Universidade do Minho
Escola de Engenharia

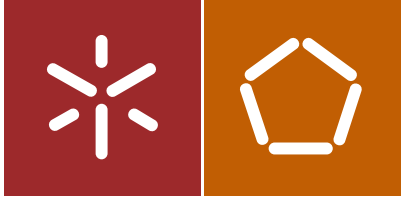
Rui Pedro Gomes de Castro

“Canibalização” de produtos – um estudo

janeiro de 2016

Rui Pedro Gomes de Castro “Canibalização” de produtos – um estudo

UMinho | 2016



Universidade do Minho
Escola de Engenharia

Rui Pedro Gomes de Castro

“Canibalização” de produtos – um estudo

Tese de Mestrado
Mestrado em Engenharia de Sistemas

Trabalho efetuado sob a orientação do
Professor Doutor Orlando Manuel de Oliveira Belo

janeiro de 2016

DECLARAÇÃO

Nome: Rui Pedro Gomes de Castro

Endereço eletrónico: ruipcastro@gmail.com Telefone: 918342564

Bilhete de Identidade/Cartão do Cidadão: 13548182

Título da dissertação: “Canibalização” de produtos – um estudo

Orientador:

Professor Doutor Orlando Manuel de Oliveira Belo

Ano de conclusão: 2016

Mestrado em Engenharia de Sistemas

Universidade do Minho, _____/_____/_____

Assinatura:

AGRADECIMENTOS

Agradeço a todas as pessoas que me apoiaram na execução deste projeto e que o tornaram possível, especialmente a:

À minha mãe, ao meu pai e à minha irmã pelo apoio constante e incansável que sempre me proporcionaram.

Ao meu orientador, o Professor Doutor Orlando Belo, por toda a colaboração, disponibilidade prestada e pela oportunidade de ter trabalhado sobre a sua tutoria.

À Profimetrics, pela oportunidade de realizar este projeto, pelos dados disponibilizados e pela maneira como fui acarinhado e recebido por todos. Em especial à Engenheira Sofia Amorim e ao Engenheiro Marco Gomes pelo apoio prestado e por se terem mostrado sempre disponíveis.

A todos os meus amigos que de algum modo sempre me apoiaram e tornaram este projeto uma realidade.

RESUMO

“Canibalização” de produtos – um estudo

A área de descobrimento de conhecimento a partir de dados de uma base de dados é uma área que nos últimos anos tem sofrido uma grande evolução e um aumento constante. É através de técnicas como a mineração de dados que conseguimos extrair conhecimento que de outra forma não seria possível devido à escala e complexidade dos sistemas de armazenamento de dados hoje em dia utilizados.

O trabalho realizado nesta dissertação visa analisar e comparar o desempenho de dois algoritmos de mineração de dados, nomeadamente métodos para regras de associação, aplicados ao registo de vendas de produtos de um retalhista brasileiro.

Os resultados obtidos mostraram um grande potencial para este tipo de casos de estudo ligados ao retalho e que mesmo sendo os algoritmos largamente diferentes entre si, estes mostram um grande potencial para serem utilizados mesmo em parceria.

Palavras-Chave: Mineração de dados, Regras de associação, CRISP-DM, Canibalização de produtos

ABSTRACT

“Cannibalization” of products – a study

The field of knowledge discovery from a data base is a field that in the last years has suffered a big evolution and constant growth. Is through an analytic process like data mining that we can extract knowledge that otherwise wouldn’t be possible due the scale and complexity of the data storage systems used now a days.

The work done in this dissertation was focused in analysing and comparing two data mining algorithms, specifically algorithms that use methods of association rules, applied to a data base containing the transaction records of a brazilian retailer.

The results obtained showed a great potential with this type of studies related to retail and that even though the algorithms are largely different from one another, they showed a great potential for them to be used side by side.

KEYWORDS: Data mining, Association Rules, CRISP-DM, Cannibalization of products

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

Lista de Abreviaturas, Siglas e Acrónimos

DW – Data Warehouse

BD – Base de Dados

MBA – Market Basket Analysis

KDD – Knowledge Discovery in Databases

DM – Data Mining

CRISP-DM - Cross Industry Standard Process for Data Mining

Capítulo 1

1. INTRODUÇÃO

1.1 Contextualização e Enquadramento

A área das Tecnologias de Informação é das áreas que, nas últimas décadas, mais evolução tem sofrido. Como sabemos, é uma área com inúmeras aplicações e que tem um impacto muito grande nas mais variadas áreas de trabalho. Hoje em dia, nenhuma empresa ou negócio pode-se dar ao luxo de não prestar atenção a esta área. Por isso, é imperativo que as empesas incorporem estas tecnologias nos seus negócios. Vivemos numa altura em que a informação é um recurso importantíssimo em qualquer ramo de atividade. O armazenamento e a organização dessa mesma informação torna-se, por isso, imperativo. Como Gu e Lin (Gu & Lin 2012) referem:

> *"With the development of information technology, each department of enterprises has generally implemented the management information system, which has improved the operation efficiency of enterprises and strengthened the competitiveness of enterprises to a great extent."*

Foi a partir deste ponto que o conceito de *data warehouse* (DW) foi introduzido no mercado. A criação deste tipo de repositório especializado é vital no armazenamento e na organização de grandes volumes de dados que suportam necessidades específicas de negócio. Com a utilização de um DW temos a facilidade de criar mecanismos especialmente orientados para atividades de suporte à decisão e, assim, ter a possibilidade de descobrir conhecimento contido nessa grande quantidade de dados que,

de uma outra forma qualquer, não seria viável. A este processo podemos designar por *Knowledge Disovery in Databases* (KDD). Um dos passos mais importantes deste processo é a mineração de dados. Por mineração de dados designamos o processo computacional de descobrir padrões em grandes volumes de dados, que à partida não seriam óbvios para um utilizador de uma DW.

A evolução do poder computacional dos sistemas, tanto em termos de *hardware* como de *software*, veio permitir que grandes quantidades de informação pudessem ser armazenadas e posteriormente processadas através de algoritmos bastante complexos e poderosos (Videla-Cavieres & Ríos 2014). As técnicas de mineração de dados mais usadas em retalho, são as regras de associação, que foram introduzidas por Agrawal (Agrawal, Imieliński, et al. 1993). Através delas é possível inferir regras de correlação, padrões e associações entre produtos. Essas regras podem ser posteriormente usadas, a título de exemplo, em áreas como marketing, publicidade ou controlo de inventário.

Uma das áreas de aplicação mais comuns destas técnicas é a *Market Basket Analysis* (MBA), isto é, a análise de compras em sistemas de retalho (Abbas et al. 2013). Este ramo trabalha com bases de dados especialmente povoadas pela informação relativa às diversas transações efetuadas numa certa loja ou conjunto de lojas. Assim, o MBA, é usada com o objetivo de extrair padrões de compras de clientes ou de grupos de clientes, identificando potenciais perfis de comportamento de cada um. Posteriormente, a informação providenciada por este tipo de análise pode ser utilizada com o objetivo de melhorar o serviço prestado pelo próprio retalhista aos seus clientes e, também, melhorar a sua própria rentabilidade.

A técnica de mineração de dados mais usual em MBA são as Regras de Associação. Um dos conceitos utilizados para fazer a previsão de vendas de um dado produto é o *Halo Effect*, isto é, o impacto positivo que as vendas acrescidas de um produto vai ter sobre outro produto. Essa influência também pode ser negativa, o que também podemos designar por canibalismo (*cannibalism*): *"Negative rules consider items that conflict with each other. In other words, negatives rules are used to represent that if product A is purchased, then product b will not be purchased."* (Agrawal et al. 2014). A previsão de vendas torna-se, assim, uma ferramenta importantíssima em sistemas de retalho, uma vez que permite prever padrões de compra de clientes, e assim, entender melhor as suas intenções comerciais. Este conhecimento adicional permite, por exemplo, fazer uma melhor gestão de stocks ou realizar melhores e mais eficazes campanhas promocionais (Fayyad et al. 1996).

1.2 Motivações e Objetivos

Na secção de contextualização e enquadramento deste capítulo são apresentadas as principais áreas de trabalho desta dissertação. Em particular, interessa utilizar técnicas de associação diferentes para prever as vendas de produtos. Com a utilização deste tipo de técnicas, espera-se poder obter regras de associação sobre a influência positiva e negativa que as vendas de um produto têm sobre as vendas de um outro produto, situações que frequentemente são referidas como sendo de "*halo effect*" para associações positivas e "canibalismo" para associações negativas. Pretende-se assim desenvolver um sistema que permita fazer a previsão de vendas num determinado domínio de negócio.

De uma forma mais concreta, pode dizer-se que os objetivos deste trabalho de dissertação são:

1. Determinar quais os algoritmos de mineração que melhor se aplicam a um problema típico de "*halo effect*" e de "canibalismo";
2. Identificar um algoritmo de associação para cada problema, para se utilizar no caso de estudo;
3. Aplicar esses algoritmos sobre os dados de uma fonte de dados;
4. Analisar os resultados obtidos e o desempenho dos algoritmos;
5. Comparar os algoritmos para estabelecer qual o melhor algoritmo a utilizar, numa determinada situação, consoante os dados disponíveis e os objetivos do problema.

1.3 Organização da Dissertação

Para além do presente capítulo esta dissertação está organizada da seguinte maneira:

- Capítulo 2, são apresentados e aprofundados os conceitos de *Data Mining* e a metodologia usada neste projeto, mais concretamente a metodologia CRISP-DM. Além disso é também apresentado o método de mineração das regras de associação, tal como os medidores usados, o processo de mineração e ainda uma análise a alguns algoritmos relevantes na área. Por fim, é analisada a aplicação destes conceitos ao ramo do retalho.

- Capítulo 3, é apresentado o caso de estudo do projeto, são analisados os dados que foram utilizados, a origem dos dados, tal como a forma como os dados estão organizados. Ainda sobre os dados é analisada a seleção que foi feita e a preparação dos mesmos para a aplicação no problema. Por fim são apresentados os algoritmos escolhidos para abordar o caso de estudo tal como as ferramentas onde estes vão ser executados.

- Capítulo 4, são analisados todos os modelos que foram construídos com as suas características e o processo de como foram construídos.

- Capítulo 5, são apresentados os resultados dos modelos e é feita uma análise crítica e comparativa entre os vários modelos.
- Capítulo 6, é feita uma conclusão sobre o trabalho realizado e uma reflexão sobre o trabalho futuro possível.

Capítulo 2

2. REGRAS DE ASSOCIAÇÃO

2.1 Mineração de Dados

Data Mining, ou em português, mineração de dados, é a etapa de extração de conhecimento do *Knowledge Discovery in Databases* ou KDD (Fayyad et al. 1996). DM é o processo computacional de descoberta de padrões em BDs de grande dimensão, ou *big data*, que de outro modo seria impossível de descobrir. Este processo envolve métodos como inteligência artificial, *machine learning*, estatística e sistemas de bases de dados e tem como principal objetivo a extração de conhecimento e não de informação *"Thus, data mining should have been more appropriately named "knowledge mining from data," which is unfortunately somewhat long"* (Han & Kamber 2001) para que depois esse conhecimento possa ser aplicado na área em estudo.

Graças ao facto de que consegue oferecer a todos os tipos de negócios, a possibilidade de transformar informação em conhecimento, o DM é uma ferramenta cada vez mais imprescindível. É uma forma bastante fiável e confiável e oferece bastante suporte nos processos de tomada de decisões. A parte de interpretação dos resultados já não pertence ao processo de *Data Mining*, mas pertence ao processo geral de KDD.

