C e foi reprovado na disciplina D, turma E no mesmo ano e semestre (B, C) E sua idade está classificada no intervalo A. O intervalo A significa que o aluno tem entre 15 e 19 anos de idade.

Essa regra sugere uma informação que pode fazer sentido, já que quando o aluno é reprovado no início do curso, ele pode se sentir desmotivado e até mesmo se perguntar se optou pelo curso correto.

Além da regra gerada, o Aleph apresenta a cobertura da mesma, isto é, quando exemplos positivos e negativos foram cobertos pela mesma. No caso da regra acima, foram cobertos 29 exemplos positivos e 10 exemplos negativos.

Para o mesmo curso, a regra abaixo também foi gerada:

```
evasao(A) :-
    classe_idade_aluno(A,'C').
```

Embora esta regra pareça simples, ela também pode fazer sentido em muitos casos, uma vez que os alunos com a classificação dada C têm entre 25 e 29 anos e, normalmente, as pessoas que fazem parte deste intervalo têm maior probabilidade de serem casadas e/ou trabalhar enquanto estudam. Nesse caso, a cobertura da regra foi de 10 exemplos positivos e apenas 1 exemplo negativo.

Ao agrupar alguns cursos juntos, algumas regras foram muito específicas para um único curso, enquanto outras não mencionavam o identificador do curso (seu código), transformando-o em uma regra mais genérica. Os exemplos para o grupo de Ciências Exatas estão listados abaixo.

```
evasao(A) :-
    cidade(A,'Macae').
```

Para essa regra, o Aleph sugere que os estudantes de *Macaé*, uma cidade localizada a 174km de distância da maior parte dos *campi* da UFF, evadem. A cobertura desta regra foi de 5 exemplos positivos e nenhum negativo.

```
evasao(A) :-
    genero_aluno(A,'M'), classe_idade_aluno(A,'G').
```

Semelhante à regra discutida anteriormente, em relação à faixa etária do aluno, ao usar apenas o identificador único do aluno como parte do *target*, essa regra sugere que os alunos entre 45 e 49 anos (classificados como classe G) tem chance de desistência. A cobertura neste caso foi de 14 exemplos positivos e nenhum negativo.

```
evasao(A) :-
    curso_aluno(A,20), genero_aluno(A,'F'), cidade(A,'Sao Goncalo').
```

Nesse caso, uma regra muito específica foi gerada, declarando que um aluno desiste se estuda $Matematica$ (identificador do curso 20) E é do sexo $feminino$ E é de *São Gonçalo*. *São Gonçalo* é uma cidade localizada a 10 km do centro de Niterói, que é próxima aos diversos *campi* dos cursos treinados. A cobertura neste caso foi de 34 exemplos positivos e 7 negativos.

As regras acima indicam que o modelo pode ser facilmente interpretado por humanos, o que é uma grande vantagem sobre muitas técnicas tradicionais de aprendizado de máquina. Ter a possibilidade de ler essas regras e verificar quais delas cobrem a maioria dos casos positivos de abandono pode ajudar a administração da Universidade a identificar possíveis grupos em risco. Não apenas o modelo poderia indicar os alunos exatos que provavelmente abandonariam, mas a administração poderia aprender os padrões, já que as regras são compreensíveis. Dessa forma, a universidade pode tomar medidas mais eficazes contra o abandono do aluno.

Quanto a cobertura das regras, algumas delas cobriram exemplos negativos, mas isso foi necessário para possibilitar maior flexibilidade na criação das regras, criando cláusulas menos específicas, conforme descrito no Capítulo 5. Apesar dessas regras cobrirem exemplos negativos, o número de exemplos positivos cobertos foi predominante em relação ao número de exemplos negativos cobertos pela mesma regra.

6.6 Tempo de execução do modelo

Como forma de avaliar o tempo de execução do modelo, alguns cursos foram escolhidos para medição do tempo de execução. O tempo total de execução leva em consideração o somátorio do tempo necessário para treinamento de cada um dos 10 *folds* utilizados na validação cruzada. Foram selecionados alguns cursos isoladamente e também os grupos

considerando taxas de evasão e áreas de pesquisa. Os dados compreendem os alunos que ingressaram nos anos de 2009 e 2010. Os resultados são apresentados nas Tabelas 6.14, 6.15 e 6.16.

