Seleção de Atributos de Dados Inconsistentes em ambiente HDF5+Python na *cloud* INCD

João Apolónia[1], Luís Cavique[2]
[1]Mestre em Tecnologias e Sistemas Informáticos WEB, Universidade Aberta
[2]Universidade Aberta, DCeT
jd.apolonia@gmail.com, luis.cavique@uab.pt

Resumo

O tratamento de conjuntos de dados de grande dimensão é uma questão que é recorrente nos dias de hoje. 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. A Análise Lógica de Dados Inconsistentes (LAID) é uma metodologia sistematizada, robusta, sendo fácil de interpretar e consegue lidar com dados inconsistentes. O paradigma, relativamente ao manuseamento de grandes volumes de dados, 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*. Este trabalho pretende validar o novo paradigma, com recurso ao sistema de dados HDF5 e ao ambiente remoto disponibilizado pela. 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.

Palavras-chave: *Data Mining*, Seleção de atributos, LAID, HDF5, *Python*, INCD

Title: Feature Selection of Inconsistent Data in HDF5+Python environment on INCD cloud

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. Logical Analysis of Inconsistent Data (LAID) is a systematic, robust methodology that is easy to interpret and can handle inconsistent data. The paradigm regarding the handling of large data has 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 present work intends to validate this new paradigm, using HDF5 data system and remote environment provided by INCD. Because HDF5 is the system adopted by Python's community to handle large datasets, this language was chosen for LAID algorithm implementation.

keywords: Data Mining, Feature selection, LAID, HDF5, Python, INCD

1. Introdução

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 está a Análise Lógica de Dados Inconsistentes (LAID). Embora teoricamente, esta técnica esteja consolidada, os estudos aplicados a conjuntos de dados de grande dimensão são escassos. 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).

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.

O algoritmo LAID é descrito na secção 2. A explanação do LAID é acompanhada por um exemplo demonstrativo.

A fase de implementação do algoritmo LAID é descortinada na secção 3. 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.

Os resultados obtidos no processo de seleção e de avaliação são analisados na secção 4. São descritas as características dos conjuntos de dados artificiais usados no processo.

Por fim, as conclusões estão na secção 5, onde se indicam os trabalhos futuros que se julgam necessários para complementar o âmbito desta publicação.

2. Metodologia LAID

A Análise Lógica de Dados Inconsistentes (LAID) surge da combinação de duas 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).

Considerem-se as seguintes definições:

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

2.1. 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:

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

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

2.2. 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:

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

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*.

A Figura 2.1 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 2.1 mostra o resultado da remoção das inconsistências na LAID quando é adicionado o atributo *jnsq*.

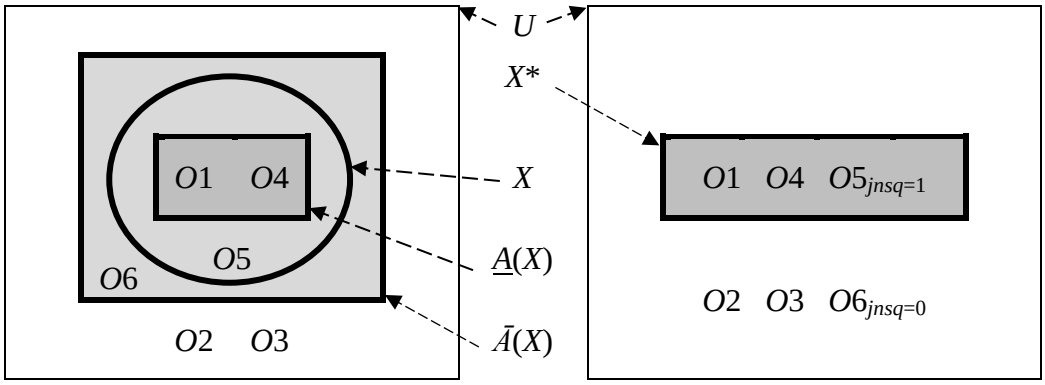

Figura 2.1. Diferença entre *Rough Sets* e a remoção das inconsistências na LAID. (Adaptado de: Cavique et al., 2013)

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 2.1 para aplicar a remoção das inconsistências segundo a metodologia LAID.