Um processo de DM passa sempre por 3 etapas, o pré-processamento, o processo de *Data Mining* concreto, e por fim, a validação dos resultados.

Pré-Processamento

Antes que quaisquer algoritmos possam ser usados é necessário construir um *data set*. O processo de DM só consegue descobrir padrões caso eles existam no *data set* ainda que estejam por descobrir. Por causa disto é necessário que o *data set* seja suficientemente grande para conter os padrões e ao mesmo tempo não demasiado grande para não tornar o processo demasiado longo. Uma fonte de dados é geralmente um *data mart* ou *data warehouse*. A fase de pré-processamento é essencial para analisar a qualidade dos dados e limpeza de dados, removendo dados que possam conter lixo e dados nulos. É com isto que esperamos aumentar a probabilidade de obtenção de resultados interessantes (Hu 2003; Crone et al. 2006).

Esta fase será abordada com mais detalhe no capítulo 3.

Data Mining

Os objetivos do processo de DM são normalmente de previsão e descrição. Previsão, consiste em usar informação existente da BD de modo a conseguir prever acontecimentos desconhecidos ou futuros, enquanto que a descrição, consiste em encontrar padrões que melhor consigam descrever os dados e que sejam interpretáveis pelo utilizador. Esta distinção nem sempre é restritiva já que há métodos que podem ser tanto de previsão como de descrição, mas são geralmente divididos em 6 métodos:

Anomaly Detection é a identificação de acontecimentos, *items* ou observações que não têm impacto suficiente nos dados para serem um padrão, ou seja, são anomalias de eventos considerando o que seria normal e espectável (Chandola et al. 2009). Este método de encontrar anomalias pode ser utilizado para encontrar fraudes bancárias, problemas médicos, erros num testo, entre outros (Hodge & Austin 2004).

Association rule learning é o método que irei abordar e por isso, será analisado na secção 2.2 deste documento.

Cluster analysis ou *clustering* é o método de agrupar uma quantidade de objetos, de modo a que esses objetos agrupados partilhem certas características que sejam de algum modo diferenciadoras das características de outros grupos, para desta formar se conseguir caracterizar e distinguir melhor os vários grupos (Kanungo et al. 2002). Este método é frequentemente utilizado para fazer análises

estatísticas e pode ser usado em vários ramos como *machine learning*, reconhecimento de padrões, recuperação de informação, bioinformática, entre outros (Dalton et al. 2009).

Classification é o método de identificar a que categorias uma nova observação de dados pertence, baseado num conjunto de categorias previamente criadas (Kesavaraj & Sukumaran 2013). Um bom exemplo deste método é a classificação de um *email* como spam ou não spam, segundo certas frases ou palavras chaves que esse *email* possa conter (Alguliev et al. 2011). Outro exemplo é a criação de um diagnóstico a um paciente, segundo certas características do paciente, como sexo ou idade e certos sintomas que o paciente possa apresentar ou não (Gupta et al. 2011).

Regression é o método estatístico de estimar os relacionamentos existentes entre as variáveis. É usado para descobrir que dependências uma variável pode ter noutra, para desta forma descobrir qual a probabilidade de quando uma variável sofre alterações outra variável também ser alterada, enquanto outras se mantêm inalteradas (Hassan et al. 2013).

Summarization é o método de reduzir um texto de modo a criar um texto sumário do texto inicial, retendo os pontos essenciais (Chandola & Kumar 2005). Este método, cada vez mais usado nos dias de hoje, devido ao grande aumento de informação que qualquer sistema cria, por exemplo, em grandes lojas que tenham muitos registos de satisfação de clientes é necessário que a análise desses registos seja sumarizada (Hu & Liu 2004).

Esta fase será abordada com mais detalhe no capítulo 4.

Análise dos Resultados

Nesta última fase, é necessário analisar os resultados obtidos, depois de aplicar um dos métodos acima referidos. Para isso, verifica-se se os resultados estão de acordo com os objetivos e com o que era esperado obter-se. É também necessário validar os resultados com os dados para se ter a certeza que são suportados pelos dados previamente utilizados.

Caso se verifique que os resultados obtidos não são suportados pelos dados, ou que não cumprem com os requisitos, é necessário reavaliar e alterar as fases de pré-processamento e *data mining*. Depois de realizar novamente todo o processo, com as devidas alterações, caso se obtenha resultados

satisfatórios, o último passo a fazer será interpretar esses resultados e assim transformá-los em conhecimento.

Esta fase será abordada com mais detalhe no capítulo 5.

2.2 Regras de Associação

Regas de associação ou em inglês, *Association Rules,* é um método popular e muito investigado para descobrir relações interessantes entre variáveis numa BD de grande dimensão. O objetivo é assim identificar regras de grande peso na BD usando diferentes medidores de interesse (Piatetsky-Shapiro 1991). Baseado no conceito de regras interessantes, em 1993, Rakesh Agrawal (Agrawal, Imieliński, et al. 1993) introduziu o conceito de regras de associação para descobrir padrões de vendas de produtos utilizando os dados capturados por sistemas ponto de venda, geralmente caixas registadoras, em BDs de grande escala. Por exemplo, a regra $\{Cerveja\} \Rightarrow \{Amendoins\}$ encontrada em vendas de um supermercado, indica que, se um cliente compra Cerveja, é também provável que ele também compre amendoins. Esta informação pode ser usada na tomada de decisões referente, por exemplo, à criação de campanhas de marketing relacionadas com o preço dos produtos, ou quanto à localização dos produtos na loja. Além deste caso, as regras de associação são usadas em muitas outras áreas como *Web mining*, sistemas de deteção de intrusos, métodos de produção contínua na indústria, Bioinformática, etc (Rajak & Gupta 2008).

O conceito de regras de associação ficou popular em 1993 graças ao artigo de Agrawal (Agrawal, Imieliński, et al. 1993), artigo esse que, segundo o *Google Scholar*, já tem mais de 16000 citações à data de Outubro de 2015. Ainda assim, o que hoje é conhecido como Regras de Associação é bastante parecido com o que foi apresentado em 1996 como GUHA (General Unary Hypotheses Automation) (Hájek et al. 1966), já que esse método pode ser comparado aos métodos de regras de associação mais recentes (Hájek et al. 2010).

A definição para o processo de regras de associação, apresentada por Agrawal nesse artigo, pode ser definida da seguinte forma:

Considerando que $I = \{i_1, i_2, \dots, i_n\}$ é um conjunto de atributos binários chamados produtos, D é uma base de dados de transações $D = \{t_1, t_2, \dots, t_m\}$ onde t é uma vetor binário em que $t[k] = 1$ se a transação t incluir o produto i_k e $t[k] = 0$ caso não inclua o produto i_k.
Uma regra de associação é definida pela implicação $X \Rightarrow Y$ em que $X, Y \subseteq I$ e $X \cap Y = \emptyset$. Cada regra é sempre composta por 2 conjuntos de produtos, $X\ e\ Y$, onde X é chamado de antecedente e Y é chamado de consequente.

Para exemplificar este conceito, usa-se o exemplo de uma BD que representa 5 transações de 5 possíveis produtos de um dito supermercado - "tabela 1". A "tabela 2" contém a representação binária das mesmas 5 transações da "tabela 1". Neste exemplo, a lista de produtos é $I = \{Pão, Leite, Fraldas, Cerveja, Ovos, Cola\}$ e para cada transação temos a lista de produtos que ela contém marcados a 1, caso o produto esteja presente na transação, ou 0, caso não esteja presente. Uma regra de associação que podemos tirar desta base de dados é $\{Fraldas\} \Rightarrow \{Cerveja\}$. A regra sugere que existe uma grande relação entre a venda de Fraldas e a venda de Cerveja, já que se observarmos a "tabela 1" e a "tabela 2" podemos verificar que os consumidores que compram Fraldas têm tendência a também comprar Cerveja. É de notar que este exemplo é pequeno de mais para poder ser considerado como válido. Em casos reais, uma regra de associação precisa de ter suporte em várias centenas de transações e que a BD tenha vários milhares ou mesmo milhões de transações.

Tabela 1 - Exemplo de BD com 5 transações e 5 produtos

Transação ID	Produtos
1	{Pão, Leite}
2	{Pão, Fraldas, Cerveja, Ovos}
3	{Leite, Fraldas, Cerveja, Cola}
4	{Pão, Leite, Fraldas, Cerveja}
5	{Pão, Leite, Fraldas, Cola}

Tabela 2 - Representação binária da Tabela 1

Transação ID	Pão	Leite	Fraldas	Cerveja	Ovos	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	0	0	0
5	1	1	1	0	0	1

Este exemplo foi em parte baseado no famoso caso de regras de associação de "cerveja e fraldas". Nesse caso, um estudo sobre a tendência de compras de consumidores de um supermercado, descobriu que consumidores, normalmente jovens e do sexo masculino, que compravam fraldas, tinham tendência a também comprar cerveja. Este caso tornou-se famoso por se mostrar como um processo de mineração de regras de associação pode gerar regras completamente inesperadas. Sobre este caso Daniel J. Power diz:

> *In 1992, Thomas Blischok, manager of a retail consulting group at Teradata, and is staff prepared an analysis of 1.2 million market baskets from about 25 Osco Drug stores. Database queries were developed to identify affinities. The analysis "did discover that between 5:00 and 7:00 p.m. that consumers bought beer and diapers". Osco managers did NOT exploit the beer and diapers relationship by moving the products closer together on the shelves. (Power 2002)*

2.2.1 Medidores de interesse de Regras de Associação

De modo a selecionar as regras mais interessantes de uma lista de possíveis regras, podem ser usadas restrições em vários medidores de interesse (Hahsler 2015). Seja X um produto ou conjunto de produtos ou *itemset*, $X \Rightarrow Y$ uma regra de associação e $D = \{t_1, t_2, \dots, t_m\}$ uma base de dados de transações :

Suporte

$$supp(X) = \frac{|\{t \in D; X \subseteq t\}|}{|D|} = P(D)$$

O valor de suporte é definido no *itemset* e dá-nos a proporção de transações que contêm X na BD D. É usado como medida de importância no *Itemset.* Como é um medidor que usa basicamente uma contagem de transações, é normalmente designado como restrição de frequência. Quando um *itemset*

tem um valor de suporte superior a um suporte mínimo definido, $supp(X) > \sigma$ chamamos a esse *itemset* de frequente ou grande.
Pegando no exemplo anterior, o *itemset* $\{Leite, Fraldas, Cerveja\}$ tem um suporte de $^{2}/_{5} = 0.4$ já que ocorre em duas das cinco transações (40%).

Para as regras, o suporte é definido como o suporte de todos os *itemsets* na regra, isto é, $supp(X \Rightarrow Y) = supp(X \cup Y) = P(X \wedge Y)$. A principal desvantagem do suporte é o problema com *itemsets* raros. Se um *itemset* ocorrer com muito pouca frequência na BD, não será considerado, uma vez que não vai ter um impacto suficientemente grande. Isto pode ser problemático já que esses casos de ocorrência rara poderiam produzir regras potencialmente interessantes. O valor de suporte varia entre 0 e 1.

Confiança

$$conf(X \Rightarrow Y) = \frac{supp(X \Rightarrow Y)}{supp(X)} = \frac{supp(X \cup Y)}{supp(X)} = \frac{P(X \wedge Y)}{P(X)} = P(X|Y)$$

A confiança de uma regra é a proporção de transações que contêm o *itemset* X, mas que também contêm o *itemset* Y. É de notar que as regras $X \Rightarrow Y$ e $Y \Rightarrow X$ podem ter valores diferentes.
As regras de associação têm de satisfazer uma restrição mínima de confiança, $conf(X \Rightarrow Y) > \gamma$.
Pegando outra vez no exemplo anterior, a regra $\{Leite, Fraldas, Cerveja\} \Rightarrow \{Pão\}$ tem uma confiança de $^{0.2}/_{0.4} = 0.5$ na BD. Isto significa que, de todas as transações que contêm $\{Leite, Fraldas, Cerveja\}$ em metade delas também foi comprado $\{Pão\}$.