Tabela 6.14: Tempos de execução do treinamento, considerando o somátorio do tempo de cada fold, por curso.

Cursos	Tempo de execução
Física	6 minutos e 12 segundos
Matemática	10 minutos e 9 segundos
Ciência da Computação	26 minutos e 1 segundo
Eng. Telecomunicações	5 minutos e 7 segundos
Eng. Produção	21 minutos e 19 segundos
Medicina	10 minutos e 7 segundos

Com relação à Tabela 6.14, apesar dos tempos de execução não serem necessariamente diretamente proporcionais ao número de exemplos positivos e negativos, o treinamento considerando os 10 *folds*, independente do curso, ocorre em tempo computacional satisfatório para o objetivo deste trabalho. Além disso, diferentemente da abordagem adotada, cada *fold* poderia ser treinado de forma paralela, o que resultaria em um tempo de execução na média 10 vezes menor, visto que o tempo de execução de cada *fold* é bastante semelhante.

Tabela 6.15: Tempos de execução do treinamento, considerando o somátorio do tempo de cada fold, por grupos separados por taxa de evasão.

Cursos	Tempo de execução
Grupo I	2 horas, 1 minuto e 48 segundos
Grupo II	2 horas e 45 minutos
Grupo III	38 minutos e 43 segundos

Tabela 6.16: Tempos de execução do treinamento, considerando o somátorio do tempo de cada fold, por grupos separados por área de pesquisa.

Cursos	Tempo de execução
Exatas	6 horas e 45 minutos
Biomédicas	3 horas, 15 minutos e 36 segundos

No caso das Tabelas 6.15 e 6.16, o tempo computacional maior quando comparado aos tempos apresentados na Tabela 6.14 faz sentido, visto que o número de predicados

é maior nesses casos. Não só o treinamento em grupo poderia se beneficiar de maior performance caso os *folds* fossem treinados em paralelo, mas também o próprio tempo em série é satisfatório uma vez que o treinamento ocorre precisa ocorrer apenas uma vez por semestre.

Capítulo 7

Conclusão e Trabalhos Futuros

Detectar efetivamente alunos em risco de abandono escolar nas universidades é essencial para otimizar a utilização dos recursos financeiros. No entanto, essa é uma tarefa desafiadora para os algoritmos clássicos de aprendizado de máquina, já que as informações demográficas e acadêmicas dispostas em várias tabelas relacionadas devem ser exploradas juntas para identificar as causas reais desse problema. Assim, este trabalho teve como objetivo aplicar um algoritmo de aprendizagem relacional para induzir modelos capazes de prevenir o abandono escolar de uma grande universidade brasileira. Para alcançar esse objetivo, o sistema de aprendizagem relacional baseada em lógica Aleph foi incorporado em um sistema chamado PRELUDE, que é responsável por coletar os dados das fontes de banco de dados e transformá-los em fatos lógicos.

Para treinar os modelos e observar sua capacidade de generalização sobre os diferentes cursos oferecidos pela universidade, três grupos foram selecionados seguindo a taxa de evasão (marcada como alta, média e baixa). Para observar o comportamento das regras aprendidas em diferentes cenários, os cursos também foram divididos em duas áreas científicas: Ciências Exatas e Biomédicas.

As regras foram aprendidas através da utilização de três estratégias diferentes: treinamento dos cursos separadamente, em grupos de acordo com as taxas de evasão, e em grupos de acordo com suas áreas científicas.

Ao analisar os resultados produzidos neste trabalho através do procedimento de validação cruzada, pode-se concluir que as regras aprendidas são capazes de classificar com precisão os alunos que provavelmente evadirão seus cursos. Além disso, as regras também são muito boas em prever os alunos em risco de abandono no próximo ano. Quando o modelo foi testado levando em consideração os agrupamentos em Ciências Exatas e Bio-

médicas, o desempenho aumentou para alguns dos cursos com taxas de abandono médias ou mesmo baixas.

