UNIVERSIDADE ABERTA

SELEÇÃO DE ATRIBUTOS DE DADOS INCONSISTENTES

João Daniel Baião de Brito Apolónia

Mestrado em Tecnologias e Sistemas Informáticos Web

Lisboa 2018

UNIVERSIDADE ABERTA

SELEÇÃO DE ATRIBUTOS DE DADOS INCONSISTENTES

João Daniel Baião de Brito Apolónia

Mestrado em Tecnologias e Sistemas Informáticos Web

Dissertação orientada pelo
Professor Doutor Luís Manuel Pereira Sales Cavique Santos

Lisboa 2018

Resumo

O tratamento de conjuntos de dados de grande dimensão é uma questão que é recorrente nos dias de hoje e cuja tarefa não é simples, dadas as limitações computacionais, ainda, existentes. Uma das abordagens possíveis passa por realizar uma seleção de atributos que permita diminuir, consideravelmente, a dimensão dos dados sem aumentar a inconsistência dos mesmos. "*Rough Sets*" é uma abordagem que difere doutras técnicas de seleção de atributos pela sua capacidade de lidar com dados inconsistentes. Outra abordagem para redução de dados é conhecida como Análise Lógica de Dados (LAD). A Análise Lógica de Dados Inconsistentes (LAID) junta as vantagens destas duas abordagens.

Com o grande aumento do volume de dados, o paradigma, relativamente ao seu manuseamento, tem-se alterado. Antes, o tratamento dos dados era efetuado num único computador e o acesso era realizado depois do seu carregamento em memória. A tendência atual é aceder aos dados em disco, num ambiente *cloud*. O trabalho realizado pretende validar este novo paradigma, com recurso ao sistema de dados HDF5 (*Hierarchical Data Format*) e ao ambiente remoto disponibilizado pela INCD (Infraestrutura Nacional de Computação Distribuída). Pelo facto de o HDF5 ser o sistema adotado pela comunidade *Python* para lidar com dados de grande dimensão, esta linguagem foi escolhida para implementação do LAID.

A presente dissertação é mais um contributo para o aprofundamento das técnicas de *Data Mining* (extração de conhecimento de dados). Nomeadamente, aborda a seleção de atributos (*feature selection*) aplicada a conjunto de dados de grande dimensão, guardados no formato HDF5, com avaliação da inconsistência dos dados, através da aplicação do algoritmo LAID, codificado em *Python*, num ambiente *cloud*.

Palavras chave: *Data Mining*; Seleção de atributos; LAID; Inconsistência dos dados; HDF5; *Python*; INCD.

Abstract

The treatment of large datasets is an issue that is often addressed today and whose task is not simple, given the computational limitations that still exist. One possible approach is to perform a feature selection that allows a considerably reduction of data size without increasing inconsistency. “Rough Sets” is an approach that differs from other feature selection techniques by its ability to deal with inconsistent data. Another approach to data reduction is known as Logical Analysis of Data (LAD). Logical Analysis of Inconsistent Data (LAID) combines the advantages of these two approaches.

With the increase of large volumes of data, its handling paradigm has been changing over. Previously, data processing was performed on a single computer, with in-memory data access. The current trend is to access data on disk, in a cloud environment. The work carried out intends to validate this new paradigm, using HDF5 data system (Hierarchical Data Format) and remote environment provided by INCD (National Distributed Computing Infrastructure). Because HDF5 is the system adopted by Python’s community to handle large datasets, this language was chosen for LAID algorithm implementation.

The present document is one more contribution for deepening research of Data Mining techniques (data knowledge extraction). It addresses the feature selection applied to large datasets, stored in HDF5 format, with the evaluation of data inconsistency, through the application of LAID’s algorithm, encoded in *Python*, in a cloud environment.

Keywords: Data Mining; Feature selection; LAID; Data inconsistency; HDF5; Python; INCD.

Agradecimentos

Em primeiro lugar, quero agradecer ao professor Luís Cavique pela disponibilidade em orientar o meu projeto de dissertação, pelo constante incentivo e pelos contributos importantes dados, que, em muito, valorizaram este trabalho.

Agradeço à INCD, nas pessoas de Mário David e João Pina, por disponibilizar as infraestruturas da instituição para o armazenamento e realização dos testes necessários sobre bases de dados de enorme dimensão. Saliento a afabilidade no trato de Mário David e de João Pina.

Por último, mas muito próximo, agradeço à minha esposa e ao meu filho pela paciência em aturarem alguma falta de disponibilidade durante o período de trabalho da dissertação e por relevarem algum mau humor meu em períodos mais stressantes.

Índice

Resumo ..... **iii**

Abstract ..... **iv**

Agradecimentos ..... **v**

Índice ..... **vii**

Índice de tabelas ..... **x**

Índice de figuras ..... **xi**

Lista de abreviaturas, siglas e acrónimos ..... **xiii**

1. Introdução **1**

1.1. Enquadramento e motivação 1

1.2. Objetivos 2

1.3. Metodologia 3

1.4. Estrutura do documento 4

2. Seleção de atributos **5**

2.1. Etapas do processo de seleção de atributos 7

2.2. Diferentes abordagens à seleção de atributos 10

2.2.1. Abordagem *Filter* 10

2.2.2. Abordagem *Wrapper* 11

2.2.3. Abordagem *Embedded* 12

2.3. Relevância dos atributos 13

2.4. Algoritmos de seleção de atributos 14

3. LAID e as metodologias subjacentes **17**

3.1. Rough Sets 18

3.1.1. Indiscernibilidade e conjuntos elementares 19

3.1.2. Conjuntos aproximados....20
3.1.3. Grau de relevância dos atributos....23
3.1.4. Seleção de atributos....25
3.1.5. Função de discernibilidade....25
3.1.6. Características dos *Rough Sets*....26
3.2. Metodologia LAD....27
3.2.1. *Binarização* de um sistema de decisão....28
3.2.2. Redução do conjunto suporte....30
3.2.3. Variantes à metodologia clássica da LAD....32
3.2.4. LAD versus *Rough Sets*....33
3.3. Metodologia LAID....34
3.3.1. Remoção das redundâncias....34
3.3.2. Remoção das inconsistências....34
3.3.3. Gerar a matriz disjunta....38
3.3.4. Redução do conjunto suporte....40
3.4. Critérios de validação....43
3.4.1. Classificação....43
3.4.2. Avaliação da performance....45
4. Implementação do LAID em HDF5+*Python* no INCD....49
4.1. HDF5....50
4.2. *Python*....54
4.3. Infraestrutura Nacional de Computação Distribuída....56
4.3.1. INCD – O que é....56
4.3.2. Serviços....58
4.3.3. Impacto previsto na comunidade científica....59
4.3.4. Facilidades concedidas....60

4.4. Estratégia na implementação do LAID 61
4.4.1. Construção da matriz disjunta 62
4.4.2. Seleção dos atributos relevantes 63
4.4.3. Avaliação do conjunto de atributos selecionados 64
5. Resultados computacionais 65
5.1. Conjuntos de dados para seleção de atributos 66
5.1.1. Conjuntos de dados artificiais da categoria PTZ 67
5.1.2. Conjuntos de dados artificiais da categoria ARC 68
5.1.3. Conjuntos de dados artificiais da categoria SVA 69
5.1.4. Conjuntos de dados artificiais gerados 70
5.2. Tempos e outros indicadores de desempenho 72
5.3. Avaliação da qualidade dos resultados 77
5.4. Argumentação final 82
5.4.1. Comparação de resultados 83
5.4.2. Novo paradigma de ambiente de desenvolvimento 85
6. Conclusão 87
6.1. Trabalho futuro 88
Referências 89

Índice de tabelas

Tabela 2.1 – Algoritmos de seleção de atributos. 15
Tabela 3.1 – Sistema de decisão para exemplificar a LAID. 37
Tabela 3.2 – Sistema de decisão com a redundância removida. 37
Tabela 3.3 – Grau de inconsistência das observações. 38
Tabela 3.4 – Remover as inconsistências do sistema de decisão. 38
Tabela 3.5 – Matriz Disjunta M. 39
Tabela 3.6 – Processo de redução – iteração 1. 40
Tabela 3.7 – Processo de redução – iteração 2. 41
Tabela 3.8 – Processo de redução – iteração 3. 41
Tabela 3.9 – Sistema de decisão após redução do conjunto suporte 42
Tabela 3.10 – Valores da classe em função dos atributos selecionados. 42
Tabela 3.11 – Novo exemplo de sistema de decisão 45
Tabela 3.12 – Exemplo do cálculo da função f. 45
Tabela 3.13 – Matriz confusão quando a classe é booleana. 46
Tabela 3.14 – Escala de avaliação do índice Kappa-statistic. 47
Tabela 3.15 – Matriz confusão quando a classe é ternária. 47
Tabela 5.1 – Conjuntos de dados artificiais gerados. 71
Tabela 5.2 – Características dos conjuntos de treino e resultados da seleção de atributos. .72
Tabela 5.3 – Avaliação da seleção de atributos obtida com base na matriz confusão. 78
Tabela 5.4 –Conjunto PTZs, igual a PTZp, mas sem inconsistências. 81
Tabela 5.5 – Resultados da seleção de atributos obtidos por Cavique *et al.* (2018). 84
Tabela 5.6 – Evolução do paradigma dos ambientes de desenvolvimento. 85

Índice de figuras

Figura 2.1 – Processo de seleção de atributos com validação. 8
Figura 2.2 – Abordagem Filter. 10
Figura 2.3 – Abordagem Wrapper. 11
Figura 2.4 – Abordagem Embedded. 12
Figura 3.1 – Esquema ilustrativo da Aproximação Inferior e Superior de X. 22
Figura 3.2 – Diferença entre os Rough Sets e a remoção das inconsistências na LAID. 36
Figura 4.1 – Esquema da estrutura de um ficheiro HDF5. 51
Figura 4.2 – Estrutura dos datasets. 51
Figura 4.3 – Leitura e escrita num dataset contínuo ou compartimentado. 52
Figura 4.4 – Sobredimensionamento dos blocos num dataset. 53
Figura 4.5 – Leitura de parte de uma coluna num dataset contínuo ou compartimentado. . 53
Figura 4.6 – Exemplo de execução de código Python no modo interativo e num ficheiro. 55
Figura 4.7 – Logótipo da INCD. 57
Figura 4.8 – Aspeto geral das infraestruturas de hardware ao serviço da INCD. 57
Figura 4.9 – Página de registo para acesso aos serviços da INCD. 58
Figura 4.10 – Terminal para acesso remoto aos serviços disponibilizados pela INCD..... 60
Figura 5.1 – Espaço em disco do ficheiro HDF5, em função da dimensão dos dados. 74
Figura 5.2 – Número de atributos selecionados, em função da dimensão dos dados. 75
Figura 5.3 – Tempo de processamento, em minutos, em função da dimensão dos dados... 76
Figura 5.4 – Kappa-statistic, em função da dimensão dos dados. 79
Figura 5.5 – Taxa de cobertura, em função da dimensão dos dados. 80

Lista de abreviaturas, siglas e acrónimos

ARC – Atributos Relevantes Conhecidos

EGI – European Grid Infrastructure

FCCN – Fundação para a Computação Científica Nacional

FCT – Fundação para a Ciência e a Tecnologia

GB – Gigabyte

H5PY – HDF5 + Python

HDF5 – Hierarchical Data Format, versão 5

HPC – High Performance Computing

HTC – High Throughput Computing

INCD – Infraestrutura Nacional de Computação Distribuída

KB – Quilobyte

LAD – Análise Lógica de Dados (Logic Analysis of Data)

LAID – Análise Lógica de Dados Inconsistentes (Logic Analysis of Inconsistent Data)

LHC – Large Hadron Collider

LIP – Laboratório de Instrumentação e Física Experimental de Partículas

LNEC – Laboratório Nacional de Engenharia Civil

MB – Megabyte

PME – Pequenas e Médias Empresas

PTZ – Primeiros Todos Zero

RAM – Random Access Memory

RCTS – Rede Ciência, Tecnologia e Sociedade

SSH – Secure Shell

SVA – Soma dos Valores dos Atributos

TB – Terabyte

TI – Tecnologias de Informação

WLCG – Worldwide LHC Computing Grid

1. Introdução

1.1. Enquadramento e motivação

Data Mining consiste num conjunto de processos e métodos destinados ao tratamento e análise de dados, procurando modelos, padrões, relacionamentos entre variáveis e validá-los, aplicando esses modelos a novos conjuntos de dados.

A existência de um grande volume de observações (registos), as quais pertencem a uma de várias classes (resultados possíveis) e que poderão ter associados muitos atributos ou caraterísticas, faz com que a dimensão do conjunto de dados seja tal que, mesmo computacionalmente, torne o seu tratamento extremamente moroso e pesado. É neste contexto que surgem os métodos de seleção de atributos (*feature selection*). O intuito é escolher, entre os atributos existentes, um conjunto mínimo que permita obter resultados o mais próximo possível dos resultados do conjunto inicial, removendo atributos que sejam irrelevantes e que poderiam originar modelos de menor qualidade. Estes processos de seleção aceleram os algoritmos de *Data Mining*, melhoram a precisão das previsões e facilitam a sua compreensão (KUMAR & MINZ, 2014).

Quando o número de observações é muito menor que o de atributos, aumenta a dificuldade de todo o processo de extração de conhecimento, porque o espaço de pesquisa, normalmente, é escassamente povoado (matriz esparsa) (KUMAR & MINZ, 2014). Outros problemas prendem-se com a possibilidade de existir uma grande quantidade de atributos irrelevantes, redundantes, ou dependentes, e dados descontextualizados (*noisy data*).

Existem várias técnicas de seleção de atributos, entre as quais estão os *Rough Sets* e a Análise Lógica de Dados (LAD), cada uma com as suas virtudes e defeitos. Da combinação destas duas técnicas, surge a Análise Lógica de Dados Inconsistentes (LAID), que vai buscar o melhor de cada uma das técnicas primárias (CAVIQUE *et al.*, 2013).

Embora teoricamente, estas técnicas estejam consolidadas, os estudos aplicados a conjuntos de dados de grande dimensão são escassos. Relativamente à LAID, foi realizado um primeiro teste sobre a hipótese de existirem diferenças significativas nos provérbios usados em cada ilha açoriana, o que possibilita identificar a ilha de origem de um indivíduo a partir do seu conhecimento de provérbios (CAVIQUE *et al.*, 2013) e um segundo trabalho com recurso a conjuntos de dados gerados artificialmente (CAVIQUE *et al.*, 2018).

Perante estes factos, é importante colmatar esta lacuna e disponibilizar um contributo para a validação e avaliação do método de seleção LAID, aplicado a conjuntos de dados de grande dimensão, com avaliação dos modelos obtidos, de forma a consolidar processos na descoberta do conhecimento em *Data Mining*, para que futuros trabalhos na área possam recorrer a estas técnicas de uma forma consolidada. O trabalho de dissertação pretende dar mais um passo na direção deste objetivo.

Outra vertente deste trabalho é testar o novo paradigma do tratamento de grandes volumes de dados, caracterizado pelo acesso aos dados em disco, num ambiente *cloud*. Para isso recorreu-se à infraestrutura INCD, para realização dos testes computacionais, com os algoritmos codificados em *Python* e os dados armazenados no formato HDF5.

1.2. Objetivos

O presente trabalho pretende fornecer um enquadramento teórico para o algoritmo LAID e a aplicação deste método de seleção de atributos a conjuntos de dados de grande dimensão, com acesso aos mesmos a partir do disco, no formato HDF5, num ambiente *cloud*. A realização de testes ao algoritmo em estudo tem por objetivo a sua sustentação, validação e avaliação, para a consolidação do mesmo. A avaliação não se restringe ao tempo de processamento, mas, também, tem em conta os resultados obtidos quando o modelo é

aplicado a outros conjuntos de dados que servirão de teste. Um aspeto importante é a validação da plataforma HDF5+*Python*, em ambiente *cloud*. Neste caso, os tempos são o principal critério.

Outro objetivo consiste em contribuir para um aprofundamento de toda a teoria subjacente ao LAID, nomeadamente, na descrição detalhada dos modelos *Rough Sets* e LAD.

O *software* desenvolvido para implementação e teste do algoritmo LAID foi codificado em *Python* e está disponível na *Internet*, no sítio http://infapol.com/jda , para análise e futuros desenvolvimentos. No mesmo sítio, está disponível a versão digital deste trabalho.

1.3. Metodologia

O trabalho de pesquisa teórica foi baseado em consultas bibliográficas e digitais, como sejam artigos, livros, teses e sítios de referência na *Internet*.

Foi desenvolvida uma implementação do algoritmo LAID para realização de testes, com recurso a conjuntos de dados de grande dimensão, gerados artificialmente, para possibilitar a alteração dos mesmos de forma a validar determinados parâmetros, como sejam, o tempo de processamento, número de atributos selecionados, dimensão dos ficheiros e avaliação dos resultados. Tudo isto, em função de diferentes dimensões dos conjuntos de dados usados.

Os conjuntos de dados foram guardados no formato HDF5 e o processamento foi realizado nas infraestruturas INCD, tendo em conta o novo paradigma de ambientes de desenvolvimento, onde o acesso aos dados em memória é substituído pelo acesso em disco e o processamento é realizado em ambiente *cloud*, com computadores de elevada performance.

1.4. Estrutura do documento

O capítulo 2 descreve o estado da arte relativamente à seleção de atributos. É feita uma apresentação teórica do assunto e são abordados os principais algoritmos.

O algoritmo LAID, objeto principal desta dissertação e os algoritmos *Rough Sets* e LAD, metodologias de suporte, são detalhadamente descritos no capítulo 3. A explanação do LAID é acompanhada por um exemplo demonstrativo. O capítulo termina com os critérios de validação.

A fase de implementação do algoritmo LAID é descortinada no capítulo 4. São explicadas as opções tomadas relativamente ao sistema de base de dados (HDF5) e à linguagem de programação (*Python*). Faz-se a apresentação da infraestrutura INCD, onde foram realizados os testes computacionais. O capítulo termina com a descrição e análise do pseudocódigo, ponto de partida no processo de codificação.

Os resultados obtidos no processo de seleção e de avaliação são analisados no capítulo 5. São descritas as características dos conjuntos de dados artificiais usados no processo.

Por fim, as conclusões estão no capítulo 6, onde se indicam os trabalhos futuros que se julgam necessários para complementar o âmbito desta dissertação.

2. Seleção de atributos

À medida que se dissemina o uso das TI no mundo empresarial, nas instituições oficiais e no dia a dia das pessoas, aumenta, de forma exponencial, a disponibilização de informação em todas as áreas do conhecimento humano. Apesar da grande evolução registada neste século, ao nível dos equipamentos informáticos, não tem sido suficiente para acompanhar as exigências de processamento dessa mesma informação digital. Para lidar com tamanho volume de dados é necessário a sua filtragem, através de algoritmos de pré-processamento.

A seleção de atributos é uma das principais técnicas de pré-tratamento de grandes volumes de dados. Segundo Kumar e Minz (2014), a seleção de atributos é um processo que permite detetar os atributos relevantes num conjunto de dados e eliminar os irrelevantes, redundantes ou causadores de ruído (onde o valor original foi modificado, por exemplo, no caso de entrevistas, a existência de falsos testemunhos). Esta seleção acelera os algoritmos de *Data Mining*, aumenta a precisão na realização de previsões e melhora a compreensão dos processos. O problema agrava-se quando, no conjunto dos dados, o número de atributos é muito maior que as observações realizadas, originando um espaço de pesquisa mais esparso e dificultando a diferenciação do que é relevante por contraponto com o que é ruído (KUMAR, MINZ, 2014). Uma boa seleção de atributos permite facilitar o processo de aprendizagem e compreensão do domínio em estudo, obter melhores previsões, simplificar modelos e reduzir custos.

Nos últimos anos, a seleção de atributos tem sido objeto de muitos estudos e artigos publicados. Tem sido aplicada em muitas áreas da atividade humana como na análise do ADN, deteção de perturbações, classificação de textos, ou músicas, tratamento de imagens, etc. (BOLÓN-CANEDO *et al.*, 2013). Por consequência, surgiram muitos algoritmos de seleção, sendo difícil estabelecer um critério de avaliação dos mesmos. Entre os autores, é unânime concluir que não existe um método que possa ser considerado "o melhor", dependendo de muitas variáveis do domínio em estudo.

2.1. Etapas do processo de seleção de atributos

A seleção de atributos pode ser realizada segundo duas filosofias distintas (GUYON, ELISSEEFF, 2003):

- Avaliação individual dos atributos, onde a cada atributo é atribuído um peso de acordo com a sua relevância;
- Avaliação de um conjunto de atributos selecionados obtidos iterativamente segundo um critério definido.

A avaliação individual não consegue remover atributos redundantes, porque tenderão a ter pesos semelhantes, ao contrário da avaliação de subconjuntos de atributos, que pode lidar com redundâncias, mas tem problemas inerentes à procura dos subconjuntos (BOLÓN-CANEDO *et al.*, 2013).

Na opinião de Dash e Liu (1997), o procedimento típico de seleção de atributos é composto por quatro etapas (ver Figura 2.1):

1. Procedimento heurístico de procura para selecionar um subconjunto de atributos, muito influenciado pelo método de escolha dos atributos e pelo ponto de partida;
2. Avaliação do subconjunto de atributos gerado, por comparação com o subconjunto previamente obtido, que poderá ser independente de algoritmos de mineração ou ser dependente destes e recorrer a dados de treino;
3. O processo é repetido até que se cumpra determinado critério de paragem definido, que poderá depender do número de iterações, do número de atributos escolhidos, se a alteração de algum atributo já não acrescenta valor, ou depender de alguma função de decisão (que poderá substituir a etapa de avaliação);
4. Validação do subconjunto obtido, através de vários testes, por comparação com resultados previamente obtidos ou comparando com os resultados de métodos alternativos.

A seleção de um subconjunto de atributos não é tarefa fácil, a começar pelo facto de, numa base de dados com n atributos, existirem 2^n-1 subconjuntos diferentes não vazios (${}^nC_1+{}^nC_2+\ldots+{}^nC_n$). Pelo facto de o problema de seleção ter um crescimento exponencial com o

número de atributos, obriga ao uso de estratégias, que poderão ser exaustivas, sequenciais ou aleatórias (DOAK, 1992).

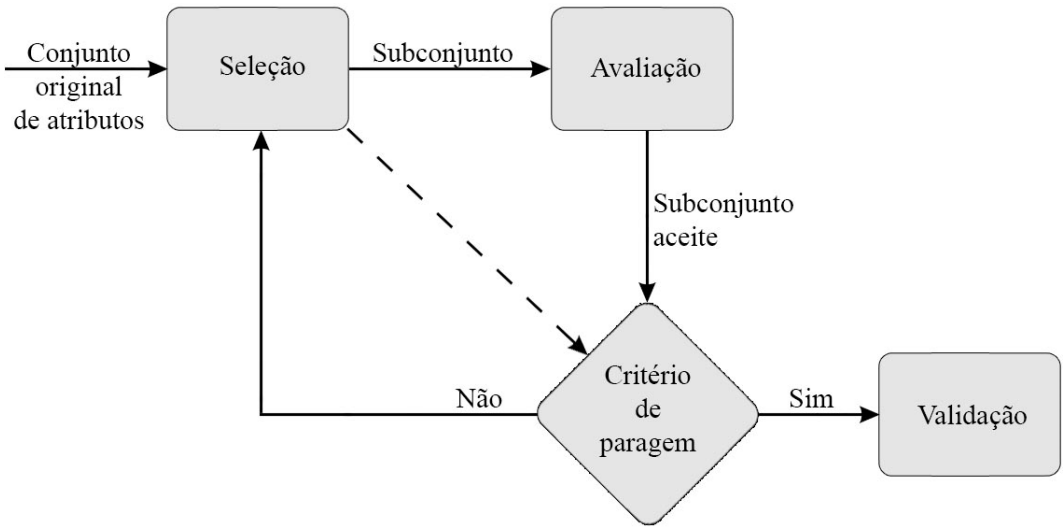

Figura 2.1 – Processo de seleção de atributos com validação.
(Adaptado de: DASH, LIU, 1997)

A pesquisa exaustiva garante a solução ótima. O algoritmo *Exhaustive Search* examina todas as possibilidades no espaço de pesquisa, mas existem outros que, efetuam a pesquisa exaustiva numa determinada vizinhança, ou dentro de certos limites, como são o caso dos algoritmos *Branch and Bound*, *Approximate Monotonicity with Branch and Bound* e *Beam Search* (DOAK, 1992).

Na pesquisa sequencial, são selecionados um ou mais atributos, num processo iterativo, onde o número máximo de iterações é $O(n)$. A pesquisa sequencial é completa, fácil de implementar, mas pode não obter o subconjunto ótimo e o espaço de pesquisa é $O(n^2)$ (KUMAR, MINZ, 2014). Pode ter muitas variantes: a *Forward Sequential Selection* começa com o subconjunto vazio e adiciona um atributo de cada vez; a *Backward Sequential Selection* começa com o conjunto total de atributos e remove um atributo de cada vez; a *PQ Sequential Selection* adiciona atributos numa iteração e remove na seguinte; a *Bi-Directional Search* efetua duas pesquisas em simultâneo, por iteração, uma para adicionar um atributo, outra para retirar (DOAK, 1992).

A pesquisa aleatória parte da um subconjunto com atributos escolhidos aleatoriamente. Durante as várias iterações, a seleção pode ser completamente aleatória, ou sequencial, mas incluindo alguma aleatoriedade na escolha (KUMAR, MINZ, 2014). A aleatoriedade tem como objetivo conseguir-se escapar a mínimos locais do espaço de pesquisa (DOAK, 1992). A ordem do espaço de pesquisa é $O(n^2)$ (KUMAR, MINZ, 2014)

No processo iterativo, após obter um subconjunto, há que avaliá-lo, para verificar se é melhor que o melhor subconjunto até então encontrado. Aqui, é essencial definir um critério de avaliação. Há que ter em conta que, um subconjunto pode preencher os requisitos de um critério e não os de outro (KUMAR, MINZ, 2014). Esta não é uma questão fácil, nomeadamente, se estivermos a lidar com bases de dados de grande dimensão, onde o número de observações é muito menor que o número de atributos (KOHAVI, JOHN, 1997), porque, neste caso, temos, relativamente, menos observações para poder avaliar a seleção obtida.