O problema com o medidor de confiança é o facto de ser sensível à frequência que o *itemset* Y ou consequente aparece na BD. Isto pode levar a que, consequentes com um suporte maior produzam regras com um valor de confiança maior, ainda que na verdade, o antecedente e o consequente dessa regra possam não ter qualquer associação entre eles. O valor de confiança varia entre 0 e 1.

Lift

$$lift(X \Rightarrow Y) = lift(Y \Rightarrow X) = \frac{conf(X \Rightarrow Y)}{supp(Y)} = \frac{conf(Y \Rightarrow X)}{supp(X)} = \frac{P(X \wedge Y)}{P(X)P(Y)}$$

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) \times supp(Y)}$$

Lift, ou como era inicialmente chamado, interesse, mede quantas mais vezes X e Y ocorrem juntos do que era esperado caso X e Y fossem estatisticamente independentes (Brin, Motwani, Ullman, et al. 1997).

Pegando novamente na regra do exemplo anterior, $\{Leite, Fraldas, Cerveja\} \Rightarrow \{Pão\}$ aqui o valor de *lift* será $\frac{0.2}{0.4 \times 0.8} = 0.625$. Uma vez que o valor é menor que 1, podemos concluir que o antecedente e o consequente são correlacionados negativamente. Caso o *lift* fosse igual a 1, iria implicar que a probabilidade dos dois acontecimentos era independente um do outro, logo nenhuma regra de associação pode ser retirada envolvendo os dois acontecimentos. Se o *lift* for maior que 1, então existe uma dependência entre os dois acontecimentos e quanto maior for o valor do *lift*, maior essa dependência. Regras onde o *lift* é alto, podem ser potencialmente usadas para fazer previsões futuras.

Ainda que o *lift* não sofra em casos onde um *itemset* possa ser mais raro, tem a desvantagem de que pode ser suscetível a ruido em BDs mais pequenas. *Itemsets* raros e com baixa probabilidade, mas que por acaso ocorram algumas vezes, podem produzir valores de *lift* muito altos.

Convicção

$$conviction(X \Rightarrow Y) = \frac{1 - supp(Y)}{1 - conf(X \Rightarrow Y)} = \frac{P(X)P(\bar{Y})}{P(X \wedge \bar{Y})}$$

Convicção compara a probabilidade de X acontecer sendo que Y não acontece se X e Y forem dependentes com a real probabilidade de X acontecer sendo que Y não acontece.

Seguindo novamente a regra do exemplo, $\{Leite, Fraldas, Cerveja\} \Rightarrow \{Pão\}$, o valor de convicção será $\frac{1-0.8}{1-0.5} = 0.4$. Como o valor é inferior a 1, tiramos a mesma conclusão já observada no *lift*, isto é, que o antecedente e o consequente são correlacionados negativamente.

Convicção também compara a dependência/independência entre os acontecimentos de uma regra tal como o *lift* mas ao contrário deste é sensível à direção da regra ($lift(X \Rightarrow Y) = lift(Y \Rightarrow X)$). Dizemos que a convicção é boa para valores infinitos e que X e Y são independentes caso a convicção seja igual a 1 (Azevedo & Jorge 2007).

Fator de Certeza

Fator de certeza (CF – *certainty factor*) mede a variação da probabilidade de que Y está numa transação quando consideramos apenas as transações onde Y está presente (Shortliffe & Buchanan 1990). O domínio é $[-1,1]$ sendo que valores negativos representam dependência negativa ou correlação negativa, zero representa independência, e valores positivos representam uma dependência positiva. Para medir o fator de certeza da regra $X \rightarrow Y$ usam-se três formulas dependendo se o valor de confiança é menor, maior ou igual ao suporte de Y.

$$CF(X \rightarrow Y) = \begin{cases} \dfrac{conf(X \rightarrow Y) - supp(Y)}{1 - supp(Y)}, & conf(X \rightarrow Y) > supp(Y) \\ \dfrac{conf(X \rightarrow Y) - supp(Y)}{supp(Y)}, & conf(X \rightarrow Y) < supp(Y) \\ 0, & conf(X \rightarrow Y) = supp(Y) \end{cases}$$

2.2.2 Processo de Mineração de Regras de Associação

Geralmente, regras de associação têm de satisfazer um mínimo de suporte e também por um mínimo de confiança especificados pelo utilizador. Deste modo o processo de mineração de regras de associação é dividido em dois passos:

1. Encontrar *itemset* frequentes, que implica encontrar todos os *itemsets* que satisfaçam o mínimo de suporte definido. Estes *itemsets* são considerados *itemsets* frequentes.
2. Gerar as regras de associação, que implica encontrar todas as regras que satisfaçam o mínimo de confiança definido, a partir dos *itemsets* frequentes encontrados no passo anterior. Estas regras são consideradas como regras fortes ou interessantes.

O segundo passo é o mais fácil de aplicar, e o primeiro passo o que precisa de mais atenção. Encontrar todos os *itemsets* frequentes numa BD é um processo difícil, longo e possivelmente muito

exigente em termos de *hardware*, já que o processo tem de encontrar todos os possíveis *itemsets* que cumprem os requisitos. (Tan et al. 2005).

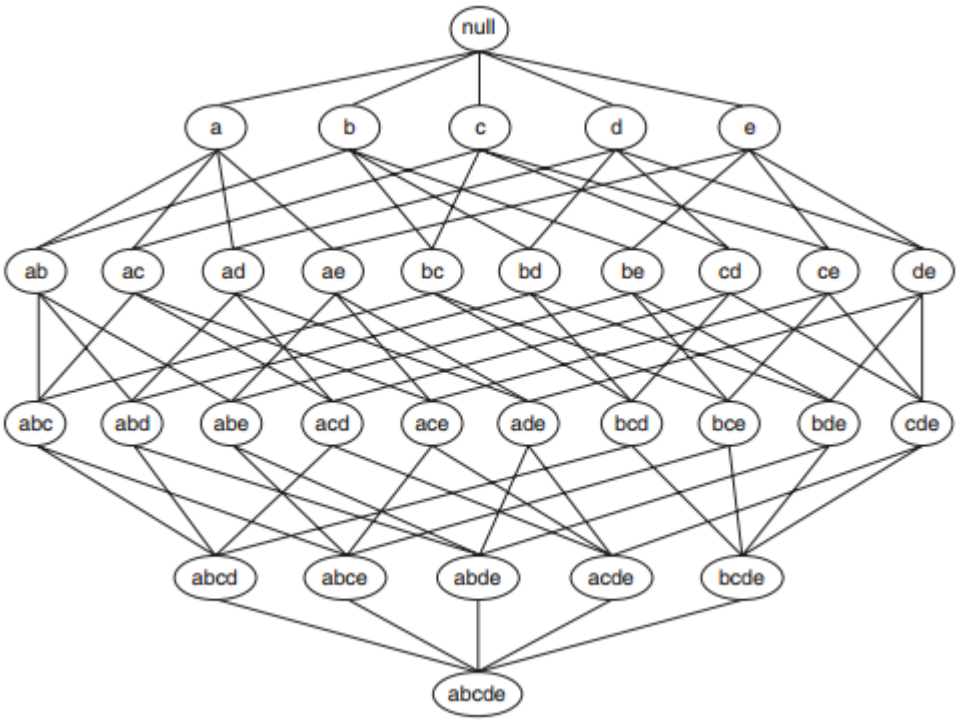


Figura 1 - Diagrama simulando possíveis *itemsets* (Tan et al. 2005)

Encontrar *Itemsets* Frequentes

Um diagrama como o da Figura 1 pode ser usado para emular a lista de todas as possíveis combinações de *itemsets*. Na Figura 1 temos os produtos $I = \{a, b, c, d, e\}$ e geralmente uma base de dados que contenha k produtos pode potencialmente gerar $2^k - 1$ *itemsets* frequentes, excluindo o conjunto vazio (*null*). Uma vez que k pode ser um valor muito grande, a quantidade de *itemsets* que têm de ser analisados é exponencialmente grande. Uma situação onde seja aplicado um método de força bruta, onde todas as combinações de *itemsets* são analisadas e comparadas entre elas, pode tornar-se um processo muito dispendioso, lento e por isso desaconselhado.

Gerar Regras de Associação

Cada *k-itemset* Y pode produzir $2^k - 2$ regras de associação, ignorando regras com antecedente ou consequente nulos $(Y \rightarrow \emptyset \; ou \; \emptyset \rightarrow Y)$. Uma regra de associação pode ser gerada dividindo um *itemset* em dois *itemsets* não nulos X e $Y - X$, sendo que a regra $X \rightarrow Y - X$ satisfaça o mínimo de confiança definido.

2.3 Regras de Associação Positivas

Ao longo dos tempos, vários algoritmos que geram regras de associação positivas, foram apresentados. De seguida são apresentados alguns desses algoritmos.

Apriori

O algoritmo Apriori foi proposto por Agarwal and Srikant in 1994 com o objetivo de encontrar *itemsets* frequentes e geração de regras de associação, operando sobre uma base de dados transacional (Agrawal & Srikant 1994). Por base de dados transacional consideramos uma base de dados que contenha transações como por exemplo os produtos comprados por clientes ou os detalhes sobre o uso de um *website*. O algoritmo identifica *items* individuais frequentes da BD e depois estende esses *items* singulares $(k-1)$ para *itemsets* cada vez maiores (k) desde que esses *itemsets* também tenham uma presença suficientemente grande na BD (ver Figura 3). Para contar estes *itemsets* candidatos eficientemente, o algoritmo usa uma procura *breadth-first* (Skiena 2008) e uma estrutura *hash tree*.

1: $k = 1$.
2: $F_k = \{ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$. {Find all frequent 1-itemsets}
3: **repeat**
4: $k = k + 1$.
5: $C_k = \text{apriori-gen}(F_{k-1})$. {Generate candidate itemsets}
6: **for** each transaction $t \in T$ **do**
7: $C_t = \text{subset}(C_k, t)$. {Identify all candidates that belong to t}
8: **for** each candidate itemset $c \in C_t$ **do**
9: $\sigma(c) = \sigma(c) + 1$. {Increment support count}
10: **end for**
11: **end for**
12: $F_k = \{ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}$. {Extract the frequent k-itemsets}
13: **until** $F_k = \emptyset$
14: Result $= \bigcup F_k$.

Figura 2 - Pseudo-code do algoritmo Apriori (Tan et al. 2005)

O Apriori, ainda que historicamente muito relevante, sofre hoje de algumas limitações ou restrições. A procura de *itemsets* candidatos gera um grande número de *subsets*. Além disso, a procura por *breadth-first* para encontrar todos os *itemsets* possíveis, pode tornar-se um processo lento e pesado. Para resolver este problema, algoritmos como o *Max-Miner*, propõem-se a extrair os melhores *itemsets* através de uma procura mais eficiente (Bayardo 1998).

ECLAT

O algoritmo ECLAT (*Equivalence CLAss Transformation*) usa um método para agrupar os *itemsets* frequentes e depois divide-os pelos vários processadores. Além disso, agrupa as transações relacionadas para depois disponibilizar aos processadores o conjunto de transações relevantes naquela instância. O algoritmo propõe-se ainda a reduzir o número de vezes que a base de dados é consultada otimizando assim todo o processo.

