Exploração de Aprendizagem Multi-relacional com a utilização da biblioteca Python RDM

Miguel Lopes Dias

Engenharia de Redes e Sistemas Informáticos

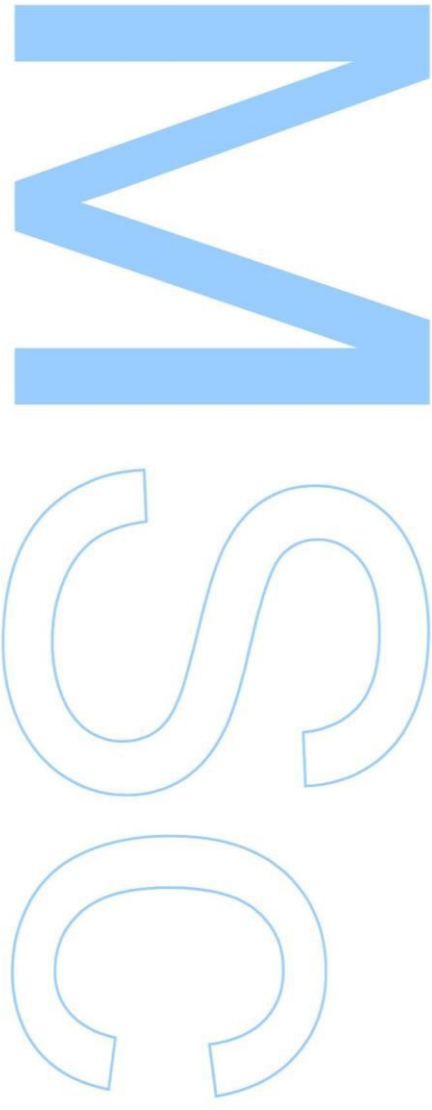
Departamento de Ciência de Computadores

2021

Orientador

Inês de Castro Dutra, Professor Auxiliar

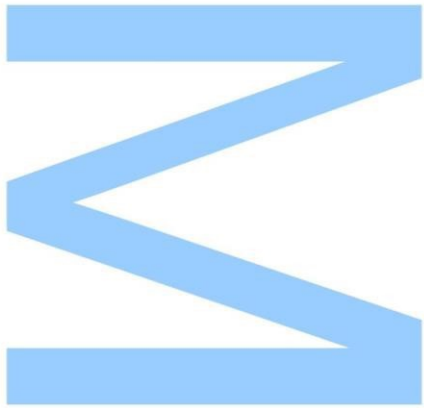
Faculdade de Ciências

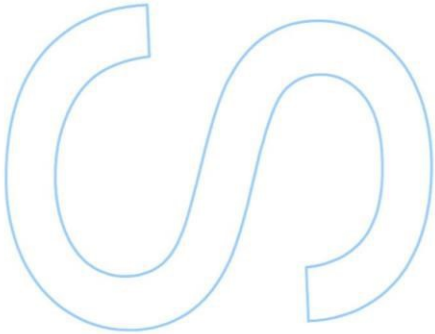


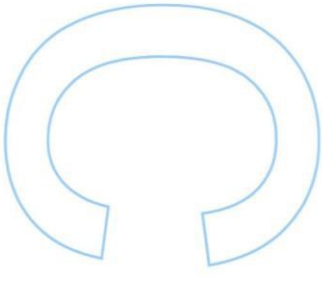
Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____







Agradecimentos

Agradeço à minha família por toda a dedicação e apoio cruciais na chegada até aqui.

Um agradecimento especial à Professora e amiga Inês Dutra, por todas as discussões e contribuições ao longo deste ano. A sua ajuda e entusiasmo constantes foram determinantes para o sucesso deste projeto.

A minha palavra de agradecimento é para todos aqueles que direta ou indiretamente contribuíram para a realização desta dissertação. Seria impensável nomeá-los a todos.

Muito obrigado!

Abstract

The information that we can extract from the various relationships between datasets is significant for decision making or the retrieval of new knowledge.

In this dissertation we explore a relational data mining Python package and suggest and implement some modifications to make it more flexible and allow the introduction of expert knowledge. We use two different datasets to illustrate how to use the package and take advantage of our modifications. Additionally, we show quantitative and qualitative results that validate our implementation.

For the breast cancer dataset, we illustrate the introduction of expert knolwedge by creating groups of patients aggregated by age and show that the learned rules are more meaningful and better than rules generated solely from the data. For the FIFA European soccer, we illustrate the introduction of expert knowledge by creating a rule to take into account temporal player events. Players' temporal events are those that represent the characteristics of these players at a certain time and the characteristics of the same players at a certain later time, these events can vary depending on the time changes or not. Results show some learned rules that use the temporal added knowledge to extract other knowledge which is relevant.

The new version of the package implements cross-validation, besides allowing the user to introduce new knowledge. We hope this can be integrated to the Python package to make it possible to use these methods to any other type of dataset where expert knowledge is available to be added.

Resumo

A informação que podemos retirar das várias relações entre conjuntos de dados é significativa para a toma de decisões ou a retirada de novo conhecimento.

Nesta dissertação exploramos um pacote Python de manipulação de dados relacional e sugerimos e implementamos algumas modificações para torná-lo mais flexível e permitir a introdução de conhecimento especializado. Usamos dois conjuntos de dados diferentes para ilustrar como usar o pacote e tirar proveito das nossas modificações. Além disso, apresentamos resultados quantitativos e qualitativos que validam a nossa implementação.

Para o conjunto de dados sobre o cancro da mama, ilustramos a introdução do conhecimento especializado com a criação de grupos de pacientes agregados por idade e mostramos que as regras aprendidas são mais significativas e melhores do que as regras geradas exclusivamente a partir dos dados. Para o vídeo-jogo FIFA European soccer, ilustramos a introdução de conhecimento especializado com a criação de uma regra para levar em consideração os eventos temporais dos jogadores. Os eventos temporais dos jogadores são aqueles que representam caracteristicas destes jogadores num determinado tempo e as caracteristicas dos mesmos jogadores num determinado tempo posterior, estes eventos podem variar conforme o tempo muda ou não. Os resultados mostram algumas regras aprendidas que usam o conhecimento temporal adicionado para extrair outro conhecimento que seja relevante.

A nova versão do pacote implementa validação cruzada, além de permitir ao usuário introduzir novo conhecimento. Esperamos que isso possa ser integrado ao pacote Python para possibilitar o uso desses métodos para qualquer outro tipo de conjunto de dados onde o conhecimento especializado está disponível para ser adicionado.

Palavras-chave: Relação de dados, programação lógica indutiva, Python, informação temporal, Aleph.

Índice

Lista de Tabelas

Lista de Figuras

Lista de Blocos de Código

Lista de Abreviaturas

Aleph	*A Learning Engine for Proposing Hypotheses*
CAD	*Computer-Aided Design*
ECG	Eletrocardiograma
FN	*False Negative*
FP	*False Positive*
FPR	*False Positive Rate*
MRDM	*Multi Relational Data Mining*
MRI	*Magnetic resonance imaging*
ILP	*Inductive Logic Programming*
RDM	*Relational Data Mining*
ROC *Curve*	*Receiver Operating Characteristic Curve*
SVM	*Support Vector Machines*
TIA	*Allen's temporal interval algebra*
TN	*True Negative*
TP	*True Positive*
TPR	*True Positive Rate*
WEKA	*Waikato Environment for Knowledge Analysis*

Capítulo 1

Introdução

A área de aprendizagem automática é um subcampo da engenharia e da ciência da computação que evoluiu do estudo de reconhecimento de padrões e da teoria de aprendizagem automática em inteligência artificial. A aprendizagem automática explora o estudo e construção de algoritmos que podem aprender através dos seus erros e fazer previsões sobre os dados. Estes algoritmos desempenham uma construção de um modelo a partir de entradas de maneira a fazer previsões sobre os dados [1].

Antes de serem aplicados aos dados, de forma a ser obtido novo conhecimento a partir destes, os dados devem receber um pré-processamento.

Em muitos casos, os dados são reduzidos apenas a uma tabela bidimensional, desta forma, a capacidade de serem obtidas novas informações relativas a estes torna-se muito difícil pois as relações que se vão criar, posteriormente, entre os dados, serão muito escassas. Com apenas uma tabela bidimensional, grande parte da informação relevante sobre os dados pode ficar oculta e a informação retirada ser muito limitada. No entanto, com o uso de várias tabelas a informação relativa aos dados está completamente presente e é possível criar relações entre estas de forma à retirada de informação ser significativa.

A manipulação de dados relacionais é uma técnica aplicada a bases de dados relacionais. Ao contrário dos algoritmos tradicionais de manipulação de dados, que procuram por padrões em apenas uma tabela, os algoritmos de manipulação de dados relacionais procuram por padrões em várias tabelas. Existem várias abordagens para manipulação de dados relacionais, aprendizagem relacional estatística, proposicionalização, aprendizagem de multivisualização, e, entre estas, a programação lógica indutiva (PLI) [2].

A programação lógica indutiva é um subcampo da inteligência artificial que usa a programação lógica como uma representação uniforme para exemplos, *background knowledge* e hipóteses. Dados *background knowledge* e um conjunto de exemplos representados como

uma base de dados, um sistema de programação lógica indutiva deriva hipóteses que cubram exemplos positivos [3].

Nesta dissertação exploramos aprendizagem relacional com o uso de um pacote *Python – Relational Data Mining* (RDM). Utilizamos dois conjuntos de dados relacionais: um conjunto de tabelas contendo dados sobre mamografias e biópsias de pacientes e um outro conjunto de tabelas relativas ao FIFA *European soccer* [32].

A utilização de aprendizagem relacional utilizando ILP permite que conhecimento extra possa ser adicionado ao conjunto de dados originais

No entanto, o pacote Python-rdm não está completamente preparado para receber múltiplas tabelas de dados, apresenta algumas restrições, e por isso, foram necessários alguns ajustes ao algoritmo, nomeadamente, ajustes às chaves primárias e estrangeiras que permitem a relação entre as tabelas, para que fosse possível a realização destas relações e posteriormente a indução de novas regras através de programação lógica indutiva.

O pacote Python-rdm utiliza o programa Aleph para permitir que sejam criadas novas regras com nova informação que não seria possível serem verificadas sem esta ajuda, no entanto, para que isto seja praticável, foram aplicadas algumas alterações ao pacote Python-rdm, sendo a mais significativa a divisão deste em dois programas. O primeiro gera os ficheiros que serão entrada para o programa Prolog Aleph e o segundo invoca o programa Prolog Aleph, através da interface dada pelo programa Python aleph.py, para aprender as regras. O utilizador poderá, desta forma, intervir, e adicionar novo conhecimento ao *background knowledge*. Este novo conhecimento poderá ajudar a obter regras mais expressivas ou com melhor qualidade preditiva.

1.1 Propósito da investigação

O objetivo deste trabalho é investigar a funcionalidade do pacote python-rdm e compreender a importância da aprendizagem relacional.

Neste trabalho é feita uma reformulação ao pacote Python-rdm para tornar possível a obtenção de novo conhecimento através das relações entre múltiplas tabelas de dados.

São obtidas regras com informação temporal relativas aos dados com o uso de *background knowledge* com informação temporal aplicado ao programa.

Este trabalho apresenta uma abordagem através de programação lógica indutiva juntamente com métodos de manipulação de dados, com o uso de um pacote de manipulação de dados relacionais – *Relational Data Mining*, RDM - que será aplicado a várias tabelas de

dados para extrair as relações existentes de forma a retirar novo conhecimento que não é visível sem esta aplicação.

Este trabalho insere-se no contexto de aprendizagem automática, mas utiliza o método baseado em programação lógica indutiva para poder trabalhar com múltiplas tabelas de dados, o que não é possível com os métodos padrão que requerem uma única tabela bidimensional. Além disso, exploramos a linguagem lógica de programação lógica indutiva, PLI, para adicionar conhecimento que pode auxiliar na obtenção de regras mais ricas que descrevem padrões para os dados.

Uma das vantagens de se introduzir conhecimento extra aos dados relacionais é poder expressar regras temporais. Estas regras permitem que um sistema de indução possa aprender relações temporais.

Dados temporais representam um estado no tempo, normalmente, os conjuntos de dados guardam informação sobre o estado atual dos dados e quando o tempo se altera, o estado em que os dados se encontram também pode ser alterado. O conhecimento sobre a evolução ao longo do tempo dos dados é, muitas vezes, necessário e, por isso, informação sobre o estado dos dados no passado é tão importante como no presente. Através de tuplos de informação, podemos relacionar dados temporais a dados categóricos, cada tuplo tem associado um período de tempo onde uma determinada informação é válida para aquele período de tempo.

De forma a retirar informação temporal e após o uso dos conjuntos de dados relativos ao cancro da mama, nesta fase da dissertação foram adicionados conjuntos de dados sobre o FIFA *European Soccer*, que contêm informação temporal e permitem retirar a importância desta informação em qualquer conjunto de dados.

1.2 Estrutura

O percurso desta dissertação é feito ao longo dos capítulos seguintes.

O segundo capítulo descreve os conceitos fundamentais necessários para o entendimento dos seguintes. Explicamos o conceito de programação lógica indutiva e o sistema que utilizamos, Aleph onde está implementada toda a parte indutiva e é executado através de um programa Python. Também discutimos sobre relational data mining e o pacote Python RDM, além de apresentar os métodos básicos de validação e métricas conhecidas de avaliação.

No capítulo três é apresentada a revisão bibliográfica sobre assuntos relacionados com o *relational data mining,* métodos de aprendizagem onde são utilizados como entrada dados relacionais. São apontados resultados obtidos na literatura e limitações.

No capítulo quatro apresenta-se o trabalho desenvolvido, descrição de algoritmos e métricas utilizadas, obstáculos que surgiram durante o trabalho e como foram superados.

No capítulo seguinte, apresentam-se os resultados e análise.

Por fim, no último capítulo é feito um resumo do trabalho realizado, confrontando-o com os objetivos iniciais e traça-se perspectivas de trabalhos futuros.

Capítulo 2

Conceitos Gerais

Neste capitulo são apresentadas definições relativas a termos e conceitos utilizados durante as secções seguintes. Inicia-se com a área de programação lógica indutiva, onde existem vários sistemas que a implementam, como por exemplo, c-progol, progol, foil, claudien, no entanto, como na dissertação é utilizado o pacote *Python*-rdm, o sistema que se apresenta nesta secção e que implementa PLI, é o Aleph.

Por fim são apresentadas métricas de validação e avaliação que foram utilizadas. Nesta parte é feita uma introdução à pesquisa em grelha pois foi importante na otimização de parâmetros.

2.1 Mineração de Dados Relacionais

As bases de dados relacionais utilizam técnicas de R*elational Data Mining,* RDM, para manipulação dos seus dados [2]. Algoritmos de manipulação de dados procuram por padrões existentes nesses dados. A maioria das abordagens de manipulação de dados que existem são proposicionais e procuram por padrões apenas numa só tabela de dados. As abordagens de manipulação de dados relacionais, RDM, são baseadas em vários métodos como grafos ou redes semânticas, neste caso, o método relevante para a dissertação é a PLI, e é através destes que procuram por padrões em várias tabelas de dados [5].

Uma base de dados relacional, normalmente, consiste em várias tabelas e as suas relações e não apenas numa só tabela. Os padrões relacionais existentes envolvem várias relações de uma base de dados relacional, estes, podem ser expressos em subconjuntos de lógica de primeira ordem, assim, os padrões são mais expressivos [5].

2.1.1. Algoritmo RDM

Os algoritmos de procura utilizados são muito semelhantes àqueles utilizados na manipulação de dados de apenas uma tabela. Esta procura tanto pode ser exaustiva como heurística, no entanto, com apenas uma tabela estes algoritmos não conseguem extrair toda a informação necessária à obtenção de novas informações relativas aos dados, pois, o uso de apenas uma tabela pode causar a perda de alguns valores ou a repetição de outros, visto que valores presentes em diferentes tabelas iriam ser juntos numa só tabela [6].

Os algoritmos de manipulação de dados multi-relacionais são criados para permitirem a extração de padrões de múltiplas relações, com eficiência e eficácia, sem a necessidade de juntar os dados numa só tabela. Estes algoritmos possibilitam a extração de conhecimento em situações onde é importante manter a estrutura ou o relacionamento entre as múltiplas tabelas e têm sido aplicados em diversas áreas.