Em geral, os cursos com média ou baixa taxa de abandono tiveram resultados piores que aqueles com maiores taxas de abandono, possivelmente devido ao baixo número de exemplos positivos, o que pode ser visto como uma limitação do método. No entanto, cursos com altas taxas de abandono são, na verdade, o foco deste trabalho, pois são eles que geram mais perdas financeiras para as universidades. Além disso, isso também pode colocar em risco a economia do país, pois as demandas da indústria e dos serviços podem não ser atendidas quando um grande número de alunos evadem.

No entanto, sempre é possível aplicar métodos desenvolvidos para lidar com conjuntos de dados altamente desbalanceados [38] para lidar com esses cursos com baixas taxas de desistência. Reequilibrar o número de exemplos positivos e negativos pode ser benéfico para o modelo e pode ser usado em um trabalho futuro. Outra direção do trabalho futuro seria criar conjuntos de dados variando o ano de ingresso do estudante na universidade para ver se o PRELUDE também é capaz de produzir regras úteis de desistência para os alunos no início de sua vida acadêmica. Mesmo que ter um aluno desistindo no início seja menos prejudicial, uma vez que a universidade gastou menos recursos com eles do que os alunos no final do curso, prever o abandono com anos de antecedência poderia ajudar a Universidade a elaborar uma estratégia de longo prazo.

Esse trabalho demonstra a viabilidade na utilização de aprendizado relacional para predição de evasão através da apresentação dos resultados gerados. No entanto, para que seja possível a atuação da administração da Universidade, um sistema *web* poderia ser construído de forma a realizar um *parsing* dos resultados das aplicações das regras geradas pelo Aleph, em cada aluno e, após, apresentar gráficos e tabelas que indiquem um "termômetro"para o curso, dependendo do número de alunos classificados em risco com relação ao número total de alunos e tabelas que exibam quais alunos estão no grupo de risco.

Quanto a escalabilidade do modelo, o Aleph não se beneficiou das arquiteturas de processadores *multi-core*, utilizando apenas um *core* durante as fases de treinamento e testes. Para o treinamento de cursos isoladamente, a validação cruzada levou um tempo da ordem de minutos para finalizar os 10 *folds*, que foram executados de forma serial. Porém, ao agrupar cursos como o que foi feito para Ciências Exatas, o tempo de execução dos *folds* foi da ordem de 6h em um computador com 8 núcleos e 16GB de RAM. Dessa forma, o agrupamento de muitos cursos no mesmo treinamento poderia levar a muitas

horas de execução, visto que não há paralelização por padrão. Isso não é exatamente um problema para este trabalho, visto que os dados sobre o desempenho acadêmico do aluno são adicionados uma vez por semestre e não há *streamming* de dados, ou seja, não há necessidade de treinamento em intervalos de tempo muito pequenos. O treinamento realizado uma vez por semestre pode ser utilizado ao longo de meses. Além disso, o tempo computacional necessário para verificar quais são os alunos que se enquadram em determinada regra gerada pode ser considerado instantâneo, dando respostas na ordem de milisegundos.

Prever se um aluno pode desistir usando as regras aprendidas com o PRELUDE pode ajudar a administração da universidade na identificação de alunos que apresentam risco de desistência, tarefa muito difícil de ser realizada sem o uso de algoritmos preditivos devido ao grande volume de dados que precisa ser analisado. Como consequência, há uma maior possibilidade de reversão da situação.

Apesar da capacidade de predição da evasão de determinados alunos, com alta acurácia e/ou *F-Measure* para cursos com muitos exemplos positivos ser muito relevante, a capacidade de identificar alunos desses cursos que não correm alto risco de evasão também é muito benéfica, e isso também é possível através do trabalho aqui realizado. Os alunos identificados como aqueles que não irão evadir podem ser utilizados pelas coordenações de curso para formarem grupos de estudo, receberam bolsas de monitoria e ajudarem os alunos com mais dificuldade, atuando de forma complementar a força de trabalho dos professores.

Referências

[1] Disponível em `http://weka.sourceforge.net/doc.dev/weka/classifiers/meta/CostSensitiveClassifier.html`.

[2] Disponível em `https://www.cs.waikato.ac.nz/ml/weka/`.

[3] Disponível em `https://www.docker.com/`.