Existem, dois tipos de critérios, baseados na sua dependência, ou não, dos algoritmos de aprendizagem (KUMAR, MINZ, 2014).

Os critérios independentes não têm em conta o algoritmo de aprendizagem, recorrendo a modelos de filtragem e aos dados de treino para avaliar os subconjuntos. Medidas de distância, medidas de informação ou incerteza, medidas de probabilidade de erro, medidas de dependência, medidas de distância interclasse e medidas de consistência são alguns, dos muitos, critérios independentes existentes (KUMAR, MINZ, 2014).

Na aplicação de critérios dependentes é necessário ter o algoritmo de aprendizagem já determinado. A avaliação dos subconjuntos de atributos é feita recorrendo aos processos do algoritmo, o que aumenta a performance deste. No entanto, é computacionalmente dispendioso, porque é efetuada uma estimativa da precisão para cada subconjunto de atributos (KUMAR, MINZ, 2014).

Por fim, o critério de paragem pode corresponder a um, ou a uma combinação, dos seguintes casos (KUMAR, MINZ, 2014):

- Limitar o número de iterações;
- Definir um número mínimo de atributos selecionados;
- Definir uma taxa mínima de erro de classificação;

- Completar a pesquisa;
- Verificar se retirar, ou acrescentar, novos atributos ao subconjunto já não origina diferenças significativas.

2.2. Diferentes abordagens à seleção de atributos

Bólon-Canedo, Sánchez-Maroño e Alonso-Betanzos (2013), Kumar e Minz (2014) indicam a existência de três diferentes abordagens à seleção de atributos: *Filter*; *Wrapper*; *Embedded*.

2.2.1. Abordagem *Filter*

Os métodos que seguem esta abordagem fazem a seleção dos atributos com base na sua importância para a previsão da variável em estudo (classe), independentemente do algoritmo de aprendizagem. Estes métodos selecionam os atributos mais relevantes para determinar o valor da classe. São rápidos, eficientes e permitem boas previsões (BOLÓN-CANEDO *et al.*, 2013). O problema desta abordagem é poder selecionar atributos redundantes e não selecionar atributos que, não tendo importância por si, são importantes quando combinados com outros (KUMAR, MINZ, 2014). São frequentemente usados para pré-processamento dos dados (BOLÓN-CANEDO *et al.*, 2013). *FOCUS, ABB, Relief* (KUMAR, MINZ, 2014), *CFS, INTERACT, Consistency-based Filter, InfoGain, ReliefF, mRMR* e M_d (BOLÓN-CANEDO *et al.*, 2013) são exemplos de métodos que usam esta abordagem.

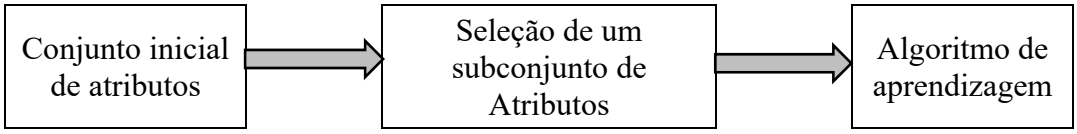

Figura 2.2 – Abordagem Filter.
(Fonte: JOHN et al., 1994)

A abordagem *Filter* tem uma boa performance, porque a filtragem redunda numa única solução, sendo mais intuitiva. No entanto, porque não tem em conta o algoritmo de aprendizagem, muitos dos atributos selecionados poderão não otimizar o modelo de previsão e os critérios de seleção são difíceis de determinar. (CAVIQUE *et al.*, 2013)

FOCUS e *Relief* são os métodos mais conhecidos que recorrem à abordagem *Filter*. *FOCUS* faz uso da existência de inconsistências, procurando grupos de atributos que minimizam a quantidade destes conflitos. Já o método *Relief* faz a avaliação de um conjunto de atributos através de um processo de avaliação baseado na diferença entre a distância das observações da mesma classe e de classe diferente, que sejam mais próximas. (CAVIQUE *et al.*, 2013)

2.2.2. Abordagem *Wrapper*

A grande diferença entre as abordagens *Wrapper* e *Filter* reside no critério de avaliação. Os métodos que seguem a abordagem *Wrapper* realizam a seleção de subconjuntos de atributos durante o processo de treino, através de um algoritmo de aprendizagem específico, ao qual recorrem para proceder à avaliação (KUMAR, MINZ, 2014), com maiores custos de processamento, logo mais lentos (BOLÓN-CANEDO *et al.*, 2013). Se o número de observações for insuficiente, aumenta o risco de *overfitting* (KUMAR, MINZ, 2014), ou seja, o modelo obtido encaixar perfeitamente nos dados de treino, mas a precisão diminuir bastante quando aplicado noutros grupos de dados. Consegue capturar as dependências entre atributos (BOLÓN-CANEDO *et al.*, 2013). Exemplos de métodos: *Wrapper-C4.5* e *Wrapper SVM* (BOLÓN-CANEDO *et al.*, 2013).

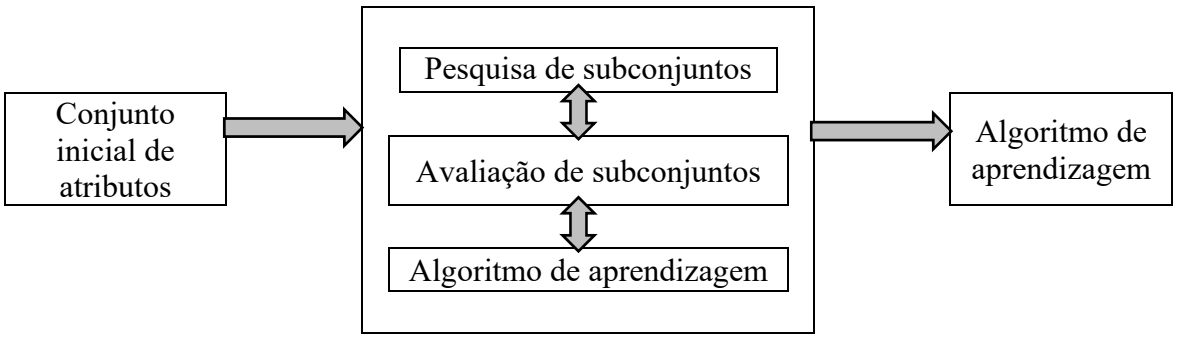

Figura 2.3 – Abordagem Wrapper.
(Fonte: JOHN et al., 1994)

Neste caso, como os critérios de seleção são determinados recorrendo ao algoritmo de aprendizagem, eles são fáceis de estimar. Mas, pela mesma razão, são computacionalmente mais pesados e pouco intuitivos. Também, não identificam dependências estatísticas entre atributos, podendo, o subconjunto determinado, não ser o que melhor explica o modelo, destruindo a semântica subjacente ao sistema. (CAVIQUE *et al.*, 2013)

2.2.3. Abordagem *Embedded*

A abordagem *Embedded* tenta combinar as vantagens das abordagens anteriores. Procura o subconjunto ótimo de acordo com o algoritmo de aprendizagem durante o processo de treino. Tem baixo custo computacional e consegue detetar dependências entre atributos (BOLÓN-CANEDO *et al.*, 2013). O algoritmo faz uso do seu próprio processo de seleção para realizar, simultaneamente, a seleção de atributos e a avaliação, ou seja, usa uma métrica própria durante a seleção de atributos (KUMAR, MINZ, 2014). *FS-Percepton* e *SVM-RFE* (BOLÓN-CANEDO *et al.*, 2013) são métodos que aplicam esta abordagem.

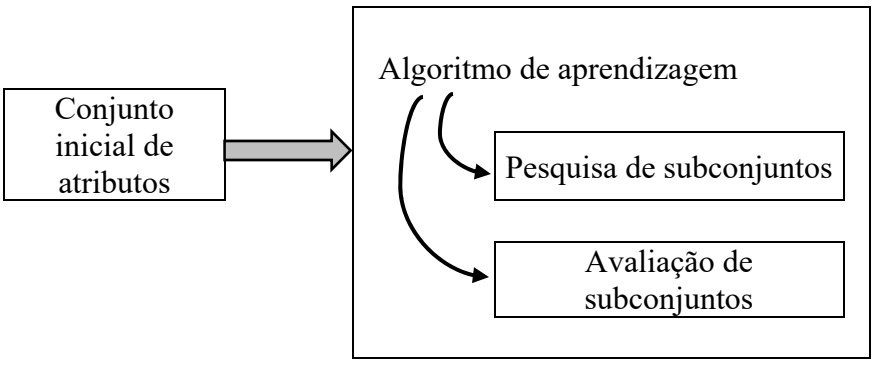

Figura 2.4 – Abordagem Embedded.
(Adaptado de: BOLÓN-CANEDO et al., 2013)

2.3. Relevância dos atributos

O ideal, no processo de seleção, para obter um subconjunto de atributos, seria que todos os atributos escolhidos fossem relevantes. Para que isto aconteça, há que definir o conceito de atributo relevante.

Kumar e Minz (2014) compilaram seis definições de relevância, de acordo com ser relevante em relação ao quê:

- Relevante para com o objetivo – um atributo x_i é relevante para um objetivo C (função que associa a cada observação a respetiva classe) se existirem duas observações (A e B) que difiram apenas no valor de x_i e $C(A) \neq C(B)$;
- Forte relevância para com os dados – um atributo x_i é fortemente relevante para com um conjunto de dados do sistema (dados de um subconjunto de atributos), se existirem duas observações (A e B) que difiram apenas no valor de x_i , relativamente ao conjunto de dados considerado, e $C(A) \neq C(B)$;
- Fraca relevância para com os dados – um atributo x_i é fracamente relevante para com os dados se existir pelo menos um subconjunto de atributos que contenha x_i, onde, nesse subconjunto, x_i não é fortemente relevante para com os dados;
- Relevância sendo uma medida de complexidade – numa amostra de dados e tendo um conjunto de objetivos C, a medida corresponde ao número de atributos relevantes, considerando a primeira definição, relativamente a um objetivo de C cujo erro sobre S seja menor e tenha menos atributos relevantes;
- Utilidade incremental – dada uma amostra de dados, um algoritmo de pesquisa e um subconjunto de atributos X', um atributo x_i é incrementalmente útil para o algoritmo, relativamente a X', se a precisão da hipótese que o algoritmo produz, usando o conjunto de atributos $\{x_i\} \cup X'$ é melhor do que a precisão alcançada apenas por X'.
- Relevância da entropia – a entropia de x para y é definida por $I(x,y) \,/\, H(y)$, onde $I(x,y) = H(x) - H(x \mid y)$ e H é a função entropia de um atributo.

A relevância de um atributo é qualificada em três níveis (KOHAVI, JOHN, 1997):

- Fortemente relevante;
- Fracamente relevante;
- Irrelevante.

Um atributo é fortemente relevante se não puder ser retirado do subconjunto de atributos sem que diminua a avaliação do subconjunto. Será fracamente relevante se, por vezes, contribuir para aumentar a avaliação do subconjunto. Um atributo é irrelevante se não for fortemente nem fracamente relevante (KOHAVI, JOHN, 1997).

2.4. Algoritmos de seleção de atributos

Têm sido propostos muitos algoritmos de seleção de atributos. A eficácia dos mesmos é difícil de determinar sem saber, de antemão, quais os atributos relevantes (BOLÓN-CANEDO *et al.*, 2013). Existem muitas medidas de performance definidas na literatura, nomeadamente, medidas de precisão, cálculo de rácios, recursos computacionais, etc. (KUMAR, MINZ, 2014). Além disso, o volume de dados pode ser enorme, existir uma grande quantidade de atributos irrelevantes ou redundantes e, ainda, uma grande quantidade de observações, ou registos, inconsistentes. Na literatura encontram-se muitos estudos comparativos de métodos de seleção.

Em consequência do acima exposto, têm surgido muitos novos algoritmos de seleção de atributos, com diferentes estratégias (KUMAR, MINZ, 2014):

- Recorrer à aplicação de vários métodos de seleção para obter melhores resultados;
- Combinar a seleção com outras técnicas, como a extração de atributos e técnicas de árvore;
- Melhorar algoritmos existentes;
- Criar novos métodos para responder a problemas que não se enquadrem em nenhum dos métodos existentes;
- Combinar dois ou mais métodos de seleção, de forma a obter o melhor de cada um.

Tabela 2.1 – Algoritmos de seleção de atributos.

			Estratégias de seleção		
			Exaustiva	Sequencial	Aleatória
Abordagens	*Filter*	Distância	B&B, BFF, BOBRO, OBLIVIIN	Relief, RelifS, SFS, Segen's, SBS	
		Informação	MDLM, CARDIE	DTM, Koller's, FG, FCBF, BSE, Dash's, SBUD	
		Dependência	Bobrowski's	CFS, RRESET, POE+ACC, DVMM, Mitra's	
		Consistência	FOCUS, ABB, MIFESI, Schlimmer's	*Rough Sets*, LAD, LAID, *Set Cover*, WINNOW	LIV, QBB, LVF
	Wrapper		BS, AMB&B, FSLC, FSBC, CARDIE, OBLIVIIN	SBS-SLASH, WSFG, WSBG, BDS, PQSS, RC, SS, Queiros's, BSE, K2-AS, RACE, SBS-W, SBS-SLASH, AICC, FSSEM, ELSA	SA, RGSS, LVW, RMHC-PF, GA, RVE
	Embedded			BBHFS, Xing's, Dash-Liu's	

(Adaptado de: KUMAR, MINZ, 2014)

3. LAID e as metodologias subjacentes

A Análise Lógica de Dados Inconsistentes (LAID) surge da combinação de duas metodologias *Filter* desenvolvidas nos finais do século XX: a Teoria dos *Rough Sets* e a LAD. A LAID herda muitas das qualidades de cada uma destas teorias, sendo capaz de lidar com dados inconsistentes e com classes não dicotomizadas, características dos *Rough Sets*, bem como, tem a eficácia computacional da LAD. (CAVIQUE *et al.*, 2011)

Neste capítulo far-se-á a descrição teórica de cada uma destas três metodologias, sendo a explanação da LAID complementada com um exemplo simples, de fácil compreensão.

3.1. Rough Sets

A teoria dos *Rough Sets* é uma abordagem matemática, proposta por Zdzislaw Pawlak, em 1982, para análise de dados inconsistentes (ZHANG *et al.*, 2016). Esta abordagem tem obtido bons resultados em muitos domínios, nomeadamente, na seleção de atributos, a encontrar dependências entre atributos e na identificação de padrões num conjunto de dados. Uma das grandes aplicações dos *Rough Sets* tem sido na área da inteligência artificial, nomeadamente, na extração de conhecimento, tratamento de dados, suporte a decisões e reconhecimento de padrões (PAWLAK, 2003).

Pelas suas características, os *Rough Sets* têm tido uma atenção crescente por parte da comunidade científica, corroborada pela grande quantidade de artigos e outras publicações (PAWLAK, 2003), em especial, nas áreas de *Data Mining*, reconhecimento de padrões e classificação, na indústria energética, processamento de imagens e outros dados de grande dimensão, desenvolvimento de sistemas inteligentes, medicina, gestão, engenharia e muitas outras (ZHANG *et al.*, 2016).

A teoria dos *Rough Sets* introduziu novos conceitos e notações. Por ser relativamente nova e por ter despertado o interesse de um grande número de investigadores, estes novos conceitos não estão estandardizados (SASSI, 2006), pelo que convém clarificá-los.

3.1.1. Indiscernibilidade e conjuntos elementares

Um conceito fundamental na teoria dos *Rough Sets* é o de indiscernibilidade (PAWLAK, 2002). A indiscernibilidade manifesta-se quando, num conjunto, existem elementos que não se conseguem distinguir, parecendo ser repetições do mesmo elemento (PAWLAK, 2002). Por exemplo, se duas observações tiverem os mesmos valores num conjunto de atributos, elas dizem-se indiscerníveis (PILA, MONARD, 2001). No entanto, poderão divergir se forem acrescentados novos atributos.

Um espaço aproximado é composto por um conjunto universo (U), finito, não vazio, e por uma operação de equivalência (R) entre dois elementos de U, que goza das propriedades reflexiva (xRx), simétrica (se xRy então yRx) e transitiva (se xRy e yRz então xRz) (PILA, MONARD, 2001). Dados dois elementos de U, x e y, se xRy então os dois elementos são indiscerníveis no espaço aproximado. Algumas extensões da teoria dos *Rough Sets* não exigem que a relação de equivalência goze da propriedade transitiva, sendo chamada relação de tolerância ou similaridade (PAWLAK, 2003).

A classe de equivalência de um elemento $x \in U$, denotada por $[x]_R$, é o conjunto de todos os elementos $y \in U$, tais que xRy. Conjuntos elementares de U são conjuntos cujos elementos são indiscerníveis, logo correspondem às classes de equivalência (SASSI, 2006).

Na seleção de atributos, U corresponde ao conjunto de observações ou registos de uma base de dados. O conjunto de atributos designa-se por A. Cada atributo tem um conjunto de valores possíveis, que poderão ser binários, numéricos ou literais. Duas observações são indiscerníveis se tiverem o mesmo valor para cada atributo. Ao conjunto das observações, com os atributos e respetivos valores, dá-se o nome de sistema de informação (PAWLAK, 2003).

A Relação de Indiscernibilidade associada ao conjunto de atributos A, $IND(A)$, é definida pela operação de equivalência (PAWLAK, 2003):

$$IND(A) = \{x,y \in U: \forall a \in A, a(x) = a(y)\} \tag{3.1}$$

Se x e y verificam a relação $IND(A)$, então são indiscerníveis, ou indistinguíveis, tendo em conta os atributos A (PAWLAK, 2003). A notação $[x]_{IND(A)}$ representa o conjunto elementar ao qual uma observação x pertence, isto é, a classe de equivalência de x.

Um conjunto elementar de um sistema de informação é um subconjunto de U que contém todos os elementos de U que verificam a Relação de Indiscernibilidade associada aos atributos A (PAWLAK, 2002). Todos os conjuntos elementares de um sistema de informação, U/A, formam uma partição de U. Qualquer união finita de conjuntos elementares designa-se por conjunto definível num sistema de informação.

Em muitos casos, a cada observação de um sistema de informação está associada uma classificação de resposta designada por classe (C). A classe poderá ser considerada como um atributo de decisão. Um sistema de informação com uma classe associada designa-se por Sistema de Decisão (PAWLAK, 2003). Cada linha do sistema de decisão representa uma regra que, dada a satisfação de determinadas condições (valores dos respetivos atributos) é obtido um resultado (valor da classe) (PAWLAK, 2002).

Duas observações com os mesmos valores para cada atributo e para a classe dizem-se redundantes:

$$x \text{ e } y \text{ são redundantes sse } (\forall a \in A,\ a(x) = a(y)) \wedge c(x) = c(y) \tag{3.2}$$

Observações inconsistentes têm os mesmos valores para cada atributo, mas classes com valores diferentes:

$$x \text{ e } y \text{ são inconsistentes sse } (\forall a \in A,\ a(x) = a(y)) \wedge c(x) \neq c(y) \tag{3.3}$$

As observações redundantes, para determinado sistema de decisão, podem ser eliminadas, ficando apenas uma delas.

3.1.2. Conjuntos aproximados

Quando um sistema de decisão apresenta inconsistências, a teoria dos *Rough Sets* consegue dar resposta e realizar a seleção de atributos (SASSI, 2006).

Num sistema de decisão, seja X um conjunto de observações de U com determinado(s) valor(es) da classe, $C_1 \subseteq C$:

$$X = \{x \in U: c(x) \in C_1\} \tag{3.4}$$

A partir de um determinado conjunto X, podem-se definir outros dois conjuntos (PAWLAK, 2002)(PAWLAK, 2003):

- Aproximação inferior $\underline{A}(X)$ – maior conjunto definível contido em X – conjunto das observações com valores dos atributos iguais aos dos elementos de $\underline{A}(X)$ terão, seguramente, valor de classe pertencente a C_1, isto é, pertencerão a X;

- Aproximação superior $\bar{A}(X)$ – menor conjunto definível que contém X – conjunto das observações com valores dos atributos iguais aos dos elementos de $\bar{A}(X)$ poderão ter valor de classe pertencente a C_1, isto é, serão possíveis elementos de X.

Usando a notação $[x]_{IND(A)}$ para representar o conjunto elementar ao qual uma observação x pertence, vem (PAWLAK, 2003):

$$\bar{A}(X) = \{x \in U: [x]_{IND(A)} \cap X \neq \varnothing\} \tag{3.5}$$

$$\underline{A}(X) = \{x \in U: [x]_{IND(A)} \subseteq X\} \tag{3.6}$$

Região negativa de X, $NEG(X)$, é o conjunto de todas as observações de U que não pertencem a $\bar{A}(X)$. Elas terão, seguramente, valor de classe que não pertencente a C_1, isto é, não pertencerão a X (PAWLAK, 2003).

Região fronteira de X, $FR(X)$, é o conjunto de todas as observações de U que pertencem a $\bar{A}(X)$ e não pertencem a $\underline{A}(X)$. A região fronteira corresponde ao resultado da operação de conjuntos $\bar{A}(X) - \underline{A}(X)$ (ZHANG *et al.*, 2016). Observações com valores de atributos iguais a um dos elementos de $FR(X)$ não poderão ser classificadas, com absoluta certeza, como pertencentes a X, nem como não pertencentes a X (PAWLAK, 2003).

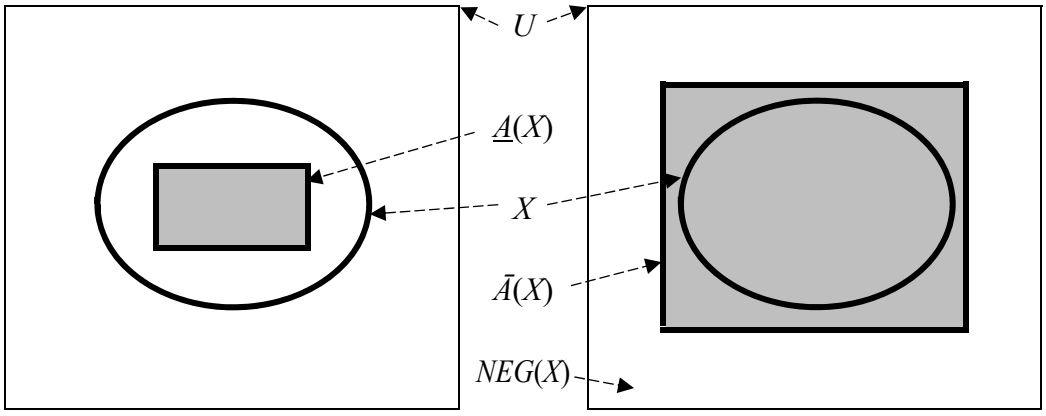

Figura 3.1 – Esquema ilustrativo da Aproximação Inferior e Superior de X.
(Adaptado de: PAWLAK, 2003)

Um conjunto X é impreciso ou aproximado (*rough*) se a sua região fronteira não for vazia, caso contrário, é preciso (*crisp*) (PAWLAK, 2002).

Dependendo das respetivas aproximações inferior e superior, podem-se definir quatro tipos de incerteza relativamente a um conjunto X (PAWLAK, 2003):

- X é grosseiramente A-definível se e só se $\underline{A}(X) \neq \varnothing \wedge \bar{A}(X) \neq U$. Neste caso, para determinado sistema de decisão, existem observações de U que pertencem, inequivocamente, a X e existem observações de U que não pertencem, inequivocamente, a X.

- X é internamente A-indefinível se e só se $\underline{A}(X) = \varnothing \wedge \bar{A}(X) \neq U$. Neste caso, para determinado sistema de decisão, existem observações de U que não pertencem, inequivocamente, a X, mas já não se pode afirmar que existam observações de U que pertençam, inequivocamente, a X.

- X é externamente A-indefinível se e só se $\underline{A}(X) \neq \varnothing \wedge \bar{A}(X) = U$. Neste caso, para determinado sistema de decisão, existem observações de U que pertencem, inequivocamente, a X, mas já não se pode afirmar que existam observações de U que não pertençam, inequivocamente, a X.

- X é totalmente A-indefinível se e só se $\underline{A}(X) = \varnothing \wedge \bar{A}(X) = U$. Neste caso, para determinado sistema de decisão, não existem observações de U que pertençam, inequivocamente, a X, nem que não pertençam, inequivocamente, a X.

Existem três medidas que permitem quantificar a qualidade das aproximações de X (PAWLAK, 2003)(RISSINO, LAMBERT-TORRES, 2009):

- Coeficiente de precisão da aproximação, $\alpha_A(X) = \frac{\#\underline{A}(X)}{\#\overline{A}(X)}$, (3.7)

  mede o grau de precisão de X. Como $\#\underline{A}(X) \leq \#\bar{A}(X)$, vem $0 \leq \alpha_A(X) \leq 1$. Se $\alpha_A(X) = 1$, X é preciso, caso contrário é aproximado. O grau de imprecisão é dado por $1 - \alpha_A(X)$.

- Coeficiente de precisão da aproximação inferior, $\alpha_A(\underline{A}(X)) = \frac{\#\underline{A}(X)}{\#U}$. (3.8)

  Como $\#\underline{A}(X) \leq \#U$, vem $0 \leq \alpha_A(\underline{A}(X)) \leq 1$. Se $\alpha_A(\underline{A}(X)) = 1$, todas as observações de U pertencem, inequivocamente, a X. Se $\alpha_A(\underline{A}(X)) = 0$, nenhuma observação de U pertence, inequivocamente a X.