Tabela 2.1. Sistema de decisão para exemplificar a LAID.

Observações	*a*1	*a*2	*a*3	*a*4	classe
*o*1	1	0	1	0	1
*o*2	0	1	1	0	1
*o*3	0	1	0	0	1
*o*4	0	1	0	0	2
*o*5	1	0	0	1	0
*o*6	1	0	1	0	2
*o*7	0	1	0	0	0
*o*8	1	0	1	0	1

Começa-se por remover as redundâncias. As observações *o*1 e *o*8 são redundantes. Ao retirar uma delas, escolha-se *o*8, a redundância desaparece.

Tabela 2.2. Sistema de decisão com a redundância removida.

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

Da observação da Tabela 2.2, identificam-se os seguintes conjuntos de observações inconsistentes: {*o*1, *o*6} e {*o*3, *o*4, *o*7}, vindo:

Tabela 2.3. 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 $A^*=\{a1, a2, a3, a4, a5^*, a6^*\}$.

Tabela 2.4. 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 2.4 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.

2.3. 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 2.4, a matriz disjunta vem:

Tabela 2.5. 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)

2.4. 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:

$$s_j = \sum_{i=1}^{n} d_{i,j} \text{ , com } j = 1,\ldots,m+m^* \qquad (2.1)$$

Tabela 2.6. Processo de redução – iteração 1.

A^*	$a1$	$a2$	$a3$	$a4$	$a5^*$	$a6^*$
	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
$d_{i,j}$	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:

$$s(a_e) = \text{máx}(s_j), \text{ com } j = 1,\ldots,m+m^* \qquad (2.2)$$

Na Tabela 2.6, o atributo escolhido é $a3$, entrando para a solução, $S=\{a3\}$. Se vários atributos cumprem a condição (2.2), existem escolhas alternativas e escolhe-se um deles.

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

Tabela 2.7. 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 2.7), é escolhido o atributo *a*5*, ficando *S* = {*a*3, *a*5*}. São eliminadas as 5 primeiras linhas da matriz *M* e a coluna do atributo escolhido.

Tabela 2.8. Processo de redução – iteração 3.

*a*1	*a*2	*a*4	*a*6*
1	1	1	1
0	0	0	1
1	1	1	2

Na terceira iteração do exemplo (Tabela 2.8), é escolhido o atributo *a*6*, ficando *S* = {*a*3, *a*5*, *a*6*}. 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 {*a*3, *a*5*, *a*6*}.

Tabela 2.9. 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 2.10. 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. Implementação do LAID em HDF5+*Python* no INCD

3.1. 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.

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.

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.

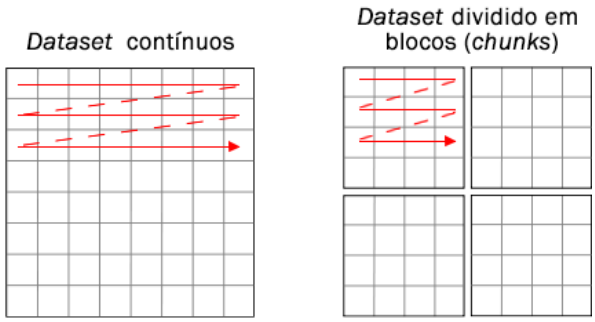

Figura 3.1. Leitura e escrita num *dataset* contínuo ou compartimentado.

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.

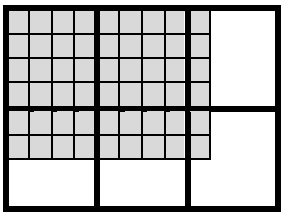

Figura 3.2. Sobredimensionamento dos blocos num dataset.

- 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.