[4] Agarap, A. F. M. On breast cancer detection: An application of machine learning algorithms on the wisconsin diagnostic dataset.

[5] Allan Sales, Leandro B. Marinho, A. C. Predicting student dropout: A case study in brazilian higher education.

[6] Balaniuk, R.; Antonio do Prado, H.; da Veiga Guadagnin, R.; Ferneda, E.; Cobbe, P. Predicting evasion candidates in higher education institutions. *Model and Data Engineering* (2011), 143–151.

[7] Barbosa, A.; Santos, E.; Pordeus, J. P. A machine learning approach to identify and prioritize college students at risk of dropping out. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)* (2017), vol. 28, p. 1497.

[8] Bonaldo, L.; Pereira, L. N. Dropout: Demographic profile of brazilian university students. *Procedia - Social and Behavioral Sciences 228* (2016), 138 – 143. 2nd International Conference on Higher Education Advances,HEAd'16, 21-23 June 2016, València, Spain.

[9] Brachman, R. J.; Levesque, H. J.; Reiter, R. *Knowledge representation.* MIT press, 1992.

[10] Cohen, W. W. Text categorization and relational learning.

[11] Craven, M.; Slattery, S. Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning 43*, 1/2 (2001), 97–119.

[12] da Silva, H. R. B.; Adeodato, P. J. L. A data mining approach for preventing undergraduate students retention. In *Neural Networks (IJCNN), The 2012 International Joint Conference on* (2012), IEEE, pp. 1–8.

[13] Daniel de Oliveira, Eduardo Ogasawara, F. B. M. M. Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows.

[14] de Carvalho Melo Lobo, M. B. Panorama da evasao no ensino superior brasileiro: Aspectos gerais das causas e solucoes.

[15] De Raedt, L. *Logical and relational learning.* Springer Science & Business Media, 2008.

[16] Dekker, G.; Pechenizkiy, M.; Vleeshouwers, J. Predicting students drop out: A case study. In *Educational Data Mining 2009* (2009).

[17] Delen, D. A comparative analysis of machine learning techniques for student retention management. *Decision Support Systems 49*, 4 (2010), 498–506.

[18] Frazier, M.; Pitt, L. Learning from entailment: An application to propositional horn sentences. In *Machine Learning, Proceedings of the Tenth International Conference* (1993), Morgan Kaufmann, pp. 120–127.

[19] Hanley, J. A.; McNeil, B. J. The meaning and use of the area under a receiver operating characteristic (roc) curve.

[20] He, J.; Bailey, J.; Rubinstein, B. I. P.; Zhang, R. Identifying at-risk students in massive open online courses. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015), AAAI'15, AAAI Press, pp. 1749–1755.

[21] Husainy, S. I. Identifying likely student dropouts using fuzzy inferencing. In *Proceedings of the 51st ACM Southeast Conference* (New York, NY, USA, 2013), ACMSE '13, ACM, pp. 42:1–42:1.

[22] Inmon, W. H. *Building the Data Warehouse.* John Wiley & Sons, Inc., New York, NY, USA, 1992.

[23] John O. Awoyemi, Adebayo O. Adetunmbi, S. A. O. Credit card fraud detection using machine learning techniques: A comparative analysis.

[24] Kohav, R. A study of crossvalidation and bootstrap for accuracy estimation and model selecti.

[25] Lakkaraju, H.; Aguiar, E.; Shan, C.; Miller, D.; Bhanpuri, N.; Ghani, R.; Addison, K. L. A machine learning framework to identify students at risk of adverse academic outcomes. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015), ACM, pp. 1909–1918.

[26] Lewis, D. D. Naive (bayes) at forty: The independence assumption in information retrieval. *Lecture Notes in Computer Science* (2005), 4–15.

[27] Manhães, L. M. B.; da Cruz, S. M. S.; Zimbrão, G. Evaluating performance and dropouts of undergraduates using educational data mining. In *Proceedings of the Twenty-Ninth Symposium on Applied Computing* (2014).

[28] Markov, Z.; Russell, I. Relational learning for web document classification.