- Coeficiente de precisão da aproximação superior, $\alpha_A(\bar{A}(X)) = \frac{\#\overline{A}(X)}{\#U}$. (3.9)

  Como $\#\bar{A}(X) \leq \#U$, vem $0 \leq \alpha_A(\bar{A}(X)) \leq 1$. Se $\alpha_A(\bar{A}(X)) = 1$, nenhuma observação de U não pertence, inequivocamente a X. Se $\alpha_A(\bar{A}(X)) = 0$, todas as observações de U não pertencem, inequivocamente, a X.

3.1.3. Grau de relevância dos atributos

Para determinar o grau de relevância de um atributo, há que, primeiro, definir a dependência entre a classe e os atributos.

Sejam A_1 e A_2 dois subconjuntos de atributos de A ($A_1 \subseteq A$ e $A_2 \subseteq A$), A_1 depende totalmente de A_2 se todos os valores dos atributos de A_1 forem univocamente determinados pelos valores dos atributos de A_2 (PAWLAK, 2003).

A dependência de um conjunto de valores da classe (C), relativamente ao conjunto de atributos (A), pode ser quantificada recorrendo ao respetivo coeficiente de qualidade da

aproximação inferior 3.8 (PAWLAK, 2003), onde U/C denota a partição de U segundo os valores de C:

$$\gamma(A,C) = \frac{\#\left(\cup_{X \in U/C}(\underline{A}(X))\right)}{\#U} = \sum_{X \in U/C} \frac{\#\underline{A}(X)}{\#U} \tag{3.10}$$

A dependência da classe, em relação aos atributos, pode ser classificada segundo o valor de $\gamma(A,C)$ (PAWLAK, 2003):

- Se $\gamma(A,C) = 1$, há total dependência desses valores da classe em relação ao conjunto de atributos, ou seja, esses valores da classe são determinados, inequivocamente, pelos valores dos atributos.

- Se $0 < \gamma(A,C) < 1$, existe dependência parcial dos valores da classe em relação ao conjunto de atributos. Neste caso, indica que existem atributos que podem ser retirados de A, sem afetar a correspondência entre os valores dos atributos e os, respetivos, valores da classe (seleção de atributos).

- Se $\gamma(A,C) = 0$, os valores da classe são independentes do conjunto de atributos.

Outro conceito importante para a seleção de atributos é a importância que cada um tem para a tomada de decisão (obter o valor da classe). Neste contexto, surge a definição de significância de um atributo (a), determinada a partir do grau de dependência de C em relação a A, por Sassi (2006):

$$\sigma_C(a) = 1 - \frac{\gamma(A-\{a\},C)}{\gamma(A,C)} \tag{3.11}$$

A significância de um atributo corresponderá ao erro que se comete na dependência de C em relação a A, quando o atributo é removido de A (SASSI, 2006). Um atributo é dispensável se a sua significância for nula (PILA, MONARD, 2001).

3.1.4. Seleção de atributos

Na seleção de atributos, um conjunto de atributos que permita extrair o mesmo conhecimento de uma base de dados que a totalidade dos mesmos, diz-se um reduto (PAWLAK, 2003). Seja $A^* \subseteq A$, A^* é um reduto de A, se todos os atributos de $A–A^*$ forem dispensáveis e $\forall x \in U$, $[x]_{IND(A^*)} = [x]_{IND(A)}$. O conjunto de todos os redutos de A é representado por $RED(A)$ (PILA, MONARD, 2001).

No processo de remoção de um atributo há que determinar os conjuntos elementares de cada atributo. Se os conjuntos elementares de um atributo forem iguais aos do conjunto total de atributos (A), então esse atributo será um reduto de A, ou seja, todos os outros atributos poderão ser retirados. Caso não haja nenhum atributo nestas condições, determinam-se os conjuntos elementares para todos os subconjuntos de dois atributos, depois para subconjuntos de três, e assim sucessivamente, até encontrar um, ou vários, subconjuntos de A cujos conjuntos elementares sejam iguais aos de A, sendo, esse subconjunto, um reduto de A. Se para nenhum subconjunto de A, exceto o próprio A, se obtém um reduto, isto significa que o conjunto de atributos não pode ser reduzido (PAWLAK, 2003).

3.1.5. Função de discernibilidade

O processo de seleção de atributos, visto atrás, tem um grande peso computacional, nomeadamente, no caso de sistemas de informação com milhares de atributos. A função de discernibilidade permite realizar a seleção de atributos de uma forma computacionalmente mais eficaz (PAWLAK, 2003). Para tal, há que construir a matriz de discernibilidade.

A matriz de discernibilidade, M, é uma matriz simétrica, de dimensão $n \times n$, onde n é o número de observações do sistema de informação ($\#U$) e cada elemento de M, $m(i,j)$, é obtido para um par de observações e corresponde ao conjunto de atributos de A cujos valores diferem nas duas observações (PAWLAK, 2003). Formalmente, será (PAWLAK, 2003):

$$m(i,j) = \{a \in A: a(o(i)) \neq a(o(j))\} \tag{3.12}$$

Determinada a matriz M, há que calcular a função de discernibilidade F, uma função booleana composta pela operação de conjunção dos elementos que correspondem à

disjunção dos atributos constantes em cada célula da matriz M (PAWLAK, 2003) (PILA, MONARD, 2001).

$$F(A) = \wedge[\ \vee[a(k) \in m(i,j)]] \tag{3.13}$$

Onde $1 \leq k \leq \#A$ e $1 \leq j < i \leq \#U$.

Aplicando, à expressão obtida para $F(A)$, as propriedades da álgebra booleana, obtém-se a expressão simplificada de F, que indica a redução, ou reduções, de A.

Na versão alargada deste trabalho, disponível no sítio da *Internet* http://infapol.com/jda , esta secção é acompanhada por um exemplo esclarecedor.

3.1.6. Características dos *Rough Sets*

Desde que foi proposta, a teoria dos *Rough Sets* tem tido um grande desenvolvimento e aplicação. Para tal contribuíram o facto de ter uma estrutura matemática maturada, que não necessita de grandes bases teóricas. Tem um modelo simples, fácil de compreender e aplicar. Pode processar dados inconsistentes, incompletos, vagos e de vários tipos. A teoria dos *Rough Sets* permite obter regras de decisão simplificadas e precisas (ZHANG *et al.*, 2016) e uma notável habilidade para lidar com dados qualitativos (RISSINO, LAMBERT-TORRES, 2009). Além disto, o método *Rough Sets* não precisa de informação preliminar sobre os dados, é objetivo a lidar com as inconsistências, permite avaliar a significância dos dados e obter um conjunto de regras de decisão (PAWLAK, 2003).

A teoria dos *Rough Sets* pode, facilmente, ser aplicada como complemento a outros métodos (ZHANG *et al.*, 2016) e tem aplicação na maioria dos casos reais (RISSINO, LAMBERT-TORRES, 2009). Os recentes desenvolvimentos da teoria têm passado pelo estudo da significância dos atributos, pela incorporação de algoritmos heurísticos e integração com o processamento paralelo. Um senão da teoria é estar limitada a dados discretos, sendo este um tópico para desenvolvimentos futuros (ZHANG *et al.*, 2016).

3.2. Metodologia LAD

No final da década de 1980, Hammer e a sua equipa deram a conhecer uma nova metodologia de seleção e classificação de dados, chamada Análise Lógica de Dados (LAD) (BRUNI, 2007). O seu suporte teórico é a matemática discreta, com destaque para a teoria das funções booleanas (BOROS *et al.*, 1997), razão pela qual, a LAD só trabalha com variáveis binárias. No entanto, raras são as bases de dados reais onde todos os atributos são binários. Quando tal não acontece, ou seja, existem atributos cujo domínio corresponde a valores qualitativos (variáveis literais, como a cor), ou quantitativos (variáveis discretas ou contínuas), há que aplicar um processo de conversão para valores binários (BOROS *et al.*, 1997). Nesta secção, a LAD vai ser analisada com o foco na seleção de atributos.

Recuperando as definições usadas na descrição dos *Rough Sets*, vem:

- Sistema de informação – conjunto das observações, com os atributos e respetivos valores.
- Sistema de decisão – sistema de informação com a classe associada.
- U – Universo – conjunto das observações (o_1, o_2, …, o_n), onde n é o número total de observações.
- A – conjunto dos atributos (a_1, a_2, …, a_m), onde m é o número total de atributos. $A = \{a_1, a_2, \ldots, a_m\}$
- C – classe.

Considere-se ainda:

- D_i – o domínio de cada atributo – conjuntos de todos os valores do atributo respetivo.

Relativamente à ação de converter os valores dos atributos para valores binários, ir-se-á usar a expressão *binarizar* e *binarização* para designar o ato de *binarizar*.

Dado um sistema de decisão, onde a classe é binária, podem-se definir dois conjuntos de observações, que formam uma partição do universo de observações (BOROS *et al.*, 1996):

- Conjunto de observações positivas (U^+), cujas observações (o^+) correspondem a um dos dois valores da classe;

- Conjunto de observações negativas (U^-), cujas observações (o^-) não correspondem ao valor da classe escolhido para definir U^+, ou seja, correspondem ao outro valor da classe;

A metodologia LAD não lida com inconsistências, logo, os conjuntos U^+ e U^- são bem definidos, sem ambiguidades (BOROS *et al.*, 1996). Outra particularidade é o facto de a LAD ser aplicada a sistemas de decisão com uma única classe, que só poderá ter dois valores possíveis, facilmente substituíveis pelos valores binários 0 e 1. No entanto, a LAD pode ser estendida a problemas onde o atributo classe tem mais de dois valores possíveis (SUBASI, AVILA-HERRERA, 2015).

3.2.1. *Binarização* de um sistema de decisão

Para que se possa aplicar a LAD, todos os atributos devem ser binários (BOROS *et al.*, 1996). Caso não sejam, há que *binarizá-los*. No processo de *binarização*, cada atributo não binário, a_i, será convertido num conjunto de atributos binários, b_j, onde $j = 1, \ldots, m^*$, sendo m^* a soma do número de atributos a_i iniciais que já eram binários, com o número total de atributos binários necessários para substituir os atributos a_i não binários (BRUNI, 2007).

Num sistema de decisão *binarizado*, o correspondente conjunto de atributos binários b_j, será $B = \{b_1; \ldots ; b_{m^*}\}$. As observações corresponderão a tuplos de valores binários, $o_b = (b_1; \ldots ; b_{m^*})$ (BRUNI, 2007).

O conjunto de atributos binários (B), resultantes da *binarização* de um sistema de decisão, é chamado de conjunto suporte (*support set*). Um conjunto suporte diz-se exatamente separado se cada tuplo o_b corresponder, sem ambiguidades, a uma observação positiva ou negativa. Isto é, se não existirem duas observações, uma positiva e a outra negativa, com iguais valores *binarizados* (BRUNI, 2007). Neste caso, não ocorrem inconsistências. Como a LAD não trabalha com dados inconsistentes, considerar-se-ão, sempre, conjuntos de suporte exatamente separados.

Quando um atributo a_i é qualitativo nominal, serão necessários tantos atributos binários (n_i) quanto o número total de elementos do, respetivo, domínio D_i (BOROS *et al.*, 1996). Supondo o atributo $a1$ (cor), com D_1 = {amarelo, verde, vermelho, azul}, vem $n_1 = \#D_1 = 4$. Desta forma, o atributo $a1$ será desdobrado em quatro atributos binários (b_j, j=1,…,4) correspondentes a cada uma das cores. O valor do atributo binário (b_j, j=1,…,4) para determinada observação o_b, será 1 se corresponder à cor do atributo respetivo e será 0 para as restantes cores.

Também, se podem *binarizar* atributos qualitativos ordinais, como por exemplo, meses do ano, ou ocorrência de um fenómeno (nunca, raramente, frequentemente, sempre) (BOROS *et al.*, 1996). Neste caso, deve-se criar uma correspondência entre os valores literais ordenados e uma sequência de números naturais, após a qual se procede à *binarização* do atributo quantitativo resultante, da forma descrita a seguir.

No caso de um atributo quantitativo, o processo de *binarização* é mais trabalhoso. Há que introduzir a noção de ponto de corte (*cut-point*), $\alpha_{i,j} \in I\!R$ (BOROS *et al.*, 1996).

Dado um atributo quantitativo, não binário, a_i, considere-se o respetivo domínio finito, D_i, cujos elementos estão ordenados por ordem crescente. Determina-se um ponto de corte entre dois valores consecutivos de D_i, sempre que um pertença a uma observação positiva, $o^+ \in U^+$ e o outro pertença a uma observação negativa, $o^- \in U^-$.

Sejam $a_i(k)$ e $a_i(k+1)$ dois valores consecutivos do domínio de um atributo, tais que $a_i(k) < a_i(k+1)$, $a_i(k) \in o^+$ e $a_i(k+1) \in o^-$. Neste caso, define-se um ponto de corte, cujo valor será a média destes dois valores de D_i (BRUNI, 2007):

$$\alpha_{i,j} = \frac{a_i(k) + a_i(k+1)}{2} \tag{3.14}$$

A cada ponto de corte do atributo a_i ($\alpha_{i,j}$) vai corresponder um atributo binário (b_j), cujo valor dependerá do valor do atributo (a_i):

$$b_j = \begin{cases} 1 & \text{se } a_i \geq \alpha_{i,j} \\ 0 & \text{se } a_i < \alpha_{i,j} \end{cases}$$

Da mesma forma se define um ponto de corte entre dois valores consecutivos do domínio do atributo, onde o primeiro pertence a uma observação negativa e o segundo a uma observação positiva. Os valores dos pontos de corte não têm de pertencer ao domínio do atributo.

Estes são os atributos binários do primeiro tipo, chamados de atributos de nível. Também são definidos os atributos binários do segundo tipo, designados por atributos de intervalo. (BOROS *et al.*, 1996)

A cada par de pontos de corte do atributo a_i ($\alpha_{i,j}$ e $\alpha_{i,k}$, com $\alpha_{i,j} < \alpha_{i,k}$) vai corresponder um atributo binário ($b_{i,jk}$), cujo valor dependerá do valor do atributo (a_i):

$$b_{i,jk} = \begin{cases} 1 & \text{se } \alpha_{i,j} \leq a_i < \alpha_{i,k} \\ 0 & \text{para outros valores de } a_i \end{cases}$$

O processo de *binarização* é analisado com grande detalhe no trabalho de Boros *et al.* (1997).

3.2.2. Redução do conjunto suporte

No processo de *binarização* de sistemas de decisão, obtidos a partir de casos reais de grande dimensão, normalmente, são determinados um grande número de pontos de corte, tornando impraticável o seu tratamento informático (BOROS *et al.*, 1996). Acontece que, muitos dos atributos binários não são necessários para classificar as observações. Daqui resulta a possibilidade de reduzir a dimensão do conjunto suporte com o objetivo de diminuir a complexidade do problema e melhorar a performance computacional. (BRUNI, 2007)

Para cada par de observações, uma positiva (o^+) e outra negativa (o^-), seja $I(o^+,o^-)$ o conjunto de pares ordenados (r, s) que correspondem aos índices do par de observações o^+ e o^- com representações binárias diferentes (BRUNI, 2007).

Considere-se, para cada atributo binário b_j, uma nova variável binária y_j, definida da seguinte forma (BRUNI, 2007):

$$y_j = \begin{cases} 1 & \text{se } b_j \text{ se mantém no conjunto suporte após a redução} \\ 0 & \text{se } b_j \text{ não se mantém no conjunto suporte após a redução} \end{cases}$$

Há que definir, também, um coeficiente binário, $d_{i,j}$, que indica se os valores binários do atributo b_j são diferentes, para cada par $(r, s) \in I(o^+,o^-)$, identificado pelo índice i (CAVIQUE *et al.*, 2011):

$$d_{i,j} = \begin{cases} 1 & \text{se } b_j(r) \neq b_j(s) \\ 0 & \text{se } b_j(r) = b_j(s) \end{cases}$$

onde, $b_j(r)$ é o valor binário do atributo b_j na observação positiva o_r e $b_j(s)$ é o valor binário do atributo b_j na observação negativa o_s.

Ou seja, $d_{i,j} = 1$ se, para duas observações o_r e o_s, os valores binários do atributo b_j são diferentes. Caso contrário, $d_{i,j} = 0$. O conjunto de todos os $d_{i,j}$ formam a matriz disjunta M.

O processo de redução do conjunto suporte pode ser matematizado através do modelo de cobertura de conjuntos (BOROS *et al.*, 1996):

$$\min \sum_{j=1}^{m^*} y_j$$

Sujeito às condições:

$$\sum_{j=1}^{m^*} d_{i,j}.y_j \geq 1 \quad , \quad \forall\, i \in I(o^+,o^-) \text{, com } o^+ \in U^+ , o^- \in U^-$$

$$y_j \in \{0,1\} \quad , \text{ com } j = 1, \ldots , m^*$$

Os atributos binários selecionados serão os b_j que na solução ótima correspondem aos valores de $y_j = 1$.

Após a redução do conjunto suporte, a metodologia LAD cria padrões que serão usados no processo de classificação. Este aspeto não será, aqui, desenvolvido, por estar fora do âmbito do tema.

Na versão alargada deste trabalho, disponível no sítio da *Internet* http://infapol.com/jda , toda esta secção é acompanhada por um exemplo ilustrativo.

3.2.3. Variantes à metodologia clássica da LAD

Há situações às quais se pode aplicar o algoritmo LAD, embora os respetivos problemas tenham variantes relativamente ao modelo clássico. É o caso da existência de custos associados aos atributos, ou existirem atributos sem valor em algumas observações, ou, ainda, classes com multivalores.

Se a cada atributo binário for imputado um custo, q_j, a formalização do problema de redução do conjunto suporte vem (CAVIQUE *et al.*, 2011):

$$\min \sum_{j=1}^{m^*} q_j \, . y_j$$

Sujeito às condições:

$$\sum_{j=1}^{m^*} d_{i,j} . y_j \;\; \geq \;\; 1 \;\; , \;\; \forall \, i \in I(o^+, o^-) \, , \text{com} \;\; o^+ \in U^+ \, , o^- \in U^-$$

$$y_j \in \{0,1\} \quad , \text{ com } j = 1, \ldots , m^*$$

A imputação de um custo a cada atributo binário quantifica a sua importância isolada para a solução final. Não tem em conta atributos que, por si só, não são significativos, mas, quando combinados, têm uma importância apreciável para a qualidade da redução do conjunto suporte (BRUNI, 2007). O recurso aos custos na formulação do problema apresenta grandes vantagens para a sua resolução computacional, nomeadamente, ao nível do tempo de execução dos algoritmos. Esta vantagem é mais notória quando o algoritmo procura uma solução através de um processo heurístico (BOROS *et al.*, 1996).

Outra variante ocorre quando existem atributos com valor omisso em algumas observações. Se, num atributo original, a_i, existem valores em falta, então nos respetivos atributos binários, b_j, também, são considerados em falta. Isto faz com que estes atributos não sejam usados. Matematicamente, corresponde a atribuir o valor zero aos respetivos coeficientes $d_{i,j}$ (BOROS *et al.*, 1996).

No caso do atributo classe de um problema ter mais de dois valores possíveis, procede-se à *binarização* da classe, como se de outro atributo se tratasse (SUBASI, AVILA-HERRERA, 2015). Este aspeto tem sido bastante estudado e existem diversas abordagens para lidar com o problema (FRIEDMAN *et al.*, 2000).

3.2.4. LAD versus *Rough Sets*

Tanto a LAD como a metodologia *Rough Sets* permitem efetuar a seleção de atributos e, ambos, são compostos por duas etapas: uma primeira etapa de transformação e uma segunda de redução do número de atributos (CAVIQUE *et al.*, 2011).

A LAD não permite a existência de inconsistências (duas observações com os mesmos valores dos atributos, mas pertencentes a classes diferentes) e só pode ser aplicada a dados binários (podendo recorrer ao processo de *binarização*) (BOROS *et al.*, 1996). Outro aspeto favorável é poder associar a cada atributo um custo, o que permite selecionar os atributos minimizando o custo total. Tudo isto faz da LAD uma metodologia mais sistematizada, robusta e sem ambiguidades, sendo fácil de interpretar (CAVIQUE *et al.*, 2013).

Já os *Rough Sets* lida com inconsistências, embora possa dificultar a interpretação dos resultados. Também, possibilita a existência de atributos classe com valores não necessariamente dicotomizados. No entanto, não permite o recurso aos custos associados a cada atributo (CAVIQUE *et al.*, 2011).

Da combinação destas duas metodologias, surge a Análise Lógica de Dados Inconsistentes (LAID), que é descrita na secção seguinte.

3.3. Metodologia LAID

Como já referido, a Análise Lógica de Dados Inconsistentes (LAID) surge da combinação das metodologias *Rough Sets* e LAD. Como características principais, a LAID permite atributos inteiros, com custos associados, lida com inconsistências e comporta classes não dicotomizadas (CAVIQUE *et al.*, 2013).

3.3.1. Remoção das redundâncias

Observações redundantes são observações com os mesmos valores dos atributos e pertencentes à mesma classe. Sendo m o número de atributos do sistema de decisão, o_x e o_y são redundantes se:

$$\begin{cases} a_i(o_x) = a_i(o_y) \ \ \forall i = 1, \dots, m \\ c(o_x) = c(o_y) \end{cases}$$

A remoção de redundâncias é simples, basta eliminar, uma a uma, as observações redundantes.

3.3.2. Remoção das inconsistências

Observações inconsistentes são observações com os mesmos valores dos atributos e pertencentes a classes diferentes. Sendo m o número de atributos do sistema de decisão, o_x e o_y são inconsistentes se:

$$\begin{cases} a_i(o_x) = a_i(o_y) \ \ \forall i = 1, \dots, m \\ c(o_x) \neq c(o_y) \end{cases}$$

Considere-se o grau de inconsistência (*gr*) de uma observação o_x, com $x = 1,\dots,n$, como o número de observações do Universo de dimensão n, inconsistentes com o_x. A definição formal é:

$$gr(o_x) = \#\{y=1,\dots,n: o_y \in U \wedge (a_i(o_x) = a_i(o_y), \forall i=1,\dots,m) \wedge \ c(o_x) \neq c(o_y)\}$$

Se $gr(o_x)=0$, significa que a observação o_x não tem inconsistências.

O grau de inconsistência do Universo, $gr(U)$ é igual ao maior dos graus de inconsistência de cada observação:

$$gr(U) = \text{máx}(\ gr(o_x)),\ \forall x=1,\ldots,n$$

O valor do grau de inconsistência do Universo é, sempre, menor ou igual ao número de valores diferentes da classe. Se $gr(U)=0$, significa que não existem inconsistências no sistema de decisão.

O processo de remoção das inconsistências existentes num sistema de decisão é surpreendentemente simples. O exemplo dado por Cavique *et al.* (2013) é bastante esclarecedor para se perceber a ideia subjacente. Quando um paciente vai a uma consulta, pode acontecer que o médico fique indeciso no diagnóstico, porque o conjunto dos sintomas relatados são comuns a várias doenças. Neste caso, o médico faz um ou mais testes para despistar o mal de que o paciente padece. Considerando os sintomas como um conjunto de atributos, o que o clínico faz é acrescentar novos atributos de forma a acabar com as ambiguidades existentes.

Assim sendo, quando existem inconsistências num conjunto de dados, elas são removidas através da adição de novos atributos binários, que vão testar "*je ne sais quoi*", para diferenciar as observações inconsistentes (CAVIQUE *et al.*, 2013). Para remover as inconsistências de um sistema de decisão são necessários $m^* = \lceil log_2(1+gr(U)) \rceil$ novos atributos binários, onde a função $\lceil x \rceil$ devolve a aproximação às unidades por excesso do valor x. Ou seja, é necessário um novo atributo binário, para distinguir duas observações com os mesmos valores dos atributos, mas pertencentes a classes diferentes. No caso de três, ou quatro, observações com valores dos atributos iguais, mas classes diferentes, são necessários 2 novos atributos binários.

Pelo facto de acrescentar novos atributos "*je ne sais quoi*" para retirar as inconsistências, faz com que qualquer subconjunto de U seja preciso (*crisp*), sendo a sua região fronteira vazia, porque $\underline{A}(X) = \bar{A}(X)$. Desta forma, evitam-se as dificuldades de interpretação destes conceitos, que são fundamentais na teoria dos *Rough Sets* (YAO, 2006).

A Figura 3.2 esquematiza a diferença entre as metodologias *Rough Sets* e LAID no tratamento das inconsistências. Neste exemplo, o Universo é composto por 6 observações (o_1, o_2, o_3, o_4, o_5, o_6). As observações o_1, o_4 e o_5 têm valor da classe igual a 1, as restantes observações têm valor da classe 0. As observações o_5 e o_6 são inconsistentes, ou seja, têm igual valor nos atributos, mas classe diferente. Seja X um subconjunto de U cujos valores dos atributos correspondem ao valor 1 da classe. Logo, $\underline{A}(X) = \{o_1; o_4\}$, $\bar{A}(X) = \{o_1; o_4; o_5; o_6\}$ e $FR(X) = \{o_5; o_6\}$. Como cada inconsistência corresponde a um par de observações, é necessário um novo atributo "*je ne sais quoi*" (a*) que toma o valor 1 para a observação da fronteira cujo valor da classe é 1 (o_5) e o valor 0 para as restantes observações. Assim, após a inserção do atributo a*, o subconjunto de U, cujos valores dos atributos correspondem ao valor 1 da classe, será X*, onde $X^* = \underline{A}(X^*) = \bar{A}(X^*) = \{o_1; o_4; o_5\}$, e $FR(X^*) = \varnothing$. A Figura 3.2 mostra o resultado da remoção das inconsistências na LAID quando é adicionado o atributo *jnsq*.