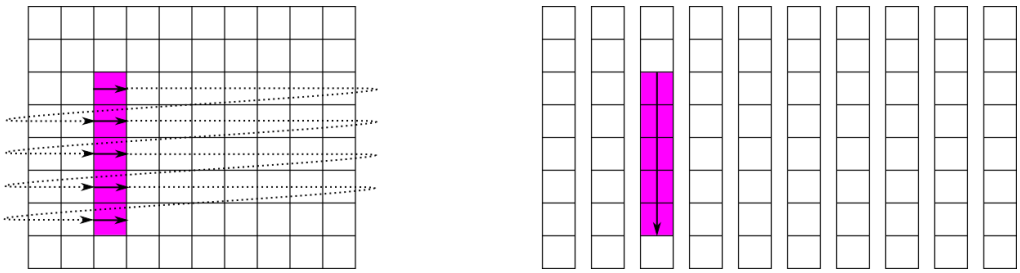

Figura 3.3. Leitura de parte de uma coluna num dataset contínuo ou compartimentado. (Fonte: HDF Group, 2018)

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.

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.

3.2. Infraestrutura Nacional de Computação Distribuída

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 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.

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).

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*.

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).

Para este trabalho, a INCD disponibilizou 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.

3.3. Estratégia na implementação do LAID

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).

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

A seleção dos atributos é realizada com recurso à matriz disjunta. O algoritmo 2 retorna o conjunto dos atributos selecionados a partir da matriz disjunta:

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

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.

4. Resultados Computacionais

Nesta secção, 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*.

4.1. Conjuntos de dados artificiais 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*.

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.

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.

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.

Tabela 4.1. 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		

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).

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 4.1). 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.

4.2. Tempo computacional e outros indicadores de desempenho

Após aplicar o algoritmo LAID a cada um dos conjuntos de dados artificiais, registaram-se os resultados obtidos. As Figuras 4.1, 4.2 e 4.3 mostram 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.

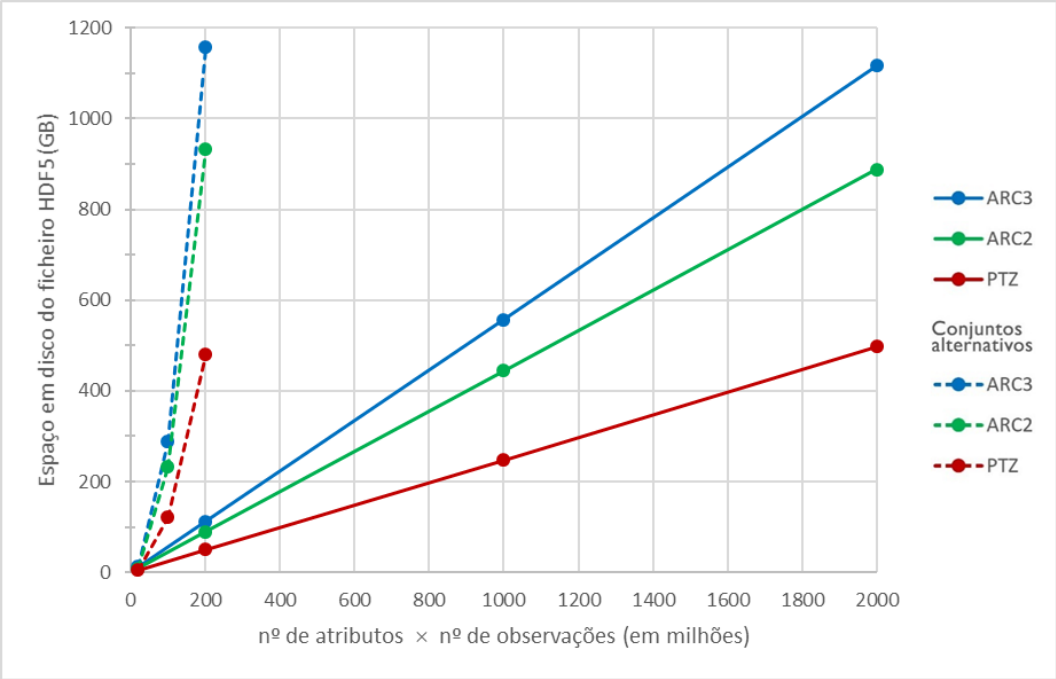

Figura 4.1. Espaço em disco do ficheiro HDF5, em função da dimensão dos dados.