[29] Márquez-Vera, C.; Cano, A.; Romero, C.; Ventura, S. Predicting student failure at school using genetic programming and different data mining approaches with high dimensional and imbalanced data. *Applied intelligence 38*, 3 (2013), 315–330.

[30] Michalski, R. S.; Carbonell, J. G.; Mitchell, T. M. *Machine learning: An artificial intelligence approach.* Springer Science & Business Media, 2013.

[31] M.S. Bartlett, G. Littlewort, M. F. Recognizing facial expression: machine learning and application to spontaneous behavior.

[32] Muggleton, S. Inductive logic programming. *New generation computing 8*, 4 (1991), 295–318.

[33] Muggleton, S. Inverse entailment and progol. *New Generation Comput. 13*, 3&4 (1995), 245–286.

[34] Muggleton, S.; Raedt, L. Inductive logic programming: Theory and methods. *Journal of Logic Programming* (1994), 629–679.

[35] Muggleton, S.; Santos, J. C. A.; Tamaddoni-Nezhad, A. Toplog: ILP using a logic program declarative bias. In *Logic Programming, 24th International Conference, ICLP 2008, Proceedings* (2008), vol. 5366 of *Lecture Notes in Computer Science*, Springer, pp. 687–692.

[36] Nandeshwar, A.; Menzies, T.; Nelson, A. Learning patterns of university student retention. *Expert Systems with Applications 38*, 12 (2011), 14984–14996.

[37] Nilsson, U. *Logic, programming and Prolog.*

[38] Oliphant, L.; Burnside, E. S.; Shavlik, J. W. Boosting first-order clauses for large, skewed data sets. In *Inductive Logic Programming, 19th International Conference, ILP 2009. Revised Papers* (2010), vol. 5989 of *LNCS*, Springer, pp. 166–177.

[39] Picado, J.; Termehchy, A.; Fern, A.; Ataei, P. Schema independent relational learning. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017* (2017), ACM, pp. 929–944.

[40] Pittman, K. *Comparison of data mining techniques used to predict student retention.* Nova Southeastern University, 2008.

[41] Raedt, L. D. *Inductive Logic Programming.* Springer US, 2010, pp. 529–537.

[42] Raedt, L. D. *Logical and Relaetional Learning.* 2010.

[43] Rumberger, R. W.; Lim, S. A. Why students drop out of school: A review of 25 years of research. Tech. rep., California Dropout Research Project Report Santa Barbara, CA, 2008.

[44] Sado Dzeroski, N. L. Inductive logic programming, techniques and applications.

[45] Sales, A.; Balby, L.; Cajueiro, A. Predicting student dropout: A case study in brazilian higher education. In *Proc. of the 3rd Symposium on Knowledge Discovery, Mining and Learning* (2015).

[46] Srinivasan, A. The aleph manual. Disponível em `http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/aleph`.

[47] Srinivasan, A. The aleph manual. Disponível em `http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/aleph.html#SEC8`.

[48] Srinivasan, A. The aleph manual: version 4 and above. *URL: http://www. comlab. o x. ac. uk/activities/machinelearning/Aleph/(accessed 10.09. 10)* (2007).

[49] Stratton, L. S.; O'Toole, D. M.; Wetzel, J. N. A multinomial logit model of college stopout and dropout behavior. *Economics of Education Review 27*, 3 (2008), 319 – 331.

[50] Struyf, J.; Blockeel, H. *Relational Learning*. Springer US, 2010.

[51] Tamhane, A.; Ikbal, S.; Sengupta, B.; Duggirala, M.; Appleton, J. Predicting student risks through longitudinal analysis. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), ACM, pp. 1544–1552.

[52] Tang, J. K. T.; Xie, H.; Wong, T.-L. A big data framework for early identification of dropout students in mooc. In *Technology in Education. Technology-Mediated Proactive Learning* (Berlin, Heidelberg, 2015), J. Lam, K. K. Ng, S. K. Cheung, T. L. Wong, K. C. Li, and F. L. Wang, Eds., Springer Berlin Heidelberg, pp. 127–132.

[53] Vítor Lourenço, Paulo Mann, A. P. e. D. d. O. Siapp: Um sistema para análise de ocorrências de crimes baseado em aprendizado lógico-relacional.