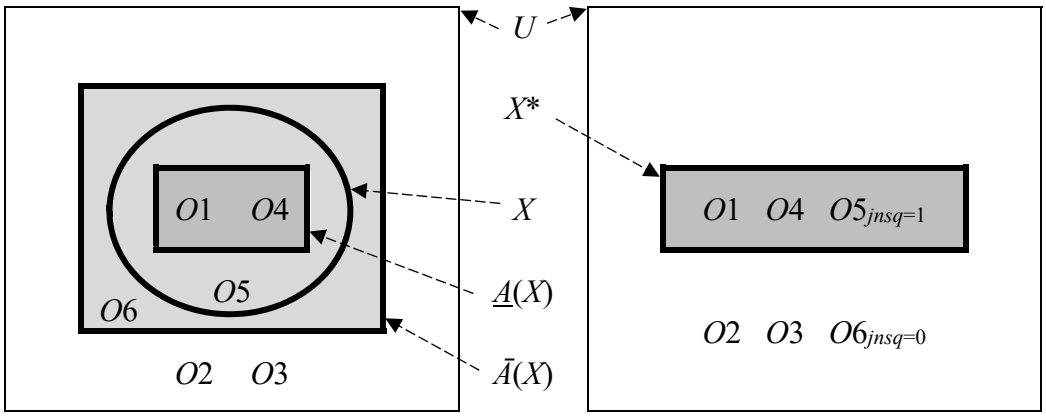

Figura 3.2 – Diferença entre os Rough Sets e a remoção das inconsistências na LAID. (Adaptado de: PAWLAK, 2003 e CAVIQUE et al., 2013)

Sistematizando todo o processo de remoção das inconsistências num sistema de decisão:

- Há que começar por remover as redundâncias.
- A quantidade de atributos "*je ne sais quoi*" é determinada por $m^* = \lceil log_2(1+gr(U)) \rceil$.
- Para cada conjunto de inconsistências, ordenam-se as observações pelos respetivos valores da classe e faz-se corresponder, a cada uma, a respetiva posição de ordem, $p_i(c(o_x))$, começando por zero.

- Para cada observação inconsistente, o valor de cada atributo "*je ne sais quoi*" é o respetivo algarismo da codificação binária de $p_i(c(o_x))$. Para as observações sem inconsistências, todos os atributos "*je ne sais quoi*" têm o valor zero.

O conjunto de todos os atributos originais, unido com o conjunto dos atributos "*je ne sais quoi*" é designado por A^*.

Considere-se o exemplo do sistema de decisão da Tabela 3.1 para aplicar a remoção das inconsistências segundo a metodologia LAID.

Tabela 3.1 – Sistema de decisão para exemplificar a LAID.

Observações	$a1$	$a2$	$a3$	$a4$	classe
$o1$	1	0	1	0	1
$o2$	0	1	1	0	1
$o3$	0	1	0	0	1
$o4$	0	1	0	0	2
$o5$	1	0	0	1	0
$o6$	1	0	1	0	2
$o7$	0	1	0	0	0
$o8$	1	0	1	0	1

Começa-se por remover as redundâncias. As observações $o1$ e $o8$ são redundantes. Ao retirar uma delas, escolha-se $o8$, a redundância desaparece.

Tabela 3.2 – Sistema de decisão com a redundância removida.

Observações	$a1$	$a2$	$a3$	$a4$	classe
$o1$	1	0	1	0	1
$o2$	0	1	1	0	1
$o3$	0	1	0	0	1
$o4$	0	1	0	0	2
$o5$	1	0	0	1	0
$o6$	1	0	1	0	2
$o7$	0	1	0	0	0

Da observação da Tabela 3.2, identificam-se os seguintes conjuntos de observações inconsistentes: {*o*1, *o*6} e {*o*3, *o*4, *o*7}, vindo:

Tabela 3.3 – Grau de inconsistência das observações.

Observações	*o*1	*o*2	*o*3	*o*4	*o*5	*o*6	*o*7
gr(o_x)	1	0	2	2	0	1	2

Como o grau de inconsistência do universo de observações é $gr(U) = 2$, são necessários dois atributos "*je ne sais quoi*" (*a*5* e *a*6*), porque $m^* = \lceil log_2(1+2) \rceil = 2$, ficando A^*={*a*1, *a*2, *a*3, *a*4, *a*5*, *a*6*}.

Tabela 3.4 – Remover as inconsistências do sistema de decisão.

Observações	*a*1	*a*2	*a*3	*a*4	*a*5*	*a*6*	classe
*o*1	1	0	1	0	0	0	1
*o*2	0	1	1	0	0	0	1
*o*3	0	1	0	0	1	0	1
*o*4	0	1	0	0	0	1	2
*o*5	1	0	0	1	0	0	0
*o*6	1	0	1	0	1	0	2
*o*7	0	1	0	0	0	0	0

O sistema da Tabela 3.4 está pronto para aplicar o algoritmo de seleção dos atributos. Sendo a LAID um algoritmo constituído por duas fases (CAVIQUE *et al.*, 2013):

1. Gerar a matriz disjunta *M*;
2. Redução do conjunto suporte.

3.3.3. Gerar a matriz disjunta

À semelhança da LAD, a metodologia LAID recorre à determinação da matriz disjunta *M*. Mas, ao contrário da primeira, a LAID permite classes com um número ilimitado de valores diferentes (CAVIQUE *et al.*, 2013).

O processo consiste em, para cada observação, comparar os valores de cada atributo com os respetivos das observações seguintes que tenham valor de classe diferente. Se o_x e o_y forem duas observações, tais que $c(o_x) \neq c(o_y)$, cuja comparação corresponde à linha i da matriz disjunta M, os elementos da matriz M ($d_{i,j}$, com j=1,…,m+m*) serão (CAVIQUE *et al.*, 2013):

$$d_{i,j} = \begin{cases} 1 & \text{se } a_j(o_x) \neq a_j(o_y) \\ 0 & \text{se } a_j(o_x) = a_j(o_y) \end{cases}$$

Como as inconsistências foram removidas, cada linha da matriz M tem de ter, pelo menos, um valor não nulo. O número de linhas da matriz M é menor, ou igual, a $n(n-1)/2$, valor que corresponderia a comparar todas as n observações. (CAVIQUE *et al.*, 2013)

Retomando o sistema da Tabela 3.4, a matriz disjunta vem:

Tabela 3.5 – Matriz Disjunta M.

Por comparação das observações	*a*1	*a*2	*a*3	*a*4	*a*5*	*a*6*
*o*1 e *o*4	1	1	1	0	0	1
*o*1 e *o*5	0	0	1	1	0	0
*o*1 e *o*6	0	0	0	0	1	0
*o*1 e *o*7	1	1	1	0	0	0
*o*2 e *o*4	0	0	1	0	0	1
*o*2 e *o*5	1	1	1	1	0	0
*o*2 e *o*6	1	1	0	0	1	0
*o*2 e *o*7	0	0	1	0	0	0
*o*3 e *o*4	0	0	0	0	1	1
*o*3 e *o*5	1	1	0	1	1	0
*o*3 e *o*6	1	1	1	0	0	0
*o*3 e *o*7	0	0	0	0	1	0
*o*4 e *o*5	1	1	0	1	0	1
*o*4 e *o*7	0	0	0	0	0	1
*o*5 e *o*6	0	0	1	1	1	0
*o*6 e *o*7	1	1	1	0	1	0

Na matriz disjunta M, se $d_{i,j} = 1$ então o atributo da coluna j diferencia as duas observações comparadas na linha i. Pelo contrário, se $d_{i,j} = 0$ então o atributo da coluna j não distingue as duas observações comparadas na linha i. (CAVIQUE *et al.*, 2013)

3.3.4. Redução do conjunto suporte

A obtenção do menor conjunto suporte usa uma heurística proposta por Chvatal (CAVIQUE *et al.*, 2013). A redução do conjunto suporte é um processo iterativo que pretende determinar um subconjunto, S, de A^*. Neste subconjunto S, inicialmente vazio, colocam-se os atributos escolhidos em cada iteração. No final, os elementos de S correspondem à seleção dos atributos que reduzem o conjunto suporte.

Começa-se por somar os valores de cada coluna da matriz M, obtendo-se o vetor s, onde:

$$s_j = \sum_{i=1}^{n} d_{i,j} \text{ , com } j = 1,\ldots,m+m^* \tag{3.15}$$

Tabela 3.6 – Processo de redução – iteração 1.

A^*	$a1$	$a2$	$a3$	$a4$	$a5^*$	$a6^*$
$d_{i,j}$	1	1	1	0	0	1
	0	0	1	1	0	0
	0	0	0	0	1	0
	1	1	1	0	0	0
	0	0	1	0	0	1
	1	1	1	1	0	0
	1	1	0	0	1	0
	0	0	1	0	0	0
	0	0	0	0	1	1
	1	1	0	1	1	0
	1	1	1	0	0	0
	0	0	0	0	1	0
	1	1	0	1	0	1
	0	0	0	0	0	1
	0	0	1	1	1	0
	1	1	1	0	1	0
s_j	8	8	9	5	7	5

As componentes do vetor s correspondem às imagens da função que decide qual o atributo escolhido para a solução, $s_j = s(a_j)$. O atributo pretendido, a_e, é aquele que melhor explica o sistema de decisão, ou seja, aquele que cumpre a condição:

$$s(a_e) = \text{máx}(s_j), \text{ com } j = 1,\ldots,m+m^* \tag{3.16}$$

No exemplo da Tabela 3.6, o atributo escolhido é $a3$, entrando para a solução, $S=\{a3\}$. Se vários atributos cumprem a condição (3.16), existem escolhas alternativas e escolhe-se um deles.

Após escolhido o atributo a entrar na solução, a_e, e atualizada a solução S, eliminam-se as linhas da matriz M que são explicadas por esse atributo, cujo valor $d_{i,e} = 1$, bem como a coluna do próprio atributo. Reinicia-se o processo com nova iteração, até eliminar todas as linhas da matriz M.

Tabela 3.7 – Processo de redução – iteração 2.

$a1$	$a2$	$a4$	$a5^*$	$a6^*$
0	0	0	1	0
1	1	0	1	0
0	0	0	1	1
1	1	1	1	0
0	0	0	1	0
1	1	1	0	1
0	0	0	0	1
3	3	2	5	3

Na segunda iteração do exemplo (Tabela 3.7), é escolhido o atributo $a5^*$, ficando $S = \{a3, a5^*\}$. São eliminadas as 5 primeiras linhas da matriz M e a coluna do atributo escolhido.

Tabela 3.8 – Processo de redução – iteração 3.

$a1$	$a2$	$a4$	$a6^*$
1	1	1	1
0	0	0	1
1	1	1	2

Na terceira iteração do exemplo (Tabela 3.8), é escolhido o atributo *a*6*, ficando $S = \{a3, a5^*, a6^*\}$. São eliminadas as restantes linhas da matriz *M* e a coluna do atributo escolhido. O processo está terminado. A redução obtida do conjunto suporte é $\{a3, a5^*, a6^*\}$.

Tabela 3.9 – Sistema de decisão após redução do conjunto suporte.

Observações	*a*3	*a*5*	*a*6*	classe
*o*1	1	0	0	1
*o*2	1	0	0	1
*o*3	0	1	0	1
*o*4	0	0	1	2
*o*5	0	0	0	0
*o*6	1	1	0	2
*o*7	0	0	0	0

Tabela 3.10 – Valores da classe em função dos atributos selecionados.

		(*a*5*, *a*6*)			
		(0,0)	(1,0)	(0,1)	(1,1)
*a*3	0	0	1	2	
	1	1	2		

As células em branco representam valores para novas observações que não são explicados pelo novo sistema de decisão. Este conceito é diferente da noção de inconsistência, porque é resultante da falta de observações (CAVIQUE *et al.*, 2013). No exemplo, dado o diminuto número de observações, era de esperar a existência de muitas células inexplicadas.

Havendo, inicialmente, inconsistências no conjunto suporte e tendo-se acrescentado os atributos "*je ne sais quoi*" para eliminar essas inconsistências, é lógico que estes atributos sejam necessários para explicar o sistema de decisão. Daí, a redução do conjunto suporte contém, sempre, todos os atributos "*je ne sais quoi*".

3.4. Critérios de validação

Depois de aplicar um determinado método de seleção de atributos, há que definir critérios para aferir a sua performance e validar os resultados obtidos.

A performance dos métodos de seleção de atributos é determinada a partir da performance dos modelos de aprendizagem aplicados após a redução, usando métricas que variam consoante o método. Os recursos computacionais (memória e tempo despendido), a precisão, o rácio de atributos selecionados, são exemplos de métricas usadas. (BOLÓN-CANEDO *et al.*, 2013)

Para dificultar este processo, muitos conjuntos de dados podem ter atributos classe multivalor, dados descontextualizados ou errados, atributos irrelevantes ou repetidos, o número de observações muito pequeno comparado com a quantidade de atributos, etc.. Todas estas dificuldades podem tornar os estudos comparativos de tal forma complexos que são impraticáveis. (BOLÓN-CANEDO *et al.*, 2013)

A maioria dos estudos foca-se no problema que se pretende resolver. Muitos são realizados recorrendo a dados gerados artificialmente. Já que os resultados são conhecidos, o método de seleção de atributos pode ser avaliado independentemente do classificador usado e permite alterar as condições experimentais. (BOLÓN-CANEDO *et al.*, 2013)

3.4.1. Classificação

O processo de classificação tem por objetivo prever o valor da classe de uma nova observação, que não pertence ao sistema de decisão usado na seleção de atributos. Normalmente, para que tal se possa realizar, o conjunto de observações do sistema é dividido, aleatoriamente, em dois subconjuntos (CAVIQUE *et al.*, 2013):

- O conjunto de treino – as observações são usadas para selecionar os atributos e determinar o modelo de aprendizagem ou classificador;

- O conjunto de teste – as observações são usadas para testar o modelo de aprendizagem determinado.

Sendo n a dimensão do universo de observações, se o conjunto de treino tem n_{tr} observações, então o conjunto de teste tem n-n_{tr} observações. Esta divisão do sistema de informação tem a desvantagem de não usar todas as observações para determinar o modelo de aprendizagem, particularmente, no caso de existirem poucas observações.

Uma forma de contornar esta desvantagem é a chamada validação cruzada (*cross-validation*) (CAVIQUE *et al.*, 2013). Esta técnica consiste em dividir o sistema de dados em p partições. Uma das partições é usada para teste, as restantes partições constituem o conjunto de treino. O processo é repetido, trocando a partição de teste por uma das de treino, até que todas as partições tenham sido usadas para teste. O modelo é avaliado pela média das performances de cada teste. Um caso particular é o chamado *Leave-One-Out Cross-Validation*, onde as partições do sistema têm apenas uma observação (CAVIQUE *et al.*, 2013).

Num sistema de decisão, onde o atributo classe pode ter k valores diferentes, a forma de prever o valor da classe de uma nova observação o_x é através da função f (CAVIQUE *et al.*, 2013), onde:

$$f(o_x) = i: H(o_x,i) = \min(H(o_x,j), \text{com } j=0,\ldots,k\text{-}1) \tag{3.17}$$

A função $H(o_x,j)$ devolve a média das distâncias de Hamming entre o_x e todas as observações do sistema de decisão cujo valor da classe é j. A distância de Hamming (dH), entre dois vetores da mesma dimensão, é o número de coordenadas cujos valores, nesses vetores, são diferentes. Dito de outra forma, a distância de Hamming é o menor número de substituições necessárias para transformar um dos vetores no outro. A função f devolve o valor da classe cujo retorno de H é menor.

Considerando o sistema de decisão da Tabela 3.11, pretende-se prever o valor da classe de uma nova observação o_x = (1,0,1). A Tabela 3.12 esquematiza o processo de cálculo e determina o resultado final. O valor dH corresponde à distância de Hamming entre a nova observação e cada uma do sistema de decisão.

Tabela 3.11 – Novo exemplo de sistema de decisão.

Observações	$a1$	$a2$	$a3$	classe
$o1$	1	0	0	1
$o2$	0	1	0	1
$o3$	0	1	1	1
$o4$	0	0	1	0
$o5$	0	0	0	0
$o6$	1	1	0	0

Tabela 3.12 – Exemplo do cálculo da função f.

Observações	vetor	classe	dH	$H(o_x,j)$
$o1$	(1,0,0)	1	1	$\frac{1+3+2}{3} = 2$
$o2$	(0,1,0)		3	
$o3$	(0,1,1)		2	
$o4$	(0,0,1)	0	1	$\frac{1+2+2}{3} = 1{,}67$
$o5$	(0,0,0)		2	
$o6$	(1,1,0)		2	
$f(1,0,1) = 0$				

O valor da classe prevista para o_x = (1,0,1) é 0.

3.4.2. Avaliação da performance

A performance de um modelo de aprendizagem é avaliada através da correção, ou não, das previsões. Esta avaliação tem por base a matriz confusão (*confusion matrix*) (CAVIQUE *et al.*, 2013), cujas linhas contabilizam o número de observações de cada um dos valores das classes no conjunto de teste, de acordo com o sistema de decisão, e as colunas contabilizam o número de observações previstas para cada classe (SANTRA, CHRISTY, 2012).

Tabela 3.13 – Matriz confusão quando a classe é booleana.

		Classe prevista	
		0	1
Classe real	0	n_{00}	n_{01}
	1	n_{10}	n_{11}

Na Tabela 3.13 (SANTRA, CHRISTY, 2012):

- n_{00} é o número de observações do conjunto de teste cujo valor da classe é 0 e a previsão foi correta (verdadeiros negativos);
- n_{01} é o número de observações cujo valor da classe é 0 e a previsão foi incorreta (falsos positivos);
- n_{10} é o número de observações cujo valor da classe é 1 e a previsão foi incorreta (falsos negativos);
- n_{11} é o número de observações cujo valor da classe é 1 e a previsão foi correta (verdadeiros positivos).

A soma $n_{00} + n_{01} + n_{10} + n_{11}$ é a dimensão do conjunto de teste.

A matriz confusão permite determinar várias medidas de performance, nomeadamente, a taxa de cobertura (*overall accuracy*) (SANTRA, CHRISTY, 2012):

$$\text{taxa de cobertura} = \frac{n_{00} + n_{11}}{n_{00} + n_{01} + n_{10} + n_{11}} \tag{3.18}$$

Esta métrica indica a frequência de acerto do modelo obtido. Um modelo de aprendizagem é melhor que um modelo aleatório se a sua taxa de cobertura é maior que a taxa da classe modal (CAVIQUE *et al.*, 2013), ou seja, é maior que a taxa de qualquer valor da classe:

$$\left(\frac{n_{00} + n_{11}}{n_{00} + n_{01} + n_{10} + n_{11}} > \frac{n_{00} + n_{01}}{n_{00} + n_{01} + n_{10} + n_{11}}\right) \wedge \left(\frac{n_{00} + n_{11}}{n_{00} + n_{01} + n_{10} + n_{11}} > \frac{n_{10} + n_{11}}{n_{00} + n_{01} + n_{10} + n_{11}}\right)$$

$$\Rightarrow \quad n_{11} > n_{01} \quad \wedge \quad n_{00} > n_{10}$$

Outra medida de performance é o *Cohen's Kappa-statistic*, que indica o grau de concordância dos dados (CAVIQUE *et al.*, 2013). Para tal, há que definir a taxa esperada (*expected rate*) que mede a precisão para um modelo de aprendizagem aleatório (JENNESS, WYNNE, 2005):

$$\text{taxa esperada} = \frac{(n_{00}+n_{01})\times(n_{00}+n_{10}) + (n_{10}+n_{11})\times(n_{01}+n_{11})}{(n_{00}+n_{01}+n_{10}+n_{11})^2} \tag{3.19}$$

O índice *Kappa-statistic* é calculado recorrendo à taxa de cobertura e à taxa esperada (JENNESS, WYNNE, 2005):

$$\textit{Kappa-statistic} = \frac{\text{taxa de cobertura} - \text{taxa esperada}}{1 - \text{taxa esperada}} \tag{3.20}$$

Os valores de *kappa* estão entre -1 e 1, inclusive, e permitem avaliar a performance de acordo com a escala seguinte (CAVIQUE *et al.*, 2013):

Tabela 3.14 – Escala de avaliação do índice Kappa-statistic.

Kappa-statistic	[-1,0; 0,0]	]0,0; 0,2]	]0,2; 0,4]	]0,4; 0,6]	]0,6; 0,8]	]0,8; 1,0]
Performance do modelo	má	fraca	sofrível	moderada	boa	excelente

O índice *Kappa-statistic* é comparável entre modelos de aprendizagem diferentes (JENNESS, WYNNE, 2005).

A matriz confusão, a taxa de cobertura e o índice *Kappa-statistic* são facilmente generalizáveis para sistemas com classes multivalor. No caso da classe poder assumir 3 valores, vem:

Tabela 3.15 – Matriz confusão quando a classe é ternária.

		Classe prevista		
		0	1	2
Classe real	0	n_{00}	n_{01}	n_{02}
	1	n_{10}	n_{11}	n_{12}
	2	n_{20}	n_{21}	n_{22}

$$\text{taxa de cobertura} = \frac{n_{00} + n_{11} + n_{22}}{n_{00} + n_{01} + n_{02} + n_{10} + n_{11} + n_{12} + n_{20} + n_{21} + n_{22}}$$

taxa esperada =

$$\frac{(n_{00}+n_{01}+n_{02})\times(n_{00}+n_{10}+n_{20})+(n_{10}+n_{11}+n_{12})\times(n_{01}+n_{11}+n_{21})+(n_{20}+n_{21}+n_{22})\times(n_{02}+n_{12}+n_{22})}{(n_{00} + n_{01} + n_{02} + n_{10} + n_{11} + n_{12} + n_{20} + n_{21} + n_{22})^2}$$

$$\textit{Kappa-statistic} = \frac{\text{taxa de cobertura} - \text{taxa esperada}}{1 - \text{taxa esperada}}$$

4. Implementação do LAID em HDF5+*Python* no INCD

4.1. HDF5

O propósito e a vocação do método LAID é, fundamentalmente, a seleção de atributos num conjunto de dados booleanos de enorme dimensão, esparsos, com classes booleanas ou multivalor e inconsistências.

Neste trabalho, o HDF5 (*Hierarchical Data Format* versão 5) foi o sistema de base de dados adotado. Engloba um modelo de dados, uma biblioteca e um formato de ficheiro, para armazenamento e gestão de dados. Esta escolha deve-se às suas características (HDF GROUP, 2018):

- Suporta uma variedade ilimitada de tipos de dados, que poderão ser objetos complexos e multidimensionais;
- É portátil, porque os ficheiros têm um formato binário *standard* amplamente usado, não havendo necessidade de conversão;
- É extensível, permitindo que as aplicações possam evoluir sem barreiras adicionais;
- Tem um formato de ficheiro autoexplicativo, o que significa que todos os dados e metadados podem ser guardados num único ficheiro;
- Corre numa grande variedade de plataformas, desde os computadores pessoais aos grandes sistemas de acesso paralelo;
- Tem alto desempenho de entradas e saídas, permitindo uma otimização do tempo de acesso e do espaço de armazenamento;
- Não tem limite quanto ao número ou dimensão dos dados, tendo grande flexibilidade para um grande volume de dados.

O formato HDF5 tem uma estrutura semelhante à de um sistema de ficheiros de um computador, o que permite organizar os dados de uma forma intuitiva e versátil. A estrutura correspondente à pasta é chamada de grupo e os *datasets* são o equivalente aos ficheiros. O ficheiro HDF5 é o grupo raiz e um grupo pode conter outros grupos ou ligações para objetos noutros grupos, ou, inclusive, noutros ficheiros HDF5. Os *datasets*, com os dados, são criados dentro dos grupos. Os *datasets* podem ser imagens, tabelas, gráficos ou documentos

(p.e. PDF, Excel). Cada uma destas estruturas pode ter metadados associados que descrevam e guardem informação sobre os dados e a estrutura dos objetos.

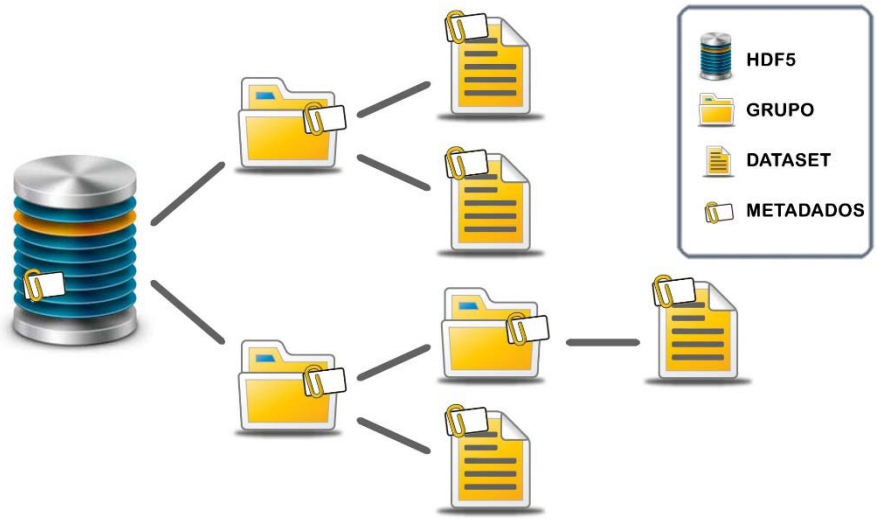

Figura 4.1 – Esquema da estrutura de um ficheiro HDF5.

Os *datasets* contêm os dados propriamente ditos e os metadados (que descrevem os dados). Os metadados atributos são definidos pelo utilizador para guardar informação extra sobre os objetos armazenados.