O algoritmo ECLAT foi apresentado por Zaki, Parthasarathy, Li and Ogihara em 1997 (M. J. Zaki et al. 1997; Mohammed Javeed Zaki et al. 1997)

FP-Growth

O algoritmo FP-growth ou *Frequent Pattern Growth* foi introduzido por Han em 2000 (Han et al. 2000).
O algoritmo FP-Growth permite uma alternativa ao processo de encontrar *itemsets* frequentes sem usar *candidate generations* melhorando assim o desempenho. A principal característica do algoritmo é o facto de usar uma estrutura de dados chamada FP-Tree (*frequente-pattern tree*), que guarda a informação sobre as associações entre os *itemsets*. O algoritmo começa por compactar a informação vinda da BD numa estrutura FP-Tree para representar os itemsets frequentes. De seguida, divide a estrutura FP-tree em BDs mais pequenas, em que em cada uma está a informação de cada padrão frequente. Finalmente, cada uma das BDs mais pequenas é minerada separadamente. A grande vantagem deste método prende-se na redução dos custos de procura para cada padrão encontrado, já que a informação necessária para analisar um certo padrão já está na BD mais pequena que foi construída.

Estrutura FP-Tree

A *Frequent-Pattern Tree* é uma estrutura de dados compacta que guarda informação sobre a frequência de *itemsets* numa Base de Dados.
Han define uma estrutura FP-Tree como uma estrutura em árvore (Han et al. 2000) em que:

1) A raiz da árvore como "*null*" com as várias "árvores filho" de *items* como ramos dessa raiz e uma tabela com a lista de items frequentes (Figura 3). Adicionalmente, a tabela também pode conter o suporte associado a cada *item*.

2) Cada nodo na árvore contém três campos:
    a. O nome do *item*
    b. O número de transações consoante o peso desse ramo.
    c. A ligação ao nodo seguinte na árvore com ligação a esse *item* ou a um *null* caso onde não exista outro *item* com ligação.
3) Cada entrada na tabela contém:
    a. Um ID com o mesmo nome do nodo
    b. Um apontador para o primeiro nodo com o mesmo ID na árvore
    c. Um contador com o suporte que esse item tem na Base de Dados

O pseudo-código do algoritmo original proposto por Han (Han et al. 2000) para a construção da *FP-Tree* pode ser visto na Figura 4. Usando este algoritmo, a estrutura FP-Tree é construída percorrendo apenas duas vezes a BD. Uma primeira vez, para recolher e organizar a lista de *items* frequentes e uma segunda vez, para contruir a FP-Tree.

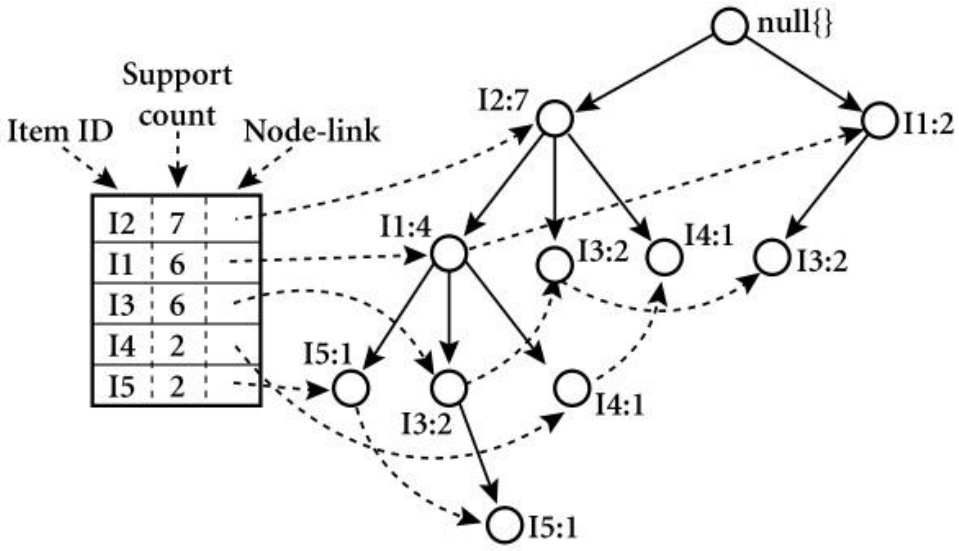


Figura 3 - Exemplo da estrutura *FP-Tree* (Han & Kamber 2001)

Input: A transaction database *DB* and a minimum support threshold δ.

Output: FP-tree, the frequent-pattern tree of DB.

Method: The FP-tree is constructed as follows.

1. Scan the transaction database *DB* once. Collect *F*, the set of frequent items, and the support of each frequent item. Sort *F* in support descending order as *L*, the *list* of frequent items.
2. Create the root of an FP-tree, *T*, and label it as "null". For each transaction *Trans* in *DB* do the following:
   - Select and sort the frequent items in *Trans* according to the order of *L*. Let the sorted frequent item list in *Trans* be [p | P], where *p* is the first element and *P* is the remaining list. Call *insert_tree*([p | P],T).
   - The function *insert_tree*([p | P],T) is performed as follows. If *T* has a child *N* such that *N.item-name = p.item-name*, then increment *N*'s count by 1; else create a new node *N*, with its count initialized to 1, its parent link linked to *T*, and its node-link linked to the nodes with the same *item-name* via the node-link structure. If *P* is nonempty, call *insert_tree*(P,N) recursively.

Figura 4 - Pseudo-codigo da construção da FP-Tree

Algoritmo FP-Growth

Depois da construção da estrutura FP-Tree, essa estrutura é minerada para se encontrar a lista de *itemsets* frequentes. Para realizar esta tarefa, Han (Han et al. 2000) apresenta o algoritmo FP-Growth tal como é apresentado na Figura 5 o seu pseudo-código.

Input: FP-Tree constructed, using database *DB* and a minimum support threshold δ.

Output: The complete set of frequent patterns.

Method: Call FP-Growth(FP-tree, null).

Procedure FP-growth(*Tree*,α) {

(1) **if** Tree contains a single path P

(2) **then for each** combination (denoted as β) of the nodes in the path P **do**

(3) generate pattern $\beta \cup \alpha$ with support = minimum support of nodes in β;

(4) **else for each** α_i in the header of Tree **do** {

(5) generate pattern $\beta = \alpha_i \cup \alpha$ with support = $\alpha_i.support$;

(6) construct β*'s* conditional pattern base and then β*'s* conditional FP-tree $Tree_\beta$;

(7) **if** $Tree_\beta \neq \emptyset$

(8) **then** call FP-growth($Tree_\beta$,β) }

}

Figura 5 - Pseudo-código do algoritmo FP-Growth

Este algoritmo foi mais aprofundado nesta secção já que foi este um dos algoritmos escolhidos para o desenvolvimento do projeto.

2.4 Regras de Associação Negativas

Até recentemente, os algoritmos de regras de associação apenas eram capazes de extrair regras de associação positivas $(X \rightarrow Y)$ sem prestarem grande atenção a regras de associação negativas. Ainda assim, regras de associação como $(X \rightarrow \neg Y)$ podem ter muito impacto e relevância para o caso em estudo (Silverstein et al. 1998). Regras de associação negativas analisam precisamente os mesmos *itemsets* analisados pelas regras positivas com o acréscimo que também analisam a negação de um *itemset*, isto é, a inexistência do mesmo. Uma regra de associação negativa pode tomar três formas: (1) o consequente é negativo $(X \rightarrow \neg Y)$, (2) o antecedente é negativo $(\neg X \rightarrow Y)$ e (3) tanto o antecedente como o consequente são negativos $(\neg X \rightarrow \neg Y)$. Aplicando estes três tipos de regras, a uma lógica de retalho, (1) seria uma situação onde a compra do produto X implica que o produto Y não seja comprado, (2) uma situação onde a não compra do produto X implica a compra do produto Y e (3)

uma situação onde a não compra do produto X implica que o produto Y também não seja comprado (Brin, Motwani & Silverstein 1997).

Para o caso das regras de associação negativas, os medidores de suporte e confiança sofrem algumas alterações, como pode ser visto nas seguintes fórmulas:

$$supp(X \Rightarrow \neg Y) = supp(X) - supp(X \cup Y)$$
$$conf(X \Rightarrow \neg Y) = \frac{supp(X \Rightarrow \neg Y)}{supp(X)}$$
$$supp(\neg X \Rightarrow Y) = supp(Y) - supp(X \cup Y)$$
$$conf(\neg X \Rightarrow Y) = \frac{supp(\neg X \Rightarrow Y)}{1 - supp(X)}$$
$$supp(\neg X \Rightarrow \neg Y) = 1 - supp(X) - supp(X) + supp(X \cup Y)$$
$$conf(\neg X \Rightarrow \neg Y) = \frac{supp(\neg X \Rightarrow \neg Y)}{1 - supp(X)}$$

Mais recentemente, com o aumento na procura de soluções que completem os já existentes algoritmos de regras positivas com regras de associação negativas, alguns investigadores, propuseram alguns métodos para conseguir extrair tanto regras de associação positivas como negativas de uma base de dados (Alataş & Akin 2006; Taniar et al. 2012; Wu et al. 2004).

De seguida são apresentados alguns desses algoritmos.

WNAIIMS

Em 2012, foi apresentado o algoritmo WNAIIMS (*Weighted Negative Association rules from Infrequent Itemsets based on Multiple Supports*) (Jiang et al. 2012).

Segundo os autores, em muitos dos algoritmos de regras de associações, é usado apenas um único valor de suporte mínimo. Isto leva a que, caso o valor de suporte seja muito alto, *itemsets* com frequência baixa sejam perdidos e que caso o valor de suporte seja muito baixo pode haver *combinatorial explosion*, isto é, o algoritmo devolve um número exponencial de resultados, fazendo com que o valor dos mesmos seja mais reduzido (Burton A. Leland et al. 1997). É mais eficaz que, para grandes bases de dados, sejam usados vários valores de suporte mínimo. O objetivo deste

método é então encontrar regras de associação negativas, baseadas em *itemsets* pouco frequentes, usando múltiplos valores de suporte mínimo.

SARIC

Em 2014 foi proposto um algoritmo, SARIC (*Set particle swarm optimization for Association Rules using the Itemset range and Correlation coeficiente*) (Agrawal et al. 2014). O algoritmo proposto usa otimização de *particle swarm* para gerar regras de associação, a partir de uma base de dados, considerando tando ocorrências positivas como ocorrências negativas dos atributos. É usada uma gama de *itemsets* e um coeficiente de correlação para que não seja preciso especificar um valor de suporte e confiança mínimo, já que o algoritmo consegue detetar os valores de forma rápida e objetiva.
O algoritmo divide-se em duas fases. Na primeira fase, a fase de pré-processamento, os dados são transformados para uma estrutura binária, de seguida é calculado uma gama de *itemsets* e por fim, são feitos os cálculos de correlação para categorizar os *itemsets* como positivos ou negativos. Na segunda fase, a fase de mineração, é aplicado um algoritmo PSO *(particle swarm optimization)* para se extrair regras tanto positivas como negativas (Figura 4).