Hoje em dia, muitos algoritmos são completamente baseados numa configuração de atributo-valor, porém, esses algoritmos restringem o seu uso para conjuntos de dados que consistem apenas numa única tabela. Os dados estruturados são armazenados em bases de dados relacionais. O RDM pode realizar a descoberta de conhecimento em ambientes multi-relacionais, regras multirrelacionais e clustering multirrelacional [7]. Há várias abordagens suportadas pela manipulação de dados relacionais [7]:

- Programação Lógica Indutiva – *Inductive Logic Programming*;
- Clustering multirrelacional – Multi-relacional clustering ;
- Modelos probabilísticos relacionais – Probabilistic relational models.

Um algoritmo RDM também pode fazer uso de aprendizagem supervisionada binária onde utiliza a tarefa de classificação com o objetivo de aprender como atribuir um rótulo de classe para exemplos do domínio do problema, por exemplo, classificar e-mails como e-mails de *spam,* ou não. Assim, é utilizada uma tabela como *target* onde esta terá a variável que será um *target attribute*, a partir da qual é feito o processo de aprendizagem, no exemplo de e-mails, a variável *target* seria a classe com valores *spam* ou *not spam*. Se o modelo classifica corretamente a classe positiva então considera-se um T*rue Positive*, da mesma maneira, se o modelo classifica corretamente uma classe negativa, então, T*rue Negative.* Pelo contrário, se o modelo classifica como positivo, incorretamente, uma classe negativa, então, *False Positive*, e de forma semelhante, se classifica como negativo, incorretamente, uma classe positiva, então, False Negative.

2.1.2. RDM – Estrutura

A estrutura desta técnica trata de coletar dados sobre os dados – metadados – de uma base de dados e escolher a melhor abordagem para obter os melhores resultados. Esta estrutura é útil para os investigadores. Antes de ser implementado qualquer algoritmo, é importante referir que a compreensão do dados e a sua preparação são altamente necessárias [7].

As técnicas de manipulação de dados podem ser divididas em duas categorias:

- Abordagem da manipulação da tabela única;
- Manipulação de dados estruturados.

O termo manipulação de dados estruturados significa lidar com a complexidade presente no conjunto de dados. Esta complexidade dá-se quando os dados não podem ser armazenados apenas numa tabela ou não têm uma única representação numa tabela na qual as linhas ou as colunas não estão relacionadas entre si. Hoje em dia, utilizam-se bases de dados multirrelacionais – bases de dados com múltiplas tabelas [7].

Existem quatro tipos de técnicas de manipulação de dados estruturados, sendo elas:

- Manipulação de grafo (Graph Mining);
- Programação lógica indutiva (*Inductive Logic Programming*);
- Manipulação de dados semi-estruturados (Semi-structured Data Mining);
- Manipulação de dados multirrelacionais (Multi Relational Data Mining).

2.1.3. Aplicações do algoritmo RDM

A utilização do RDM possibilitou a sua aplicação em áreas ricas com dados estruturados e conhecimento de domínio, por outro lado, abordagens de apenas uma tabela, seriam bastante mais complicadas de aplicar nestas áreas pois as relações entre os dados seriam mais complicadas de obter.

Com a utilização deste algoritmo deixa de ser necessário, por parte do utilizador, a conversão das tabelas para linguagem prolog, a serem utilizadas posteriormente pelo programa Aleph. O algoritmo faz todo este processo automaticamente.

O RDM é usado em várias áreas, desde a análise de dados de negócios assim como áreas da bioinformática [8].

2.1.4. RDM – Exemplo

Considerado o exemplo dos comboios de Michalski, onde existem vários comboios diferentes a dirigirem-se para oeste e outros para este, como pode ser observado posteriormente na Fig. 2.1, para ser feita a representação destes numa só tabela, esta representação seria complicada, confusa e de difícil extração de novo conhecimento. Nesta tabela teriam de constar todas as condicionantes dos comboios, a direção, este ou oeste, os tetos, abertos ou fechados, a forma das carruagens, a quantidade de carruagens, a quantidade de objetos que transportam em cada uma delas, ou seja, seriam bastantes as caracteristicas que devem estar presentes nesta única tabela.

Por outro lado, se estes comboios forem representados em múltiplas tabelas, ainda assim todas as caracteristicas devem estar presentes, no entanto, a visualização destas fica mais simples e melhor estruturada de forma a que a sua compreensão seja facilitada. Posteriormente, é possível fazer uso de algoritmos de manipulação de dados multi-relacionais para a extração de novo conhecimento destes dados, ou seja, garantir nova informação que não é possível verificar mesmo quando estes dados estão bem estruturados em múltiplas tabelas.

2.2 Programação Lógica Indutiva

A programação lógica indutiva – *Inductive Logic Programming* (ILP), é uma área onde está presente o desenvolvimento de técnicas e ferramentas para manipulação relacional de dados. Para além de lidarem com dados armazenados em várias tabelas, a utilização de ILP permite a capacidade de considerar aquilo que é denominado como *background knowledge*, na forma de um programa lógico e fazer uso do poder de expressão da linguagem lógica para aprender padrões de primeira ordem, interpretáveis, a partir dos dados [2].

Esta área teve origem no cruzamento de *machine learning* – aprendizagem automática com programação lógica. Com base em *machine learning,* a ILP herda técnicas e ferramentas para induzir hipóteses a partir de exemplos. Em relação à programação lógica, a ILP herda o seu formalismo de representação [9]. A ILP faz uso de *machine learning* e tem a capacidade de usar o conhecimento no domínio, este conhecimento é muito importante para a pessoa que está a aprender e também para encontrar, a partir de exemplos, uma relação desconhecida em termos de relações já conhecidas daquele domínio [10].

A construção de hipóteses é uma característica muito importante de ILP e é feita com base no conhecimento do domínio, este conhecimento pode melhorar os resultados aprendidos [9].

2.2.1. Especificação formal de PLI

Um problema de aprendizagem em programação lógica indutiva, pode ser formulado como [11]: dado algum *background knowledge*, *B*, um conjunto de exemplos, *E* onde *E* pode ser dividido em dois subconjuntos E^+ (exemplos positivos, que deveriam ser provados pelas hipóteses encontradas) e E^- (exemplos negativos, que não deveriam ser provados pelas hipóteses encontradas), associados às classes, o objetivo é construir n cláusulas que constituem a hipótese, *H*, como se segue,

Suficiência: $B \wedge H \models E^+$.

Todos os exemplos em E^+ (exemplos positivos) podem ser logicamente derivados de B ∧ H.

Consistência forte: $B \wedge H_i \not\models E_j$ para todo j≠i, em que i varia entre 1 e |H| e j varia entre 1 e |E| e onde E_j e H_i representam um exemplo do conjunto de exemplos e uma hipótese do conjunto de hipóteses, respetivamente. A consistência forte proíbe a geração de qualquer hipótese h que seja inconsistente com os exemplos negativos E-, dado o conhecimento prévio B.

Normalmente, *B*, E_i e H_i são representados como programas lógicos e, assim, trazem uma linguagem de representação versátil para todos os constituintes do problema.

Uma boa característica de ILP é que as condições de suficiência e consistência podem ser verificadas por sistemas que provem este teorema tais como programação lógica, Prolog [12].

2.2.2. *Background knowledge*

A possibilidade de ser utilizado *background knowledge* durante a aprendizagem traz dois benefícios principais, um desses benefícios é o utilizador poder fornecer conhecimento sobre o domínio, o que pode tornar o processo de aprendizagem mais rico pois não é necessário começar do zero e o outro benefício é o facto de que o *background knowledge* especifica o vocabulário da linguagem definida bem como a semântica dos predicados associados e as suas relações com os exemplos.

O *background knolwedge* é apresentado no ficheiro com extensão “.b” e utiliza a sintaxe Prolog.

2.3 Aleph - *A Learning Engine for Proposing Hypotheses*

O sistema Aleph foi desenvolvido para ser um protótipo com a função de explorar ideias em ILP e foi escrito em Prolog. Modelos anteriores surgiram em 1993 como parte de um projeto realizado por Ashwin Srinivasan e Rui Camacho na Universidade de Oxford. Hoje em dia o Aleph usa funcionalidades de vários sistemas ILP como: Progol, FOIL, FORS, Indlog, MIDOS, SRT, Tilde e WARMR [4]. O Aleph possui uma poderosa linguagem de representação que permite representar expressões complexas e, simultaneamente, incorporar novos conhecimentos prévios facilmente. Este sistema é de código aberto, tornando-o um poderoso recurso a todos os investigadores de ILP [2]. Ao ser aplicado a dados presentes num dado contexto, revela a informação que é possível retirar destes, informação esta que só é possível retirar devido à utilização de um algoritmo de relação de dados, o qual faz a ligação de todos os conjuntos presentes nas diferentes tabelas.

2.3.1. Programa Aleph

O sistema Aleph segue um procedimento simples que pode ser descrito em quatro passos [2]:

1. Seleciona um exemplo positivo que deve ser generalizado. Se não houver nenhum, termina.
2. Constrói a cláusula mais específica que cobre o exemplo selecionado e que está dentro das restrições fornecidas. Chamámos a esta de “*bottom clause*”. Este passo é muitas vezes chamado de “saturação”.
3. Encontrar uma cláusula mais geral do que a *bottom clause*. Isto é feito com a pesquisa de um subconjunto dos literais presentes na *bottom clause* que contenham a "melhor" pontuação. Este passo é às vezes chamado de passo de "redução".
4. Adicionar a cláusula com melhor pontuação à teoria e remover todos os exemplos redundantes, ou seja, remover os exemplos positivos que são cobertos por aquela teoria para que na próxima iteração outros exemplos positivos, ainda não cobertos, possam ser escolhidos para a saturação. Voltar para o passo 1.

O uso do Aleph segue, normalmente, a seguinte estrutura:

1. Construir três ficheiros de dados, “ficheiro_de_dados.b”, “ficheiro_de_dados.n”, “ficheiro_de_dados.f”.
2. Ler todos os dados com o uso do comando *read_all(ficheiro_de_dados)*.
3. Construir a teoria com o uso do comando *induce.*

Relativamente aos ficheiros de dados:

- ficheiro_de_dados.b – contém todo o *background knowledge*. O *background knowledge* está na forma de cláusulas de Prolog que codificam informações relevantes ao domínio.
- ficheiro_de_dados.f – contém todos os exemplos positivos, que são simplesmente factos.
- ficheiro_de_dados.n – contém todos os exemplos negativos, que são, também, factos.

É importante referir que o ficheiro_de_dados.b contém restrições de linguagem para o Aleph, sendo elas os *mode declarations, types specifications* e *determinations:*

- *mode declarations* – descrevem as relações entre os tipos de dados. Estas podem aparecer em qualquer cláusula proveniente de uma hipótese do Aleph.
- *types specifications* – devem ser especificados para cada argumento de todos os predicados que serão usados na construção de uma hipótese.
- *determinations* – definem os predicados que podem ser usados para construção de uma hipótese.

Antes de avançar, é importante referir como é que a *bottom clause* é definida. O Aleph seleciona um dos exemplos positivos, constrói a *bottom clause,* e, com o uso dos *determinations* e dos *modes*, pesquisa no conjunto de factos todos aquele relacionados ao exemplo escolhido primeiramente. Assim, constitui-se a *bottom clause* que será utilizada para fazer a busca pela melhor teoria que cobre, idealmente, todos os exemplos positivos e nenhum dos exemplos negativos. Por outras palavras, o algoritmo assume que, se existe um padrão

entre os exemplos positivos, este deve estar “escondido” na definição de um dos exemplos positivos. Daí a escolha de um deles para construir a *bottom clause*.

2.3.2. Construção de uma teoria

O comando para selecionar os exemplos e construir a teoria é o *induce*, quando posto em prática, o Aleph aplica os quatro passos descritos inicialmente. Antes de ser executado o método *induce* é importante referir que são passados parâmetros ao programa de forma a garantir que este executa com uma variedade de restrições no espaço das hipóteses a serem aprendidas. O resultado, normalmente, é uma pesquisa que lista as cláusulas reveladas juntamente com uma contagem dos exemplos positivos e negativos cobertos por aquela teoria. Como ilustração mostramos como exemplo os comboios de Michalski. Na Fig. 2.1 observam-se diferentes comboios que seguem para este e para oeste.

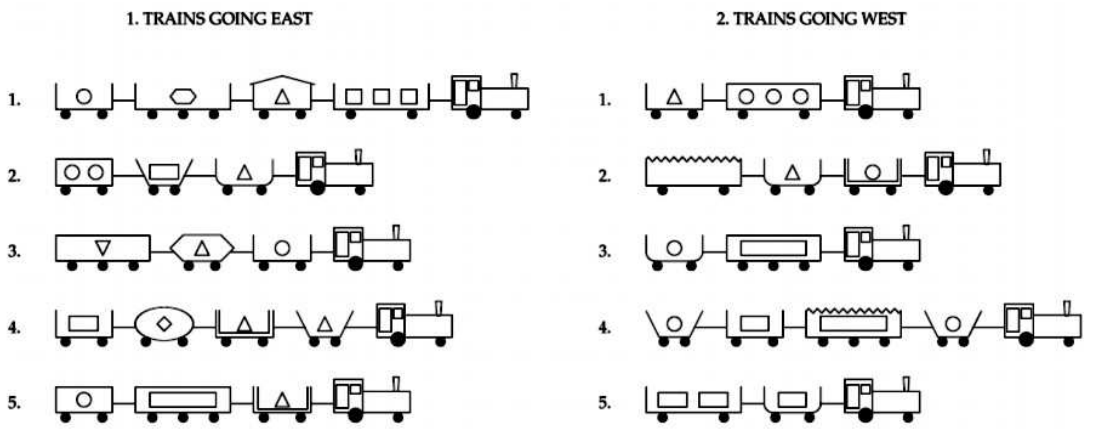


Figura 2.1: Comboios de Michalski, retirado de [31].

De seguida, verificam-se regras de primeira ordem geradas a partir de um exemplo positivo que foi saturado, juntamente com a sua cobertura de exemplos positivos e negativos como mostram os números dentro de parêntesis:

eastbound(A) :- has_car(A,B). [5/5]

eastbound(A) :- has_car(A,B), short(B). [5/5]

eastbound(A) :- has_car(A,B), open_car(B). [5/5]

eastbound(A) :- has_car(A,B), shape(B,rectangle). [5/5]

Com o uso das restrições da linguagem, o Aleph pesquisa por uma cláusula que melhor separa os exemplos positivos (este) dos exemplos negativos (oeste), escolhendo literais da *bottom clause*:

[teoria]

[Rule 1] [Pos cover = 5 Neg cover = 0]

eastbound(A) :- has_car(A,B), short(B), closed(B).

[pos-neg] [5]

O *induce* também informa sobre o desempenho nos dados de treino com uma matriz de confusão, este tipo de matriz é uma tabela que permite a visualização do desempenho de um algoritmo de classificação, neste caso, o “Pred” é o que foi previsto pelo Aleph como sendo positivo e negativo e o “Actual” são os casos positivos e negativos reais, só assim é possível retirar o valor de a*ccuracy,* algo como:

[Training set performance]

		Actual		
		+	-	
Pred	+	5	0	5
	-	0	5	5
		5	5	10

Accuracy = 100%

2.4 Métodos de validação e métricas de avaliação

2.4.1 Validação Cruzada – *k-fold cross-validation*

A validação cruzada é uma técnica para avaliar a capacidade de generalização de um modelo, a partir de um conjunto de dados [13].

O objetivo principal desta técnica é dividir o conjunto de dados em subconjuntos e utilizar alguns destes para estimar os parâmetros do modelo. Aos subconjuntos que irão estimar os parâmetros do modelo dá-se o nome de conjunto de treino, os restantes, que irão validar o modelo, dá-se o nome de conjunto de teste.

O método de validação cruzada denominado *k-fold* consiste em dividir o conjunto total de dados em k subconjuntos mutuamente exclusivos do mesmo tamanho e, a partir daí, um subconjunto é utilizado para teste e os k-1 restantes são utilizados para estimação dos parâmetros, calcula-se a precisão do modelo, esta indica um desempenho geral do modelo, ou seja, de todas as classificações, quantas o modelo classificou corretamente. Este processo é realizado k vezes onde se alterna, de forma circular, o subconjunto de teste [13]. Em cada fold um modelo é construído e avaliado. É possível então calcular uma métrica por fold ou somar o número de erros e acertos nos conjuntos de teste de todos os folds e calcular a métrica a partir destes contadores.