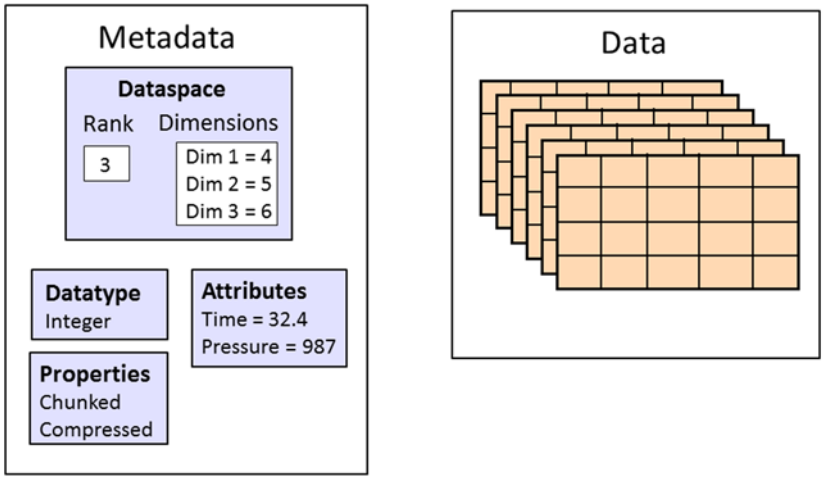

Figura 4.2 – Estrutura dos datasets.
(Fonte: HDF GROUP, 2018)

As propriedades, nos metadados, definem como os valores dos dados são fisicamente armazenados no disco. Existem três formas de organizar um conjunto de dados: contínuo, em blocos e compactado.

Se o armazenamento for contínuo, os valores dos dados são armazenados sequencialmente, ficando adjacentes entre si no ficheiro HDF5. Esta é a forma padrão de armazenamento.

Uma poderosa característica do HDF5 é a compartimentação dos dados, ou seja, os *datasets* podem ser divididos em vários blocos (*chunks*), de igual dimensão, definidos pelo utilizador. Nas operações de escrita e leitura, apenas os blocos necessários são carregados. Isto significa que não é preciso aceder a todo o *dataset*, permitindo uma maior eficiência quando se trabalha com conjuntos de dados de muito grande dimensão.

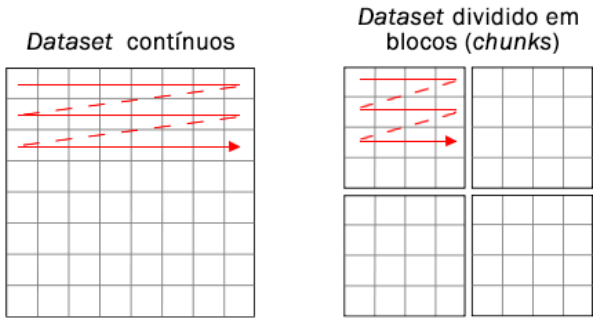

Figura 4.3 – Leitura e escrita num dataset contínuo ou compartimentado.

A compartimentação de um *dataset* é obrigatória quando se pretende a compressão, ou para criar conjuntos de dados de dimensão extensível ou ilimitada. A compartimentação pode melhorar muito o desempenho de conjuntos de dados de grande dimensão, porque só os blocos necessários é que serão acedidos. No entanto, caso a compartimentação seja mal dimensionada, pode prejudicar bastante o desempenho. O número máximo de elementos num bloco é $2^{32} - 1$ e a dimensão máxima é 4 GB (HDF GROUP, 2018).

Problemas que podem surgir quando se usa a compartimentação:

- Se a dimensão dos blocos for muito pequena, pode originar uma quantidade de blocos excessiva, especialmente no caso de conjuntos de grande dimensão e originar uma degradação no desempenho durante os processos de leitura e de escrita ou na procura de um determinado bloco.

- Se a dimensão dos blocos for muito grande, pode causar uma degradação do desempenho na leitura de um pequeno subconjunto de dados, especialmente se a dimensão do bloco for substancialmente maior que a do subconjunto cuja leitura se pretende fazer. Além disso, um *dataset* pode ficar maior do que o esperado, se existirem blocos que contenham uma pequena quantidade de dados.

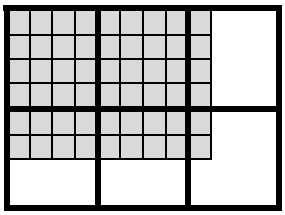

Figura 4.4 – Sobredimensionamento dos blocos num dataset.

- Cada *dataset* compartimentado tem uma *chunk cache* associada, com um volume padrão de 1 MB. O objetivo da *cache* é melhorar o desempenho mantendo em memória os blocos que são acedidos com maior frequência, para que não seja necessário lê-los em disco. Se os blocos forem muito grandes, poucos caberão na *chunk cache*, e o desempenho será, significativamente, prejudicado. O volume da *cache* pode ser aumentado.

Uma boa prática é evitar blocos muito pequenos e testar os dados com blocos de diferentes dimensões para determinar qual a ideal. Há que ter em conta a forma mais frequente de leitura e escrita dos dados. Por exemplo, se a maior parte das leituras forem realizadas ao longo das colunas da tabela dos dados, os blocos deverão corresponder às colunas da tabela. Neste caso, cada coluna será armazenada contiguamente e a leitura de uma parte da coluna será realizada numa única operação, com acesso, apenas, a um bloco.

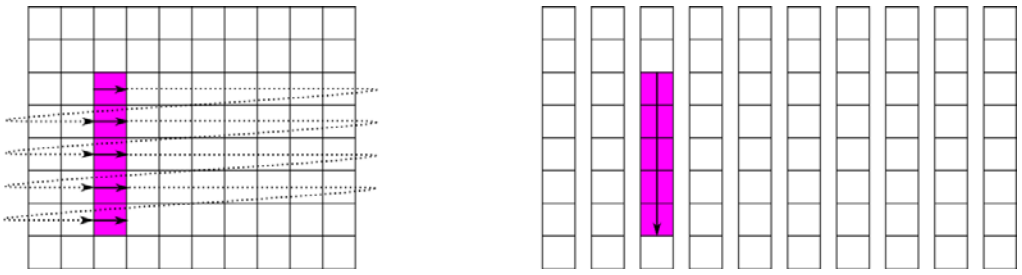

Figura 4.5 – Leitura de parte de uma coluna num dataset contínuo ou compartimentado. (Fonte: HDF GROUP, 2018)

Num *dataset* compactado, os dados são armazenados no cabeçalho do *dataset*, onde estão os metadados. Este *layout* é para conjuntos de dados muito pequenos, que podem caber, facilmente, no cabeçalho do objeto, cujo volume máximo é 64 KB (HDF GROUP, 2018).

O espaço em disco de um ficheiro HDF5 é otimizado. Mesmo assim, é possível comprimir os *datasets*, otimizando, ainda mais, o espaço, mas, penalizando o tempo de acesso. Existem filtros para compressão predefinidos (ZLIB e SZIP) ou podem ser aplicados filtro definidos pelo utilizador. Para comprimir um *dataset* é necessário que este esteja compartimentado em blocos (*chunks*). Nas operações de leitura e escrita, só os blocos cujos dados se pretendam aceder serão descomprimidos e, depois, comprimidos. A definição e dimensionamento dos blocos influência bastante a performance no caso de *datasets* comprimidos.

4.2. *Python*

Dado já existir uma versão do LAID em linguagem C, por um critério de inovação, que deve prevalecer num trabalho de dissertação, optou-se por programar em *Python*3, de forma a aumentar o leque de implementações do LAID. Além do que, em condições normais e em igualdade de circunstâncias, a curva de aprendizagem e o tempo de desenvolvimento da linguagem *Python* são, substancialmente, menores que em C, proporcionando, também, um menor número de *bugs* e uma mais fácil leitura do código (W3RESOURCE, 2018), nomeadamente, se for usada por terceiros. Por exemplo, programar em *Assembly* pode aumentar a velocidade e a precisão do código, mas será que compensa o aumento exponencial do tempo de desenvolvimento e da dificuldade de manutenção, ou alteração, do código? Outra razão, é o facto de nas universidades, em especial nos Estados Unidos, se estar a generalizar o ensino da linguagem *Python*. Quando se tem em vista trabalhos futuros, é uma linguagem a ter em conta.

O *Python* é uma linguagem de alto nível, *open source*, desenvolvida por Guido Van Rossum no início dos anos 90. Corre na maioria dos sistemas Unix, no Windows e no Mac. Tem uma

vasta biblioteca nativa (*PYTHON* S.F., 2018). No modo interativo, as instruções em *Python* são digitadas na consola e interpretadas imediatamente. Alternativamente, um conjunto de instruções em *Python* pode ser escrito num ficheiro com extensão ".py". Ao correr o interpretador, o código é traduzido para uma linguagem de baixo nível (*byte code*) e é este código que é interpretado (*PYTHON* S.F., 2018). Este género de compilação *Just In Time* é mais rápida que a execução no modo interativo.

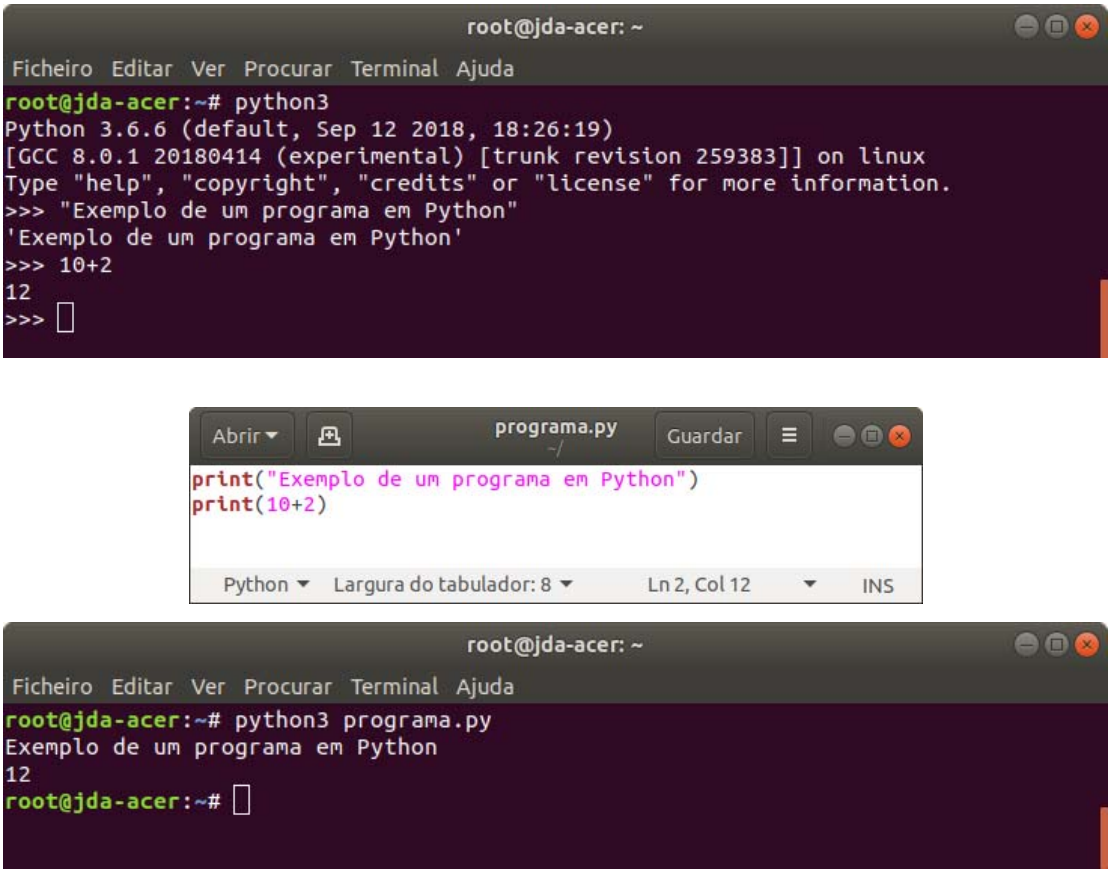

Figura 4.6 – Exemplo de execução de código Python no modo interativo e num ficheiro.

Sendo C uma linguagem compilada, com criação de um executável, seria mais rápida em tempo de execução. Sendo o objetivo principal deste trabalho validar a eficácia e eficiência do algoritmo LAID na seleção dos atributos, a rapidez de execução não deve ser negligenciada. Muitas funções usadas estão definidas na biblioteca nativa da linguagem *Python* e no pacote *Numpy*, mas não em C, como sejam algumas funções de manuseamento de matrizes, seleção, ordenação e contagem, tendo de ser codificadas pelo programador, o que, muito provavelmente, não seriam tão eficientes quanto as definidas nativamente em *Python*.

Foi instalado o pacote H5PY, que é uma interface *Python* para o formato de dados binários HDF5. Tem a vantagem de usar a linguagem *Python* e o pacote *Numpy*, que suporta vetores e matrizes multidimensionais, possuindo uma grande coleção de funções para lidar com estas estruturas. (COLLETTE *et al.*, 2018)

4.3. Infraestrutura Nacional de Computação Distribuída

> "Com a expansão das fronteiras do conhecimento surgem novos e mais complexos desafios científicos. A investigação de ponta requer meios de cálculo e processamento de dados cada vez mais poderosos, e que não estão ao alcance de centros de investigação isolados. É necessária colaboração e uma abordagem integrada da disponibilização e utilização das tecnologias de informação, que permita a exploração e partilha de dados e meios computacionais através da *Internet*.
>
> […] A simulação, processamento e análise de dados do LHC desenvolveu novas tecnologias, e levou à criação do Worldwide LHC Computing Grid (WLCG), a maior infraestrutura de computação científica distribuída jamais construída. Requisitos semelhantes ou até superiores estão agora a surgir noutros domínios científicos tais como as ciências da vida, ciências dos materiais, astronomia, biodiversidade e observação da terra entre muitos outros."
>
> (LNEC, 2018)

4.3.1. INCD – O que é

A Infraestrutura Nacional de Computação Distribuída (INCD) tem por missão permitir o acesso a recursos computacionais e de armazenamento de elevada capacidade e desempenho, por parte da comunidade científica e académica sediada em Portugal (FCT-FCCN, 2018), quando os respetivos centros de investigação não têm possibilidade de fornecer tais recursos. É uma infraestrutura digital aprovada no âmbito do Roteiro de infraestruturas estratégicas de investigação da Fundação para Ciência e Tecnologia (FCT) (LIP, 2018)

A INCD surgiu em 2015 (PIRES, PAGAIME, 2015) e é herdeira da infraestrutura de computação, iniciada em 2008, no âmbito da Iniciativa Nacional Grid (FCT-FCCN, 2018). É resultado da parceria entre o LIP (Laboratório de Instrumentação e Física Experimental de

Partículas), a FCCN (unidade da FCT responsável pela gestão e operação de Rede Ciência, Tecnologia e Sociedade) e o LNEC (Laboratório Nacional de Engenharia Civil) e com a colaboração de outras entidades (INCD, 2018). Está localizada em vários locais de Portugal, interligados entre si por uma infraestrutura de redes de última geração.

Figura 4.7 – Logótipo da INCD.

(Fonte: INCD, 2018)

A INCD faz parte de uma rede de infraestruturas internacionais mais vasta, com as quais partilha recursos e permite que investigadores, em Portugal, possam usar recursos computacionais existentes noutros países agregados. A infraestrutura portuguesa colabora com o European Grid Infrastructure (EGI), a infraestrutura ibérica IBERGRID, e o Worldwide LHC Computing Grid (WLCG) (INCD, 2018).

Figura 4.8 – Aspeto geral das infraestruturas de hardware ao serviço da INCD.

(Fonte: INCD, 2018)

4.3.2. Serviços

A principal vocação da INCD é dar apoio a investigadores, doutorandos, mestrandos e comunidade científica em geral, em projetos nacionais ou internacionais, com grande exigência de cálculos, dimensão de dados, capacidade de processamento e espaço de armazenamento (INCD, 2018). Está vocacionada para a comunidade servida pela RCTS (Rede Ciência, Tecnologia e Sociedade), nomeadamente, centros de investigação e estabelecimentos do ensino superior (FCT-FCCN, 2017). O acesso faz-se através de um registo realizado no sítio da INCD, na *Internet* (Figura 4.9).

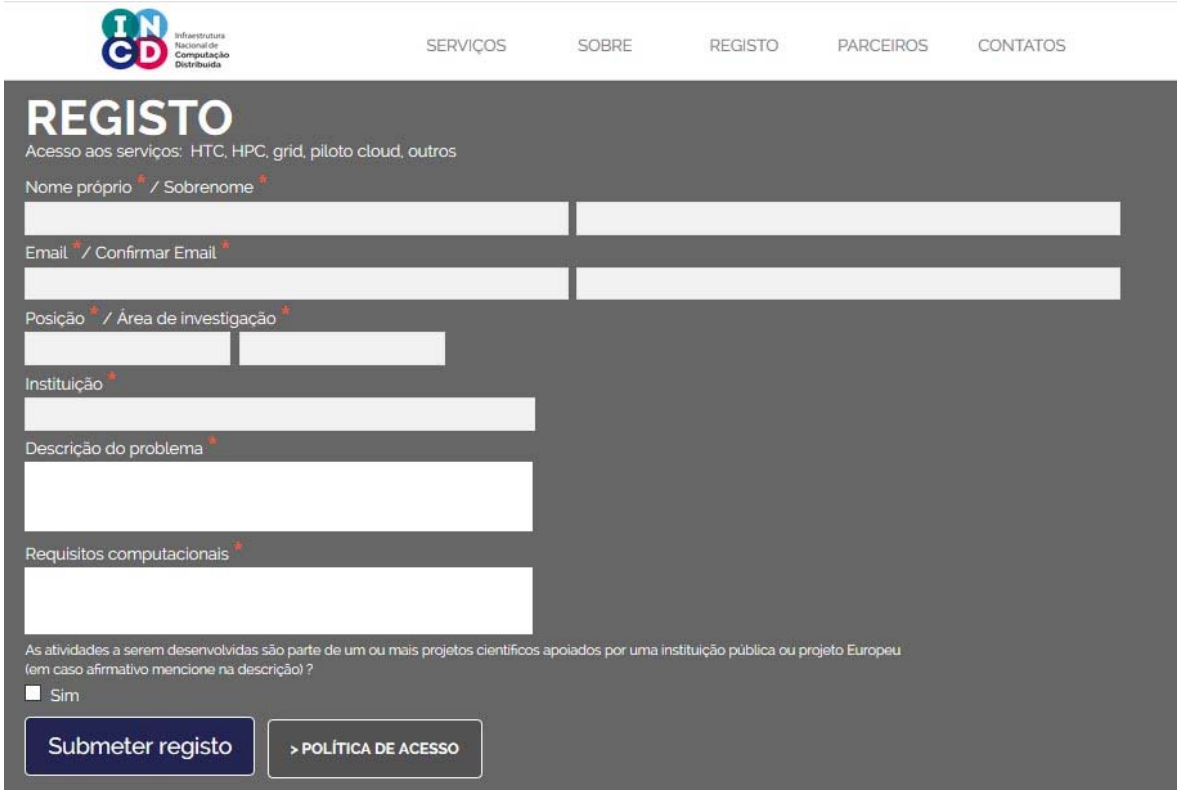

Figura 4.9 – Página de registo para acesso aos serviços da INCD.
(Fonte: INCD, 2018)

Disponibiliza serviços de *cloud computing*, processamento sequencial de elevado débito (HTC) e processamento paralelo de elevado desempenho (HPC) (FCT-FCCN, 2018).

O serviço de *cloud computing* está numa fase piloto. É baseado em *OpenStack*, permitindo maior flexibilidade na gestão dos recursos computacionais. O âmbito deste serviço é assistir

projetos que não se enquadrem nos paradigmas de computação HTC ou HPC atualmente suportados. É possível pedir acesso para realização de testes (INCD, 2018).

O HTC disponibiliza aos utilizadores nacionais, através de submissão direta ou via *Grid Middleware*, um processamento sequencial de elevado débito. A gestão dos recursos computacionais é realizada através de um sistema de *batch* (INCD, 2018).

O HPC disponibiliza um processamento paralelo de elevado desempenho, através de sistemas *batch* equipados com baixa latência *Infiniband* ou Gigabit Ethernet. Este serviço é complementado por um sistema de ficheiros de elevado desempenho configurado para I/O paralelo intenso (INCD, 2018).

Também fornece serviços de armazenamento, virtualização e soluções de *hardware* especializado (FCT-FCCN, 2017).

4.3.3. Impacto previsto na comunidade científica

O impacto previsto desta infraestrutura será vasto. Em primeiro lugar, permite aos investigadores nacionais aprofundarem os seus projetos científicos e, aos estudantes, um maior conhecimento sobre computação distribuída, que será uma mais valia para o mundo empresarial que os venham a recrutar (FCT, 2014). Os colaboradores dos centros especializados aumentam os seus conhecimentos e fornecem informação especializada a quem necessitar. À medida que estes centros especializados crescem, criam mais emprego, que, mais tarde, se disseminará pela indústria ou por outras entidades públicas ou privadas (FCT, 2014).

Por exemplo, o LNEC fornece muitos e importantes serviços à sociedade, dependentes ou com recurso à INCD, em múltiplas áreas, nomeadamente, na hidrodinâmica, qualidade da água, derramamentos de óleo, simulações de trânsito, no comportamento e segurança das barragens, etc (FCT, 2014). A infraestrutura, também, presta apoio à gestão do risco em trabalhos de engenharia civil. A INCD vem possibilitar a implementação de algumas diretivas comunitárias em Portugal. Os serviços de *cloud* abrirão horizontes, e novas

oportunidades, às PME (Pequenas e Médias Empresas) e aos serviços governamentais disponibilizados na *Internet*. (FCT, 2014)

4.3.4. Facilidades concedidas

O contacto com a INCD ocorreu por intermédio do professor doutor Luís Cavique, tendo proposto que os testes computacionais, no âmbito da presente dissertação, fossem realizados nas infraestruturas dessa entidade, devido ao enorme volume de dados a processar. Após reunião com os representantes da INCD, Mário David e João Pina, foi disponibilizada uma máquina virtual com acesso remoto e dispondo das seguintes características:

- Sistema operativo instalado – Ubuntu 16.04 LTS;
- 1 processador com 2 núcleos;
- 8 Gigabytes de memória RAM;
- 2 TeraBytes de espaço em disco.

O acesso, através do protocolo SSH, foi disponibilizado de imediato e nunca houve problemas com o mesmo. Esta experiência reforçou o facto da INCD ser uma infraestrutura sólida, muito bem construída, mantida e coordenada.

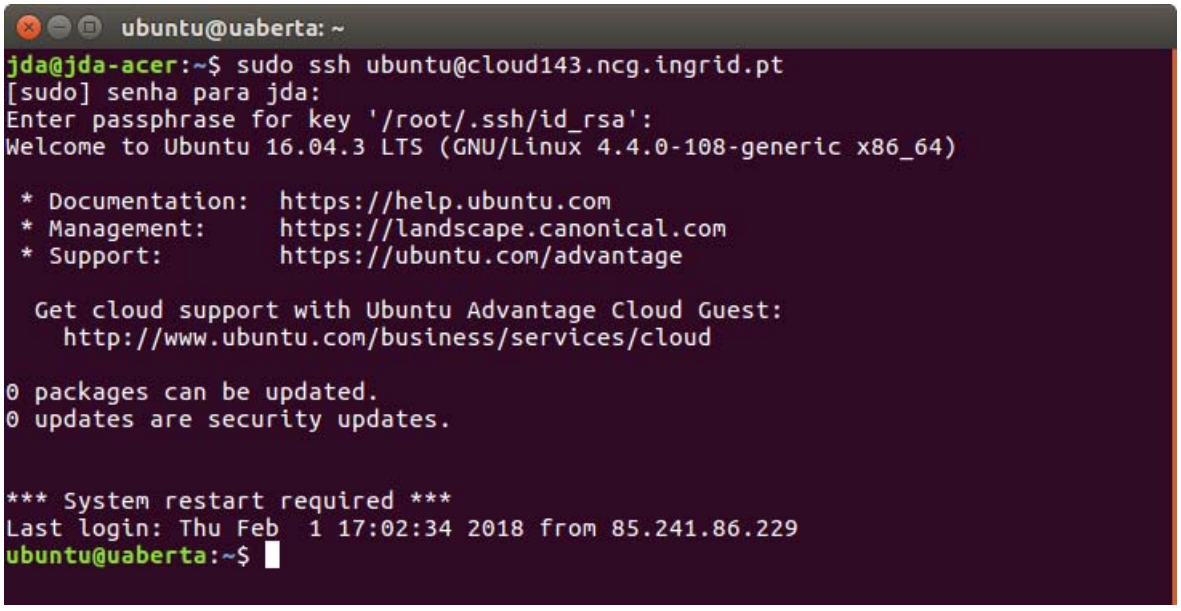

Figura 4.10 – Terminal para acesso remoto aos serviços disponibilizados pela INCD.

4.4. Estratégia na implementação do LAID

Nesta secção, pretende-se descrever o pseudocódigo que serve de orientação para o desenvolvimento do código.

O processo de implementação do algoritmo LAID é composto por três fases sequenciais:

1. Construção de bases de dados com conjuntos de dados artificiais para treino e teste;
    1.1. Criação da estrutura das bases de dados;
    1.2. Preenchimento das bases de dados com conjuntos de dados artificiais, de forma a cumprirem uma lista de requisitos;
    1.3. Inserir redundâncias e inconsistências nos dados;

2. Seleção de atributos, pela aplicação do algoritmo LAID aos conjuntos de dados de treino, dividido em duas etapas;
    2.1. Construção da matriz disjunta;
    2.2. Processo iterativo de seleção dos atributos;

3. Validação dos resultados obtidos, com recurso aos conjuntos de teste;
    3.1. Determinação da matriz confusão;
    3.2. Cálculo da taxa de cobertura e do índice *Kappa-statistic*.

O conjunto de dados é colocado numa matriz, onde as linhas correspondem às observações e as colunas aos atributos (as últimas colunas guardam os atributos *jnsq* e o atributo classe).

Embora os ficheiros HDF5 permitam comprimir os *datasets*, esta opção não é usada para não penalizar a performance e porque o espaço em disco não é uma questão crítica. Quanto à compartimentação dos *datasets* (*chunks*), após muitos testes realizados, a melhor opção foi compartimentar por linhas, ou seja, no *dataset* dos dados, definir *chunks* de dimensão 1×(número de atributos + número de atributos *jnsq* + 1 do atributo classe) e, no *dataset* da

matriz disjunta, definir *chunks* de dimensão 1×(número de atributos + número de atributos *jnsq*).