Numa primeira análise, destaca-se a eficiência das bases de dados HDF5 no armazenamento dos elementos. 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 $O(n)$.

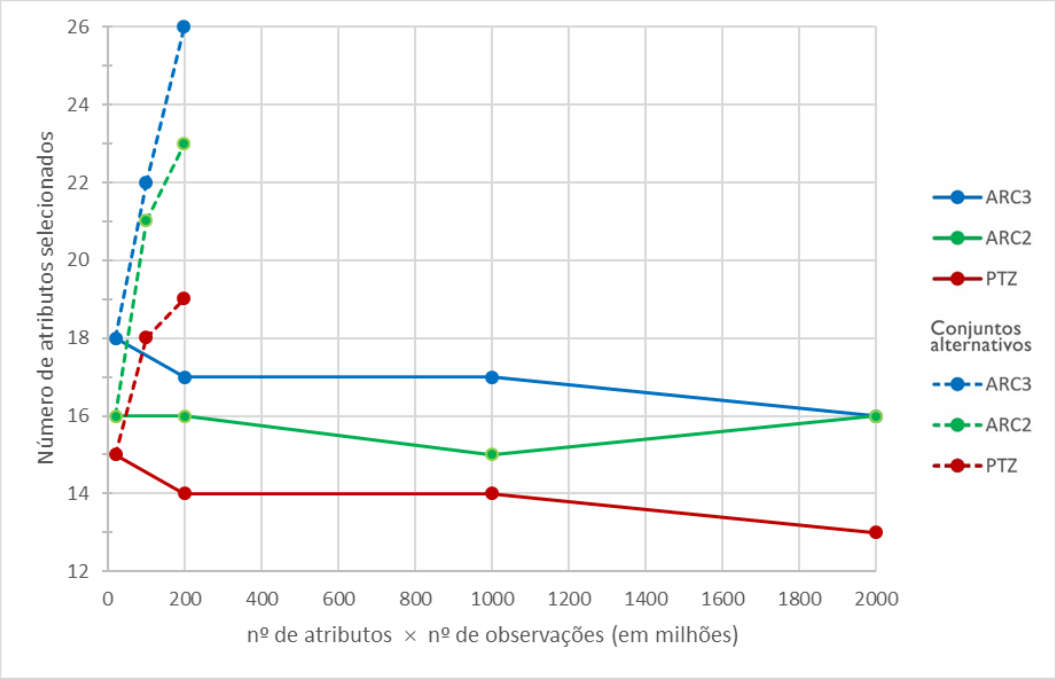

Figura 4.2. Número de atributos selecionados, em função da dimensão dos dados.

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.

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.