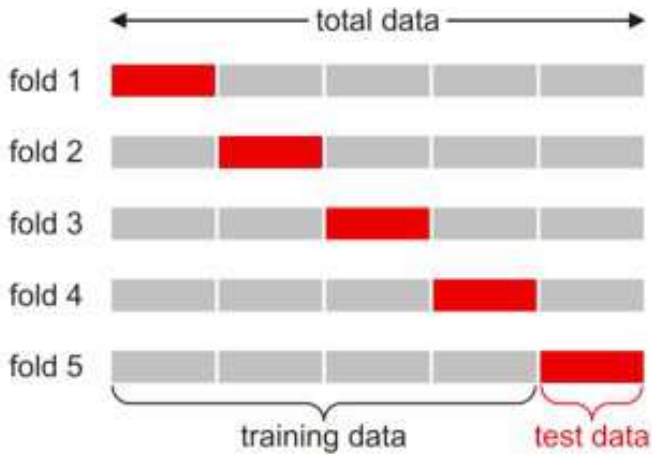


Figura 2.2: Divisão de dados em conjuntos independentes de treino e teste em *5-fold cross-validation*, retirado de [14].

2.4.2. *Stratified cross-validation*

Stratified k-fold cross-validation é uma variante da *cross-validation*. Os subconjuntos são selecionados para que a frequência de valores de cada classe seja aproximadamente igual em todos os subconjuntos. Por exemplo, no caso de classificação binária, isto significa que cada subconjunto contém, aproximadamente, as mesmas proporções de cada valor da classe.

2.5 Pesquisa em grelha – Grid Search

A forma tradicional de realizar a otimização de parâmetros tem sido a pesquisa em grelha.

Esta pesquisa é, simplesmente, uma busca exaustiva através de um subconjunto especificado manualmente do espaço de parâmetros de um algoritmo de aprendizagem. Um algoritmo que utiliza pesquisa em grelha deve ser encaminhado por alguma métrica de desempenho, normalmente medida por validação cruzada no conjunto de treino [15].

Uma vez que o espaço de parâmetros de *machine learning* pode incluir espaços de valor real ou ilimitado para certos parâmetros, pode ser necessário definir limites manualmente antes de ser aplicada a pesquisa em grelha.

A pesquisa em grelha sofre da maldição da dimensionalidade, mas muitas vezes pode ser feita recorrendo à paralelização do código porque as configurações dos parâmetros, que estão a ser avaliados, são, geralmente, independentes umas das outras [16].

Capítulo 3

Revisão da literatura

Nesta secção será apresentada a literatura com estudos semelhantes a este. Foi utilizado o Google Scholar como software de pesquisa dos artigos, onde foram retornados inúmeros artigos relacionados com o tema. Foram utilizados vários artigos para a escrita desta dissertação e são apresentados de acordo com a sua relevância para o estudo.

No entanto, não existem estudos sobre a utilização do pacote Python-rdm noutros trabalhos, daí a aplicação deste pacote nesta dissertação ser importante.

3.1 ILP e o Aleph

Stephen Muggleton é um conhecido fundador da PLI. Neste campo, fez contribuições para a teoria com a invenção de predicados, vinculação inversa e programas de lógica estocástica. Esteve presente no desenvolvimento de sistemas onde foi fundamental nos sistemas Duce, Cigol, Golem, Progol e Metagol e aplicações - especialmente tarefas de previsão biológica. Trabalhou num robô cientista juntamente com Ross D. King que é capaz de combinar a programação lógica indutiva com o aprendizado ativo [34].

Um estudo em que foi utilizado ILP foi publicado em 2016 por Ferreira et al. [17], utilizou ILP para prever regras relativas a dados de pacientes com cancro da mama. Foram usados os mesmos conjuntos de dados aplicados em [18]. Todo o conjunto de dados foi dividido em conjunto de treino e teste os quais são independentes um do outro.

Fizeram experiências com PLI, o programa Aleph e com a ferramenta *Waikato Environment for Knowledge Analysis* (WEKA) – uma coleção de algoritmos de aprendizagem de máquina para tarefas de manipulação de dados. Usaram o melhor classificador obtido em [19] para prever tumores malignos.

Consideram-se duas hipóteses:

- Podem remover variáveis e obter os mesmos resultados com o conjunto de teste?
- Podem produzir mais classificadores interpretáveis que podem ter um desempenho melhor do que aquele do modelo SVM?

Da primeira hipótese, com a utilização das ferramentas já mencionadas, retiram que com as variáveis e sem elas os resultados parecem ser muito semelhantes o que sugere que pode ser feita uma remoção das variáveis. No entanto, de acordo com outros estudos verifica-se que tal não é verdade, então o assunto é controverso [20][17].

Para a segunda hipótese, foram utilizadas curvas ROC para visualização dos desempenhos do Aleph e do SVM. Através de várias experiências com o programa Aleph e com a alteração dos parâmetros aplicados a este como *coverage, cost* ou *mestimate*, é possível concluir que os resultados obtidos mostram que o desempenho do Aleph está muito próximo do melhor classificador do SVM e até mesmo ultrapassar o desempenho do melhor classificador baseado em árvores de decisão com a vantagem de fornecer regras interpretáveis e mais compactas que ajudam a explicar as razões para um determinado caso ser maligno.

Apesar de utilizarem ILP, nenhum destes dois trabalhos tiraram partido de múltiplas tabelas relacionais, porém aproveitam o facto do conhecimento ser representado através de regras para adicionar conhecimento extra aos dados.

Em 2003, Sašo Džeroski publicou um estudo onde fez uso de algoritmos de manipulação de dados para procurar padrões entre os dados. Enquanto a maior parte das abordagens existentes de manipulação de dados procuram por padrões numa só tabela, as abordagens de manipulação de dados relacionais procuram por padrões em múltiplas tabelas a partir de uma base de dados relacional. Os tipos mais comuns de padrões e abordagens considerados em manipulação de dados foram extrapolados para manipulação de dados relacionais e, assim, os algoritmos de manipulação de dados relacionais utilizam árvores de decisão, métodos baseados em distâncias entre outros para descoberta de novas regras. Estas abordagens relacionais foram, com sucesso, aplicadas a um grande número de problemas em várias áreas sendo uma das mais notáveis a bioinformática [21].

Como a maior parte das técnicas relacionais têm a sua origem em abordagens de lógica de primeira ordem para aprendizagem, Sašo Džeroski apresenta uma introdução a PLI, posteriormente demonstra três abordagens principais em algoritmos relacionais, regras de

associação relacional, árvores de decisão relacionais e abordagens baseadas em distâncias relacionais [21].

Pahdy e Panigrahi em 2012 também apresentam um estudo relativo a abordagens relacionais onde referem as vantagens de aplicação destas, pois, tais como referido no estudo de Sašo Džeroski, ao utilizarem várias tabelas, e não apenas uma, a eficiência, o tempo de execução e a memória utilizada por estas abordagens são, de facto, melhores do que outros tipos de abordagens quando o objetivo é a retirada de nova informação de entre os dados [7].

3.2 Dados Temporais

Os dados temporais podem ser relevantes para qualquer estudo onde são utilizados ILP e *machine learning.*

Carrault et al. [12] utilizaram ILP para reconhecimento de arritmias através de eletrocardiogramas onde exploram as vantagens de ILP na introdução de conhecimento temporal existente no conjunto de dados. Têm como objetivo integrar abstração temporal e técnicas de diagnóstico em dois passos: o primeiro, é feito offline, e a ideia é construir um conjunto de caracterizações simbólicas de arritmias cardíacas. O segundo, é um passo online, que está encarregue de analisar o sinal e identificar arritmias, fazendo correspondência da representação do sinal às caracterizações pré-guardadas.

O principal objetivo é transformar a série temporal de entrada associada ao eletrocardiograma numa sequência de eventos simbólicos para: detetar e identificar os pontos de atividade cardíaca, caracterizar cada onda por um vetor de características, e classificar as ondas em classes normais ou anormais. Esta informação (tempo ocorrência, morfologia, etc.) é usada para gerar a sequência de eventos simbólicos que será usada como entrada pelo módulo de reconhecimento de crónicas que representa um raciocinador temporal. O reconhecimento crónico é um método de reconhecimento temporal que, de forma eficiente, faz reconhecimento online e temporal de padrões de eventos num fluxo de intervalos de tempo de eventos dado por sensores de um dispositivo que monitoriza um sistema ou um paciente [22].

O uso de *background knowledge* juntamente com o PLI, possibilita a análise de padrões em eletrocardiogramas que não são observados a olho nu, nem mesmo por peritos. Os resultados obtidos permitem verificar, numa grande quantidade de pacientes, quais aqueles que são propícios a desenvolver problemas cardíacos e, assim, garantir que estes começam um tratamento apropriado de forma a prevenir estes problemas. Desta forma, Carrault et al. [12] contribuíram para ser possível obter conhecimento de eletrocardiogramas

automaticamente sem ajuda de nenhum perito. Assim, é possível aceder facilmente, por parte de especialistas, a relatórios de fácil leitura [22].

Num outro estudo verifica-se a álgebra de intervalos temporais de *Allen - Allen's temporal interval algebra* (TIA), que é um formalismo conhecido, para representar e raciocínar sobre conhecimento temporal. TIA lida com a noção de tempo e a sua popularidade é devido ao facto de ser um dos primeiros formalismos temporais propostos e é de fácil implementação [23].

Lisboa et al. [24] teêm o objetivo de explorar as relações temporais aprendidas automaticamente dos dados usando um sistema de PLI.

A álgebra de intervalos temporais de *Allen* é baseada num modelo linear de tempo abordado por um conceito primitivo para representar tempo, o período de tempo, e uma relação binária entre dois períodos, a relação *meets*. O trabalho de *Allen* permitiu que este representasse um período de tempo por um intervalo real definido por dois pontos de fim.

Dois números a e b, pertencem ao conjunto dos números reais, R. Sistemas computacionais baseados em TIA de *Allen*, geralmente representam o período de tempo T como um intervalo de dois pontos de tempo, T = [t-, t+] que satisfazem t- < t+.

Relações temporais básicas de *Allen - Allen's temporal interval algebra*, TIA, é um formalismo baseado em relações binárias entre dois períodos de tempo, adequados para representar e raciocinar sobre conhecimento temporal. TIA é definido por 7 relações binárias básicas: *meets*, *before*, *overlaps*, *during*, *equal*, *starts* e *finishes* [25]. Duas incertezas podem ser relacionadas com o formalismo de TIA, uma tem a ver com a incerteza associada com representar números reais. Representar tempo como um intervalo de números reais num sistema de computador, por exemplo, tem o efeito colateral do aspeto discreto de tal representação, herdado da representação de números em computadores. A outra incerteza tem a ver com a inclusão, ou não, do intervalo de pontos fim para representar períodos de tempo.

Tabela 3.1: Relações temporais de Allen entre dois períodos representados por intervalos de tempo, retirado de [24].

TIA Relation	Conditions on both interval endpoints	Pictorial Representation
meets(A,B)	a– < b–, a– < b+ **a+ = b–**, a+ < b+	AAAA BBBBBBB
before(A,B)	a– < b–, a– < b+ **a+ < b–**, a+ < b+	AAAA BBBBBBB
overlaps(A,B)	**a– < b–**, a– < b+ **a+ > b–**, **a+ < b+**	AAAA BBBBBBB
starts(A,B)	**a– = b–**, a– < b+ a+ > b–, **a+ < b+**	AAAA BBBBBBB
finishes(A,B)	**a– > b–**, a– < b+ a+ > b–, **a+ = b+**	AAAA BBBBBBB
during(A,B)	**a– > b–**, a– < b+ a+ > b–, **a+ < b+**	AAAA BBBBBBB
equal(A,B)	**a– = b–**, a– < b+ a+ > b–, **a+ = b+**	AAAAA BBBBB

Lisboa et al. [24] concluem que sem uma escolha cuidadosa de exemplos positivos e negativos um sistema de aprendizagem automático dificilmente atinge uma expressão que pode ser útil. Verifica-se neste estudo a importância de algumas noções de relações temporais existentes. É importante reter alguma desta informação para a abordagem aos dados temporais presentes nesta dissertação.

3.3 Validação cruzada *k-fold*

Para treinar um modelo com dados de um conjunto de dados é necessária uma maneira ideal para tal. O treino deve ser feito de forma a que, embora o modelo contenha instâncias suficientes para treino, estas não devem se sobrepor ao modelo e, ao mesmo tempo, considera-se que, se não houver instâncias suficientes para treinar, o modelo não será treinado adequadamente e dará resultados insatisfatórios quando usado para teste. A precisão é importante quando se trata de classificação e deve-se sempre esforçar para obter a mais alta.

Ao trabalhar em pequenos conjuntos de dados, as escolhas ideais são validação cruzada *k-fold* ou validação cruzada *leave-one-out.* Em conjuntos de dados colossais, o primeiro pensamento é usar validação *hold-out*, em geral.

Yadav e Shukla [26] estudam as diferenças entre os dois esquemas de validação, fazem uma análise da possibilidade de usar a validação cruzada k-fold sobre a validação hold-out, mesmo em grandes conjuntos de dados. A experiência foi realizada em quatro grandes conjuntos de dados e os resultados mostram que até um certo limite, a validação cruzada *k-fold* pode, de facto, ser usada sobre a validação *hold-out* para classificação de qualidade.

Depois de analisar o uso de validação cruzada *k-fold* sobre validação *hold-out* em conjuntos de dados colossais, retira-se que é preferível utilizar validação cruzada *k-fold* para classificação de qualidade.

É realmente preferível utilizar validação cruzada *k-fold,* mesmo após ser feita uma compensação de tempo, porque ao utilizar este tipo de validação obtém-se um resultado mais preciso relativamente à validação *hold-out.*

Nesta dissertação será utilizado o método de validação baseado em *stratified cross-validation,* uma derivação de validação cruzada onde os conjuntos de treino e teste apresentam um balanceamento dos elementos de classe.

Capítulo 4

Mineração de Dados Multirrelacionais

Neste capítulo apresentamos a nossa experiência com a utilização de ILP com a utilização de um pacote Python específico, o Python-RDM. Será descrito todo o trabalho realizado, as ideias que foram desenvolvidas, os obstáculos que foram surgindo e devidamente ultrapassados, e os resultados atingidos dos quais foram retiradas conclusões.

4.1 Python-RDM

O Python-RDM [33] é um pacote de código cuja contribuição para o seu desenvolvimento foi dada por Anže Vavpetič, Nicolas Lachiche, Alain Shakour, Matic Perovšek e Vid Podpečan. Este projeto Python foi criado para permitir, mais facilmente, a utilização de várias implementações de algoritmos de programação lógica indutiva e de manipulação de dados relacionais. Um objetivo importante deste projeto é oferecer uma ponte comum entre bases de dados relacionais e implementações ILP e manipulação de dados relacionais, através de linguagem Python. Assim é estabelecida mais uma abordagem para utilizadores com preferência pela linguagem Python.

Instruções de como obter o pacote e instalar podem ser obtidas no apêndice A.

Atualmente este projeto oferece suporte para algumas bases de dados e vários algoritmos, entre eles o Aleph.

Uma vez que para esta dissertação utilizamos o sistema Aleph, no pacote do Python-rdm procuramos pelo código referente a este sistema, o qual apresenta 251 linhas de código e que nos permite aplicar o método *induce* deste sistema. Além deste e dos seus componentes,

também é necessário utilizar um dos códigos apresentados nos exemplos deste pacote para, posteriormente, serem aplicadas as devidas mudanças de forma a executar o programa conforme o desejado. O exemplo escolhido é um pequeno código em que o método *induce* do programa que executa o Aleph é aplicado, no qual são utilizadas tabelas com dados referentes aos comboios de Michalski, o qual é apresentado no bloco de código 4.1.

```
1  from rdm.db import DBVendor, DBConnection, DBContext, AlephConverter
2  from rdm.wrappers import Aleph
3
4  # Provide connection information
5  connection = DBConnection(
6      'ilp',             # User
7      'ilp123',          # Password
8      'workflow.ijs.si', # Host
9      'ilp',             # Database
10 )
11
12 # Define learning context
13 context = DBContext(connection, target_table='trains', target_att='direction')
14
15 # Convert the data and induce features using Aleph
16 conv = AlephConverter(context, target_att_val='east')
17 aleph = Aleph()
18 theory, features = aleph.induce('induce_features', conv.positive_examples(),
19                                 conv.negative_examples(),
20                                 conv.background_knowledge())
21 print(theory)
```

Bloco de Código 4.1: Código presente no pacote Python-rdm referente a um exemplo aplicado com o sistema Aleph aos comboios de Michalski.

No bloco de código 4.1 verifica-se que o sistema Aleph é utilizado e o método *induce* deste, é a partir deste código que são feitas as alterações para que este se aplique ao estudo desta dissertação.