No *dataset* dos dados, os metadados são usados para guardar informação do número de atributos, de observações, de atributos *jnsq* e da quantidade de valores distintos da classe).

4.4.1. Construção da matriz disjunta

A construção da matriz disjunta (*M*) é antecedida por um pré-processamento para eliminar as redundâncias, identificar as inconsistências e preencher os valores dos atributos *jnsq*. Para tal, optou-se por ordenar as linhas (observações) da matriz do conjunto de dados.

Uma vez ordenada, é fácil detetar as redundâncias, bastando comparar com a linha anterior. Se uma linha é igual à anterior (incluindo a classe), está detetada uma redundância. A linha marcada como redundante é a primeira do par detetado, porque a segunda servirá para comparação com a linha seguinte.

Da mesma forma, é fácil encontrar inconsistências. A cada linha é associado o grau de inconsistência zero. Se uma linha é igual à anterior, exceto no valor da classe, está detetada uma inconsistência de grau 1 que fica associada a esta linha. No caso de conjuntos de dados com classe multivalor, se a linha seguinte, ainda, é igual, exceto na classe, o grau dessa inconsistência é incrementado uma unidade e este valor fica associado à linha. O número de colunas *jnsq* efetivas depende do grau de inconsistência máximo encontrado. Após a determinação deste valor, são preenchidos os valores *jnsq* das linhas de cada inconsistência, com o número binário correspondente ao grau de inconsistência associado à linha respetiva.

A matriz disjunta é determinada comparando cada linha da matriz dos dados, com todas as linhas seguintes. Desta forma, conseguem-se comparar todas as linhas duas-a-duas. Dado que, o resultado da comparação dos pares de linhas só é registado na matriz disjunta se pertencerem a classes diferentes, o número de linhas da matriz disjunta, no caso de classes booleanas, é igual ao produto do número de linhas classe 1 pelo número de linhas classe 0.

O algoritmo 1 retorna a matriz disjunta a partir do conjunto de dados, incluindo o atributo classe:

Algoritmo 1: Matriz disjunta

Parâmetros: conjunto de dados $D = \{O, A \cup C\}$
Retorno: matriz disjunta M

1. eliminar observações redundantes
2. identificar as inconsistências e o respetivo grau de inconsistência
3. para $i = 0$ até $\lceil log_2(1+ \text{máximo(grau de inconsistência)}) \rceil$:
   3.1. adicionar um atributo *jnsq*
4. preencher as colunas dos atributos *jnsq*
5. construir a matriz disjunta M

4.4.2. Seleção dos atributos relevantes

A seleção dos atributos é realizada com recurso à matriz disjunta. O processo inicia-se pela contagem do número de elementos não nulos em cada coluna da matriz. É selecionada a coluna à qual corresponde a maior contagem. São eliminadas todas as linhas cujo elemento da coluna selecionada é 1. A posição desta coluna é guardada na lista de atributos selecionados e retirada da matriz M. O processo é repetido até que a contagem do número de elementos não nulos seja zero para todas as colunas restantes.

O algoritmo 2 retorna o conjunto dos atributos selecionados a partir da matriz disjunta construída no algoritmo 1:

Algoritmo 2: Seleção dos atributos

Parâmetros: matriz disjunta M
Retorno: S = colunas selecionadas

1. $S = \{\}$
2. continuar = verdade
3. enquanto continuar:
   3.1. para $i = 0$ até número de colunas de M:
      3.1.1. contagem(i) = número de elementos não nulos na coluna i
   3.2. se todos os elementos de contagem são nulos:
      3.2.1. continuar = falso
   3.3. senão:
      3.3.1. coluna selecionada = coluna cuja contagem é máxima
      3.3.2. eliminar as linhas de M cujo elemento da coluna selecionada não é nulo
      3.3.3. eliminar a coluna selecionada em M
      3.3.4. acrescentar a identificação da coluna selecionada à lista S

4.4.3. Avaliação do conjunto de atributos selecionados

Determinar a matriz confusão é o primeiro passo para poder avaliar o resultado da seleção dos atributos. São construídas duas matrizes, uma com os dados de treino, que foram usados no processo de seleção, e outra com os dados de teste, mas descartando, em ambas, as colunas correspondentes aos atributos não selecionados. Para cada linha da matriz de teste, calculam-se as médias das distâncias de Hamming, uma para cada valor da classe, relativamente às linhas da matriz de treino respetivas. A classe prevista para uma linha de teste corresponde à classe de treino com menor distância média. Comparando com o valor real da classe dessa linha de teste, incrementa-se o respetivo elemento da matriz confusão. Após percorrer todas as linhas de teste, a matriz confusão está finalizada, seguindo-se o cálculo da taxa de cobertura, da taxa esperada e do índice *Kappa-statistic*.

Recorrendo ao conjunto de dados de treino usados na seleção dos atributos, a um conjunto de dados de teste para permitir construir a matriz confusão e à lista dos atributos selecionados, o algoritmo 3 retorna os indicadores que permitem avaliar o processo de seleção:

Algoritmo 3: Avaliação do conjunto dos atributos selecionados

Parâmetros: conjunto de dados de treino $D = \{O, A \cup C\}$;
conjunto de dados de teste $Dt = \{Ot, A \cup C\}$;
lista de atributos selecionados S

Retorno: (taxa de cobertura, taxa esperada, *kappa statistic*)

1. eliminar as colunas de D que não pertencem a S, exceto a classe, ficando assim definido $D = \{O, S \cup C\}$
2. eliminar as colunas de Dt que não pertencem a S, exceto a classe, ficando assim definido $Dt = \{Ot, S \cup C\}$
3. determinar a classe prevista para cada linha de Dt
4. construir a matriz confusão
5. calcular a taxa de cobertura, a taxa esperada e o índice *kappa statistic*

A implementação do código foi desenvolvida em linguagem *Python*3, com o pacote *Numpy* instalado. Todo o código está disponível na *Internet*, no sítio http://infapol.com/jda .

5. Resultados computacionais

Neste capítulo, o método LAID, o foco deste trabalho, é aplicado a vários conjuntos de dados artificiais. Pretende-se testar o algoritmo em várias condições: variação do número de observações e de atributos; classes booleanas ou multivalor; forma de determinar o valor da classe; existência de atributos relevantes. A implementação do algoritmo LAID é codificada em *Python*, os dados são armazenados no formato HDF5 e o processamento é efetuado na INCD, com acesso remoto, via SSH. São registados os atributos selecionados e a respetiva quantidade, o espaço em disco ocupado pelos ficheiros de dados e os tempos de processamento. A avaliação dos resultados, é feita através da matriz confusão e das medidas taxa de cobertura e *Kappa-statistic*.

5.1. Conjuntos de dados para seleção de atributos

Os primeiros testes de um algoritmo de seleção de atributos devem ser realizados com recurso a dados artificiais, uma vez que são conhecidos os atributos relevantes. Acresce o facto de se poderem alterar as condições experimentais, permitindo obter conclusões mais seguras (BOLÓN-CANEDO *et al.*, 2013). Outro ponto a favor é poder usar-se todo o conjunto de dados para selecionar os atributos e criar novos para validar a seleção obtida, sem necessidade de estratégias como a *cross-validation*.

Na realização dos testes ao algoritmo LAID, pretendiam-se conjuntos de dados que abarcassem várias características, como: grande número de atributos de valor booleano; existência de atributos irrelevantes; redundâncias (dados repetidos); classes multivalor; inconsistências (associadas a valores de atributos ou de classe errados); um número de atributos muito maior que o de observações; uma matriz de valores de atributos esparsa (valores zero em número muito maior que valores 1). Estas características podem complicar e influenciar bastante a seleção de atributos. (BOLÓN-CANEDO *et al.*, 2013)

Não se consideraram observações com valores de atributos desconhecidos. Caso existissem, essas observações deveriam ser eliminadas, como acontece com as observações redundantes.

O facto de os atributos, muitas vezes, corresponderem a características, ou sintomas, que são detetados, ou não, leva a que os seus valores sejam booleanos. A classe multivalor pode estar associada a vários graus de uma doença. Esses valores deverão ser inteiros consecutivos, com o valor inicial igual a zero. As inconsistências podem ser consequência de resultados errados, falsos testemunhos, ou bivalência de conclusões. As mesmas características, ou sintomas, levam a conclusões iguais, ou seja, valores de classe iguais, originando as redundâncias.

5.1.1. Conjuntos de dados artificiais da categoria PTZ

Um conjunto de dados da categoria PTZ (Primeiros Todos Zero) tem como características:

- Classe booleana;
- 50 redundâncias;
- 50 inconsistências;
- Matriz de dados esparsa, onde cerca de 20% dos dados têm valor 1;
- Os primeiros 100 atributos de uma observação têm o valor 0;
- Se todos os valores da segunda centena de atributos de uma observação são 1, à classe associada é atribuído o valor 1, caso contrário, é o valor 0, ou seja, o valor da classe é determinado pelo valor lógico da condição: $a_{100} \wedge a_{101} \wedge \ldots \wedge a_{198} \wedge a_{199}$, onde a_i é o valor do atributo colocado na posição i;
- Aproximadamente, 85% das observações têm valor da classe 1;
- Valores dos atributos obtidos aleatoriamente, condicionados ao cumprimento dos requisitos anteriores.

Numa situação real, esta categoria de dados pode simular um diagnóstico de determinada doença, numa população onde a doença prevalece (razão pela qual 85% das observações são classe 1). A classe corresponde ao resultado do teste clínico de despiste da doença (doente ou não doente), os atributos são uma lista exaustiva de sintomas e as observações indicam se cada paciente apresenta, ou não, cada um dos sintomas e qual o resultado do teste de despiste realizado. Neste caso, a seleção de atributos representa a escolha de quais os sintomas relevantes para o diagnóstico da doença. Uma redundância ocorre quando dois pacientes têm igual sintomatologia e o mesmo diagnóstico. Uma inconsistência ocorre

quando dois pacientes têm diferentes diagnósticos resultantes dos testes clínicos realizados (um é indicado como doente, o outro é não doente), mas ambos apresentam os mesmos sintomas. A razão pela qual os primeiros 100 atributos são zero, para todas as observações, é garantir a existência de atributos, inquestionavelmente, irrelevantes.

As características PTZ, assim definidas, permitem a comparação com os resultados obtidos por Cavique *et al.* (2018), onde foram gerados conjuntos de dados artificiais, com classe booleana, 2000 observações, das quais 1700 eram de classe 1 (85%). O número de atributos variou, tendo sido gerados conjuntos com 100, 500, 1000, 2000 e 5000 atributos. O código foi escrito em C, com processamento sequencial, compilados através da aplicação GCC/Dev-C++, na infraestrutura INCD, onde foi disponibilizado um processador Intel Core Duo 3,0 GHz, com 4 GB de RAM e o sistema operativo Windows 10. Foi gerado um sexto conjunto com 1 000 000 de atributos, mas com processamento paralelo, em 10 máquinas.

5.1.2. Conjuntos de dados artificiais da categoria ARC

Um conjunto da categoria ARC (Atributos Relevantes Conhecidos), tem como características:

- Classe booleana ou ternária com domínio {0, 1, 2};
- 50 redundâncias;
- 100 inconsistências;
- O valor da classe é 1 se a condição seguinte for verdadeira:
  $(a_{1000} \wedge a_{2000} \wedge a_{3000} \wedge a_{4000}) \vee (a_{2000} \wedge a_{4000} \wedge a_{6000} \wedge a_{8000}) \vee (a_{3000} \wedge a_{6000} \wedge a_{9000})$
  onde a_i é o valor do atributo colocado na posição i;
- O valor da classe é 2 se a classe for multivalor e a condição seguinte for verdadeira:
  $(a_{100} \wedge a_{200} \wedge a_{300} \wedge a_{400}) \vee (a_{200} \wedge a_{400} \wedge a_{600} \wedge a_{800}) \vee (a_{300} \wedge a_{600} \wedge a_{900})$
- O valor da classe é 0 se as duas condições anteriores forem falsas.
- Matriz de dados esparsa, onde cerca de 20% dos dados têm valor 1;
- Aproximadamente, 50% das observações têm valor da classe 1, se a classe é booleana. Se a classe é multivalor, 30% têm valor da classe 1 e 20% têm valor da classe 2. Os restantes 50% têm valor da classe 0;
- Valores dos atributos obtidos aleatoriamente, cumprindo os requisitos anteriores.

Neste caso, os atributos relevantes são conhecidos e são eles que determinam o valor da classe. O número de inconsistências é relativamente grande (num conjunto de 2000 observações, 100 inconsistências (5%) podem corresponder a 10% de observações inconsistentes).

O ARC pode simular uma situação idêntica à indicada no PTZ, onde, no caso da classe ser booleana, existem três conjuntos de sintomas que permitem diagnosticar uma certa doença. Metade da população está doente.

Se a classe é multivalor, existem três conjuntos de sintomas que permitem diagnosticar uma certa doença, e outros três conjuntos que permitem diagnosticar uma segunda doença. Metade da população não apresenta qualquer uma das doenças. Além das situações descritas em PTZ, que originam redundâncias e inconsistências, estas, também, podem surgir quando duas observações tornam ambas as condições verdadeiras, podendo corresponder ao valor da classe 1 ou 2.

5.1.3. Conjuntos de dados artificiais da categoria SVA

Um conjunto de dados da categoria SVA (Soma dos Valores dos Atributos) tem as seguintes características:

- Classe booleana;
- 10 observações redundantes;
- Ausência de inconsistências;
- O valor da classe é função da soma dos valores dos dados de uma observação:
  - soma < 20% do total de atributos => valor da classe = 0;
  - soma ≥ 20% do total de atributos => valor da classe = 1;
- Matriz de dados esparsa, onde cerca de 20% dos dados têm valor 1;
- Aproximadamente, 40% das observações têm valor da classe 1;
- Valores dos atributos obtidos aleatoriamente, cumprindo os requisitos anteriores.

O objetivo deste conjunto de dados é testar o algoritmo quando o valor da classe não depende de determinados atributos, mas do seu conjunto, na ausência de inconsistências e quando a

quantidade de observações, correspondentes a cada valor da classe, não é, significativamente, diferente.

Esta categoria de conjunto de dados pode simular uma lista de vocábulos (atributos), menos conhecidos, de determinada língua que são, ou não, usados por uma amostra de indivíduos (observações). Se um indivíduo usar mais de 20% desses vocábulos, então tem um vocabulário rico (classe = 1), nos restantes casos, o vocabulário é restrito (classe = 0). O resultado da seleção de atributos indica quais os vocábulos mais representativos para aplicar o estudo à totalidade da população, sem necessidade de um inquérito tão extenso e exaustivo. As redundâncias correspondem a indivíduos que conhecem os mesmos vocábulos. Pela forma como é definido o valor da classe, é impossível ocorrerem inconsistências.

5.1.4. Conjuntos de dados artificiais gerados

Para testar o algoritmo LAID, geraram-se seis conjuntos de dados artificiais de cada categoria, diferenciados pelo número de observações e de atributos (Tabela 5.1). As categorias ARC2 e ARC3 cumprem os requisitos de ARC, tendo a primeira uma classe booleana e a segunda uma classe ternária. O objetivo é comparar a performance da implementação do LAID em conjuntos com diferentes características e dimensões.

Os conjuntos “grande” e “extra grande” pretendem testar o algoritmo LAID para conjuntos de muito grande dimensão, com matriz esparsa, onde o número de atributos é muito maior que o número de observações. As características do conjunto PTZx são semelhantes às do conjunto de dados artificiais CAVx, usado por Cavique *et al.* (2018) (ver Tabela 5.5).

Os conjuntos “extra grande”, “grande”, médio” e “pequeno” divergem, apenas, no número de atributos. Os conjuntos “alternativos” e “pequeno” servem para analisar o efeito da variação do número de observações nos testes ao algoritmo.

Para cada conjunto de treino, usado na seleção de atributos, é criado um conjunto de teste, para avaliação do resultado obtido (Tabela 5.1). Têm as mesmas características dos conjuntos de treino respetivos, mas todos com 1 000 observações. O conjunto de teste “pequeno” é partilhado com os conjuntos “alternativos” porque têm o mesmo número de atributos.

Tabela 5.1 – Conjuntos de dados artificiais gerados.

Categoria	***Conjunto de treino***	***Sigla***	***Nº de observações***	***Nº de atributos***	***Conjunto de teste***	***Nº de observações***
PTZ	PTZ extra grande	PTZx	2 000	1 000 000	PTZx_t	1 000
	PTZ grande	PTZg	2 000	500 000	PTZg_t	1 000
	PTZ médio	PTZm	2 000	100 000	PTZm_t	1 000
	PTZ pequeno	PTZp	2 000	10 000	PTZp_t	1 000
	PTZ alternativo	PTZa	10 000	10 000		
	PTZ alternativo extra	PTZax	20 000	10 000		
ARC2	ARC2 extra grande	ARC2x	2 000	1 000 000	ARC2x_t	1 000
	ARC2 grande	ARC2g	2 000	500 000	ARC2g_t	1 000
	ARC2 médio	ARC2m	2 000	100 000	ARC2m_t	1 000
	ARC2 pequeno	ARC2p	2 000	10 000	ARC2p_t	1 000
	ARC2 alternativo	ARC2a	10 000	10 000		
	ARC2 alternativo extra	ARC2ax	20 000	10 000		
ARC3	ARC3 extra grande	ARC3x	2 000	1 000 000	ARC3x_t	1 000
	ARC3 grande	ARC3g	2 000	500 000	ARC3g_t	1 000
	ARC3 médio	ARC3m	2 000	100 000	ARC3m_t	1 000
	ARC3 pequeno	ARC3p	2 000	10 000	ARC3p_t	1 000
	ARC3 alternativo	ARC3a	10 000	10 000		
	ARC3 alternativo extra	ARC3ax	20 000	10 000		
SVA	SVA médio	SVAm	2 000	100 000	SVAm_t	1 000
	SVA pequeno	SVAp	2 000	10 000	SVAp_t	1 000
	SVA alternativo	SVAa	10 000	10 000		

Da categoria SVA foram gerados, apenas, três conjuntos de dados, porque, não tendo atributos relevantes, destinam-se a servir de controle aos testes efetuados ao algoritmo LAID. Os gráficos disponibilizados não apresentam dados relativos aos conjuntos SVA.

5.2. Tempos e outros indicadores de desempenho

Após aplicar o algoritmo LAID a cada um dos conjuntos de dados artificiais, registaram-se os resultados obtidos. A Tabela 5.2 mostra os principais indicadores do processo de seleção de atributos, nomeadamente, o espaço em disco dos ficheiros HDF5, o número de atributos selecionados e o tempo despendido no processamento.

Tabela 5.2 – Características dos conjuntos de treino e resultados da seleção de atributos.

Conjunto de dados de treino	Observações	Atributos	Redundâncias	Inconsistências	Nº de classes	Espaço em disco inicial (MB) [1]	Espaço em disco final (GB) [2]	Atributos selecionados	Nº de atributos selecionados	Tempo (min) [3]
PTZx	2 000	1 000 000	50	50	2	1907	497	**134, 179**, 275, 89513, 216246, 239053, 370301, 437858, 597497, 716979, 929762, 947482, 986177	13	707
PTZg		500 000				954	248	**124, 130**, 75690, 82884, 115016, 126723, 137635, 190456, 226903, 280460, 285711, 333785, 383665, 456163	14	357
PTZm		100 000				191	50	**106, 179**, 247, 534, 1158, 3394, 83 4, 10229, 26811, 34959, 39401, 44059, 74602, 76610	14	50
PTZp		10 000				20	5	**140, 183**, 321, 1484, 1775, 2205, 2496, 3474, 3505, 4372, 4610, 4682, 7593, 8174, 8713	15	10
PTZa	10 000	10 000				96	121	**119, 143, 149, 165**, 274, 386, 607, 609, 1841, 3626, 4396, 5441, 5447, 5759, 6161, 7626, 8116, 9277	18	420
PTZax	20 000					192	480	**134, 140, 146**, 554, 3062, 3749, 4668, 5552, 5924, 6087, 6428, 6564, 6636, 7602, 7922, 8387, 9114, 9310, 9562	19	1997
ARC2x	2 000	1 000 000	50	100	2	1908	888	**2000**, **3000**, **6000**, 42673, 49772, 80580, 935561, 135655, 320657, 542813, 608776, 685945, 779972, 824923, 834223, 866673	16	1231
ARC2g		500 000				954	444	1112, **2000**, **3000**, **4000**, **6000**, 8530, 41877, 101468, 103151, 114315, 124184, 196984, 234310, 359859, 436825	15	640
ARC2m		100 000				191	89	0, **2000**, **3000**, **4000**, **6000**, 14307, 16712, 18362, 20668, 27682, 34618, 35440, 38237, 76011, 79494, 97986	16	104
ARC2p		10 000				20	9	217, 426, 901, **2000**, 2867, **3000**, 3355, 3420, **4000**, 4253, 5524, **6000**, 6117, 7389, 8080, 8558	16	23
ARC2a	10 000	10 000				96	232	165, 356, 1516, **2000**, **3000**, **4000**, 4896, 5010, 5174, 5722, 5800, 5803, **6000**, 7150, 7282, 7639, **8000**, 8692, **9000**, 9093, 9305	21	690
ARC2ax	20 000					192	932	151, 210, 889, **1000**, 1828, **2000**, **3000**, 3560, 3612, 3624, **4000**, 5573, 5603, 5675, 5697, 5947, **6000**, 6136, 6462, **8000**, 8528, **9000**, 9894	23	2916

Conjunto de dados de treino	Observações	Atributos	Redundâncias	Inconsistências	Nº de classes	Espaço em disco inicial (MB) [1]	Espaço em disco final (GB) [2]	Atributos selecionados	Nº de atributos selecionados	Tempo (min) [3]
ARC3x	2 000	1 000 000	50	100	3	1907	1117	**200, 300, 400, 600, 800, 2000, 3000, 4000, 6000**, 176767, 304081, 443389, 636546, 641079, 993850, 999273	16	2103
ARC3g		500 000				954	557	**300, 400, 600, 3000, 4000, 6000, 9000**, 12620, 46512, 184104, 185032, 271936, 308324, 335148, 371948, 390127, 458752	17	1001
ARC3m		100 000				191	112	**200, 300, 400, 600**, 1492, **2000**, **3000**, **4000**, **6000**, 15393, 16671, 27984, 38506, 43959, 46009, 67843, 93239	17	188
ARC3p		10 000				20	12	**100**, **200**, **300**, **400**, 433, **600**, 1076, 1206, 1861, **2000**, 2055, **3000**, 3484, **4000**, 5219, **6000**, 6872, **9000**	18	34
ARC3a	10 000	10 000				96	288	**200**, **300**, **400**, **600**, 666, **900**, **1000**, **2000**, 2977, **3000**, **4000**, 4545, 4570, **6000**, 6056, 6683, 7655, 7664, 8560, **9000**, 9484, 9776	22	1222
ARC3ax	20 000					192	1156	27, 180, **200**, **300**, **400**, **600**, **800**, **900**, 1682, **2000**, 2523, 2628, **3000**, 3297, **4000**, 4187, 5175, **6000**, 6215, 6381, 6546, 7561, **8000**, 8137, 8353, **9000**	26	7144
SVAm	2 000	100 000	10	0	2	191	89	0, 705, 1290, 5194, 11560, 13742, 17140, 19085, 36978, 39584, 43792, 44548, 56471, 57938, 78356, 86277, 93577	17	188
SVAp		10 000				20	9	0, 829, 1056, 1890, 2077, 3031, 3196, 3400, 3833, 4096, 4183, 4584, 4602, 4744, 5374, 5667, 6081, 9291	18	33
SVAa	10 000	10 000				96	225	150, 444, 535, 1525, 2711, 2976, 3601, 3963, 4047, 4516, 4523, 4693, 5187, 5674, 5902, 6052, 6169, 6593, 6943, 7031, 7157, 7284, 7376, 7950, 8624, 8872	26	1999

(1) Espaço em disco do ficheiro HDF5 antes da construção da matriz adjunta, em MegaBytes.
(2) Espaço em disco do ficheiro HDF5 após a construção da matriz adjunta, em GigaBytes.
(3) Tempo de processamento da implementação do LAID, na INCD, em minutos.

Numa primeira análise, destaca-se a eficiência das bases de dados HDF5 no armazenamento dos elementos. Por exemplo, o conjunto PTZx tem 2 000 observações, 1 000 000 atributos, um atributo *jnsq* (porque o número de valores diferentes da classe é 2) e um atributo classe. A quantidade de dados é 2 000×(1 000 000+1+1) = 2 000 004 000. Como cada dado ocupa um Byte (do tipo int8), o conjunto de dados ocupa 2 000 004 000 Bytes (1907 MegaBytes). O ficheiro inicial ocupa 2 000 023 041 Bytes (1907 MB), com os dados e a estrutura completa da base de dados. Neste caso, a dimensão dos metadados é inferior a 19 KB.

Ao preencher a matriz disjunta, o espaço ocupado em disco tem um aumento descomunal. Este facto evidencia que o grande custo computacional do algoritmo LAID reside na

construção e manuseamento da matriz disjunta. Retomando o exemplo do conjunto de dados PTZx, com 1950 linhas (às 2000 observações, há que retirar as 50 redundantes) e 1 000 001 colunas (1 000 000 atributos + 1 atributo *jnsq*), a dimensão máxima da matriz disjunta é $^{1950}C_2 \times 1000001$ = 1 900 276 900 275 = 1770 GB. No entanto, 85% das observações são classe 1 (1950×0,85=1658), logo 1950–1658=292 são classe 0 e as combinações são feitas entre linhas de classes diferentes. Além disso, o processo de inserção das 50 inconsistências corresponde a copiar os dados de uma observação para outra e trocar o valor da classe, na mesma proporção da distribuição das classes, logo, ao número de observações classe 1 há que retirar 50×(0,85–0,15)=35 observações que passam a ser classe 0. O espaço em disco do ficheiro final deverá rondar (1658–35)×(292+35)× 1000001 = 495 GB. A este resultado há que acrescentar os 1,9 GB iniciais. Este valor é, precisamente, o espaço em disco do ficheiro final (497 GB). Optou-se por não usar a compressão, possível neste tipo de base de dados, para não prejudicar o tempo de acesso aos dados e por o espaço disponível ser suficiente.