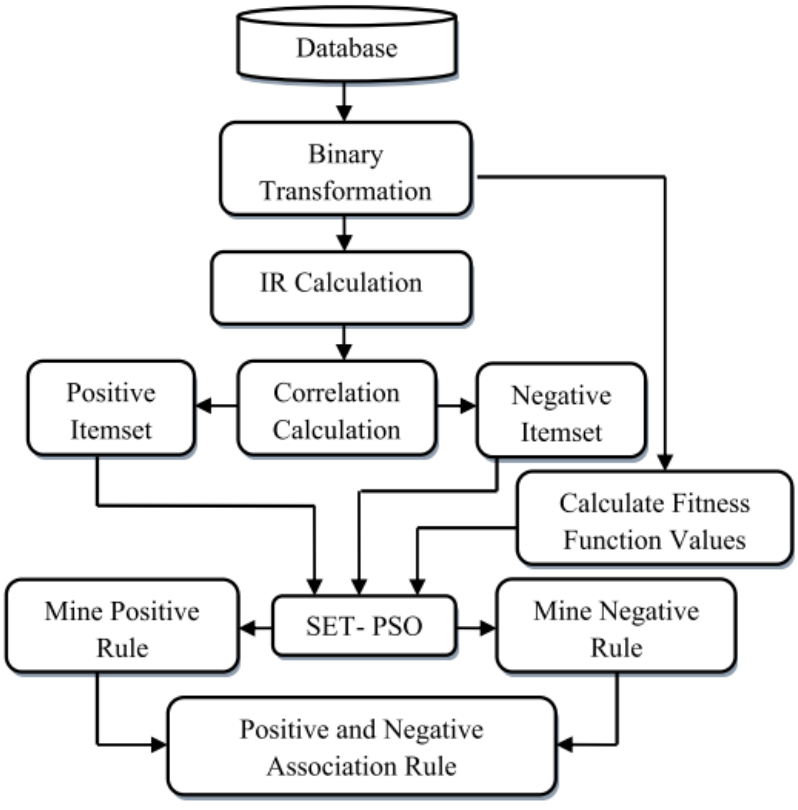


Figura 6 - Ilustração do algoritmo SARIC (Agrawal et al. 2014)

IMLMS-PANR-GA

Também em 2014, foi apresentado o algoritmo IMLMS-PANR-GA, baseado no algoritmo min-max e na fórmula MLMS (*Multi Level Minimum Support*) (Poundekar et al. 2014). Inicialmente, os dados são analisados e divididos em *itemsets* frequentes e *itemsets* não-frequentes, estando os dados divididos, o tempo de procura na BD é bastante reduzido. Por fim, é usado o algoritmo min-max durante o processo de otimização das regras. A grande vantagem deste método deve-se ao facto de permitir extrair regras de associação negativas, tanto de *itemsets* frequentes como *itemsets* infrequentes da base de dados.

MIPNAR-GA

Com o algoritmo MIPNAR-GA (*Mining Interesting Positive and Negative Association Rules – Genetic Algorithm*) os autores propõem-se a eliminar problemas que outros algoritmos apresentavam, como múltiplas pesquisas na base de dados e o grande número de regras negativas aí extraídas. Para isso baseia-se nos algoritmos MLMS (*Multi Level Minimum Support*) e genético (Rai et al. 2014). O algoritmo está dividido em três fases: uma primeira, onde são extraídos padrões frequentes e infrequentes usando o método Apriori; uma segunda fase, onde são geradas regras de associação e por fim são selecionadas as regras mais interessantes aplicando medidores de interesse. Este último passo é feito recorrendo ao algoritmo genético.

MOPNAR

Em 2014, Diana Martín propôs o algoritmo MOPNAR (*Multiobjective Evolutionary Algorithm for Mining Positive and Negative Quantitative Association Rules*) (Martin et al. 2014). Segundo a autora, o objetivo do algoritmo, é encontrar uma lista de regras de associação positivas e negativas mais pequena com um baixo custo de computação.

De forma a criar um estudo evolutivo, a autora pegou no algoritmo MEOA/D-DE (Li & Zhang 2009) adicionando uma população externa (EP – *External Population*) e um processo de reiniciação. O MEOA/D-DE decompõe o problema de otimização multiobjectivo em N partes mais pequenas e utiliza um algoritmo evolutivo para trabalhar essas N partes mais pequenas simultaneamente. Para guardar todas as regras não-dominadas encontradas, provocar variedade na população e melhorar a cobertura dada aos dados, é introduzido o EP e o processo de reiniciação. O EP vai guardar todas as regras não-dominadas e atualizá-las sempre que for gerada uma nova solução para cada população. Caso

apareçam regras redundantes, elas serão removidas do EP. Uma vez que não existem limites no tamanho do EP, este permite obter um grande número de regras independentemente do tamanho da população e reduzir o tamanho da população para criar independência da base de dados. O processo de reiniciação vai criar mais diversidade na população e é usado sempre que o número de novos indivíduos da população, numa nova geração, for inferior a $\alpha\%$ do tamanho atual da população. Neste caso, os casos abrangidos pelas regras no EP são marcados e o processo de reiniciação da população é aplicado de modo a que os casos não abrangidos pelas regras no EP possam ser analisados. Assim, o EP será atualizado com uma nova população seguindo o critério de não-dominância e as regras que sejam redundantes serão também removidas.

Os três principais objetivos do MOPNAR são: interesse, compreensão e desempenho. Desempenho representa a intensão de melhorar o alcance da leitura dos dados de modo a extrair conhecimento mais interessante. Desempenho é o produto do suporte e do fator de certeza (secção 2.2.1) para que seja possível retirar regras fortes. Interesse é o modo pelo qual podemos medir quão interessante uma regra é, usando o medidor *lift* (secção 2.2.1). Finalmente, compreensão (*comprehensibility*) tenta quantificar quão fácil é perceber uma regra (Fidelis et al. 2000). A medição da compreensão de uma regra $(X \rightarrow Y)$ é feita consoante o número de atributos envolvidos nessa regra $(Attr_{X \rightarrow Y})$:

$$comprehensibility(X \rightarrow Y) = {}^{1}/_{Attr_{X \rightarrow Y}}$$

Input:

1) N population size;
2) $nTrials$ number of evaluations;
3) m number of objectives;
4) P_{mut} probability of mutation;
5) $\lambda 1, \ldots \lambda N$ a set of N weight vectors;
6) T the number of weight vectors in the neighborhood of each weight vector;
7) δ the probability that parent solutions are selected from the neighborhood;
8) η_r the maximal number of solutions replaced by each child solution;
9) γ factor of amplitude for each attribute of the dataset;
10) α difference threshold.

Output: EP

Figura 7 - Input e Output do MOPNAR

Step 1: Initialize.

a) Compute the Euclidean distances between any two weight vectors and then work out the T closest weight vectors to each weight vector. For each $i = 1, \ldots, N$ set $Bi = \{ii, \ldots iT\}$ where $\lambda^{i_1}, \ldots \lambda^{i_T}$ are the T closest weight vectors to λ^i.

b) Generate the initial population with N chromosomes.

c) Evaluate the initial population.

d) Initialize $z = (z1, \ldots, zm)$ by setting $zj = max1 \leq i \leq N\ fj(xi), j = 1, \ldots, m$.

e) Initialize the EP

Step 2: Update. For each $i = 1, \ldots, N$ do the following.

a) Uniformly randomly generate a number rand from $[0,1]$. Then set

$$P = \begin{cases} B(i), & if\ rand < \delta \\ \{1, \ldots, N\}, & otherwise. \end{cases}$$

b) Set $r_1 = i$ and randomly select r_2 from P. The solutions x^{r_1} and x^{r_2} are crossed, generating two offspring g: y^1 and y^2. Next, the mutation and repairing operators are applied for the two offspring.

c) Evaluate the new individuals. For each $y_k, k \in \{1, 2\}$.

i. Update of z: For each $j = 1, \ldots m$, if $z_j < f_j(y_k)$, then set $z_j = f_j(y_k)$.

ii. Update solutions: Set $c = 0$ and then do the following.

A. If $c == \eta_r$ or P is empty go to Step 3. Otherwise randomly pick an index l from P.

B. If $g(y_k|\lambda^l, z) \leq g(x^l|\lambda^l, z)$, then $x^l = y_k$ and $c = c + 1$

C. Remove l from P and go to a).

Step 3: Update of EP: remove from EP all the vectors dominated by $i, i = 1, \ldots, N$, then add i to EP if no vectors in EP dominate it.

Step 4: Remove redundance from the EP.

Step 5: If the difference between the current population and previous population is less than α%, restart the population.

Step 6: If the maximum number of evaluations is not reached, go to Step 2.

Step 7: Remove redundance from the EP.

Step 8: The EP is returned.

Figura 8 - Pseudo-código do Algoritmo MOPNAR

O algoritmo pode ser descrito resumidamente através da figura 7, onde temos todos os *inputs* que o algoritmo necessita para iniciar o processo de mineração, tal como o *output* gerado, e através da figura 8, onde temos o pseudo-código do processo de mineração do MOPNAR.

Este algoritmo foi mais aprofundado já que foi este um dos algoritmos escolhidos para o desenvolvimento do projeto (secção 3.4.4).

2.5 Market Basket Analysis

Análise de afinidade (*Affinity Analysis*) é uma técnica de mineração de dados e análise de dados para descobrir relacionamentos entre ocorrências que foram executadas por agentes individuais ou grupos de agentes. Pode, portanto, ser aplicada a qualquer processo onde os agentes podem ser identificados singularmente e a informações sobre as suas ações pode ser guardada. Na área do retalho, a análise de afinidades é usado para fazer *Market Basket Analysis*, isto é, uma análise ao carrinho de compras de modo a tentar descobrir padrões ou hábitos de compras por parte dos consumidores. Executar estudos de MBA sobre os dados de vendas dos clientes, permite aos retalhistas aplicar esse conhecimento para fazer *cross-selling*, *up-selling*, promoções de saldos, programas de lealdade de clientes, alterar o *design* da loja, alterar o posicionamento dos produtos na loja, fazer simples baixas de preços nos produtos, entre outros (Gutierrez 2006; Haughton et al. 2003).

MBA é uma das técnicas mais usadas em bases de dados transacionais. O grande objetivo da utilização destas técnicas, é simplesmente extrair conhecimento interessante sobre os padrões de vendas dos consumidores para com isso tentar com que o consumidor gaste mais dinheiro e consuma mais, maioritariamente através de duas técnicas: *up-selling*, que consiste em tentar fazer com que o cliente compre uma quantidade maior de um mesmo produto ou adicionando ao produto extras como extensão de garantias e seguros, ou através de *cross-selling*, que consiste em fazer com que o consumidor compre também outros produtos de outras categorias (Videla-Cavieres & Ríos 2014). Este conhecimento é recolhido através da utilização de algoritmos de regras de associação (Agrawal & Srikant 1994; Agrawal, Imielinski, et al. 1993).

Antigamente, apenas pequenas quantidades de dados sobre as vendas numa loja eram guardadas e acessíveis em sistemas informáticos. Mais recentemente, tecnologias como o código de barras e

informatização das caixas registadoras e a geral evolução das tecnologias informáticas, veio permitir que grandes quantidades de dados sobre as transações de produtos numa loja pudessem ser armazenados (Chen et al. 2005). Sistemas que permitam analisar e extrair conhecimento a partir destas bases de dados de transações, têm o potencial de gerar informação de alto valor para os retalhistas (Agrawal, Imieliński, et al. 1993).

MBA pode ser usado como uma ferramenta para a tomada de decisão quanto ao posicionamento dos produtos na loja. Sabendo que a venda de um produto está diretamente associado à venda de outro produto, aproximar a localização desses produtos na loja, pode fazer com que a venda desses produtos aumente. Aproximar a localização de dois produtos pode ser uma estratégia interessante, mas afastar esses produtos pode ser igualmente interessante. Se a relação desses dois produtos for forte e dependente entre si, afastar esses produtos na loja pode fazer com que o cliente tenha de percorrer uma maior extensão da loja, incentivando a que assim, o cliente compre outros produtos que estejam posicionados entre os outros dois (Vindevogel et al. 2005; Abbas et al. 2013).