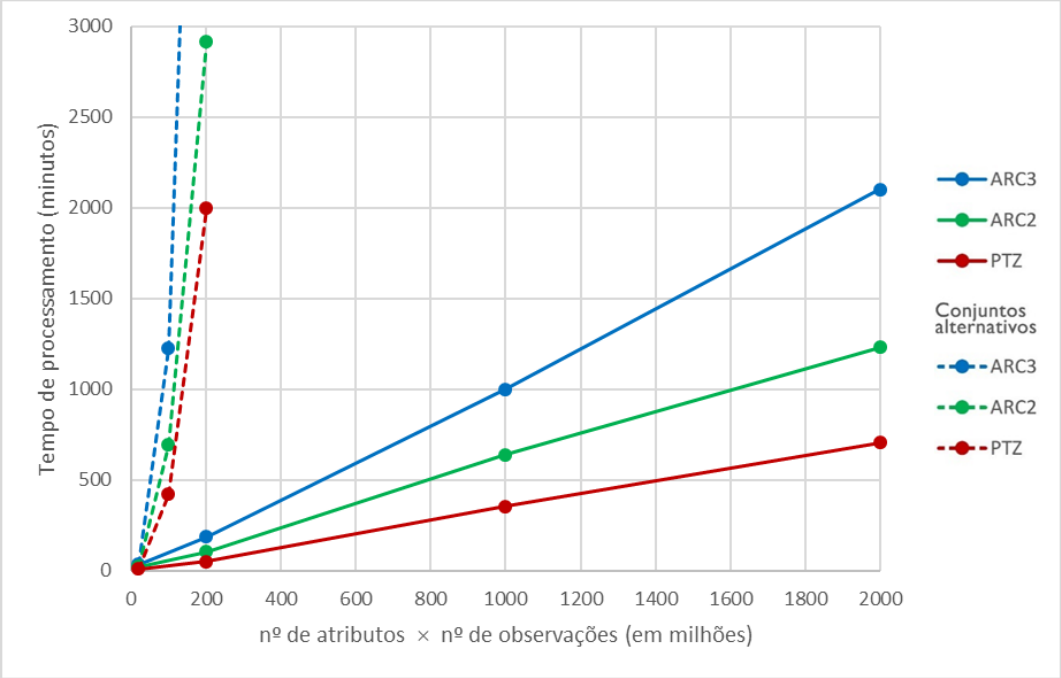

Figura 4.3. Tempo de processamento, em minutos, em função da dimensão dos dados.

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) (**Erro! A origem da referência não foi encontrada.**), com particular destaque para a comparação entre os conjuntos PTZx e CAVx.

4.3. Avaliação da qualidade dos resultados

Após a seleção dos atributos, a Figura 4.4 mostra a qualidade dos resultados obtidos.

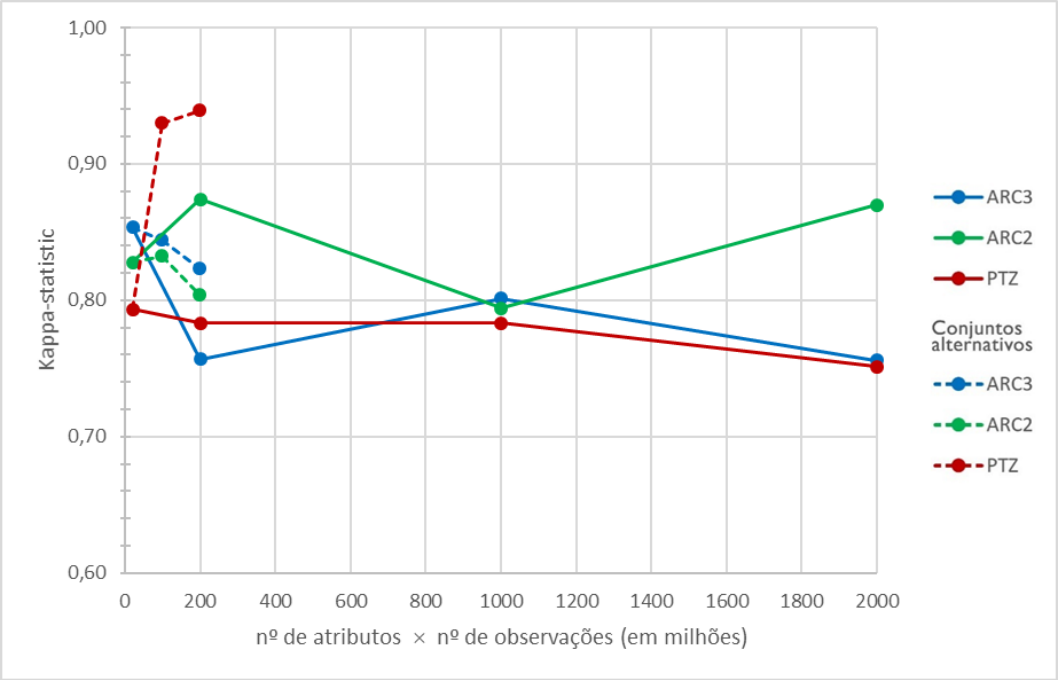

Figura 4.4. *Kappa-statistic* em função da dimensão dos dados.

Quanto aos conjuntos PTZ, 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).

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.