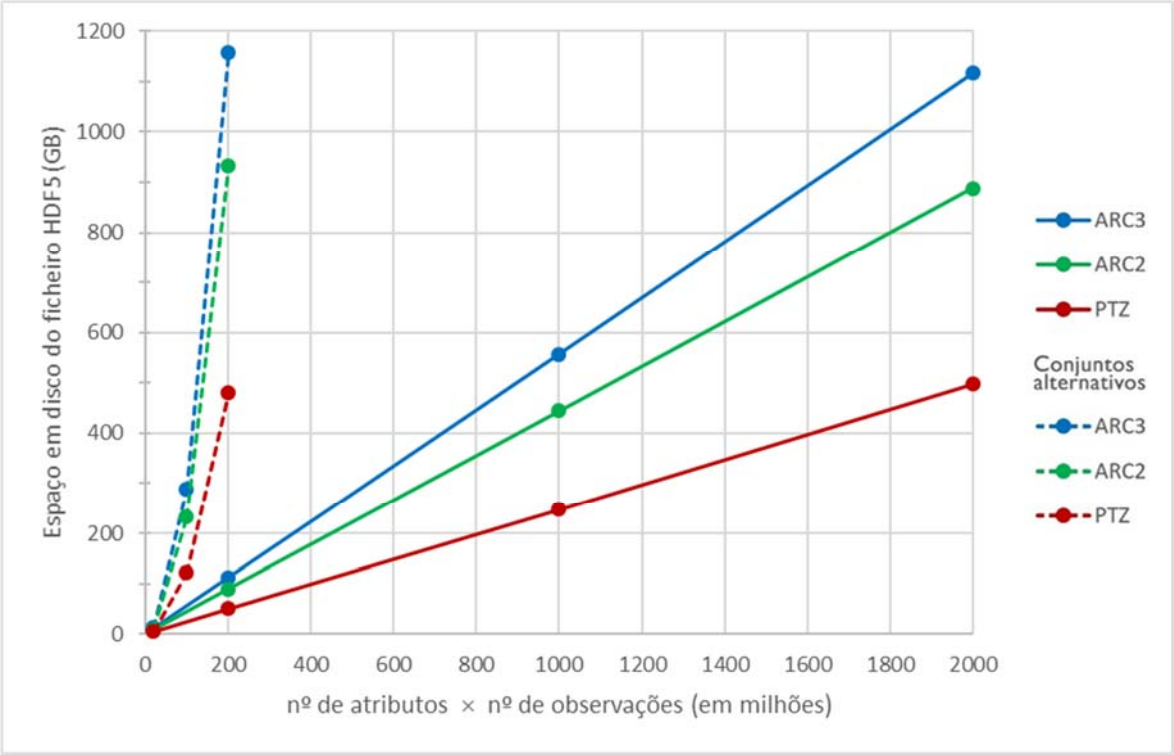

Figura 5.1 – Espaço em disco do ficheiro HDF5, em função da dimensão dos dados.

O espaço em disco dos ficheiros dos conjuntos ARC2 é superior ao dos conjuntos PTZ, porque, nos primeiros, as proporções de observações classe 1 e classe 0 são semelhantes (50%), enquanto nos PTZ, existe uma desproporção entre o número de observações classe 1 e classe 0, respetivamente, 85% e 15%. Nos conjuntos ARC3, a classe é ternária (assume um de três valores possíveis). Dada a forma como a matriz disjunta é construída, a sua dimensão aumenta com a cardinalidade do domínio de valores da classe.

Da mesma forma, é evidente que os conjuntos de dados "alternativos extra" ocupam um espaço em disco muito superior aos conjuntos "médios" com igual dimensão dos dados. Esta diferença deve-se, novamente, à construção da matriz disjunta, cuja dimensão é, particularmente, sensível à variação do número de observações, na ordem de grandeza $O(n^2)$. Quanto à variação dos atributos, a dimensão da matriz é $O(n)$.

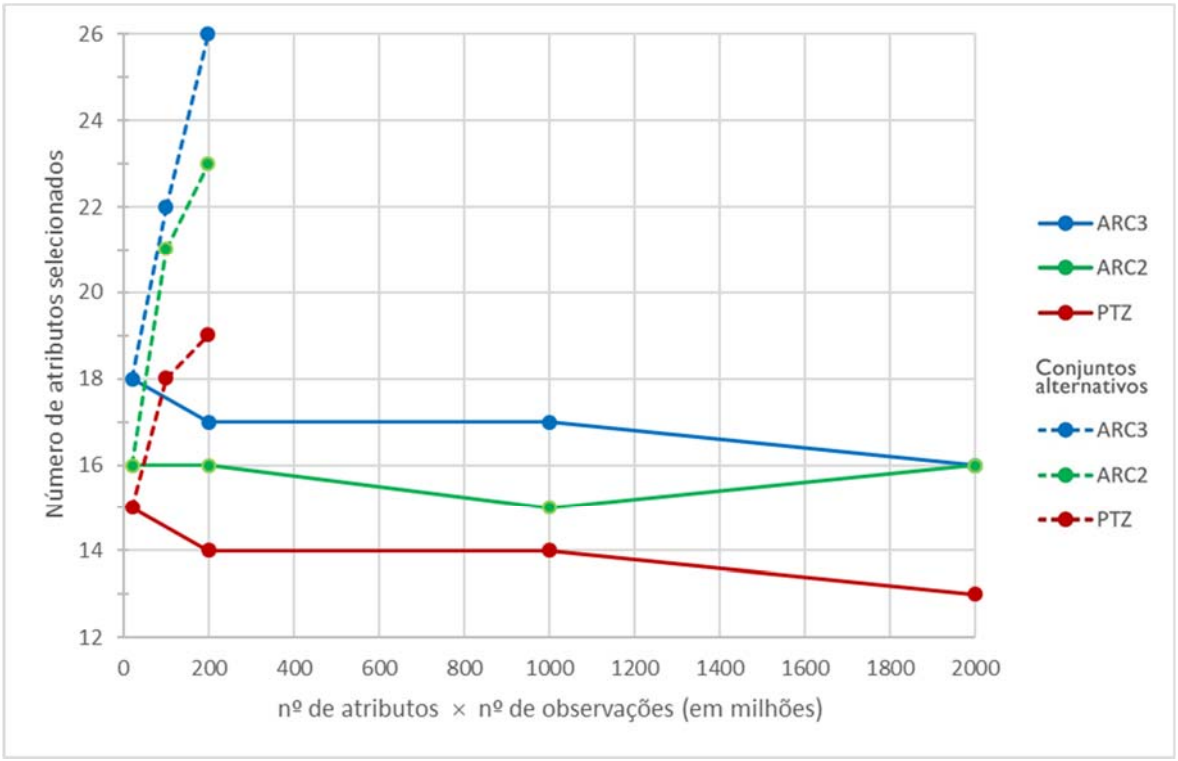

Figura 5.2 – Número de atributos selecionados, em função da dimensão dos dados.

No que respeita aos atributos selecionados, o algoritmo LAID consegue obter um número muito reduzido de atributos na solução final. Os conjuntos "alternativos" selecionam mais atributos que os restantes da mesma categoria. É de destacar o conjunto de dados PTZx.

Partindo de um milhão de atributos, a solução é composta por apenas 13 (0,0013%). Os conjuntos PTZ nunca selecionaram os primeiros cem atributos, que não distinguem quaisquer observações, mas selecionaram poucos atributos relevantes (entre a_{100} e a_{199}) porque, relativamente à escolha da classe, nada os diferencia. O resultado do conjunto ARC3x é significativo, porque, dos 16 atributos selecionados, 9 fazem parte dos atributos relevantes para a definição da classe. Mais significativo é o facto de, no conjunto ARC2ax, terem sido selecionados todos os atributos relevantes. No caso dos conjuntos SVA, cujo valor da classe não depende de determinados atributos, mas da soma dos valores de todos eles, é lógico que a solução final seja mais ampla. No entanto, seria de esperar muito mais atributos selecionados. A categoria SVA não está representada nos gráficos, porque, pelas suas características, o processo de seleção é quase aleatório.

Quanto ao tempo de processamento da implementação do algoritmo LAID, os resultados são considerados, relativamente, bons, fruto da otimização do código descrita no capítulo anterior. Verifica-se que, para cada categoria, o tempo tem uma variação quase linear com a variação do número de atributos. Em relação ao número de observações, a variação do tempo parece quadrática (conjuntos "alternativos"), à semelhança do espaço em disco.

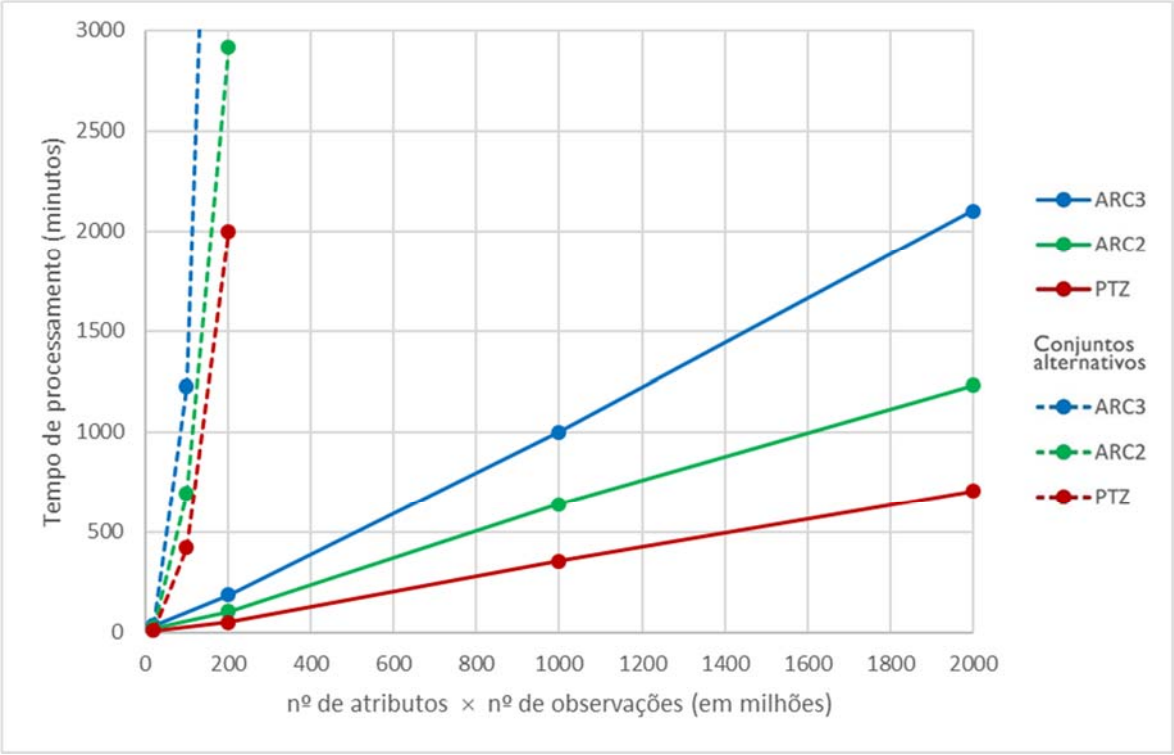

Figura 5.3 – Tempo de processamento, em minutos, em função da dimensão dos dados.

Como a maior parte das leituras na base de dados é realizada por linhas, optou-se por compartimentar (*chunk*) o *dataset* dos dados por linhas, aumentando a performance. No entanto, a leitura por colunas é penalizada. Acresce o facto de um maior número de observações originar uma matriz disjunta com muito mais linhas, logo, colunas de maior dimensão. Estas são as principais razões para o elevado tempo de processamento dos conjuntos "alternativos". O número de valores da classe, também, gera matrizes disjuntas maiores, originando maiores tempos de processamento dos conjuntos ARC3.

Dado os tempos de processamento dos conjuntos "alternativos extra" serem muito maiores, quando comparados com os conjuntos "médios", da mesma dimensão, associado ao, também, muito maior espaço em disco dos ficheiros HDF5 dos conjuntos "alternativos", vem reforçar a ideia de que o algoritmo LAID está vocacionado para conjuntos de dados de grande dimensão, onde o número de atributos é muito maior que o número de observações.

Estes tempos mostram a eficiência dos acessos aos dados no formato HDF5, quando comparados com os resultados obtidos por Cavique *et al.* (2018) (Tabela 5.5), com particular destaque para a comparação entre os conjuntos PTZx e CAVx.

5.3. Avaliação da qualidade dos resultados

Após a seleção dos atributos, segue-se a avaliação da qualidade dos resultados obtidos.

Na Tabela 5.3, observando as matrizes confusão, sobressaem os conjuntos SVA, porque a quase totalidade das observações de teste foram classificadas, pelo modelo obtido, com classe 0. Isto deve-se, muito provavelmente, ao valor da classe depender da soma de todos os atributos e as matrizes dos dados serem esparsas. Daí, a avaliação dos resultados para estes conjuntos ser "Fraca". O índice *Kappa-statistic* destes conjuntos é próximo de zero, indicando que o modelo obtido pouco difere de um modelo puramente aleatório (a taxa de cobertura é pouco maior que a esperada).

Tabela 5.3 – Avaliação da seleção de atributos obtida com base na matriz confusão.

Conjunto de dados de treino	Conjunto de dados de teste	Observações de teste	Atributos	Nº de classes	Nº de atributos selecionados	Matriz confusão	Taxa de cobertura	Taxa esperada	*Kappa-statistic*	Avaliação [1]
PTZx	PTZx_t	1 000	1 000 000	2	13	96 | 54 0 | 850	0,946	0,783	0,751	**B**
PTZg	PTZg_t		500 000		14	102 | 48 0 | 850	0,952	0,779	0,783	**B**
PTZm	PTZm_t		100 000		14	102 | 48 0 | 850	0,952	0,779	0,783	**B**
PTZp	PTZp_t		10 000		15	104 | 46 0 | 850	0,954	0,777	0,794	**B**
PTZa					18	133 | 17 0 | 850	0,983	0,757	0,930	**E**
PTZax					19	135 | 15 0 | 850	0,985	0,756	0,939	**E**
ARC2x	ARC2x_t	1 000	1 000 000	2	16	435 | 65 0 | 850	0,935	0,500	0,870	**E**
ARC2g	ARC2g_t		500 000		15	465 | 35 68 | 432	0,897	0,500	0,794	**B**
ARC2m	ARC2m_t		100 000		16	440 | 60 3 | 497	0,937	0,500	0,874	**E**
ARC2p	ARC2p_t		10 000		16	435 | 65 21 | 479	0,914	0,500	0,828	**E**
ARC2a					21	465 | 35 49 | 451	0,916	0,500	0,832	**E**
ARC2ax					23	480 | 20 78 | 422	0,902	0,500	0,804	**E**
ARC3x	ARC3x_t	1 000	1 000 000	3	16	413 | 27 | 60 53 | 242 | 5 6 | 3 | 191	0,846	0,369	0,756	**B**
ARC3g	ARC3g_t		500 000		17	397 | 49 | 54 0 | 294 | 6 0 | 19 | 181	0,872	0,355	0,801	**E**
ARC3m	ARC3m_t		100 000		17	420 | 14 | 66 58 | 238 | 4 5 | 6 | 189	0,847	0,371	0,757	**B**
ARC3p	ARC3p_t		10 000		18	451 | 31 | 18 0 | 294 | 6 35 | 1 | 164	0,909	0,378	0,854	**E**
ARC3a					22	459 | 19 | 22 47 | 244 | 9 0 | 0 | 200	0,903	0,378	0,844	**E**
ARC3ax					26	459 | 17 | 24 37 | 257 | 6 21 | 4 | 175	0,891	0,383	0,823	**E**

Conjunto de dados de treino	Conjunto de dados de teste	Observações de teste	Atributos	Nº de classes	Nº de atributos selecionados	Matriz confusão	Taxa de cobertura	Taxa esperada	*Kappa-statistic*	Avaliação [1]
SVAm	SVAm_t	1 000	100 000	2	17	599 | 1 360 | 40	0,639	0,592	0,116	**F**
SVAp	SVAp_t		10 000		18	600 | 0 369 | 31	0,631	0,594	0,092	**F**
SVAa					26	600 | 0 383 | 17	0,617	0,597	0,051	**F**

(1) **Ma** – má; **F** - fraca; **S** – sofrível; **Mo** – moderada; **B** – boa; **E** – excelente.

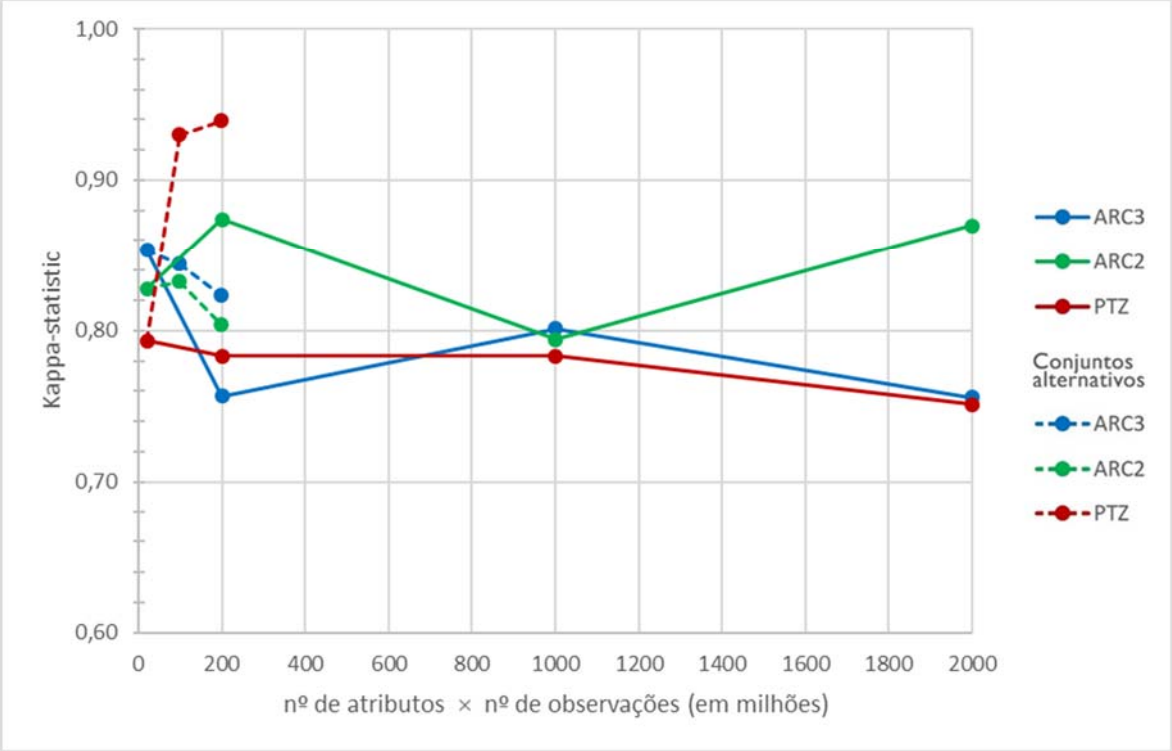

Figura 5.4 – Kappa-statistic, em função da dimensão dos dados.

Quanto aos conjuntos PTZ, é de notar que todas as observações de classe 1 foram classificadas corretamente pelo modelo. A avaliação dos conjuntos "alternativos" é superior ("Excelente"), fruto do maior número de observações do conjunto de treino e do menor número de atributos, e da grande assimetria entre a quantidade de observações classe 0 e

classe 1. É de registar o excelente índice *Kappa-statistic* para estes conjuntos, o que indica que o modelo consegue classificar os dados de teste com uma margem de erro residual. Dado o enorme número de atributos relativamente ao número de observações, os resultados obtidos nos testes do conjunto PTZx são muito bons. Consegue classificar corretamente 94,6% das observações de teste (taxa de cobertura). A taxa esperada é elevada devido à desproporção entre o número de observações classe 1 e classe 0.

Os resultados dos conjuntos ARC, também, são muito bons. No caso de ARC2x, o modelo consegue classificar corretamente 93,5% das observações de teste (taxa de cobertura), com um índice *Kappa-statistic* igual a 0,87. O modelo tem uma eficácia "Excelente" no processo de classificação destes conjuntos. Nos conjuntos ARC3, com um atributo classe multivalor, a eficácia não é, significativamente, penalizada.

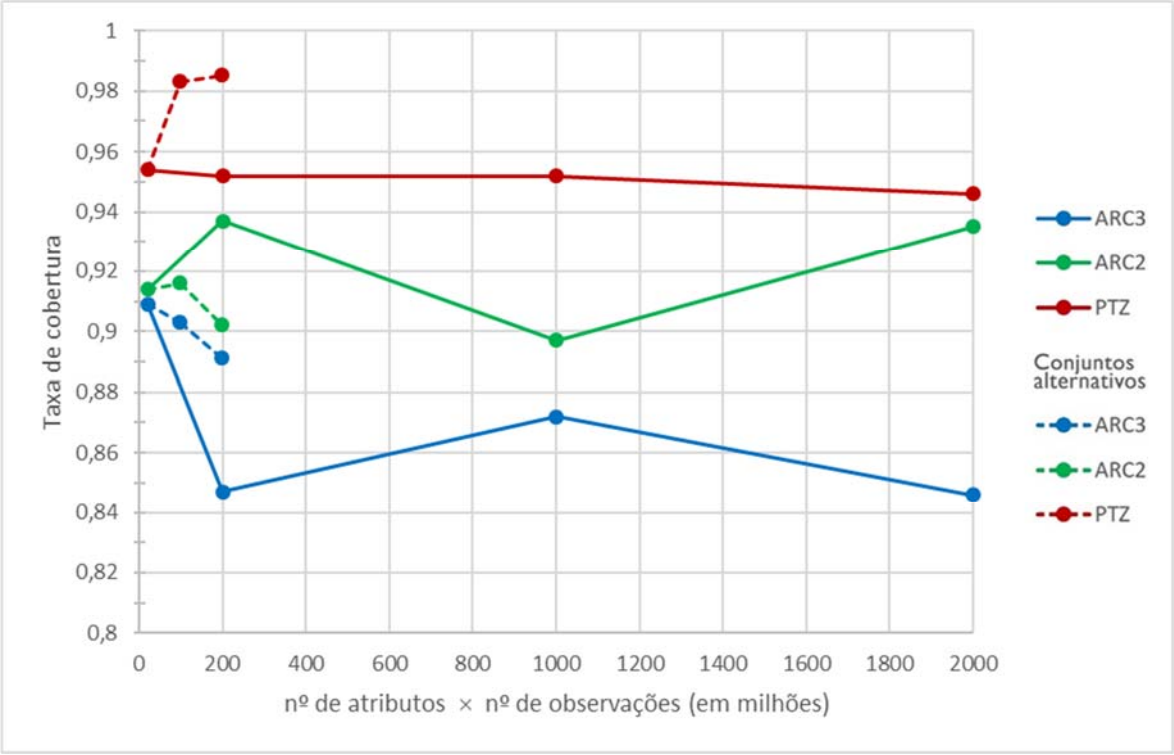

Figura 5.5 – Taxa de cobertura, em função da dimensão dos dados.

Para estudar o efeito da existência de inconsistências na qualidade dos resultados, vai-se considerar um novo conjunto PTZs, com os mesmos dados de PTZp, mas onde não foram acrescentadas inconsistências (Tabela 5.4).

Tabela 5.4 –Conjunto PTZs, igual a PTZp, mas sem inconsistências.

Conjunto de dados de treino	Observações	Atributos	Redundâncias	Inconsistências	Nº de classes	Atributos selecionados	Nº de atributos selecionados	Tempo (min) [1]	Matriz confusão	Taxa de cobertura	Taxa esperada	*Kappa-statistic*
PTZp	2 000	10 000	50	50	2	**140, 183**, 321, 1484, 1775, 2205, 2496, 3474, 3505, 4372, 4610, 4682, 7593, 8174, 8713	15	10	104 | 46 0 | 850	0,954	0,777	0,794
PTZs				0		**100, 140, 183**	3	8	131 | 19 0 | 850	0,981	0,758	0,921

(1) Tempo de processamento da implementação do LAID, na INCD, em minutos.

A diferença mais significativa está no número de atributos selecionados. No PTZs, o algoritmo selecionou, apenas, 3 atributos, todos relevantes para a definição da classe. Os indicadores de qualidade (taxa de cobertura e *Kappa-statistic*) de PTZs são superiores aos de PTZp. Estes resultados são expectáveis, visto não haver inconsistências, os atributos relevantes justificam o valor da classe sem ambiguidades e, como a matriz de dados é esparsa, um pequeno número de atributos relevantes é suficiente para classificar as observações.

Os conjuntos ARC "não alternativos", onde a proporção de inconsistências é maior (100 inconsistências podem equivaler a 200 observações inconsistentes, ou seja, 10% do total), não é detetada qualquer penalização relativamente aos conjuntos "alternativos", visto os valores de *Kappa-statistic*, serem semelhantes. Nos conjuntos PTZ, isto já não se verifica, mas a proporção de inconsistências é metade. A diferença, nestes conjuntos, deve-se à distribuição muito assimétrica dos valores da classe.

O acima exposto reforça uma das principais características do algoritmo LAID: lidar bem com as inconsistências.

O gráfico da taxa de cobertura (Figura 5.5) apresenta a mesma tendência de variação que o gráfico do índice *Kappa-statistic*. O índice *Kappa-statistic* faz uma correção da taxa de cobertura, tendo em conta a distribuição dos valores da classe e a aleatoriedade do modelo, logo é o melhor indicador da qualidade dos atributos selecionados, quando se pretende comparar conjuntos com diferentes características.