Uma área onde MBA é muito usado para descobrir relação entre produtos e padrões de compras de clientes é no comércio *online*, (Leskovec et al. 2007) e melhor exemplo é o caso da Amazon. Negócios *online*, como a Amazon, estão muito dependentes de sistemas de sugestão de produtos e para isso é necessário existir bons sistemas de mineração para conseguir extrair relações de produtos e padrões de compras dos clientes interessantes (Basuchowdhuri et al. 2014; Oestreicher-Singer & Sundararajan 2006).

Capítulo 3

3. Caso de Estudo

3.1 Contextualização

Neste caso de estudo foram utilizados os dados fornecidos por um retalhista brasileiro referentes aos registos das vendas. Tal como foi explicado na secção 1.2, um dos objetivos desta dissertação passa pela escolha de dois algoritmos para utilizar sobre os dados disponíveis. Um dos algoritmos para descobrir regras de associação positivas e outros para negativas. Associado à escolha dos dois algoritmos é também necessário escolher qual a melhor plataforma/ferramenta para a execução desses mesmos algoritmos.

Depois de escolhidos os algoritmos mais apropriados a este caso de estudo, será então necessário aplicar esses algoritmos sobre os dados para tentar obter uma coleção de regras interessantes. O objetivo é analisar as relações que a compra de um, ou grupo de produtos, pode ter sobre outro produto, ou grupo de produtos. Descobrir até que ponto a compra de um produto pode influenciar o cliente a comprar outro produto. Uma vez que este problema vai ser analisado utilizando algoritmos de regras positivas e negativas, vai-se também tentar descobrir situações onde a compra de um produto pode influenciar negativamente a compra de outro produto ou vice-versa (secção 2.4).

Assim, os algoritmos vão ser executados em várias situações diferentes para melhor se poder analisar o seu desempenho e resultados. Essas situações podem ser por exemplo, quantidades de dados diferentes.

A última fase do estudo será uma análise dos resultados obtidos e também uma análise crítica aos algoritmos consoante os seus pontos fortes e fracos. Para a realização deste caso de estudo foi necessário a adoção de uma metodologia. A metodologia escolhida foi o CRISP-DM, que é explicada na secção seguinte.

3.2 CRSIP-DM

Para facilitar todo o processo de DM foram criados ao longo do tempo vários *standards* como o *Java Data Mining standard* (JDM) (Hornick et al. 2007) ou o *Cross Industry Standard Process for Data Mining* (CRISP-DM) (IBM n.d.). Segundo vários estudos realizados pelo *site* KDNuggests, a metodologia mais usada para projetos de *data mining* é a metodologia CRISP-DM, ganhando esta sempre por uma larga vantagem face às outras metodologias (KDnuggets 2014). A CRISP-DM chegou mesmo a ser descrita como *"de facto standard for developing data mining and knowledge discovery projects."* (Kesavaraj & Sukumaran 2013). Uma vez que é a metodologia mais usada na indústria, foi também esta a metodologia que foi usada.

A metodologia CRISP-DM apareceu em 1996 e em 1997 no projeto apoiado pela União Europeia dentro da Iniciativa ESPRIT (*European Strategic Programo n Research in Information Technology*). A primeira versão da metodologia foi apresentada em março de 1999, em Bruxelas, na *4th CRISP-DM SIG* (CRISP-DM *Special Interest Group*) *Workshop* (Chapman 1999), sendo que em 2000 seria publicado como guia passo-a-passo (Chapman et al. 2000).

Entre os anos de 2006 e 2008 a CRISP-DM 2.0 SIG foi formada com o intuito de se renovar a metodologia CRISP-DM (Shearer 2006). Neste momento não se sabe muito sobre o estado atual da nova metodologia sendo que até o *site* da metodologia original (*crisp-dm.org*) e o da nova metodologia (*crisp-dm.org/new.html*) já não se encontram ativos.

A utilização da metodologia CRISP-DM traz várias vantagens, como o facto de ser independente da indústria onde é aplicada, isto é, o mesmo processo pode ser aplicado a dados comerciais, financeiros, industriais, etc. e além disso é também independente da ferramenta utilizada. Outra vantagem advém de que a metodologia está fortemente ligada ao processo de KDD. Estas vantagens e ainda o facto de esta metodologia estar muito bem documentada e por ser de fácil aplicação, fazem com que ela torne a implementação dos projetos de DM mais baratos, mais rápidos e mais fáceis de gerir.

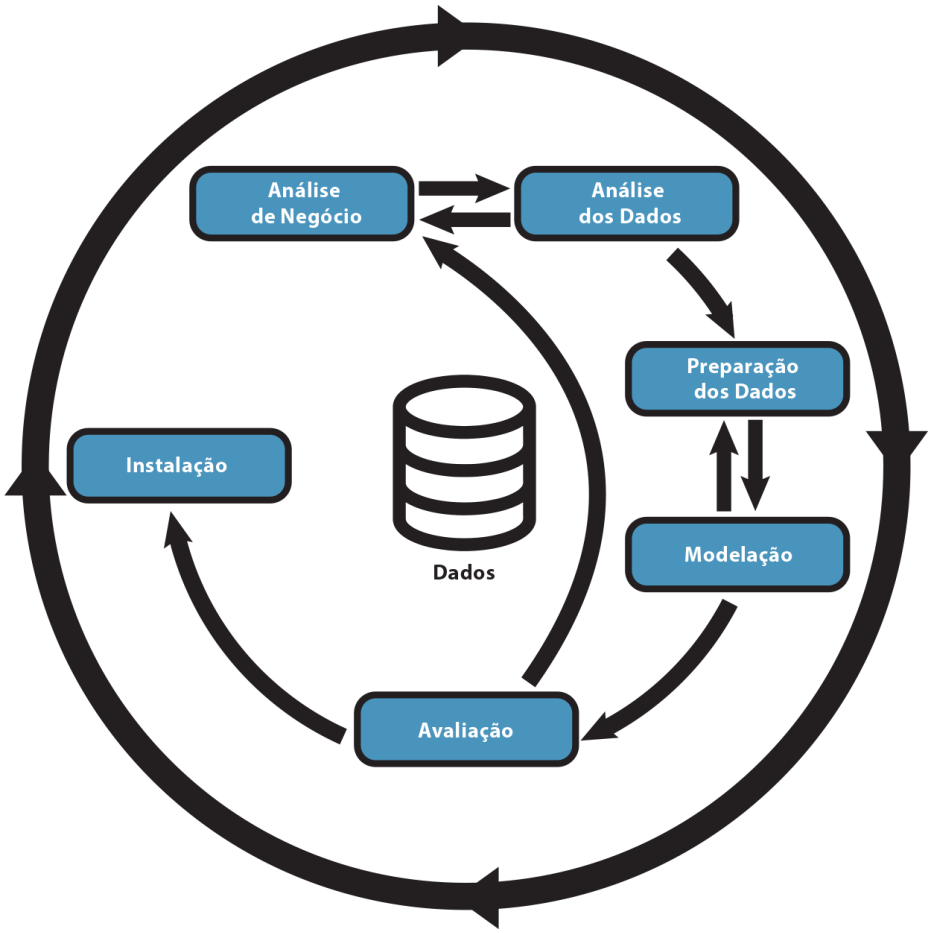


Figura 9 - Fases do CRISP-DM – imagem adaptada de (Chapman et al. 2000)

A metodologia CRISP-DM trata-se de um processo cíclico onde as várias fases têm dependências entre elas, sendo que este processo cíclico não é restritivo, já que é possível voltar atrás numa determinada fase do processo. As várias setas entre as fases, presentes na Figura 9, representam as dependências mais importantes e frequentes entre as fases, enquanto a seta de fora simboliza a natureza cíclica do processo de DM em si. O processo não acaba depois de a solução ter sido encontrada e aplicada já que todo o conhecimento recolhido de todo o processo pode ser utilizado para criar questões mais

específicas e assim, o processo DM pode continuar a ser desenvolvido para beneficiar do conhecimento obtido em iterações anteriores.

Tal como pode ser visto na Figura 9, a metodologia CRISP-DM divide-se em seis fases. A fase de análise de negócio já foi abordada nos capítulos anteriores. As fases de análise dos dados e preparação dos dados serão abordados na secção seguinte. A fase de modelação será analisada no capítulo 4. Finalmente, a fase de avaliação será analisada no capítulo 5. Uma vez que este projeto não pressupõe uma fase de instalação dos modelos numa infraestrutura, a última fase da metodologia CRISP-DM não será considerada.

3.3 Dados Utilizados

3.3.1 Introdução aos dados

Para o trabalho desenvolvido nesta dissertação, foram utilizados dados provenientes da base de dados de vendas de um retalhista brasileiro de grande dimensão com várias lojas espalhadas pelo Brasil. Esses dados consistem nos registos de vendas de todos os produtos vendidos durante um período de aproximadamente dois anos.

Sendo este retalhista, de dimensão considerável, a diversidade de produtos presentes no negócio é bastante considerável. No total são 76454 produtos diferentes divididos e agrupados por várias categorias (níveis mercadológicos).

Tabela 3 - Estrutura Mercadológica

Nível	Descrição
Merch_L1	Companhia
Merch_L2	Área de Negócios
Merch_L3	Departamento
Merch_L4	Seção
Merch_L5	Linha
Merch_L6	Nível
Merch_L7	Sub-nível

Os produtos estão organizados em sete níveis diferentes “tabela 3”, em que cada nível representa um agrupamento de vários produtos segundo a lógica comercial adotada pelo retalhista, isto é, uma

estrutura mercadológica. Os produtos ficam assim organizados em níveis do mais abstrato ao mais específico e restritivo.

3.3.2 Descrição dos dados

Os dados são registados pelo retalhista ao nível do produto, ou seja, cada registo na base de dados, representa um produto vendido pertencente a um carrinho específico num dado momento temporal. Assim, como o registo está ao nível do produto, um carrinho de compras ou talão poderá ter vários registos, caso ele tenha mais do que um produto. A base de dados tem uma grande quantidade de atributos, especificamente cinquenta e nove, para descrever cada produto vendido o mais pormenorizadamente possível, como por exemplo, a semana, dia e hora da venda, ID e descrição do produto, a estrutura mercadológica ao qual o produto pertence, código da loja, quantidade vendida, valor da venda, etc. O Anexo I possui a lista completa dos atributos e a descrição de cada atributo.

3.3.3 Seleção dos dados

Se considerarmos o tamanho do retalhista, o facto de ele possuir várias lojas e também que os registos de vendas têm um intervalo de tempo de dois anos, é fácil imaginar que o tamanho total desta base de dados seria realmente muito extensa e isso ia tornar este projeto inexequível numa primeira iteração. Por estas razões, para este projeto, apenas foram trabalhados os dados referentes à loja de código 93 e apenas com uma parte dos dados dessa loja. Assim, foram selecionados um conjunto de algumas categorias de nível 4 com o qual se iria trabalhar, ainda que o intervalo de tempo da base de dados não tenha sido alterado para que os dados continuassem a ter a mesma longevidade. Esta nova base de dados teria todas as vendas registadas dos produtos pertencentes às categorias selecionadas.