Neste programa, são introduzidas tabelas, manualmente ou através de uma conexão a uma base de dados, e, a partir destas, com a aplicação do sistema Aleph, é retirada uma teoria, a partir da qual é possível a retirada de regras com informação relativa à relação que os dados apresentam entre si. A partir destas regras retiram-se novas informações presentes nestes dados não visíveis através de inspeção visual ou manual.

4.2 Relações entre tabelas

A importância do algoritmo RDM, o qual serve para criar relações entres os dados presentes em múltiplas tabelas, para a realização desta dissertação é, de facto, significativa.

Inicia-se esta dissertação a partir de um programa base, presente no repositório do RDM apresentado no bloco de código 4.1, onde neste algoritmo temos um exemplo de duas tabelas relativas a um conjunto de comboios, com informação sobre estes.

Através de um carregamento, com a utilização de código *Python*, dos dados presentes nas tabelas e com o uso de bibliotecas presentes no repositório do RDM, é possível definir um contexto de aprendizagem.

São usados *targets*, sendo eles, uma *target_table,* tabela bidimensional na qual estão presentes um identificador de um determinado exemplo e o valor de classe a ser estudado referente a esse exemplo.

No caso de dados relativos ao *breast cancer*, seriam valores de positivo ou negativo caso exista cancro ou não em determinado paciente e o outro *target* é um *target_attribute* que representa a coluna na qual está presente o valor a ser estudado na *target_table,* e, ao aplicar os recursos do programa Aleph juntamente com aqueles presentes no RDM, os dados são convertidos, assim, podemos induzir as características presentes nestes dados.

4.2.1. Aplicação do RDM aos dados

Depois de uma compreensão do programa referido anteriormente, é então necessária uma nova escrita deste algoritmo de forma a que possam ser criadas relações entre tabelas que contém dados de análises clínicas de um número extensivo de pacientes, dados estes na forma numérica. Para tornar isto possível foram necessárias alterações às várias tabelas, pois, para haver relação entre estas é necessário que exista ligação entre todas, sem esta particularidade, o programa não obtém resultados.

É apresentado de seguida o código com algumas alterações feitas de forma a serem aplicadas às várias tabelas onde estão presentes grandes quantidades de dados e a utilização do sistema Aleph para aplicação do método *induce*, de forma a serem induzidas as regras que contém nova informação presente nos dados. As alterações feitas dividiram o código em dois passos de forma a que, no primeiro passo, o programa lê as tabelas de dados e a partir destas, gera regras e ficheiros Prolog. De seguida, o utilizador tem a oportunidade de intervir manualmente com a inserção de novos predicados ou modificar os que já lá se encontram. No

segundo passo, o programa lê os ficheiros Prolog e, com o uso de *cross-validation,* gera novas regras e novos ficheiros.

A partir dos blocos de código 4.2, 4.3 e 4.4 é possível verificar as várias modificações que foram feitas ao código inicial, presente no bloco de código 4.1, de forma a garantir o sucesso desta dissertação.

```
from rdm.db import DBVendor, DBConnection, DBContext, AlephConverter, CSVConnection
from rdm.wrappers import Aleph
import csv
import os
import codecs
import numpy as np
import sys
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold


# Load data from a set of csv files
connection = CSVConnection(['/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/proc_new.csv',
                            '/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/proc_biopsyabnNewID.csv',
                            '/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/proc_abnormalityNewID.csv',
                            '/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/proc_mammoNewID.csv',
                            '/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/proc_upgradeNewID.csv',
                            '/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/proc_pathtypeNewID.csv',
                            '/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/proc_patientNewID.csv',
                            ])

# Define learning context
context = DBContext(connection, target_table='proc_new', target_att='ResultantUpgrade')
```

Bloco de código 4.2: Código com algumas alterações, introdução de tabelas e aplicação da tabela *target*.

```
aleph = Aleph()

# Get the data from the file
with open('/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/tmp/default.f', 'rb') as fp:
  PosFile = fp.read()

with open('/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/tmp/default.n', 'rb') as fp:
  NegFile = fp.read()

with open('/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/tmp/default.b', 'rb') as fp:
  BFile = fp.read()

def last(n):
  return n[0]

def sort(tuples):
  return sorted(tuples, key=last)
```

Bloco de código 4.3: Continuação do código apresentado em Bloco de Código 4.2 com aplicação do sistema Aleph.

```
45  for m in range(0, 1):
46    for minpos in range(2, 3):
47      for noise in range(10, 11):
48          filename = 'trace:' + str(m) + str(minpos) + str(noise)
49          aleph.set('depth', 10)
50          aleph.set('evalfn', 'mestimate')
51          aleph.set('m', m)
52          aleph.set('verbosity', 3)
53          aleph.set('minpos', minpos)
54          aleph.set('noise', noise)
55          aleph.set('record', True)
56          aleph.set('recordfile', filename)
57          aleph.set('test_pos', 'test.f')
58          aleph.set('test_neg', 'test.n')
59
60          theory, features = aleph.induce('induce_features', PosFile, NegFile, BFile)
61          print(theory)
62          print(features)
```

Bloco de código 4.4: Continuação do código apresentado em Bloco de Código 4.3.

Nos blocos de código 4.2, 4.3 e 4.4 é possível verificar que são várias as tabelas utilizadas, que são criados três ficheiros com extensões “.f”, “.b” e “.n” que contém casos positivos, *background knowledge* e casos negativos, respetivamente e também é possível visualizar o sistema Aleph a ser posto em prática com alguns dos parâmetros que este pode conter. É importante referir que inicialmente o código original do pacote Python-rdm não apresenta os ficheiros “.f”, “.b” e “.n”, estes ficheiros seriam temporários e, assim, eram apagados tornando impossível a eventual intervenção do utilizador.

4.2.2. *Datasets*

Os dados estão agrupados em sete tabelas e os conjuntos de dados referem-se a características de pacientes que têm ou não cancro da mama. Em cada tabela existem vários atributos sendo que cada um deles é relativo a diferentes características de cada paciente. De seguida serão utilizados outros conjuntos de dados referentes a vídeo-jogo denominado FIFA *European Soccer*, em que estes contém informação temporal que, posteriormente, será importante para a contínua obtenção de novo conhecimento através dos dados.

As várias tabelas de dados de entrada estão em formato csv e nas várias tabelas é possível verificar que cada uma delas apresenta, na segunda linha, os tipos das variáveis e, na terceira linha, as chaves primárias e secundárias que irão permitir e facilitar a realização das relações entres as várias tabelas, como pode ser observado nas tabelas 4.1 e 4.2.

4.2.3. Estudo de caso

1. Breast Cancer [18]

Os sinais que refletem o aparecimento do cancro da mama são vários: um nódulo na mama, uma mudança no formato da mama, ondulações na pele, fluido vindo do mamilo, um mamilo recém-invertido ou uma mancha de pele vermelha ou escamosa [27]. Pessoas com a doença, podem vir a sofrer dor nos ossos, falta de ar ou pele amarelada [28].

São várias as técnicas de deteção de cancro da mama, algumas delas como, exames clínicos da mama - mamografias, mamografia digital de campo completo, deteção auxiliada por computador - *Computer-Aided Design* (CAD), Magnetic Resonance Imaging - MRI, entre outras. A deteção auxiliada por computador, CAD, é um software de reconhecimento de padrões que identifica e seleciona anomalias - que possam ser suspeitas - em imagens posteriormente analisadas por um radiologista. O CAD é também um sistema que identifica imagens de tumores malignos e benignos, é possível observar as imagens retiradas de um CAD na Fig. 4.1, no entanto, não marca todas as áreas suspeitas, portanto não pode ser o único sistema a ser usado na deteção de cancro da mama [29].

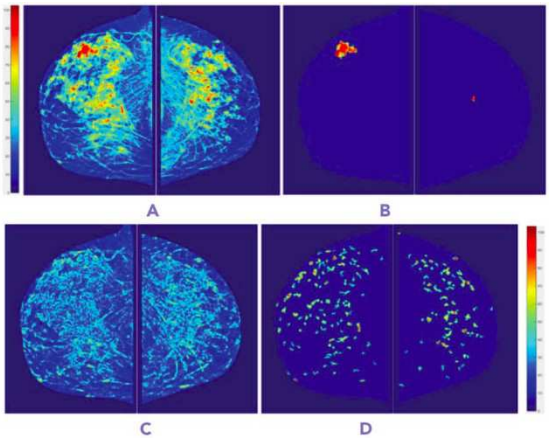


Figura 4.1: Resultados intermediários das etapas de processamento de imagem, incluindo (A) os mapas de densidade mamária calculados, (B) regiões de densidade focal detectadas, (C) mapas de variação de densidade local e (D) mapas de imagem desenvolvidos com o uso de filtragem *bandpass* gaussiana. Retirado de [30].

Primeiramente, foi criada uma tabela onde foram colocados os casos de pacientes onde é conhecida a existência de tumor, ou não, com um respetivo número de identificação, ID. De seguida, foram alteradas as chaves primárias das tabelas, ou, caso não existissem, foram criadas, para que estas representem sempre um número de identificação naquela tabela.

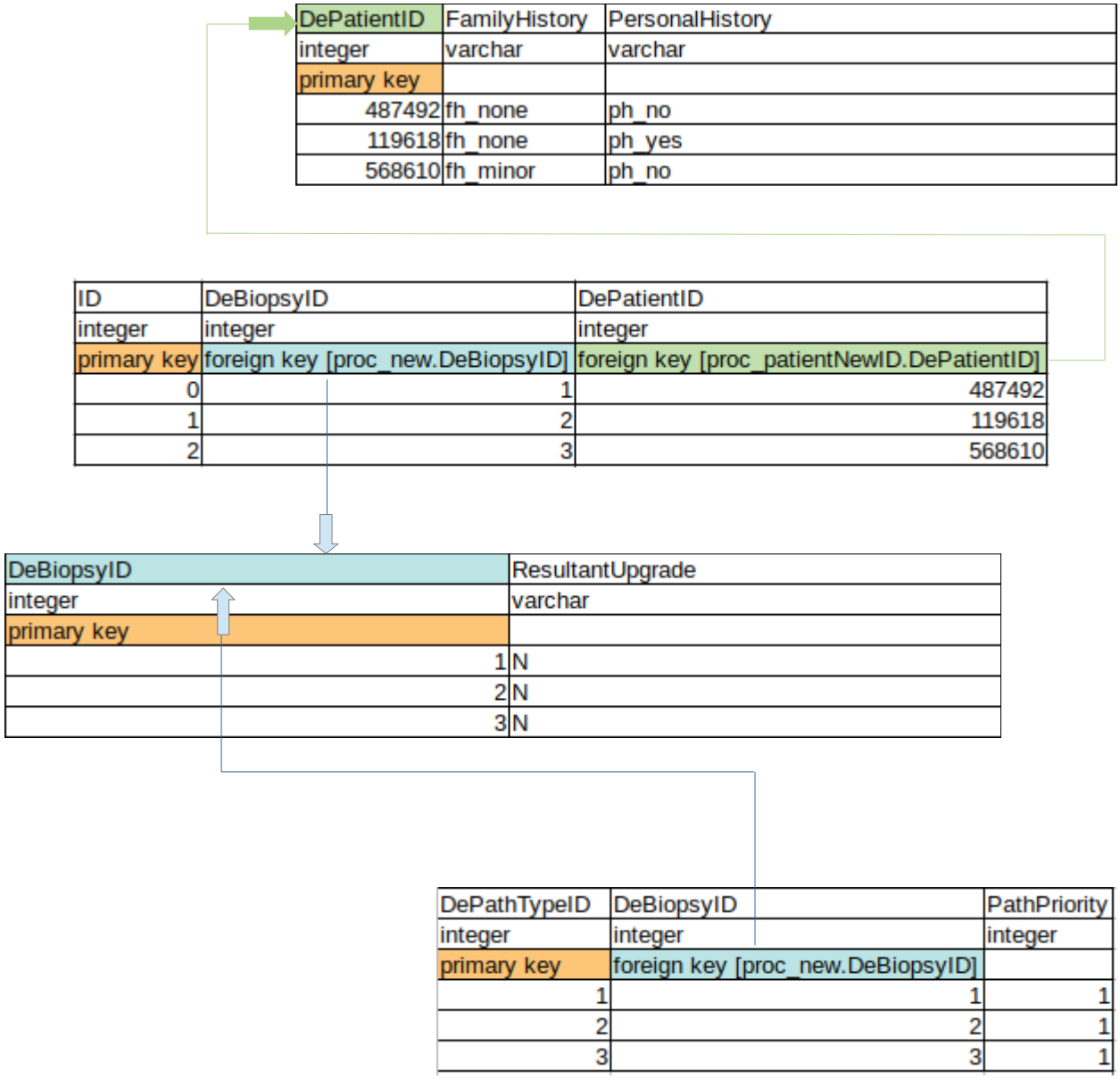
Posteriormente, foram estudadas as ligações entre as tabelas e, após estudo e compreensão, acrescentadas chaves estrangeiras que constituem a tabela à qual o atributo, presente na coluna onde foi acrescentada a chave, representa.

Como são utilizadas múltiplas tabelas, este pré-processamento é fundamental para garantir uma ligação entre os dados de todas as tabelas quando for executado o código RDM que irá fazer com que estas ligações sejam criadas, pois só assim é possível uma extração de conhecimento.

Na tabela 4.1 verificam-se as várias tabelas com alguns dos seus dados e as chaves primárias de cada tabela. Em cada uma delas existe uma célula cor-de-laranja que nos mostra a coluna onde está presente a chave primária dessa mesma tabela e esta encontra-se sempre no topo dessa coluna. Também se pode observar, em algumas delas, que existem chaves estrangeiras que correspondem à chave de outra tabela, a qual irá ter uma ligação com esta. As setas representam as ligações feitas entre as chaves das tabelas.

Por questões de praticabilidade e facilidade em serem observadas estas ligações, são apresentadas duas tabelas iguais, as quais contém a chave primária *DeBiopsyID*, pois esta tabela mostra a tabela *target* a utilizar assim como o *target_attribute*, neste caso *ResultantUpgrade*.

Tabela 4.1: Conjunto de tabelas com as suas relações representadas por ligações e cores correspondentes.



DePatientID	FamilyHistory	PersonalHistory
integer	varchar	varchar
primary key		
487492	fh_none	ph_no
119618	fh_none	ph_yes
568610	fh_minor	ph_no

ID	DeBiopsyID	DePatientID
integer	integer	integer
primary key	foreign key [proc_new.DeBiopsyID]	foreign key [proc_patientNewID.DePatientID]
0	1	487492
1	2	119618
2	3	568610

DeBiopsyID	ResultantUpgrade
integer	varchar
primary key	
1	N
2	N
3	N

DePathTypeID	DeBiopsyID	PathPriority
integer	integer	integer
primary key	foreign key [proc_new.DeBiopsyID]	
1	1	1
2	2	1
3	3	1

DeBiopsyID	ResultantUpgrade
integer	varchar
primary key	
1	N
2	N
3	N

ID	DeBiopsyID	DeAbnID
integer	integer	integer
primary key	foreign key [proc_new.DeBiopsyID]	foreign key [proc_abnormalityNewID.DeAbnID]
0	1	976654
1	2	651738
2	3	451285

DeAbnID	DeMammoID	BiradsCat
integer	integer	varchar
primary key	foreign key [proc_mammoNewID.DeMammoID]	
976654	720231	b0
651738	165357	b8
451285	659410	b8

DeMammoID	PriorMammo	ReasonMammo
integer	varchar	varchar
primary key		
720231	pm_yes	s
165357	pm_yes	v
659410	pm_yes	v

2. FIFA *European Soccer* [32]

Antes de continuar com os dados sobre o cancro da mama, é importante referir que para além destes dados serão utilizados outros de um ambiente diferente.

Serão utilizados, posteriormente, conjuntos de dados, fornecidos pela plataforma *kaggle*, referentes a um vídeo-jogo, FIFA *European Soccer*. A mudança para este tipo de dados deve-se ao facto de serem escassos os conjuntos de dados com conteúdo temporal relativos ao cancro da mama e, como neste tipo de dados é fácil de retirar conteúdo temporal, determinou-se que seriam usados para esse fim [32].

No entanto, nesta dissertação foram utilizados primeiramente os dados relativos ao cancro da mama e foram trabalhados de forma a serem obtidas todas as informações possíveis com estes, ou seja, verificar se era possível criar as relações entre as várias tabelas, aplicação dos métodos *prepare* e *induce* do Aleph e retirada dos melhores parâmetros a aplicar ao programa, estudo das regras aprendidas de forma a garantir que era possível a extração de novo conhecimento e verificar se seria executável a aplicação de informação temporal ao programa de forma a garantir a extração de novo conhecimento. Depois dos dados relativos ao cancro da mama serem utilizados, são então trabalhados os dados referentes ao FIFA *European Soccer*. Assim, esta dissertação continua com a explicação do que foi feito com os primeiros conjuntos de dados e, em seguida, serão explorados os dados do FIFA *European Soccer*, dados esses com conteúdo temporal.