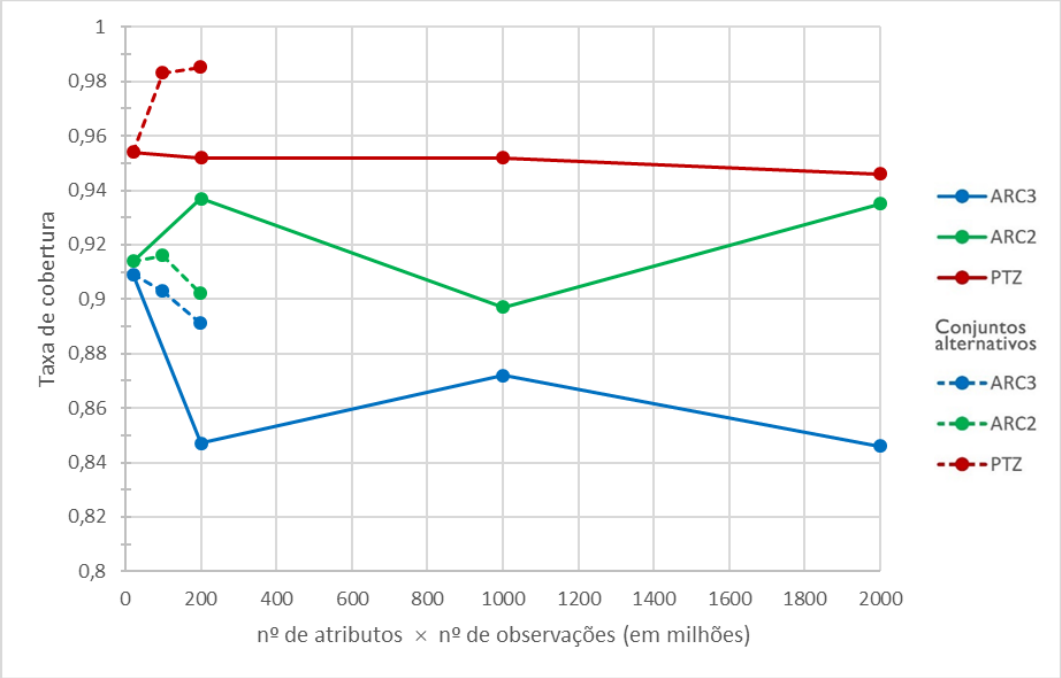

Figura 4.5. Taxa de cobertura, em função da dimensão dos dados.

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. Esta característica reforça uma das principais características do algoritmo LAID: lidar bem com as inconsistências.

O gráfico da taxa de cobertura (Figura 4.5) apresenta a mesma tendência de variação que o gráfico do índice *Kappa-statistic* (Figura 4.4). 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.

4.4. 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").
Após a dos gráficos anteriores, 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.

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.

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.

Comparação de resultados

Cavique *et al.* (2018) já haviam testado o algoritmo LAID com conjuntos de dados de várias dimensões. 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. Como os tempos de execução têm um comportamento, aproximadamente, 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).

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.

4.5. 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 4.2. 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.

Neste trabalho, o processamento em paralelo não foi desenvolvido, estando previsto que se realizará em futuros trabalhos.

5. Conclusões

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, 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 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 $O(n^2)$.

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.

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.

(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.

(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.

(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/ >.

(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/ >.

(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.

(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 >.

João Apolónia, Programador. Licenciado em Engenharia Informática em 2008 pela EST-IPS. Obteve o grau Mestre em Tecnologias e Sistemas Informáticos Web, pela Universidade Aberta, Portugal, em 2019. Tem como áreas de interesse, o "Design Gráfico" e processos de "Data Mining", nomeadamente, tratamento de conjuntos de dados de grande dimensão.

Luís Cavique, Professor Auxiliar no Departamento de Ciências e Tecnologia (DCeT), Secção de Informática, Física e Tecnologia (SIFT). Licenciado em Engenharia Informática em 1988 pela FCT-UNL. Obteve o grau Mestre em Investigação Operacional e Eng. Sistemas pelo IST-UTL em 1994. Obteve o grau de Doutor em Eng. Sistemas pelo IST-UTL em 2002. Tem como áreas de interesse, a intersecção da Informática com a Engenharia de Sistemas designadamente a área de "Data Mining".