Tabela 4 - Estrutura mercadológica de um produto

Produto	***LEITE FERM BOB ESPONJA MOR CJ C/6 480G***
Merch_L1	"Companhia"
Merch_L2	Perecíveis
Merch_L3	FLC
Merch_L4	LEITE FERMENTADO
Merch_L5	INFANTIS
Merch_L6	NATURAL
Merch_L7	NATURAL

A seleção de dados com base no nível 4 deve-se ao facto de este ser o nível onde é mais fácil identificar famílias de produtos, sendo mais fácil de entender por um utilizador. Olhando para a "tabela 4" que usa o produto *"LEITE FERM BOB ESPONJA MOR CJ C/6 480G",* como exemplo, é fácil de ver que o nível 4 é o mais fácil para se trabalhar, não sendo nem muito vago nem muito restritivo.
Neste subconjunto ficaram um total de 860 produtos diferentes, agrupados por 83 categorias diferentes de nível 4. Ficou-se assim com uma base de dados com um total de 2860049 registos de produtos vendidos. Este número traduz-se em 1576179 carrinhos de compras diferentes, com uma média de 1.81 produtos por carrinho de compras. Este número pode parecer baixo e explica-se pelo facto de que apenas só foram trabalhados um subconjunto de produtos. Por exemplo, um consumidor poderia ter feito uma compra com dez produtos, mas se desses produtos apenas um pertence ao subconjunto previamente escolhido, a amostra terá apenas um produto referente a esse carrinho de compras.

3.3.4 Preparação dos dados

Primeira análise e limpeza dos dados

Numa primeira análise aos dados deparou-se com a existência de registos de produtos repetidos dentro do mesmo carrinho de compras. Esta repetição acontecia 1259 vezes e nenhum produto repetia mais do que uma vez. Uma vez que cada carrinho só pode ter um registo para cada produto diferente, era necessário verificar o porquê desta repetição acontecer.

Depois de se consultar a lógica de negócio do retalhista, constatou-se que esta repetição deve-se à existência de uma lei brasileira que impossibilita qualquer retalhista de vender produtos por custo zero. Assim, quando o retalhista pretende fazer uma promoção do tipo "leve 3 pague 2", esse terceiro produto não é vendido por custo zero mais sim pelo menor custo possível (um cêntimo). Assim, nestes casos, aparece um registo para o produto A com um valor de venda (TRAN_VAL) igual a 20 reais (10 reais cada produto) e quantidade vendida (TRAN_QTY) igual a dois e outro registo também para o produto A mas com um valor de venda igual a um cêntimo e quantidade vendida igual a um.

Para resolver este problema, para os casos onde isto acontecia, decidiu-se agrupar estes registos e somar a quantidade do produto vendida e também o valor da venda. Assim, e pegando no exemplo anterior, ficaria apenas um registo do produto A mas agora com um valor de venda (TRAN_VAL) igual a

20.01 reais e quantidade vendida (TRAN_QTY) igual a três. Depois disto, o número de registos desceu para 2858790.

Transformação dos dados

Outra situação que os dados apresentavam era não terem um atributo que identificasse o carrinho de compra de forma singular. Existe um atributo chamado "COUNPON_ID" mas este não é único, já que carrinhos diferentes podem ter o mesmo "COUNPON_ID". Assim, para criar um identificador único para o carrinho, foi necessário concatenar alguns atributos, nomeadamente "SALES_DATE_KEY", "SALES_HOUR_KEY", "REGISTER_KEY" e "COUNPON_ID". A concatenação destes atributos é necessária já que a caixa registadora que atribui o "COUNPON_ID" ao carrinho fá-lo de forma independente de outras caixas registadoras e pode repetir os códigos no mesmo dia e até na mesma hora. É de notar ainda que, como para este projeto só foram trabalhadas as vendas da loja 93, não foi necessário concatenar também o atributo identificativo de loja ("LOC_KEY").

Esta concatenação teve de ser feita sempre que era efetuada uma extração de dados da BD para serem depois processados pelas ferramentas utilizadas.

3.4 Algoritmos escolhidos

Tal como mencionado no início desta secção, um dos objetivos era selecionar um algoritmo para analisar regras de associação positivas (*halo effect*) e outro para analisar regras de associação negativas (canibalismo). Esses algoritmos teriam de ser atuais e relevantes num estudo deste tipo.
Para capturar e analisar regras de associação positivas optou-se pela solução oferecida pela aplicação RapidMiner para estudos de M*arket Basket Analysis* que usa o algoritmo FP-Growth. O RapidMiner é umas das ferramentas mais usadas tanto a nível empresarial como académico para estudos de mineração de dados e por isso pareceu uma opção bastante interessante.

Para capturar e analisar regras de associação negativas, selecionou-se o algoritmo MOPNAR proposto por Diana Martín em 2014, que, além de analisar regras negativas, também analisa regras positivas (Martin et al. 2014). Este método foi desenvolvido com o recurso à ferramenta KEEL e por isso também se usou essa mesma ferramenta.

3.4.1 Ferramenta RapidMiner

O RapidMiner é uma plataforma de *software* desenvolvida pela empresa com o mesmo nome que fornece ferramentas para *machine learning*, *data mining*, *text mining*, *predictive analysis* e *business analytics*. É bastante usada em ambientes empresariais, industriais e de estudo científico, mas também em áreas de educação. Em termos de DM, a aplicação que suporta todos os passos do processo de DM, incluindo a visualização de resultados, validação de resultados e otimização (Hofmann & Klinkenberg 2013). O RapidMiner é desenvolvido num modelo de negócio e por isso apenas a versão anterior à versão mais atual se encontra disponível como *open-source* (RapidMiner n.d.). Existe uma versão *Starter Edition* que está disponível para *download* grátis e depois as versões *Personal Edition, Professioanl Edition* e *Enterprise Edition* com preços diferentes dependendo das circunstâncias e necessidades do utilizador.

Ferramenta inicialmente conhecida como YALE (*Yet Another Learning Enviroment*) foi desenvolvida no início de 2001 por Ralf Klinkenberg, Ingo Mierswa e Simon Fisher na *Artificial Intelligence Unit* da Universidade Tecnica de Dortmund (Deutsch 2010). A partir de 2006, o desenvolvimento passou a estar sob a alçada da Rapid-I, empresa fundada por Ralf Klinkenberg e Ingo Mierswa (KDnuggets 2010). Em 2007 o nome da ferramenta passou a chamar-se RapidMiner (KDnuggets 2007).

Hoje, o RapidMiner, é das plataformas de análise mais utilizadas com mais de 250000 utilizadores. Grande parte dos modelos disponíveis no Rapidminer não requerem conhecimentos de programação já que os operadores são do tipo *drag-and-drop*. Isto torna a aplicação bastante popular (ButleAnalytics 2015).

Neste projeto a versão do RapidMiner utilizada foi a RapidMiner 5.3.015.

3.4.2 Ferramenta KEEL

KEEL (*Knowledge Extraction based on Evolutionary Learning*) é uma ferramenta *open-source* desenvolvida em *Java* que pode ser usada para realizar um grande número de tarefas de descobrimento de conhecimento em bases de dados. Foi desenvolvida pelo grupo de investigação da Universidade de Granada - *Soft Computing and Intelligent Information Systems*, ou SDI2S.
O KEEL utiliza uma interface simples (Figura 10) baseada no fluxo de dados para desenhar experiências com o possível recurso a várias BDs e utilizando vários algoritmos. Contém uma grande

variedade de algoritmos na sua biblioteca, desde os algoritmos mais clássicos de extração, pré-processamento, aprendizagem, estatísticos, entre outros. No total são mais de 500 algoritmos que podem ser usados e comparados de várias formas (Alcalá-Fdez et al. 2011).

Figura 10 - Janela principal do KEEL

O KEEL foi desenvolvido com dois objetivos, educação e o estudo científico (Alcalá-Fdez et al. 2008). No caso de estudo científico, o principal uso do investigador será automação de execução de experiencias e a análise estatística dos resultados. A ferramenta permite ao investigador fazer testes e comparações entre os algoritmos e os resultados. No caso educacional, os requisitos dos alunos são bastante diferentes dos de um investigador e por isso, o mais importante, passa a ser o acompanhamento em tempo real da evolução do algoritmo para que depois o aluno possa fazer ajustes nos parâmetros dos algoritmos. Além disso, é possível aceder aos resultados finais através da mesma interface.

Neste projeto a versão do KEEL utilizada foi a KEEL Suite 3.0.

3.4.3 Algoritmo FP-Growth

A ferramenta RapidMiner, para realizar o estudo de MBA, utiliza dois módulos, o FP-Growth e o Create Association Rules (Figura 11).

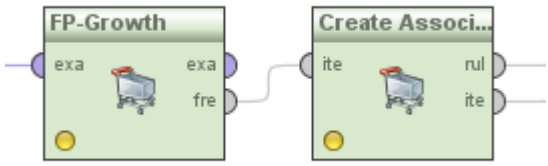


Figura 11 - Processo de MBA do RapidMiner

Módulo FP-Growth

Este módulo calcula todos os *itemsets* frequentes a partir de um *ExampleSet* que recebe como *input* (RapidMiner 2015b).

Características do Módulo:

Input:

- *example set (Data Table)*: dados em formato binário a ser processados.

Output:

- *example set (Data Table)*: os mesmos dados do *input* saem também como *output* caso o utilizador pretenda utilizar esses dados para outra função.
- *frequent sets (Frequent Item Sets)*: lista dos *itemsets* frequentes.

Parâmetros:

- *find_min_number_of_itemsets*: caso este parâmetro seja posto a verdadeiro, o módulo encontra um número mínimo de *itemsets*. Nesta situação, o valor de suporte mínimo é em parte ignorado já que ele vai ser diminuído até ter sido encontrado o mínimo de *itemsets* desejado.

  Tipo: boleano
- *min_number_of_itemsets*: número mínimo de *itemsets* que é pretendido encontrar. Este parâmetro só é considerado caso o anterior seja verdadeiro.

  Tipo: inteiro

- *max_number_of_retries*: este parâmetro define quantas vezes o valor do suporte pode ser diminuído para encontrar o número mínimo de *itemsets* (20% de cada vez). Este parâmetro também só é considerado caso o primeiro parâmetro seja verdadeiro.
  Tipo: *string*
- *positive_value*: este parâmetro determina quais os valores binominais que devem ser considerados como positivos. Os atributos com esses valores são considerados como parte da transação. Se for deixado vazio o módulo decide que valores deve usar.
  Tipo: *string*
- *min_support*: valor mínimo de suporte desejado.
  Tipo: real
- *max_items*: quantidade máxima de *itemsets* que o módulo deve devolver. Se este parâmetro for -1 nenhum máximo é definido.
  Tipo: inteiro
- *must_contain*: caso seja pretendido, o utilizador pode especificar que *items* devem estar contidos nos *itemsets* que o módulo devolve.

FP-Growth

Tal como o próprio nome do módulo indica, este módulo usa o algoritmo FP-Growth para determinar a lista dos *itemsets* mais frequentes segundo os parâmetros acima descritos. Uma descrição mais alargada sobre o funcionamento do algoritmo pode ser consultada na secção 2.3 deste documento.

Módulo Create Association Rules

Este módulo gera a lista de regras de associação a partir de uma lista de *itemsets* frequentes (RapidMiner 2015a).

Características do Módulo:

Input:

- *item sets (Frequent Item Sets)*: lista de itemsets frequentes

Output:

- *item sets (Frequent Item Sets)*: os mesmos dados do *input* saem também como *output* caso o utilizador pretenda utilizar esses dados para outra função.

- *rules (Association Rules)*: lista de regras de associação.