Primeiramente, foi criada uma tabela onde foram colocados os jogadores onde é conhecida a existência da variável *potential* e o seu valor é acima de 75 ou não, com um respetivo número de identificação, ID. De seguida, foram alteradas as chaves primárias das tabelas, ou, caso não existissem, foram criadas, para que estas representem sempre um número de identificação naquela tabela. Posteriormente, foram estudadas as ligações entre as tabelas e, após estudo e compreensão, acrescentadas chaves estrangeiras que constituem a tabela à qual o atributo, presente na coluna onde foi acrescentada a chave, representa.

Como são utilizadas múltiplas tabelas, este pré-processamento é fundamental para garantir uma ligação entre os dados de todas as tabelas quando for executado o código RDM que irá fazer com que estas ligações sejam criadas, pois só assim é possível uma extração de conhecimento.

De seguida é apresentada a tabela 4.2 semelhante à tabela 4.1 apresentada na secção 4.2.3 com as mesmas características desta, no entanto, referente aos dados do FIFA *European Soccer.*

Tabela 4.2: Conjunto de tabelas com as suas relações representadas por ligações e cores correspondentes referentes aos dados do FIFA *European Soccer*.

ID	Potential
integer	varchar
primary key	
1	M
2	M
3	B
4	M

ID	DID	Player_Api_ID
integer	integer	integer
primary key	foreign key [nova.ID]	foreign key [player.Player_Api_ID]
1	1	71557
2	2	728414
3	3	693138

Player_Api_ID	Player_Name	Birthday
integer	varchar	text
primary key		
505942	Aaron Appindangoye	29-02-92 00:00
155782	Aaron Cresswell	15-12-89 00:00
30572	Aaron Galindo	08-05-82 00:00

CID	Name
integer	varchar
primary key	
1	Belgium
1729	England
4769	France

LID	Country_ID	Name
integer	integer	varchar
primary key	foreign key [country.CID]	
1	1	Belgium Jupiler League
1729	1729	England Premier League
4769	4769	France Ligue 1
7809	7809	Germany 1. Bundesliga

Team_Api_ID	Team_Long_Name	Team_Short_Name
integer	varchar	varchar
primary key		
9987	KRC Genk	GEN
9993	Beerschot AC	BAC
10000	SV Zulte-Waregem	ZUL
9994	Sporting Lokeren	LOK

ID	Team_Api_ID	Date	buildUpPlaySpeed
integer	integer	text	integer
primary key	foreign key [team.Team_Api_ID]		
1	9930	2010-02-22 00:00:00	60
2	9930	2014-09-19 00:00:00	52
3	9930	2015-09-10 00:00:00	47

4.2.4. Aleph – *SET*

Um dos primeiros obstáculos encontrados durante o seguimento da dissertação foi, após as relações das tabelas criadas e aplicadas ao programa, o resultado não era o esperado. O programa apenas dava como output uma regra relacionada com os dados das tabelas, apresentada na figura 4.2, no entanto, isto não é considerado um problema do programa mas um problema de ajuste de parâmetros. Sabendo que a quantidade de dados presentes nas tabelas é bastante extensiva e que o número de tabelas a considerar também é significativo, retira-se que pode existir algum tipo de problema, pois o número de regras atingido é muito baixo e, esta não fornecia nenhum tipo de nova informação, assim, o conhecimento obtido é escasso, o que não deveria acontecer quando são usadas grandes quantidades de dados.

```
[features from good clauses]
[pos cover = 2 neg cover = 0] [pos-neg] [2]
feature(1,(target_y(A):-proc_new_has_proc_upgradeNewID(A,B),proc_upgradeNewID_Age(B,64))).
```

Figura 4.2: Primeiro obstáculo, resultado obtido foi apenas uma regra.

Para melhor compreensão de todo o algoritmo RDM, foi feito um estudo de alguns dos componentes que este utiliza, nomeadamente, um dos códigos importados para o programa será o do Aleph. Assim, após uma análise ao código do Aleph e ao manual disponível online [4], são modificados alguns parâmetros que estavam definidos de forma padrão. Para alteração destes parâmetros é utilizado o método *set* presente no Aleph, um predicado que forma a base para definir uma série de valores de parâmetros utilizados pelo Aleph, onde os parâmetros alterados foram, profundidade, método de avaliação (de acordo com este pode ser alterada a variável m) e ruído – *noise.* Os parâmetros são configurados para os valores da seguinte forma:

set(Parâmetro, Valor).

Um parâmetro também pode deixar de estar configurado, da seguinte forma:

noset(Parâmetro).

No manual do Aleph podem ser encontrados vários parâmetros que podem ser definidos com a utilização do método *set.*

Para esta dissertação os parâmetros definidos manualmente foram os seguintes:

- set(‘*depth*’*,* 10)
- *set*(‘*evalfn*’, ‘*mestimate*’)

- *set*(‘m’, m)
- *set*(‘*verbosity*’, 3)
- set(‘*minpos*’, minpos)
- *set*(‘*noise*’, noise)
- *set*(‘i’, 5)

Estes parâmetros estão explicados no manual do Aleph disponibilizado online, onde:

- *depth* será um limite definido de profundidade;
- o método *evalfn* define a função de avaliação para a pesquisa, neste caso, *mestimate;*
- o valor de ‘m’, utilizado pela função de avaliação;
- o valor de *minpos,* que será o número mínimo de exemplos positivos para que a procura continue apenas a partir de hipóteses que cubram aquele número mínimo de exemplos positivos. Todos os caminhos que tenham hipóteses que cubram menos do que aquele número são descontinuados;
- o valor de *noise*, um limite superior para o número de exemplos negativos que podem ser cobertos para uma cláusula ser considerada aceitável.

Inicialmente são utilizados intervalos para estes parâmetros onde o valor do ‘m’ varia entre 1 e 20, o valor do ‘minpos’ varia entre 1 e 10 e o valor de *noise* varia entre 10 e 100, estes intervalos foram retirados de um estudo feito em [17]. Posteriormente serão alcançados valores que nos garantem a obtenção das melhores regras para aqueles conjuntos de tabelas pois são estes que nos permitem o melhor valor de *accuracy* na matriz de confusão e, assim, estes intervalos já não serão necessários. Outros dois valores de parâmetros que são aplicados são:

- *verbosity,* para ajustar a quantidade de mensagens de *debugging* e *trace* que são emitidas pelo programa.
- valor de ‘i’, que define um limite de novas variávies ao criar ou refinar uma regra.

Após a alteração dos parâmetros são obtidas várias regras relacionadas com os dados presentes nas tabelas, regras estas que mostram relações e características que são retiradas dos múltiplos cruzamentos dos dados das diferentes tabelas tais como podem ser observadas na Fig. 4.3.

```
[pos cover = 2 neg cover = 7] [m estimate] [0.212727272727273]
feature(28,(target_y(A):-proc_new_has_proc_upgradeNewID(A,B),proc_new_has_proc_pathtypeNewID(A,C),proc_pathtypeNewID_PathDxAbbr(C,'FC'))).
[pos cover = 2 neg cover = 3] [m estimate] [0.334285714285714]
feature(29,(target_y(A):-proc_new_has_proc_pathtypeNewID(A,B),proc_new_has_proc_pathtypeNewID(A,C),proc_pathtypeNewID_PathDx(C,'Sclerosing adenosis'))
).
[pos cover = 2 neg cover = 7] [m estimate] [0.212727272727273]
feature(30,(target_y(A):-proc_new_has_proc_pathtypeNewID(A,B),proc_new_has_proc_pathtypeNewID(A,C),proc_pathtypeNewID_PathDxAbbr(C,'FC'))).
[pos cover = 2 neg cover = 3] [m estimate] [0.334285714285714]
feature(31,(target_y(A):-proc_new_has_proc_pathtypeNewID(A,B),proc_pathtypeNewID_PathPriority(B,1),proc_pathtypeNewID_PathDx(B,'Sclerosing adenosis'))
).
[pos cover = 2 neg cover = 7] [m estimate] [0.212727272727273]
feature(32,(target_y(A):-proc_new_has_proc_pathtypeNewID(A,B),proc_pathtypeNewID_PathPriority(B,1),proc_pathtypeNewID_PathDxAbbr(B,'FC'))).
[pos cover = 2 neg cover = 7] [m estimate] [0.212727272727273]
feature(33,(target_y(A):-proc_new_has_proc_upgradeNewID(A,B),proc_upgradeNewID_BiopsySide(B,'L'),proc_upgradeNewID_DisappearanceAbn(B,'Y'))).
[pos cover = 3 neg cover = 8] [m estimate] [0.256923076923077]
feature(34,(target_y(A):-proc_new_has_proc_upgradeNewID(A,B),proc_upgradeNewID_NeedleGauge(B,9),proc_upgradeNewID_DisappearanceAbn(B,'Y'))).
```

Figura 4.3: Várias regras obtidas após alteração dos parâmetros.

4.3 Dividir o programa

Ao executar o programa para serem obtidas tanto a teoria como a matriz booleana de caracteristicas relativas aos dados, é chamado o método *induce* do programa Aleph, este método está presente no código que é importado para o programa.

Através da chamada da função *induce* é feita uma chamada à função *prepare* a qual inicia três ficheiros, tendo eles extensões já referidas, ‘.n’, ‘.b’, ‘.f’, guardados num diretório. Os ficheiros ‘.n’ e ‘.f’, guardam informação dos dados que são positivos e negativos, em relação à presença, ou não, de tumor, respetivamente, e o ficheiro ‘.b’ guarda o *background knowledge* retirado a partir dos dados e das relações entre as tabelas.

Adicionar relações aos dados pode ajudar um sistema de aprendizagem relacional a encontrar regras mais ricas e informativas. Para fazer isto, o utilizador deveria poder ter acesso aos ficheiros de dados convertidos das tabelas para notação lógica (relações). Para isso, é fundamental uma divisão do programa em dois programas. No primeiro, deve constar o método *prepare* do Aleph, apresentado no bloco de código 4.5, pois obtemos os ficheiros anteriormente referidos e é num deles, nomeadamente o ficheiro ‘.b’, que deve ser possível a sua alteração antes do restante programa ser executado. No segundo programa deve constar o restante código que executa o método *induce,* apresentado no bloco de código 4.6, a partir do qual são obtidas as características referentes aos dados, este método utiliza os ficheiros obtidos através do método *prepare,* os quais são guardados num diretório.

```
84
85  connection = CSVConnection(['/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/fold' + str(c) + '/fold' + str(c
86                             '/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/proc_biopsyabnNewID.csv',
87                             '/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/proc_abnormalityNewID.csv',
88                             '/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/proc_mammoNewID.csv',
89                             '/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/proc_upgradeNewID.csv',
90                             '/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/proc_pathtypeNewID.csv',
91                             '/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/proc_patientNewID.csv',
92                             ])
93
94  # Define learning context
95  context = DBContext(connection, target_table='fold' + str(c) + '_treino2', target_att='ResultantUpgrade')
96
97  # Convert the data and induce features using Aleph
98  conv = AlephConverter(context, target_att_val='Y')
99
100 aleph = Aleph()
101
102 filestem = 'default'
103
104 PosFile, NegFile, BFile = aleph.prepare(filestem, conv.positive_examples(),
105                                         conv.negative_examples(),
106                                         conv.background_knowledge())
```

Bloco de código 4.5: Primeira parte do programa onde é chamado o método *prepare* do programa Aleph e são criados os ficheiros '.b', '.f' e '.n'.

```
78  for m in range(1, 2):
79    for minpos in range(1, 2):
80      for noise in range(10, 11):
81        #filename = 'trace:' + str(m) + str(minpos) + str(noise)
82        aleph.set('depth', 10)
83        aleph.set('evalfn', 'mestimate')
84        aleph.set('m', m)
85        aleph.set('verbosity', 3)
86        aleph.set('minpos', minpos)
87        aleph.set('noise', noise)
88
89        shutil.rmtree('/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/tmp')
90        shutil.copytree('/home/mdias23i/Faculdade/5º/Tese/toMiguelDias/fold' + str(c) + '/tmp'
91
92        theory, features = aleph.induce('induce_features', PosFile_treino,
93                                        NegFile_treino,
94                                        BFile)
95        print(theory)
96        print(features)
```

Bloco de código 4.6: Segunda parte do programa onde é chamado o método *induce* do programa Aleph e são feitos dois *prints*, um da teoria e outro das regras associadas aos dados.

4.3.1. Matriz – Positivos e Negativos

Além das características obtidas dos dados das tabelas, o método *induce* também origina uma teoria. Essa teoria é apresentada através de uma matriz, onde cada coluna corresponde a uma regra e os símbolos positivos (+) e negativos (-) representam se uma determinada regra presente na teoria é de facto verdadeira para um determinado exemplo. Isto só é possível determinar pois na última coluna da matriz verifica-se para cada linha uma *string,* sendo ela, *negative* ou *positive,* que indica se naquela linha os símbolos deveriam ser positivos ou negativos. No caso onde surge um símbolo positivo e a última coluna contém *positive* então a regra é de facto verdadeira, *True Positive – TP*, no entanto, um símbolo negativo e uma última coluna *negative*, então a regra está correta porém é um caso *True Negative – TN*. Além disso, também podem aparecer símbolos positivos em casos *negative* e símbolos negativos em casos *positive*, representam assim *False Positive – FP* – e *False Negative – FN* – respetivamente.

```
-,+,+,+,+,+,+,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,positive
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,positive
-,-,-,-,-,-,-,+,+,+,+,+,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,positive
-,-,-,-,-,-,-,-,-,-,-,-,+,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,positive
-,-,-,-,-,-,-,-,-,-,-,-,-,+,+,+,+,+,+,+,+,+,+,+,+,+,+,+,+,+,+,+,+,-,-,-,-,positive
-,-,-,-,-,-,-,-,+,+,+,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,+,+,-,-,positive
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,+,+,positive
-,-,-,-,-,-,-,-,+,+,+,+,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,+,+,-,-,positive
-,-,-,-,-,-,-,+,-,+,+,-,+,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,+,-,-,positive
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,positive
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,+,positive
-,-,-,-,-,-,-,-,-,-,-,-,-,-,+,+,+,+,+,-,-,+,+,-,+,+,-,+,+,+,+,+,+,-,-,-,-,positive
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,positive
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,positive
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,+,positive
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,+,+,positive
-,+,+,+,+,+,+,-,-,-,-,-,-,+,-,-,-,-,-,+,+,-,-,+,-,-,+,-,-,-,-,-,-,-,-,-,-,positive
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,negative
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,negative
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,+,-,-,-,-,-,+,+,-,-,+,-,-,+,-,+,-,+,-,-,-,-,negative
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,negative
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,negative
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,negative
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,negative
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,+,-,-,-,-,-,+,+,-,-,+,-,-,+,-,+,-,+,-,-,-,-,negative
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,negative
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,+,negative
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,negative
-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,negative
```

Figura 4.4: Exemplo de matriz de negativos e positivos que representa uma teoria.

4.3.2. Pesquisa em grelha

É fundamental identificar quais os parâmetros a fornecer ao código para que este retorne o melhor resultado possível, ou seja, as características que demonstram as relações dos dados possíveis.

Tal como referido anteriormente, num estudo realizado por Ferreira et al. [17], são sugeridos intervalos de valores para vários parâmetros do Aleph. Utilizamos estes mesmos intervalos no nosso trabalho, deste modo, são reduzidos em grande número os valores dos parâmetros.

No entanto, embora haja uma redução aos números de parâmetros a utilizar, ainda existem vários valores que os parâmetros podem tomar, e, para isso, é necessária uma pesquisa em grelha – *grid search* – onde foram criados três ciclos, cada um deles para um intervalo de valores de um determinado parâmetro, de forma a serem obtidos os melhores valores. Esta pesquisa dedica-se a chamar o método *induce* do Aleph fornecendo a este uma combinação de todos os valores que podem ser tomados pelos parâmetros de acordo com o intervalo de valores.

4.3.3. *Cross-validation* aplicado a uma conjunto de dados

Após ser determinado que a pesquisa em grelha estava bem implementada, é necessário proceder à verificação de quais os parâmetros a escolher. Para isso, foi decidido que na tabela onde estão presentes os dados sobre a informação de ocorrência de tumor, ou não, deve ser aplicada validação cruzada. Neste caso, a validação cruzada foi posta em prática com a divisão do *dataset* em cinco *folds,* (K=5), sendo que, em cada um deles, estão presentes dois conjuntos de dados, um para treino e outro para teste. É importante referir que o método de validação cruzada utilizado foi o *StratifiedKFold*, uma variação do *Kfold* onde em cada *fold* criado é preservada a percentagem de exemplos em cada um deles.