5.4. Argumentação final

As características dos conjuntos de dados gerados abrangem a relevância de alguns atributos na determinação do valor da classe, matrizes esparsas, um número de atributos muito maior que o de observações, classes booleanas ou multivalor, a existência de observações redundantes e outras inconsistentes. Foram testados cenários onde o número de atributos varia, sendo fixo o número de observações, e outros onde se fixa o número de atributos, variando a quantidade de observações (conjuntos "alternativos"). Os conjuntos de dados estão descritos na Secção 5.1 e na Tabela 5.1.

A Tabela 5.2 mostra os resultados obtidos pelo LAID aplicado a diferentes conjuntos de dados. A Tabela 5.3 mostra o resultado dos testes efetuados às seleções obtidas para cada conjunto de dados e respetiva avaliação.

Após a análise das tabelas e dos gráficos adjacentes, o tempo de processamento e o espaço em disco dos ficheiros HDF5 dos conjuntos "pequeno", "médio", "grande" e "extra grande" (com número de observações constante), evidenciam um comportamento linear em função do número de atributos. Comparando os conjuntos de igual número de atributos ("pequeno", "alternativo" e "alternativo extra"), conclui-se que o aumento do número de observações penaliza bastante os tempos e os espaços em disco. O número de valores que a classe pode tomar, também, influi nas medidas analisadas. É notório o aumento dos tempos e dos espaços em disco dos conjuntos ARC3, onde o domínio da classe tem cardinalidade 3 (o espaço em disco do conjunto de maior dimensão chega a ser superior a 1 TB). Já os conjuntos PTZ têm tempos e espaços em disco menores devido à prevalência da classe 1 (85% das observações). Estes resultados eram de esperar, dada a forma como é construída a matriz disjunta, no algoritmo LAID.

No que respeita à seleção de atributos propriamente dita, a análise das taxas de cobertura e dos índices *Kappa-statistic* mostra que o algoritmo LAID obteve resultados a rondar a excelência quando os modelos foram aplicados aos conjuntos de teste PTZ e ARC, onde, o valor da classe dependia de determinados atributos de referência. No caso dos conjuntos ARC3, é impressionante que, num total de 14 atributos de referência, todos selecionaram,

pelo menos, metade desses atributos. O destaque vai para o conjunto ARC3x, onde o LAID selecionou 16 atributos, dos quais 9 eram de referência. O algoritmo, aplicado ao conjunto ARC2ax, conseguiu selecionar todos os 7 atributos relevantes, por ter mais observações e maior número de atributos selecionados.

Uma das vantagens do LAID é conseguir lidar com a inconsistência dos dados. É de referir que os conjuntos com maior proporção de inconsistências (categoria ARC) não foram penalizados na sua avaliação, denotando que a existência de observações inconsistentes não diminui a eficácia do algoritmo.

Relativamente aos conjuntos SVA, a avaliação dos resultados obtidos foi fraca. Este facto era expectável, porque o valor da classe de cada observação dependia da totalidade dos atributos, implicando que a seleção de atributos seja, neste caso, um processo quase aleatório. Este argumento é corroborado pelas taxas de cobertura, pouco superiores a 0,6 (muito próximas da taxa esperada), e pelos *Kappa-statistic*, que rondaram o valor 0,1.

Assim, relativamente aos atributos selecionados, o algoritmo LAID obteve resultados muito bons, reforçando a sua validação como método de seleção de atributos. Está vocacionado para conjuntos de dados de grande dimensão, onde o número de atributos é muito maior que o número de observações. É mais eficiente para classes booleanas e consegue lidar com as inconsistências dos dados.

5.4.1. Comparação de resultados

Cavique *et al.* (2018) já haviam testado o algoritmo LAID com conjuntos de dados de várias dimensões (Tabela 5.5). O maior deles (CAVx) continha dados gerados artificialmente, simulando um estudo sobre pacientes com uma doença rara, com 1 000 000 atributos e 2000 observações, das quais, 1700 eram classe 1 (85%) e 300 eram classe 0 (15%). O código foi escrito na linguagem C e os dados carregados em memória, a partir de ficheiros de texto. A matriz disjunta de CAVx foi decomposta em 200 sub-matrizes, dividindo os atributos, sendo o algoritmo LAID aplicado a cada um destes sub-problemas. Numa fase posterior, as sub-soluções foram agregadas e otimizadas. Para este conjunto de dados, o algoritmo foi

executado em paralelo, no *cluster* da INCD, distribuído por 10 máquinas e demorou mais de 16 horas.

Tabela 5.5 – Resultados da seleção de atributos obtidos por Cavique et al. (2018).

Conjunto de dados de teste	Observações	Atributos	Nº de classes	Nº de atributos selecionados	Taxa de cobertura	Tempo de processamento (minutos)[1]
CAVx	2 000	1 000 000	2	19		966,0 [2]
CAV50		5 000		19	0,84	50,6
CAV20		2 000		[19; 21]	[0,74; 0,84]	20,0
CAV10		1 000		[19; 21]	[0,74; 0,84]	11,5
CAV5		500		[19; 21]	[0,74; 0,84]	7,1
CAV2		200		[19; 21]	[0,74; 0,84]	3,2
CAV1		100		[19; 21]	[0,74; 0,84]	1,5

(1) Tempo aproximado de processamento da implementação do LAID, na INCD, em minutos.
(2) Tempo obtido com processamento paralelo.

Os restantes conjuntos CAV, também gerados artificialmente, apresentam uma distribuição dos valores da classe igual à de CAVx. No entanto, nestes conjuntos, o processamento foi sequencial. Os resultados mostram que o tempo de execução se aproxima de uma função linear em relação ao número de atributos. Cavique *et al.* (2018) estimaram que o tempo de processamento sequencial, para o conjunto CAVx, seria de quase sete dias.

O conjunto de dados PTZx é usado neste trabalho, precisamente, para reproduzir o conjunto de dados CAVx. Não sendo os conjuntos iguais, não se podem comparar as avaliações obtidas, mas o tempo de execução é comparável porque depende, essencialmente, da dimensão dos conjuntos. Comparando a execução sequencial, os resultados são esmagadores. Mesmo comparando o processamento em paralelo do CAVx (mais de 16 horas) com o sequencial do PTZx, este consegue um tempo melhor (menos de 12 horas).

A Tabela 5.5 mostra os resultados obtidos por Cavique *et al.* (2018). Comparando os tempos de processamento com os obtidos no presente trabalho, nomeadamente, os respeitantes aos conjuntos PTZ, por terem uma estrutura idêntica, é notória a melhor qualidade destes últimos. Tomando como exemplo os conjuntos PTZx e CAVx, cujas dimensões são iguais,

e apesar do processamento paralelo aplicado ao conjunto CAVx, os tempos de processamento sequencial de PTZx são menores. Este facto confirma a vocação do sistema HDF5 para armazenar conjuntos de grande dimensão, com acesso aos dados em disco e alta performance de leitura e escrita. Também, mostra que a linguagem de programação *Python*, sendo interpretativa, com tradução inicial *just-in-time* para uma linguagem de baixo nível (*byte code*), conseguiu responder a todas as exigências da implementação do algoritmo, com performance assinalável.

5.4.2. Novo paradigma de ambiente de desenvolvimento

Recentemente, surgiu um novo paradigma de ambiente de desenvolvimento, com o foco em três aspetos: os dados, o algoritmo e o ambiente (CAVIQUE *et al.*, 2018).

Tabela 5.6 – Evolução do paradigma dos ambientes de desenvolvimento.

	dados	algoritmos	ambientes
antes	em memória	sequencial	computador pessoal
agora	em disco	paralelo	*cloud*

Nos conjuntos de grande dimensão, o carregamento dos dados para memória está a ser substituído pelo acesso em disco. O formato HDF5 está projetado para realizar o acesso diretamente em disco, com várias configurações possíveis.

O tempo de execução dos algoritmos é crítico quando se aplica a conjuntos de grande dimensão. O processamento em paralelo tem, aqui, um papel importante. Os algoritmos sequenciais, com processamentos pesados, como sejam as meta-heurísticas, devem ser decompostos, dando lugar a heurísticas mais simples e ao paralelismo.

A execução em computadores pessoais está a ser substituída pelo ambiente *cloud*, que permite uma computação de alto desempenho. A infraestrutura INCD disponibiliza este ambiente.

Dos três focos do novo paradigma, dois integraram este trabalho: o acesso aos dados em disco, com o armazenamento em HDF5; processamento da implementação do algoritmo em ambiente *cloud*, através da infraestrutura INCD.

Ficou a faltar o processamento em paralelo que deverá ser realizado em futuros trabalhos.

6. Conclusão

A seleção de atributos tem uma importância inquestionável na área dos *Data Mining*. Tem sido objeto de estudo e desenvolvimento, mostrando-se eficaz na criação de modelos com complexidade, cada vez, mais reduzida e com aumento da precisão nas previsões efetuadas.

Este trabalho está focado no algoritmo LAID, que é detalhadamente descrito, incluindo os algoritmos *Rough Sets* e LAD, que estiveram na sua génese, e implementado na linguagem de programação *Python*. O LAID foi aplicado a vários conjuntos de dados gerados artificialmente, para testar o seu desempenho e performance em diversas situações que possam influenciar o processo de seleção. Recorreu-se ao sistema de base de dados HDF5, para validar o novo paradigma de acesso aos dados em disco, num ambiente *cloud* (INCD), por contraponto ao acesso em memória, num computador pessoal.

Os dados gerados artificialmente incluíam um conjunto de treino, destinado ao processo de seleção dos atributos e um conjunto de teste, com as mesmas características, destinado ao processo de avaliação da seleção obtida. Foram registados os tempos de processamento do algoritmo LAID, os atributos selecionados e o espaço em disco dos ficheiros. Na fase de teste, para avaliação, foi determinada a matriz confusão, a partir da qual, foi calculada a taxa de cobertura (*overall accuracy*), a taxa esperada (*expected rate*) e o índice *Kappa-statistic*.

Os testes computacionais foram realizados na INCD, remotamente, via SSH. Durante todo o processo, não houve problemas a registar. Esta infraestrutura mostrou ser robusta e fiável.

Os objetivos iniciais foram cumpridos, podendo-se destacar como contribuições do trabalho realizado:

- A implementação em linguagem de programação *Python*, com armazenamento de dados no formato HDF5, usando o pacote H5PY, revelou-se uma combinação excelente, tendo sido bem-sucedida;
- A melhoria dos resultados computacionais do algoritmo LAID confirmam a validade da implementação H5PY;

Em jeito de conclusão final, os resultados reforçam a validade do algoritmo LAID, vocacionado para conjuntos de dados onde o número de observações é muito menor que o de atributos, e a convicção de que o recurso ao pacote H5PY é de repetir em futuras implementações que lidem com um grande volume de dados e cujos algoritmos tenham complexidade $O(n^2)$.

6.1. Trabalho futuro

O presente trabalho não esgota todas as temáticas constantes no mesmo.

A validação de um algoritmo é um processo contínuo, sendo este trabalho mais um passo nesse sentido. Em particular, o LAID deve ser testado em conjuntos de dados reais de grande dimensão. Este intuito não é fácil de concretizar, porque é difícil obter este tipo de conjuntos. Na área da saúde, uma das razões apontadas para a não partilha de dados é a sua confidencialidade.

Como referido na Secção 5.4.2, dos três focos visados pelo novo paradigma de ambientes de desenvolvimento, dois foram considerados neste trabalho (acesso aos dados em disco e correr os códigos em ambiente *cloud*). O processamento em paralelo deverá ser realizado em futuros trabalhos, com a disponibilização de vários processadores, num ambiente *cloud*.

Referências

(BOLÓN-CANEDO *et al.*, 2013) BOLÓN-CANEDO, Verónica; SÁNCHEZ-MAROÑO, Noelia; ALONSO-BETANZOS, Amparo – A review of feature selection methods on synthetic data. In **Knowledge and Information Systems**. Springer. Vol. 34, nº. 3, março 2012. ISSN 0219-1377. P. 483-519. Disponível em WWW: <URL: https://www.researchgate.net/publication/257482053_A_review_of_feature_selection_methods_on_synthetic_data > (último acesso em 07-11-2018).

(BOROS *et al.*, 1996) BOROS, Endre; HAMMER, Peter L.; IBARAKI, Toshihide [*et al.*] – An Implementation of Logical Analysis of Data. In **IEEE Transactions on Knowledge and Data Engineering**. IEEE. Vol. 12-2, março-abril 2000. ISSN 1041-4347 (impressão). P. 292-306. Disponível em WWW: <URL: https://pdfs.semanticscholar.org/051d/a38e4d6e07ed36520e313a0fc2cf6219efa4.pdf > (último acesso em 07-11-2018).

(BOROS *et al.*, 1997) BOROS, Endre; HAMMER, Peter L.; IBARAKI, Toshihide; KOGAN, Alexander – Logical Analysis of Numerical Data. In Mathematical Programming. Springer. Vol. 79(1-3), outubro 1997. ISSN 0025-5610 (impressão), 1436-4646 (eletrónica). P. 163-190. Disponível em WWW: <URL: https://pdfs.semanticscholar.org/843a/0dc559ad1277c1f71e0460173697f5cbea01.pdf > (último acesso em 07-11-2018).

(BRUNI, 2007) BRUNI, Renato – Reformulation of the Support Set Selection Problem in the Logical Analysis of Data. In **Annals of Operations Research**. Springer. Vol. 150(1), março 2007. ISSN 0254-5330 (impressão), 1572-9338 (eletrónica). P. 79-92. Disponível em WWW: <URL: http://www.dis.uniroma1.it/~bruni/files/bruni04anor.pdf > (último acesso em 07-11-2018).

(CAVIQUE *et al.*, 2011) CAVIQUE, Luís; MENDES, Armando B.; FUNK, Matthias – Logical Analysis of Inconsistent Data (LAID) for a Paremiologic Study. In **15th Portuguese Conference on Artificial Inteligence - EPIA 2011**. Lisboa: EPIA, 2011. ISBN 978-989-95618-4-7. Disponível em WWW: <URL: https://repositorio.uac.pt/bitstream/10400.3/2905/1/2011%20EPIA%2057%20LAID%20v3%20final.pdf > (último acesso em 07-11-2018).

(CAVIQUE *et al.*, 2013) CAVIQUE, Luís; MENDES, Armando; FUNK, Matthias; SANTOS, Jorge – A feature selection approach in the study of azorean proverbs. In MASEGOSA, Antonio; VILLACORTA, Pablo; CRUZ-CORONA, Carlos; GARCÍA-CASCALES, M. Socorro; LAMATA, Maria; VERDEGAY, José – **Exploring Innovative and Successful Applications of Soft Computing**. Hershey PA: Igi Global, 2013. IBSN-13: 9781466447855. P. 38-58. Disponível em WWW: <URL: http://hdl.handle.net/10400.2/2795 > (último acesso em 07-11-2018).

(CAVIQUE *et al.*, 2018) CAVIQUE, Luís; MENDES, Armando; MARTINIANO, Hugo; CORREIA Luís – A biobjective feature selection algorithm for large omics datasets. In **Expert Systems – Special issue paper**. Wiley, Junho 2018. P. 14. Disponível em WWW: <URL: https://onlinelibrary.wiley.com/doi/pdf/10.1111/exsy.12301 > (último acesso em 07-11-2018).

(COLLETTE *et al.*, 2018) COLLETTE, Andrew [et. al] – **HDF5 for *Python*** [em linha]. Andrew Collette & Contributers, 2018 [última consulta em 07-11-2018]. Disponível em WWW: <URL: http://www.h5py.org/ >.

(DASH, LIU, 1997) DASH, M.; LIU, H. – Feature Selection for Classification. In **Intelligent Data Analysis**. Elsevier, 1997. P. 131-156. Disponível em WWW: <URL: https://pdfs.semanticscholar.org/c7ed/d5641f979ebf5d2f7c3a3ee05c6741693205.pdf > (último acesso em 07-11-2018).

(DOAK, 1992) DOAK, Justin – **An evaluation of feature selection methods and their application to computer security**. Davis CA: Escholarship - Universidade da Califórnia, 01-01-1992. CSE 92-18. Disponível em WWW: <URL: https://escholarship.org/content/qt2jf918dh/qt2jf918dh.pdf > (último acesso em 07-11-2018).

(FCT-FCCN, 2017) FCT-FCCN – **Infraestrutura Nacional de Computação Distribuída** - Computação para a Ciência e Ensino [Em linha]. FCT-FCCN, 2017, 2 p [última consulta em 07-11-2018]. Disponível em WWW: <URL: https://www.fccn.pt/wp-content/uploads/2017/05/FS-INCD_v1.pdf >.

(FCT-FCCN, 2018) FCT-FCCN – **INCD** [em linha]. FCT-FCCN, 2018 [última consulta em 07-11-2018]. Disponível em WWW:<URL: https://www.fccn.pt/computacao/incd/ >.

(FCT, 2014) FCT – **Portuguese Roadmap of Research Infra-structures 2014-2020** [Em linha]. FCT, julho 2014, 100 p [última consulta em 07-11-2018]. Disponível em WWW: <URL: https://www.fct.pt/apoios/equipamento/roteiro/2013/docs/Portuguese_Roadmap_of_Research_Infrastructures.pdf >

(FRIEDMAN *et al.*, 2000) FRIEDMAN, Nir; LINIAL, Michal; NACHMAN, Iftach; PEÉR, Danna – Using Bayesian Networks to Analyze Expression Data. In **Journal of Computational Biology**. Mary Ann Liebert, Inc.. Vol. 7(3-4), agosto 2000. P. 601–620. Disponível em WWW: <URL: http://www.cs.huji.ac.il/~nir/Papers/FLNP1Full.pdf > (último acesso em 07-11-2018).

(GUYON, ELISSEEFF, 2003) GUYON, Isabelle; ELISSEEFF, André – An Introduction to Variable and Feature Selection. In **Journal of Machine Learning Research**. JMLR.Org. ISSN: 1532-4435. Vol. 3 março 2003. P. 1157-1182. Disponível em WWW: <URL: http://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf > (último acesso em 07-11-2018).

(HDF GROUP, 2018) HDF GROUP – **What is HDF5?** [em linha]. Champaingn: HDF Group, 2018 [última consulta em 07-11-2018]. Disponível em WWW: <URL: https://hdfgroup.org >.

(INCD, 2018) INCD – **Computação para a Ciência e Ensino** [em linha]. INCD, 2018 [última consulta em 07-11-2018]. Disponível em WWW: <URL: http://www.incd.pt/ >.

(JENNESS, WYNNE, 2005) JENNESS, Jeff; WYNNE, J. Judson – **Cohen's Kappa and Classification Table Metrics 2.0** - An ArcView 3x Extension for Accuracy Assessment of Spatially Explicit Models - U.S. Geological Survey Open-File Report Of 2005-1363. Flagstaff, AZ: U.S. Geological Survey, Southwest Biological Science Center, 2005, 91 p. Disponível em WWW: <URL: https://www.fs.fed.us/rm/pubs_other/rmrs_2005_jenness_j001.pdf > (último acesso em 07-11-2018).

(JOHN *et al.*, 1994) JOHN, George H.; KOHAVI, Ron; PFLEGER, Karl – Irrelevant Features and the Subset Selection Problem. In COHEN, William W.; HIRSH, Haym – **Machine Learning: Proceedings of the Eleventh International Conference**. São Francisco CA: Morgan Kaufmann Publishers, 1994. ISBN: 1-55860-335-2. P. 121-129. Disponível em WWW: <URL: https://pdfs.semanticscholar.org/a83b/ddb34618cc68f1014ca12eef7f537825d104.pdf > (último acesso em 07-11-2018).

(KOHAVI, JOHN, 1997) KOHAVI, Ron; JOHN, George H. – Wrappers for feature subset selection. In **Artificial Intelligence**. Elsevier. Vol. 97, dezembro 1997. P. 273-324. Disponível em WWW: <URL: https://ac.els-cdn.com/S000437029700043X/1-s2.0-S000437029700043X-main.pdf?_tid=c1932680-099d-11e8-b18e-00000aab0f02&acdnat=1517743560_36d04b415cf78fbc3ead78205910999c > (último acesso em 07-11-2018).

(KUMAR, MINZ, 2014) KUMAR, Vipin; MINZ, Sonajharia – Feature Selection: A literature Review. **Smart Computing Review**. Vol. 4, nº. 3, janeiro 2014. ISSN Nº. 2234-4624. P. 211-229. Disponível em WWW: <URL: www.smartcr.org/view/download.php?filename=smartcr_vol4no3p7.pdf > (último acesso em 07-11-2018).

(LNEC, 2018) LNEC – **Projetos INCD-Infraestrutura Nacional de Computação Distribuída** [em linha]. LNEC, 2018 [última consulta em 07-11-2018]. Disponível em WWW:<URL: http://www.lnec.pt/hidraulica-ambiente/pt/estudos/detalhes.php?tipo=0&id=280 >.

(LIP, 2018) LIP – **Computação – Infraestruturas** [em linha]. LIP, 2018 [última consulta em 07-11-2018]. Disponível em WWW: <URL: https://www.lip.pt/index.php?section=infrastructures&page=infrastructures-computing&projectid=12 >.

(PAWLAK, 2002) PAWLAK, Zdzislaw – Rough set theory and its applications. In **Journal of Telecommunications and Information Technology**. National Institute of Telecommunications. Vol. 3, março 2002. P. 7-10. Disponível em WWW: <URL: https://www.itl.waw.pl/czasopisma/JTIT/2002/3/7.pdf > (último acesso em 07-11-2018).

(PAWLAK, 2003) PAWLAK, Zdzislaw – **Rough Sets** [brochura]. Gliwice: 2003, 51 p. Disponível em WWW: <URL: https://wiki.eecs.yorku.ca/course_archive/2010-11/W/4403/_media/roughsetsrep29.pdf > (último acesso em 07-11-2018).

(PILA, MONARD, 2001) PILA, Adriano D.; MONARD, M. Carolina – Teoria de Rough Sets Aplicada à Data Mining. In **II Congresso de Lógica Aplicada à Tecnologia, LAPTEC'2001**. São Paulo: Plêiade, 2001. P. 203-211. Disponível em WWW: <URL: http://conteudo.icmc.usp.br/pessoas/mcmonard/public/laptec2001A.pdf > (último acesso em 07-11-2018).

(PIRES, PAGAIME, 2015) PIRES, Esmeralda; PAGAIME, João – **Piloto Experimentação Cloud Computing** [em linha]. FCT-FCCN, 2015 [última consulta em 07-11-2018]. Disponível em WWW:<URL: https://confluence.fccn.pt/pages/viewpage.action?pageId=13238497 >.

(*PYTHON* S.F., 2018) *PYTHON* SOFTWARE FOUNDATION – ***Python*** [em linha]. *Python* Software Foundation, 2018 [última consulta em 07-11-2018]. Disponível em WWW: <URL: https://*Python*.org >.

(RISSINO, LAMBERT-TORRES, 2009) RISSINO, Silvia; LAMBERT-TORRES, Germano – Rough Set Theory - Fundamental Concepts, Principals, Data Extraction, and Applications. In PONCE, Julio; KARAHOCA, Adem – **Data Mining and Knowledge Discovery in Real Life Applications**. Viena: InTech, 2009. ISBN 978-3-902613-53-0. P. 35-59. Disponível em WWW: <URL: http://cdn.intechopen.com/pdfs/5939/InTech-Rough_set_theory_151_fundamental_concepts_principals_data_extraction_and_applications.pdf > (último acesso em 07-11-2018).

(SANTRA, CHRISTY, 2012) SANTRA, A. K.; CHRISTY, C. Josephine – Genetic Algorithm and Confusion Matrix for Document Clustering. In **IJCSI International Journal of Computer Science Issues**. IJCSI. Vol. 9-1, nº. 2, janeiro 2012. ISSN 1694-0814 (eletrónico). P. 322-328. Disponível em WWW: <URL: http://www.ijcsi.org/papers/IJCSI-9-1-2-322-328.pdf > (último acesso em 07-11-2018).

(SASSI, 2006) SASSI, Renato J. – **Uma Arquitetura para Descoberta de Conhecimento em Bases de Dados: Teoria dos Rough Sets e Redes Neurais Artificiais**. São Paulo: Escola Politécnica da Universidade de São Paulo, 2006. 186 p. Tese de doutoramento. Disponível em WWW: <URL: http://www.teses.usp.br/teses/disponiveis/3/3142/tde-16032007-163930/publico/teseversaorevisada.pdf > (último acesso em 07-11-2018).

(SUBASI, AVILA-HERRERA, 2015) SUBASI, Munevver M.; AVILA-HERRERA, Juan F. – Logical Analysis of Multiclass Data. In **Latin American Computing Conference** (CLEI). IEEE, dezembro 2015, 10 p.. ISBN 978-1-4673-9143-6 (eletrónica). Disponível em WWW: <URL: http://isaim2016.cs.virginia.edu/papers/ISAIM2016_Boolean_Subasi_Herrera.pdf > (último acesso em 07-11-2018).

(W3RESOURCE, 2018) W3RESOURCE – ***Python* Tutorial** [em linha]. W3Resource, 2018 [última consulta em 07-11-2018]. Disponível em WWW: <URL: https://www.w3resource.com/*Python*/*Python*-tutorial.php >.

(YAO, 2006) YAO, Y. Y. – Neighborhood systems and approximate retrieval. In **Information Sciences**. Elsevier. Vol. 176-23, dezembro 2006. P. 3431-3452. Disponível em WWW: <URL: https://www.sciencedirect.com/science/article/pii/S0020025506000405 > (último acesso em 07-11-2018).

(ZHANG *et al.*, 2016) ZHANG, Qinghua; XIE, Qin; WANG, Guoyin – A survey on rough set theory and its applications. In **Transactions on Intelligence Technology**. CAAI. Vol. 1, outubro 2016. P. 323-333. Disponível em WWW: <URL: https://ac.els-cdn.com/S2468232216300786/1-s2.0-S2468232216300786-main.pdf?_tid=1c079c6f-951a-4472-ab65-5e5fa6d15246&acdnat=1528130321_973403687cf807226b867b1135761f43 > (último acesso em 07-11-2018).