Sendo que são criados cinco conjuntos de treino e teste para cada *fold*, torna-se necessária a formação de cinco diretórios, os cinco *folds*, onde estes conjuntos serão guardados.

Após executado o referido, são então criadas novas tabelas, para cada *fold*, que contém os exemplos extraídos, respetivamente, da validação cruzada estratificada.

4.3.4. *True Positive Rate* e *False Positive Rate*

Os *folds* estão criados, cada um deles inclui duas tabelas, com exemplos retirados da validação cruzada, uma com exemplos de treino e outra com exemplos de teste.

De seguida foram trabalhados os ciclos pertencentes a cada *fold*, onde cada um deles executa uma chamada ao método *induce* do Aleph, neste caso, com a utilização, apenas, dos exemplos presentes na tabela de treino do determinado *fold*. Ao serem obtidas tanto a teoria, como as *features*, foram calculados, através da matriz, a soma de todos os positivos e negativos de uma determinada coluna, o que corresponde a uma determinada regra, e os valores de *True Positive* (TP)*, False Positive* (FP)*, True Negative* (TN) e *False Negative* (FN)*,* isto para, a partir destes, serem calculados dois rácios, *True Positive Rate* (TPR) e *False Positive Rate* (FPR) de cada coluna*.*

Os dois valores de rácio são calculados da seguinte forma:

- TPR = TP / P
- FPR = FP / N

Sendo assim obtemos estes dois valores para cada coluna e, portanto, é fundamental guardar estes dois valores num vetor de tuplos. Assim, para cada *fold* são criadas tantas teorias como o número de ciclos a executar e, para cada teoria, é criado um vetor de tuplos com os rácios TPR e FPR correspondentes.

Por fim, depois de percorrer os três ciclos, com o vetor completo com os rácios de cada teoria, foi criado um gráfico de pontos, com a utilização de ferramentas presentes no *Python*, onde contém os valores presentes no vetor de tuplos, assim, a visualização do melhor tuplo, aquele que contém os melhores valores, o que por conseguinte, contém os melhores valores de parâmetros a utilizar, torna-se mais fácil. Para se saber quais os melhores valores de tuplo, basta visualizar no gráfico o ponto cujo TPR é mais alto e o FPR é mais baixo.

Desta forma é, de facto, mais fácil visualizar o melhor ponto, porém, quando estão presentes vários *folds* onde, em cada um deles, são executados três ciclos, cada um deles com um intervalo de valores, surge um problema de visualização pois a quantidade de gráficos gerados é bastante extensa.

Nas figuras 4.5 e 4.6, visualizam-se os gráficos onde, em 4.5 verifica-se o gráfico onde foi utilizado apenas um fold e em 4.6 verificam-se vários folds sobrepostos sendo assim mais difícil a visualização.

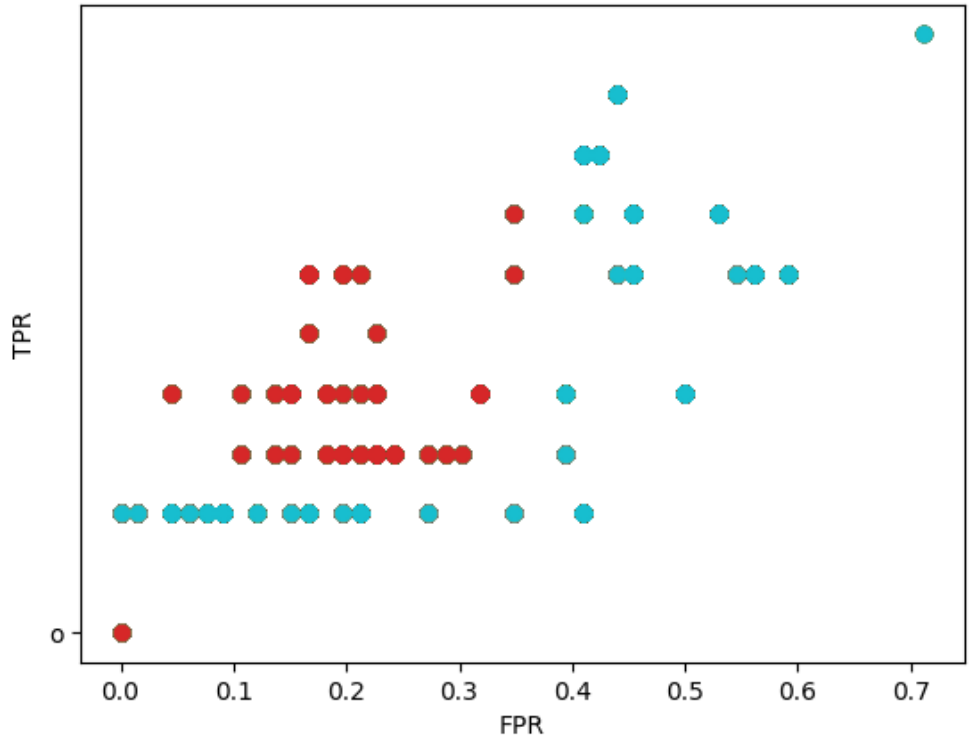


Figura 4.5: Visualização, em gráfico, dos pontos que representam os tuplos – TPR e FPR -, apenas com duas cores, ou seja, dois conjuntos de parâmetros diferentes.

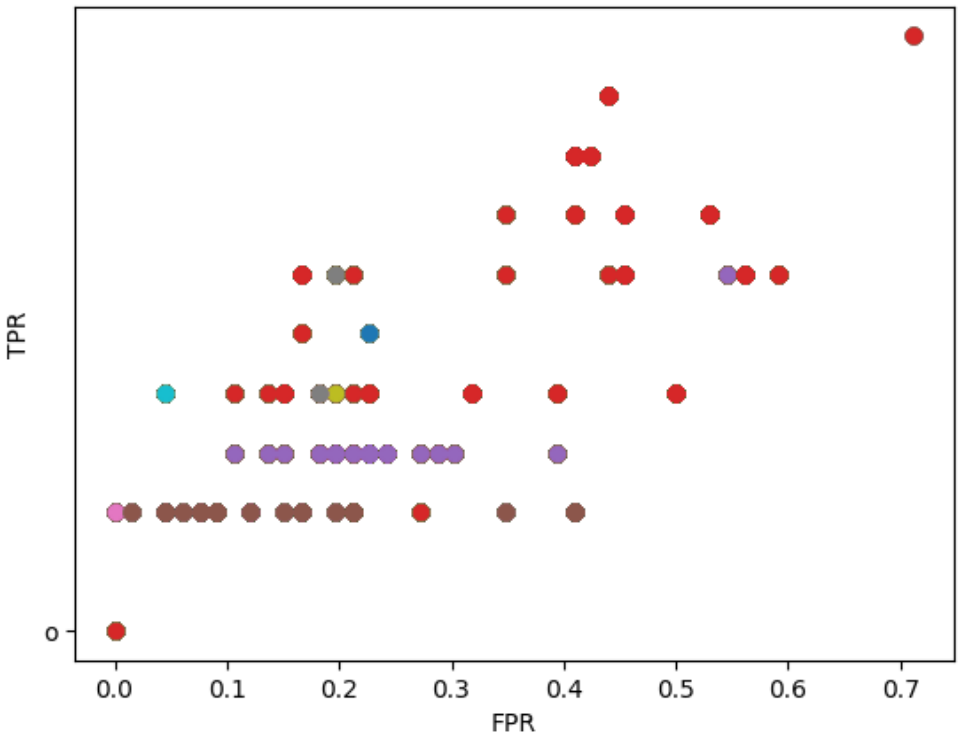


Figura 4.6: Visualização, em gráfico, dos pontos que representam os tuplos – TPR e FPR , onde já se encontram vários conjuntos de parâmetros e alguns pontos se sobrepõem. É mais difícil a visualização do melhor ponto.

4.3.5. Aquisição da melhor regra

Como está referido anteriormente, através dos gráficos criados e com uma grande quantidade de teorias geradas pelo programa, a análise destes gráficos para obter o melhor tuplo acaba por ficar bastante complicada.

Em vez de serem utilizados métodos de visualização, visto que a quantidade de informação é extensiva, foi criada uma nova variável onde fica guardado o melhor tuplo. Para ser obtido o melhor tuplo adicionam-se ao programa condições para verificar qual o tuplo que tem um valor de TPR superior ao valor de FPR, caso isto aconteça, é usada outra condição para observar se o valor de TPR é superior a uma variável iniciada em zero e, se assim se verifique, então esta variável passa a ser igual ao valor de TPR e na nova variável é guardado o tuplo que se está a utilizar neste momento.

```
172                max1 = 0
173                for tupl in array_TPR_FPR:
174                    if tupl[1] > tupl[0]:
175                        if tupl[1] > max1:
176                            max1 = tupl[1]
```

Bloco de código 4.7: Aquisição do melhor valor de TPR onde tupl[1] = TPR e tupl[0] = FPR.

Após ser conseguido o melhor valor de TPR para a teoria que advém dos parâmetros a utilizar no momento, retira-se o índice a que corresponde o valor de TPR no vetor de tuplos pois este irá corresponder à regra presente na teoria. Assim, para uma teoria gerada é possível verificar na figura 4.7 o índice do melhor tuplo:

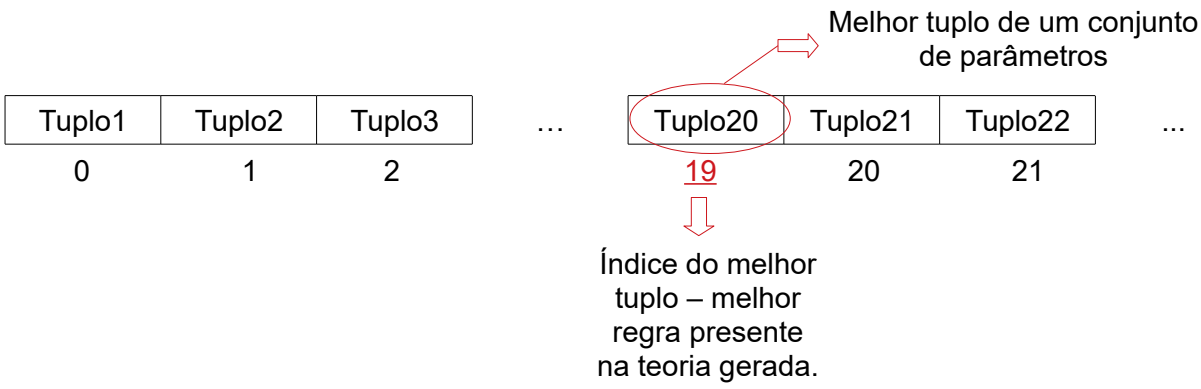


Figura 4.7: Visualização do melhor tuplo no vetor de tuplos de uma teoria gerada e o seu índice correspondente.

Para cada teoria gerada pelo programa, obtém-se a regra que corresponde ao melhor tuplo, no entanto, aquilo que é necessário retirar são os melhores parâmetros a fornecer ao Aleph para que o programa dê origem a nova informação obtida através dos dados. Para isso, é necessário retirar do programa o tuplo que corresponde ao melhor ponto do gráfico, no conjunto de todos os melhores tuplos de cada teoria. À medida que foram retirados os melhores tuplos de cada teoria, foram guardados num vetor e, também, os seus índices correspondentes, assim, para cada melhor tuplo de cada teoria tem-se a regra correspondente.

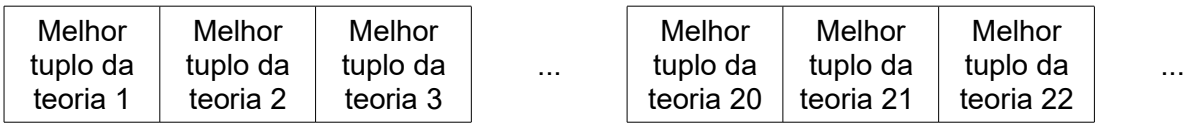


Figura 4.8: Representação do vetor com os melhores tuplos de cada teoria.

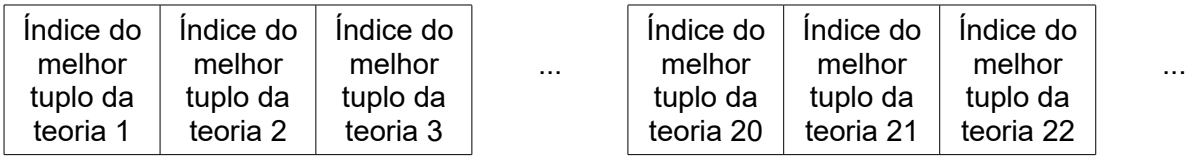


Figura 4.9: Representação do vetor com os índices dos melhores tuplos de cada teoria – correspondem à melhor regra de cada teoria.

Neste conjunto de melhores tuplos, é importante retirar o melhor dos melhores e, para isso, verifica-se qual deles contém um intervalo maior entre os valores de TPR e FPR, pois o que contém este intervalo tem um valor de TPR alto e um valor de FPR baixo, exatamente aquilo procurado. Retirado o melhor tuplo do vetor de melhores tuplos, também se retira o seu índice neste vetor e, para ser retirada a melhor regra, utiliza-se o vetor de índices anteriormente referido, onde o valor da posição a utilizar é o índice do melhor tuplo agora retirado, assim, a melhor regra do conjunto de teorias é verificada.

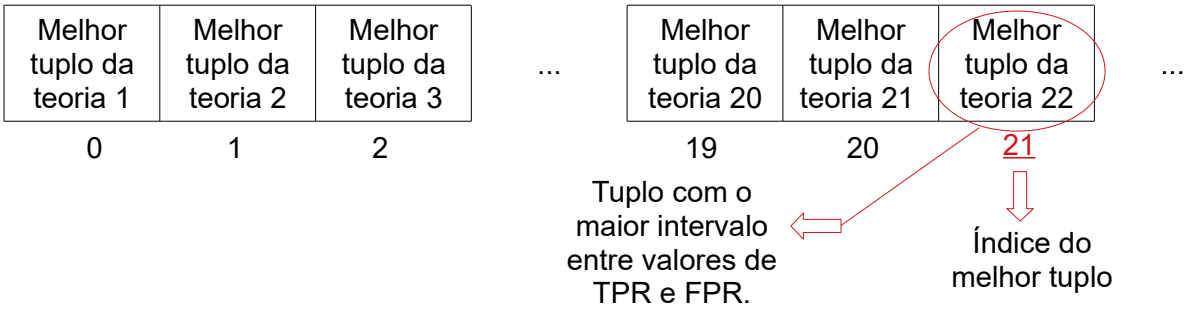


Figura 4.10: Representação do vetor com os melhores tuplos e o melhor de entre estes.

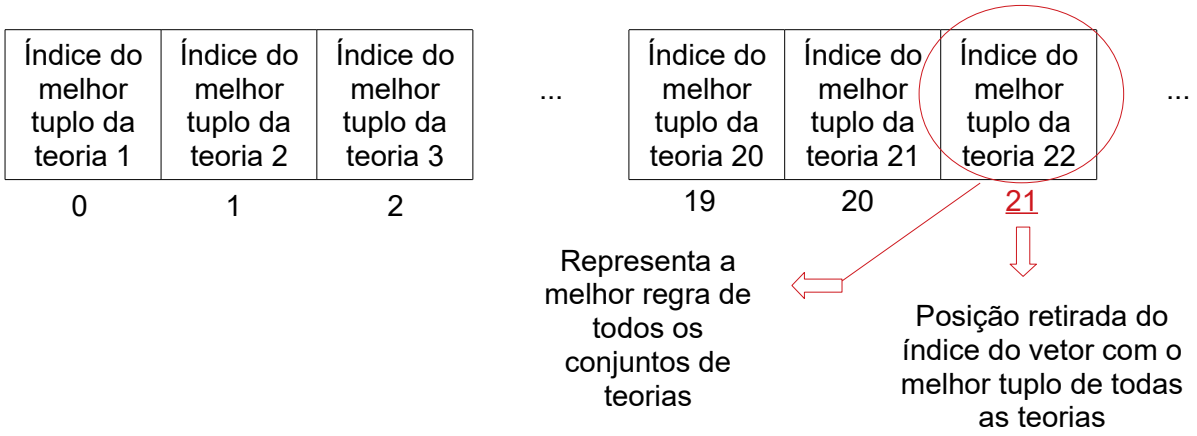


Figura 4.11: Representação do vetor com os índices dos melhores tuplos de cada teoria – correspondem à melhor regra de cada teoria.

Para serem retirados os parâmetros que correspondem à regra verificada agora, bastou criar um vetor com todos os parâmetros das teorias geradas. A partir do momento em que se obtém o índice do melhor tuplo e consequentemente da melhor regra, basta percorrer o vetor de parâmetros e procurar pela posição referente a este índice, ao retornar os valores presentes nesta posição, retiram-se os parâmetros correspondentes à melhor regra de todas as teorias geradas, assim, identificam-se aqueles que se devem utilizar na restante dissertação.

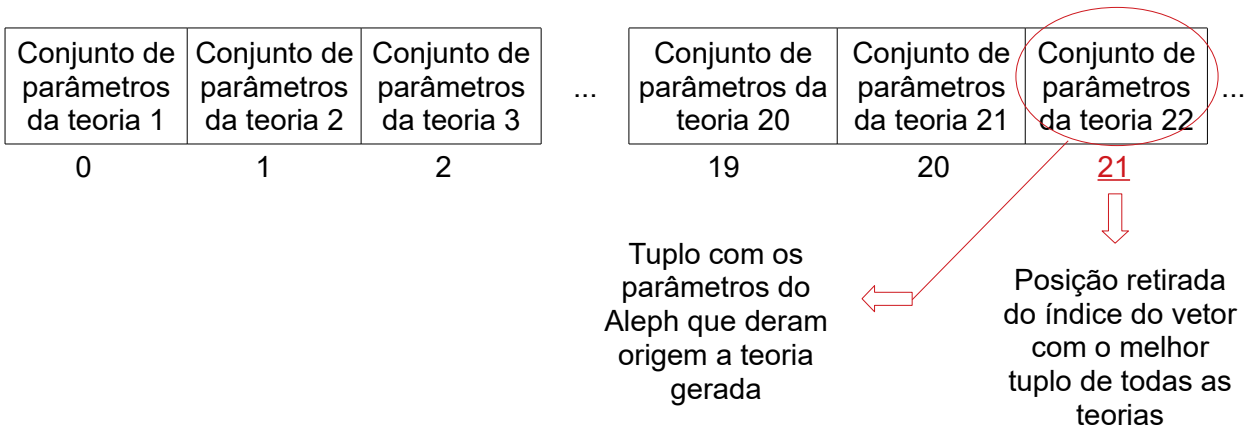


Figura 4.12: Representação do vetor com os parâmetros de cada teoria.

4.3.6. Conjunto de teste

Obteve-se o melhor tuplo de todas as teorias geradas, consequentemente, a melhor regra e os parâmetros ideiais a utilizar no restante trabalho.

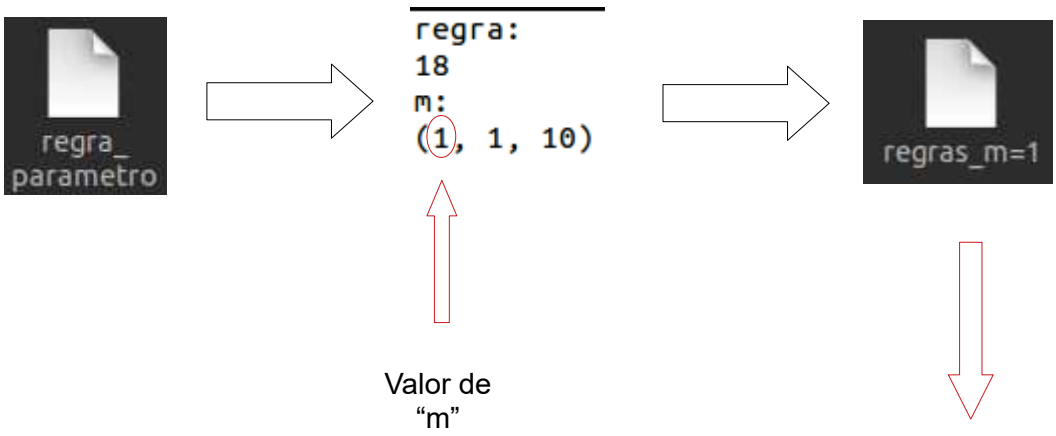
Para cada *fold* foram retiradas várias teorias a partir do Aleph e, dessas teorias, foi retirado o melhor tuplo de valores TPR e FPR. Relembrando que foram criados cinco *folds*, pois foi feita uma *stratified cross-validation* com cinco divisões, para cada um destes *folds* são verificados cinco tuplos, retirados através das teorias geradas em cada um deles. Destes cinco tuplos podemos retirar as regras a que correspondem e os parâmetros que as geram.

Para cada *fold* foi obtida a melhor regra de todas as teorias geradas pelo Aleph, assim, é importante verificar se o poder preditivo deste programa é, de facto, forte. Para tal, foi criado um programa onde, para cada *fold,* retira-se o número da melhor regra e os parâmetros que a geram, que são dados pelo Aleph e foram guardados num ficheiro de texto.

Visto que foram utilizados apenas os conjuntos de treino, é a altura de se utilizarem os conjuntos de teste. Para isso, no programa são criados dois vetores onde são colocados os valores presentes no conjunto de teste ou seja, os exemplos que devem ser verificados e os valores correspondentes.

As regras de cada teoria foram guardadas em ficheiros onde cada um corresponde a uma teoria gerada por um conjunto de parâmetros.

Através do ficheiro onde foi guardado, é verificado qual o melhor conjunto de parâmetros e obtém-se o valor de “m”. O programa criado agora, procura no conjunto de ficheiros, com as regras, qual aquele que contém o nome com o valor do “m” e, dentro deste ficheiro, procura a regra que contém o valor do número da melhor regra, também retirado do ficheiro onde se encontram o número da regra e os seus parâmetros.




```
[features from good clauses]
[pos cover = 2 neg cover = 9] [m estimate] [0.18125]
feature(1,(target_y(A):-fold1_treino2_has_proc_pathtypeNewID(A,B),proc_pathtypeNewID_PathDx(B,'Lobular carcinoma in situ (LCIS)'))).
[pos cover = 2 neg cover = 9] [m estimate] [0.18125]
feature(2,(target_y(A):-fold1_treino2_has_proc_pathtypeNewID(A,B),proc_pathtypeNewID_PathDx(B,'Lobular carcinoma in situ
(LCIS)'),proc_pathtypeNewID_PathDxAbbr(B,'LCIS'))).
[pos cover = 2 neg cover = 9] [m estimate] [0.18125]
feature(3,(target_y(A):-fold1_treino2_has_proc_pathtypeNewID(A,B),proc_pathtypeNewID_PathDx(B,'Lobular carcinoma in situ
(LCIS)'),proc_pathtypeNewID_PathCategory(B,'high risk'))).
```

Figura 4.13: Ficheiro com o valor da melhor regra e quais os parâmetros que a desenvolveram – regra_parametros, a partir do qual se retira o valor de “m”, retirado do primeiro valor apresentado dentro do parêntesis, para ser aberto o ficheiro com o nome do valor de “m” - regras_m=1, onde é possível obter a melhor regra.

Cada regra é constituida pelo seu número, correspondente à coluna da matriz gerada pelo Aleph, e pelas tabelas às quais foram retiradas as informações que deram a sua origem. Portanto, assim que o programa encontra a regra no ficheiro já referido, agora procura pela primeira tabela que surge na regra, quando encontrada é guardado o seu nome numa variável. Logo a seguir ao nome da tabela, verificam-se a coluna de onde foi retirada informação e qual o valor que esta deve ter, ambos guardados em variáveis. É importante referir que em cada regra podem ser encontradas várias tabelas ou até mesmo, a mesma tabela e várias colunas com diferentes valores.

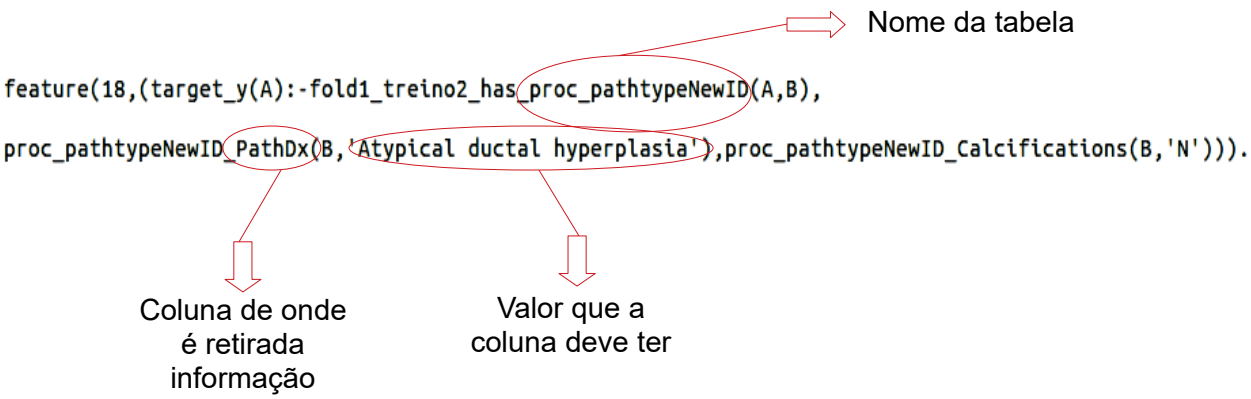


Figura 4.14: Representação da regra escolhida e identificação dos valores contidos nesta.

O programa procura pela tabela, guardada na variável, no diretório onde estas estão guardadas, e, após abrir esta, verifica a coluna correspondente e os valores presentes nesta coluna para os exemplos da tabela de teste. Por fim, basta verificar se o valor contido na tabela para o exemplo presente na tabela de teste é igual àquele presente na regra. Se o valor for igual, então caso o valor de um determinado exemplo for positivo na tabela de teste adicionam-se TP's, pelo contrário, se for negativo, então adicionam-se FP's. No entanto, se o valor da coluna, para determinado exemplo, não for igual àquele presente na regra e o exemplo é negativo então adicionam-se TN's, caso contrário, o valor da coluna é igual àquele presente na regra e o exemplo é negativo então adicionam-se FN's.

Tabela 4.3: Representação da tabela, coluna e valor referentes à regra apresentada em Figura 4.8.

Identificador da tabela

Coluna visualizada na regra

DePathTypeID	DeBiopsyID	PathPriority	PathDx	PathDxAbbr
integer	integer	integer	varchar	varchar
primary key	foreign key [proc_new.DeBiopsyID]			
1	1	1	Unclassifiable	UNC
2	2	1	Benign breast tissue	Normal
3	3	1	Apocrine metaplasia	FC
4	4	1	Post-op/post-biopsy change	POC
5	5	1	Lobular carcinoma in situ (LCIS)	LCIS
6	6	1	Benign breast tissue	Normal
7	7	1	Ductal hyperplasia	Normal
8	8	1	Benign breast tissue	Normal
9	9	1	Atypical ductal hyperplasia	ADH

Valor que o exemplo deve tomar de acordo com a regra.

Para terminar, o programa retira os valores que foram adicionados aos valores de TP, FP, TN e FN e guarda-os num ficheiro para cada *fold.* Desta forma é possível retirar uma *confusion matrix*, de onde podem ser retirados valores de *accuracy, precision* e *recall,* valores estes que nos mostram se o RDM juntamente com o Aleph obtém as regras com um forte poder preditivo.

Tabela 4.4: Representação da confusion matrix.

	Positivos	Negativos	
Positivos	TP = 8	FN = 19	Recall = 0.3
Negativos	FP = 28	TN = 105	
	Precision = 0.22		Accuracy = 0.71

4.3.7. Introdução aos conjuntos de dados temporais

Um dos pontos mais importantes desta dissertação são os dados temporais. Após retirados os melhores parâmetros a utilizar no Aleph e o poder preditivo do RDM juntamente com o Aleph, é a altura de serem aplicados conjuntos de dados temporais para ser constatado se de facto, através destes dados, podem ser verificadas alterações nas observações que são retiradas dos dados, ou seja, se a aplicação de dados temporais afeta a informação que se pode obter de determinado paciente.

Devido à escassez de conjuntos de dados temporais relacionados com o cancro da mama e visto que o objetivo é perceber se estes dados têm influência, podem ser utilizados dados de outro ambiente, desde que possuam informação temporal. Assim, para esta dissertação foram utilizados dados relacionados com um jogo, designadamente um vídeo jogo de futebol, FIFA *European Soccer*. Nestes conjuntos de dados, existe uma variedade de informações relativas aos jogadores presentes no FIFA *European Soccer*, e, nesta variedade estão presentes dados com informação temporal.

Tabela 4.5: Representação de alguns dados com características de jogadores de futebol relacionados com o FIFA *European Soccer*.

id	player_fifa_api_id	player_api_id	date	overall_rating
1	218353	505942	2016-02-18 00:00:00	67
2	218353	505942	2015-11-19 00:00:00	67
3	218353	505942	2015-09-21 00:00:00	62
4	218353	505942	2015-03-20 00:00:00	61
5	218353	505942	2007-02-22 00:00:00	61
6	189615	155782	2016-04-21 00:00:00	74
7	189615	155782	2016-04-07 00:00:00	74
8	189615	155782	2016-01-07 00:00:00	73
9	189615	155782	2015-12-24 00:00:00	73
10	189615	155782	2015-12-17 00:00:00	73

Visto que a quantidade de dados é muito extensa, foram aplicados alguns ajustes a estes, pois o objetivo é perceber se a informação temporal tem consequências. Os conjuntos de dados foram reduzidos para um determinado ano, 2016, a um subconjunto de jogadores, guarda-redes, e posteriormente, foram aplicados alguns métodos, tais como aqueles aplicados aos conjuntos de dados utilizados no início desta dissertação, como por exemplo, foi criada uma tabela com todos os exemplos a serem utilizados e os seus valores de uma determinada característica, neste caso, *potential.* Essa característica foi discretizada em dois valores, bom e mau, B e M respetivamente, sendo que o B será para valores acima de 75, visto que os valores

para esta caracteristica apresentam-se entre 50 e 100, e os restantes para o valor M. Tal como nos primeiros dados, as chaves das tabelas foram alteradas ou criadas para que o RDM consiga estabelecer relações entre todas as tabelas.

Tabela 4.6: Representação da nova tabela criada com a variável *potential* de cada jogador discretizada em dois valores, B e M.

id	potential
integer	varchar
primary key	
1	M
2	M
3	B
4	M
5	M
6	M
7	M
8	M
9	B
10	B

Na classe “id” presente na tabela nova com a variável *potential* de cada jogador discretizada em dois valores, B e M, que representa um identificador único para cada jogador, estão presentes 4262 valores, embora não seja possível visualizar na tabela 4.6 apresentada. Para a classe “*potential*” estão presentes 2525 valores “M” e 1737 valores “B”. O que nos indica que existem mais jogadores onde a classe *potential* é inferior ao valor 75.

Após uma preparação aos dados é agora necessário aplicar o primeiro programa criado anteriormente, este tem como função o método *prepare* do programa Aleph para que sejam gerados os ficheiros ‘.f’, ‘.n’ e ‘.b’ o qual contém o *background knowledge*. É a partir do ficheiro de *background knowledge* que irão ser feitas alterações tais como foram feitas anteriormente para visualização dos efeitos da introdução de dados temporais após execução do segundo programa que tem como função o método *induce* do programa Aleph.

Gerados os três ficheiros anteriormente referidos, é necessária uma introdução de nova informação ao ficheiro que contém *background knowledge.* Esta nova informação irá permitir ao programa Aleph gerar regras com informação temporal.

```
before(Player_Attributes1,Player_Attributes2) :-
      player_attributes_has_player(Player_Attributes1,Player),
      player_attributes_has_player(Player_Attributes2,Player),
      player_attributes_date(Player_Attributes1,Date1),
      player_attributes_date(Player_Attributes2,Date2), Date1 @< Date2.
```

Figura 4.15: Informação adicionada ao ficheiro que contém *background knowledge.*

Na figura 4.15, verifica-se informação relacionada com o mesmo jogador, *Player*, do qual se pretende informação da tabela *Player_Attributes*, informação essa diferente para dois tempos distintos, um será *Date1*, o outro *Date2*, sendo que *Date1* é anterior a *Date2*.

As tabelas estão relacionadas e os ficheiros '.f', '.n' e '.b' estão criados. O ficheiro '.b' que contém o *background knowledge* já foi alterado de forma a conter informação temporal, assim, resta agora correr o programa Aleph com a utilização deste *background knowledge* para serem obtidas regras com informação temporal sobre os dados.

Capítulo 5

Resultados e Análise

5.1 Resultados – Cancro da mama

Como foi visto anteriormente, os dados relativos ao cancro da mama permitiram a descoberta de valores para os parâmetros a utilizar no Aleph, além disso, através da *confusion matrix* é possível verificar que o modelo a ser utilizado é apropriado visto que obtemos um valor de *accuracy* de 0.71.

Antes de serem apresentados resultados com informação temporal referentes ao conjunto de dados do FIFA *European Soccer*, é importante referir as regras obtidas através dos conjuntos de dados relativos ao cancro da mama, são apresentados dois exemplos que podem ser observados nas figuras 5.1 e 5.2. Na Fig. 5.1 são observadas duas regras que apresentam características presentes em todas as regras obtidas ao executar o programa sem a introdução de *background knolwedge.*

```
[pos cover = 4 neg cover = 9] [m estimate] [0.298214285714286]
feature(2683,(target_y(A):-proc_biopsyAbnNewID(A,B),proc_abnormalityNewID_MassDensity(B,den_high),proc_abnormalityNewID_SkLes(B,not_present))).
[pos cover = 4 neg cover = 8] [m estimate] [0.321153846153846]
feature(2684,(target_y(A):-proc_biopsyAbnNewID(A,B),proc_abnormalityNewID_MassDensity(B,den_high),proc_abnormalityNewID_AxAdnp(B,not_present))).
```

Figura 5.1: Representação de uma regra resultante dos conjuntos de dados do cancro da mama sem a introdução de *background knowledge.*

Em nenhuma das regras é possível retirar nova informação presente nos dados.

Assim, como o programa foi dividido de forma a ser possível a introdução de *background knolwedge* por parte do utilizador, foi introduzido conhecimento ao ficheiro a ser utilizado pelo programa. Com a introdução de uma nova característica, *myage,* no ficheiro de *background knolwedge,* que divide os pacientes num intervalo de idades, *maiorque45* para os pacientes com idades acima dos 45 anos e *menorque45,* para os pacientes com idades abaixo dos 45 anos, de forma a gerar novas regras onde esta característica tenha influência, pois a partir de agora o programa faz uma busca tendo em conta pacientes acima dos 45 anos e

abaixo dos 45 anos e deixa de utilizar a idade específica de cada paciente. Executou-se o programa novamente e foram obtidas regras semelhantes àquela apresentada na Fig. 5.2.

```
[pos cover = 5 neg cover = 0] [m estimate] [0.8625]
feature(2812,(target_y(A):-proc_upgradeNewID_NeedleType(A,'vacuum-assisted'),proc_upgradeNewID_NeedleGauge(A,10),proc_upgradeNewID_SamplesNum(A,3))).
```

Figura 5.2: Representação de uma regra resultante dos conjuntos de dados do cancro da mama com a introdução de *background knowledge.*

Na regra apresentada na Fig 5.2, é possível observar que para 5 casos positivos e 0 casos negativos, resulta que em cada paciente que tem cancro da mama foi utilizada um *NeedleType* do tipo *vaccum-assisted,* o *NeedleGauge* tem o valor 10 e o *SamplesNum* tem o valor 3, ou seja, para os pacientes com cancro da mama, estes valores verificam-se e não se verificam para pacientes que não têm cancro da mama.

5.2 Resultados – FIFA *European Soccer*

Foram colocadas todas as ideias desenvolvidas até aqui. Visto que já foram obtidos os melhores parâmetros a serem aplicados ao programa Aleph, e todos os conjuntos de dados já foram devidamente preparados e por sua vez, relacionados de forma a fornecerem a informação necessária, resta executar o método *induce* aos ficheiros para obtenção das regras com a devida informação.

Após execução do segundo programa anteriormente criado, o qual aplica o método *induce* do programa Aleph, com a utilização dos novos dados, com o conjunto de treino gerado pelo primeiro programa, o qual aplica o método *prepare* do Aleph, e com o *background knowledge* atualizado, foram geradas regras com informação temporal.

É importante referir que o conjunto de treino em cada *fold,* criado pela aplicação de *stratified cross-validation* aos dados*,* contém 4308 exemplos de valores da variável *potential* para cada jogador, discretizada em dois valores, B e M, esta variável será a variável *target* para este conjunto de dados.

O primeiro *fold* contém no seu conjunto de treino, 2558 valores de *potential* iguais a M e 1750 valores de *potential* iguais a B. Este conjunto foi utilizado para serem geradas as primeiras regras obtidas através do método *induce* do programa Aleph, algumas com informação temporal, como por exemplo as que se apresentam na figura 5.3.

```
feature(55,(target_b(A):-fold1_treino2_has_player_attributes(A,B),before(B,C),player_attributes_aggression(C,43))).
[pos cover = 6 neg cover = 3] [m estimate] [0.640795228628231]
feature(56,(target_b(A):-fold1_treino2_has_player_attributes(A,B),before(B,C),player_attributes_interceptions(C,34))).
[pos cover = 14 neg cover = 8] [m estimate] [0.626432708099231]
feature(57,(target_b(A):-fold1_treino2_has_player_attributes(A,B),before(B,C),player_attributes_interceptions(C,39))).
```

Figura 5.3: Representação das regras com informação temporal geradas pelo Aleph.

Na Fig. 5.3, podem ser observadas regras com informação temporal. Na figura é possível observar uma regra com o valor 55, onde nessa regra verifica-se que em seis medidas da variável *potential,* foi medido um valor B (Bom) para esta variável e os *players* que contêm este valor tiveram duas medidas B e C, onde B ocorreu antes de C e também contêm um valor de agression de 43, além disso estes valores foram medidos após terem sido medidos valores da variável *potential* para o mesmo jogador, daí *before(*B,C*)* e depois *player_attributes_agression(*C, 43*).*

Na Fig. 5.3 são mostradas estas três regras pois uma delas foi calculada como sendo a melhor regra no conjunto de todas as obtidas, neste caso a *feature* 55. Foi calculada através de uma soma de valores TP, FP, TN e FN que posteriormente deram origem a valores de TPR e FPR tal como foi referido no capítulo 4 onde este cálculo já tinha sido feito para conjuntos de dados relativos ao cancro da mama. Assim, torna-se essencial introduzir esta figura com esta regra e com mais duas para tornar mais fácil a visualização do novo conhecimento obtido.

5.2.1. Casos positivos

Para além das regras apresentadas anteriormente, também foram verificadas regras que apenas cobrem casos positivos. Seguem-se os exemplos do treino representados nas figuras 5.4 e 5.5.

```
[pos cover = 24 neg cover = 0] [m estimate] [0.976318091451292]
feature(1089,(target_b(A):-fold1_treino2_has_player_attributes(A,B),player_attributes_preferred_foot(B,right),player_attributes_long_passing(B,80))).
```

Figura 5.4: Regra com cobertura de 24 casos positivos e 0 negativos.

```
[pos cover = 15 neg cover = 0] [m estimate] [0.962997017892644]
feature(2159,(target_b(A):-fold1_treino2_has_player_attributes(A,B),player_attributes_stamina(B,75),player_attributes_vision(B,65))).
```

Figura 5.5: Regra com cobertura de 15 casos positivos e 0 negativos.

Nas duas regras presentes nas figuras 5.4 e 5.5 é possível observar que nenhuma delas cobrem casos negativos, ou seja, como neste caso verificam-se regras referentes aos dados relativos ao FIFA *European Soccer* e o caso positivo nestes conjuntos de dados é o valor da variável *potential* ser B, ou seja, bom, a partir das regras constata-se que para jogadores com valor B na variável *potential,* estes jogadores apresentam, no caso da Fig. 5.3, o pé preferido é o direito e tem o valor de passes longos de 80. No caso da Fig. 5.4, jogadores com valor de *potential* bom, apresentam valores de stamina de 75 e uma visão com valor de 65.

Para finalizar, é importante referir que em modelos de *machine learning* que não utilizam ILP e múltiplas tabelas, e portanto, não utilizam MRDM, não é possível retirar regras deste tipo pois este conhecimento só é possível através da relação de múltiplas tabelas de dados e de programação lógica indutiva, juntamente com *machine learning*.

Capítulo 6

Conclusão

Este trabalho teve ínicio com uma pesquisa e estudo de artigos que estão direta ou indiretamente relacionada com a dissertação, o que permitiu um conhecimento crucial para o seu desenvolvimento.

O principal objetivo foi modificar o pacote Python-rdm para ser possível flexibilizar o processo de adição de conhecimento extra e desta forma, retirar novo conhecimento presente nos dados. Tal objetivo foi cumprido uma vez que, após esta abordagem ser feita, foram verificadas novas informações geradas pelo programa executado, assim, é feita uma contribuição para o estudo que é desenvolvido relativamente a relações de dados temporais contínuos. Além disso, para além do conhecimento com conteúdo temporal retirado, também foram observadas novas informações retiradas do relacionamento entre os dados em que algumas não continham conteúdo temporal e foram relevantes para o estudo.

A utilização do pacote Python-rdm, para relacionamento de dados, permitiu uma ligação entre os dados de diferentes conjuntos o que, só assim, suporta o restante estudo para obtenção das regras com informação relativa a estes mesmos dados. Para além deste algoritmo, também o programa Aleph foi uma ajuda crucial e o seu uso poderá ser aproveitado para o desenvolvimento de muitos outros estudos, fundamentalmente aqueles onde existem grandes quantidades de dados.

Na última etapa deste trabalho, com todo o conhecimento que foi desenvolvido e aplicado nesta fase foi possível observar nova informação retirada do relacionamento dos conjuntos de dados e, devido a serem utilizados dados que não se enquadravam no cancro da mama mas que continham informação temporal, é assim possível afirmar que o trabalho agora desenvolvido irá ter influência nos estudos futuros de deteção precoce deste cancro, possivelmente de outras doenças e em todos os estudos relativos a aprendizagem relacional.

Terminada a dissertação, o conhecimento obtido através deste irá contribuir firmemente para a inclusão de ideias em trabalhos futuros, por exemplo, poderá haver uma continuidade de trabalho com os dados de pacientes com cancro da mama, perceber de que forma alguns pacientes podem vir a desenvolver este cancro e prosseguir com o seu tratamento numa brevidade tal que permita que esta doença não afete o paciente.

Visto que cada vez mais é percetível que através de informação proveniente de dados já conhecidos é possível obter nova informação que pode ser fundamental para a vida humana, existem várias áreas onde o conhecimento aqui obtido pode ser aplicado, sendo uma delas a área da saúde, onde a aprendizagem relacional de dados temporais contínuos permite a cientistas de dados ajudar a melhorar a realização de decisões clínicas sem a qual não seriam possíveis.

Referências

[1] Kohavi Ron, Provost, Foster. Glossary of terms, Machine Learning. Kluwer Academic Publishers, Boston p.271-274, **1998.**

[2] Saso Dzeroski, Nada Lavrac. Relational Data Mining. Springer-Verlag Berlin Heidelberg **2001**.

[3] Shapiro, Ehud Y.. Inductive inference of theories from facts. Yale University, Department of Computer Science, **1981**.

[4] A. Srinivasan. The Aleph Manual. **2007.**

[5] Sašo Džeroski, Jožef Stefan. Multi-relational data mining. ACM SIGKDD Explorations Newsletter, **2003**.

[6] E. Shapiro.. Algorithmic Program Debugging. MIT Press, Cambridge, **1983**.

[7] Neelamadhab Padhy, Rasmita Panigrahi. Multi Relational Data Mining Approaches: A Data Mining Technique. IJCA Journal, vol. 57, **2012.**

[8] S. Džeroski. Relational Data Mining Applications: An Overview. Relational Data Mining, **2001**.

[9] Mariza Ferro. Aquisição de conhecimento de conjuntos de exemplos no formato atributo valor utilizando aprendizado de máquina relacional. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, **2004.**

[10] Saso Dzeroski, Nada Lavrac. Inductive Logic Programming, Techniques and Applications. Prentice Hall, **1994.**

[11] Van Laer W, de Raedt L, Dzeroski S.. On multi-class problems and discretization in inductive logic programming. International Symposium on Methodologies for Intelligent Systems, Vol. 1325, **1997**.

[12] G. Carrault, M.O. Cordier, R. Quiniou, F. Wang.. Temporal abstraction and inductive logic programming for arrhythmia recognition from electrocardiograms. Artificial Intelligence in Medicine, Vol. 28, **2001**.

[13] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. 14Th international joint conference on Artificial intelligence, **1995**.

[14] Nikolaus Kriegeskorte. Crossvalidation in Brain Imaging Analysis. Medical Research Council, Cognition and Brain Sciences Unit, **2015**.

[15] Chin-Wei Hsu, Chih-Chung Chang and Chih-Jen Lin, A practical guide to support vector classification. Department of Computer Science National Taiwan University, **2010.**

[16] James Bergstra, Yoshua Bengio. Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research, **2012**.

[17] Pedro Ferreira, Inês Dutra, Rogerio Salvini, Elizabeth Burnside. Interpretable Models to Predict Breast Cancer. IEEE International Conference on Bioinformatics and Biomedicine, **2016.**

[18] P. Ferreira, N. A. Fonseca, I. de Castro Dutra, R. W. Woods, E. S. Burnside. Predicting malignancy from mammography findings and image-guided core biopsies. Int J Data Min BioInform., **2015**.

[19] Ryan TJ, Antman EM, Brooks NH, et al.. ACC/AHA guidelines for the management of patients with acute myocardial infarction. Journal of the American College of Cardiology, Vol. 34, **1999**.

[20] N. Lavrac and S. Dzeroski. Inductive Logic Programming: Techniques and Applications. Ellis Horwood, New York, **1994.**

[21] Sašo Džeroski. Multi-relational data mining. ACM SIGKDD Explorations Newsletter. **2003**.

[22] Dousson C, Gaborit P, Ghallab M, Situation recognition: representation and algorithms. Proceedings of IJCAI-93, Chambéry, France: Morgan Kaufman, **1993**.

[23] J. F. Allen. Towards a general theory of action and time. Artificial Intelligence, Vol. 23, **1984**.

[24] M. C. Nicoletti, F. O. S. S. Lisboa, E. R. Hruschka Jr., O. L. de Oliveira. Representation and Automatic Learning of Temporal Relations Between Time Periods with Uncertain Boundaries. International Conference on Intelligent Systems Design and Applications, **2012.**

[25] J. F. Allen. Maintaining knowledge about temporal intervals. Communications of the ACM, Vol. 26, **1983**.

[26] S. Yadav and S. Shukla, Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification. IEEE 6th International Conference on Advanced Computing (IACC), **2016**.

[27] Nacional Cancer Institute, Breast Cancer Treatment (PDQ®), **2014**.

[28] Saunders C, Jassal S. Breast cancer: The Facts. Oxford: Oxford University Press (1.ed.), **2015**.

[29] R. A. Castellino. Computer aided detection (CAD): an overview. Cancer imaging : the official publication of the International Cancer Imaging Society vol. 5, **2005**.

[30] Zheng, B.. Assessing performances of Computer-Aided Diagnosis of breast cancer. Research outreach, **2019.**

[31] Jens Lehmann, Pascal Hitzler. Concept learning in description logics using refinement operators. Machine Learning Journal, **2010.**

[32] Kaggle, European Soccer. https://www.kaggle.com/hugomathien/soccer. Acedido em fevereiro de **2020.**

[33] GitHub, Python-RDM. https://github.com/xflows/rdm. Acedido em fevereiro de **2020**.

[34] Muggleton S. Inductive logic programming. New Generation Computing, **1991**.

Apêndice A

Neste apêndice é apresentado como obter o pacote Python-rdm e como se procede à sua instalação.

Antes de obter o pacote Python-rdm é necessário garantir que a versão do Python instalada na máquina suporta o pacote. A versão utilizada para a realização desta dissertação é a Python 2.7.17.

Para a realização desta dissertação o pacote Python-rdm foi obtido através da execução na linha de comandos do seguinte:

pip install https://github.com/xflows/rdm/archive/master.zip

Para além da instalação do pacote Python-rdm também é preferível instalar o Yap Prolog pois a sua utilização pode ser útil. Esta instalação é feita através da linha de comandos com a execução do seguinte:

apt install yap

Por fim, para a execução de qualquer programa Python com extensão .py, como por exemplo, a execução do programa que é dado como exemplo pelo pacote Python-rdm, trains.py, devem ser executados na linha de comandos, o Python juntamente como o nome do programa e a sua extensão, por exemplo:

Python trains.py

também é fundamental ter no mesmo diretório, onde é executado o programa, as tabelas em formato csv com os dados a serem utilizados pelo programa.