Parâmetros:

- *criterion*: lista de critérios que serão usados para selecionar as regras. O utilizador só pode selecionar um critério (confiança, *lift*, convicção, ganho, Laplace, ps).
- *min_confidence/min_criterion_value*: valor mínimo do critério escolhido.
  Tipo: real
- *gain_theta*: este parâmetro só é usado quando o critério Ganho é selecionado e especifica o valor de *Theta*.
  Tipo: real
- *laplace_k*: este parâmetro só é usado quando o critério Laplace é selecionado e especifica o valor de k.
  Tipo: real

Depois de encontrados todos os *itemsets* frequentes que respeitam o valor de suporte mínimo definido, através do módulo "FP-Growth" do RapidMiner, o módulo seguinte, "*Create Association Rules*" vai pegar nos *itemsets* do FP-Growth aplicando um valor mínimo de confiança e devolver a lista de regras de associação.

Uma descrição mais alargada sobre o funcionamento da geração de regras de associação pode ser consultada na secção 2.2.2 deste documento.

3.4.4 Algoritmo MOPNAR

O MOPNAR é um dos algoritmos presentes do repositório de algoritmos da ferramenta KEEL. Por requisito da ferramenta todos os dados a serem processados têm de ser previamente carregados para a ferramenta de modo a utilizarem a estrutura de dados *standard* do KEEL. O processo de mineração a utilizar na ferramenta KEEL é bastante simples bastando selecionar os dados a analisar e utilizar o módulo MOPNAR-A (Figura 12).

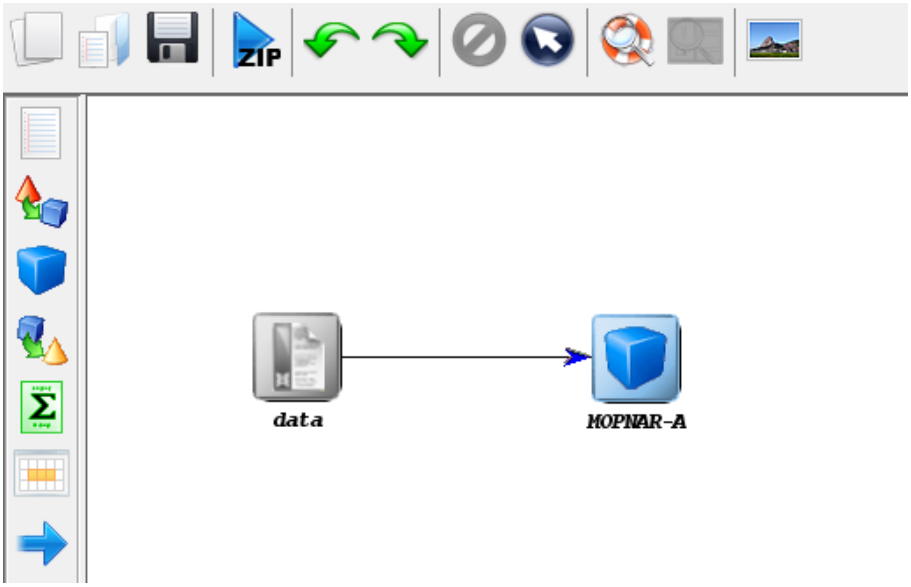


Figura 12 - Processo KEEL

Uma descrição mais alargada sobre o funcionamento do algoritmo MOPNAR pode ser consultada na secção 2.4 deste documento.

Capitulo 4

4. MODELOS DE MINERAÇÃO

Tal como foi referido na secção anterior, foram escolhidos dois algoritmos de regras de associação, um para regras positivas e outro para regras negativas. A partir destes dois algoritmos e das usas respetivas ferramentas, foram criados vários modelos fazendo variar, por exemplo, os tipos de dados selecionados ou o tamanho dos mesmos.

Lista de modelos:

1. Modelo R1 – Algoritmo: FP-Growth; Categoria de Produtos: culinários
2. Modelo R1.1 – Algoritmo: FP-Growth; Categoria de Produtos: doces
3. Modelo R1.2 – Algoritmo: FP-Growth; Categoria de Produtos: sanduiches
4. Modelo R1.3 – Algoritmo: FP-Growth; Categoria de Produtos: bolos
5. Modelo R1.4 – Algoritmo: FP-Growth; Categoria de Produtos: cafés
6. Modelo R1.5 – Algoritmo: FP-Growth; Categoria de Produtos: ovos
7. Modelo R1.6 – Algoritmo: FP-Growth; Categoria de Produtos: pães
8. Modelo R2 – Algoritmo: FP-Growth; Quantidade de registos: 250000
9. Modelo R2.1 – Algoritmo: FP-Growth; Quantidade de registos: 500000
10. Modelo R2.2 – Algoritmo: FP-Growth; Quantidade de registos: 750000
11. Modelo R2.3 – Algoritmo: FP-Growth; Quantidade de registos: 1000000
12. Modelo R2.4 – Algoritmo: FP-Growth; Quantidade de registos: 1250000
13. Modelo R2.5 – Algoritmo: RapidMiner; Quantidade de registos: todos os registos

14. Modelo K1 – Algoritmo: MOPNAR; Categoria de Produtos: culinários
15. Modelo K1.1 – Algoritmo: MOPNAR; Categoria de Produtos: doces
16. Modelo K1.2 – Algoritmo: MOPNAR; Categoria de Produtos: sanduiches
17. Modelo K1.3 – Algoritmo: MOPNAR; Categoria de Produtos: bolos
18. Modelo K1.4 – Algoritmo: MOPNAR; Categoria de Produtos: cafés
19. Modelo K2 – Algoritmo: MOPNAR; Categoria de Produtos: culinários (com atributos de data)
20. Modelo K2.1 – Algoritmo: MOPNAR; Categoria de Produtos: doces (com atributos de data)
21. Modelo K2.2 – Algoritmo: MOPNAR; Categoria de Produtos: sanduiches (com atributos de data)
22. Modelo K2.3 – Algoritmo: MOPNAR; Categoria de Produtos: bolos (com atributos de data)
23. Modelo K2.4 – Algoritmo: MOPNAR; Categoria de Produtos: cafés (com atributos de data)
24. Modelo K3 – Algoritmo: MOPNAR; Categoria de Produtos: bolos (15 atributos)
25. Modelo K3.1 – Algoritmo: MOPNAR; Categoria de Produtos: bolos (20 atributos)
26. Modelo K3.2 – Algoritmo: MOPNAR; Categoria de Produtos: bolos (25 atributos)
27. Modelo K3.3 – Algoritmo: MOPNAR; Categoria de Produtos: bolos (30 atributos)
28. Modelo K4 – Algoritmo: MOPNAR; Categoria de Produtos: bolos (20% dos registos)
29. Modelo K4.1 – Algoritmo: MOPNAR; Categoria de Produtos: bolos (40% dos registos)
30. Modelo K4.2 – Algoritmo: MOPNAR; Categoria de Produtos: bolos (60% dos registos)
31. Modelo K4.3 – Algoritmo: MOPNAR; Categoria de Produtos: bolos (80% dos registos)

4.1 Os Modelos R1, R1.1, R1.2, R1.3, R1.4, R1.5, R1.6

Os modelos R1, R1.1, R1.2, R1.3, R1.4, R1.5 e R1.6 foram implementados na ferramenta RapidMiner com o recurso ao algoritmo FP-Growth. Os modelos distinguem-se pelas categorias de produtos que são usadas nos modelos.

Uma vez que estes modelos usam um algoritmo que apenas analisa regras positivas, decidiu-se filtrar os dados de modo a que carrinhos de compras/transações que apenas continham um produto não fossem considerados. Esta decisão está relacionada com o objetivo deste caso estudo (secção 3.1). O objetivo principal do caso de estudo é descobrir a influência que um produto pode ter sobre a compra de um outro produto. Estamos portanto a tentar descobrir relações de influência entre produtos e para que A influencie B, esse produto B precisa de também de estar na mesma transação que A. Deste

modo, transações que só tenham um produto, não são capazes de oferecer nenhum tipo de relação entre produtos. A existência destas transações só vai sobrecarregar o algoritmo com dados sem valor para o objetivo deste caso de estudo.

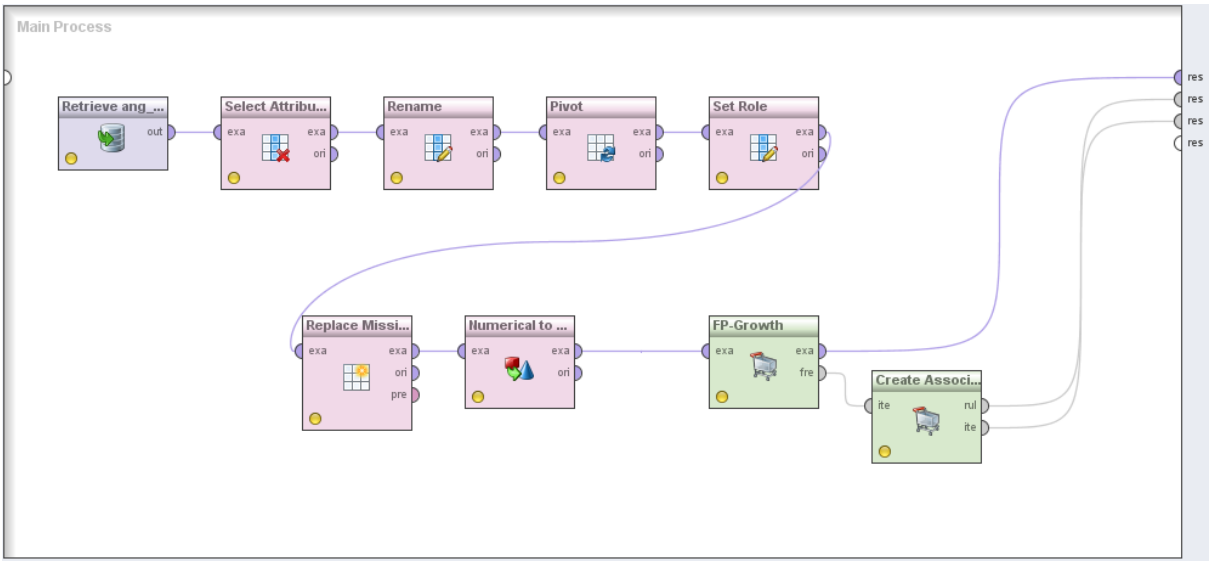


Figura 13 - Processo RapidMiner

O processo de execução destes modelos pode ser observado na Figura 13. O processo começa por carregar os dados pré-selecionados de modo a tornar o processo na ferramenta mais rápido. Caso necessário, o processo de seleção e extração dos dados podia ser todo executado dentro do RapidMiner. Por motivos de acesso aos dados da BD, optou-se, por primeiro fazer a seleção dos dados e extrair os dados da BD para um local de mais fácil acesso. Desta forma não houve limitações no acesso aos dados.

Os modelos R1, R1.1, R1.2, R1.3, R1.4, R1.5 e R1.6 podem agrupar-se juntos já que têm características muito semelhantes. O que distingue estes modelos são os dados selecionados. Cada modelo contém apenas as transações de uma categoria de produtos. Deste modo, o domínio destes modelos é mais restritivo, pois só queremos encontrar relações entre produtos que pertençam à mesma categoria. A seleção dos modelos foi feita com base no nível mercadológico quatro. Como alguns dos grupos de nível quatro são bastante parecidos e relacionados entre si, alguns dos modelos podem conter vários grupos de produtos de nível quatro - “tabela 5”.

Tabela 5 - Lista de Merch_L4s de cada modelo

Modelo	Nome	IDs Merch_L4	Descrições Merch_L4
R1	Culinários	13034	"CULINARIOS"
R1.1	Doces	115	"DOCES"
		15574	"DOCES"
R1.2	Sanduiches	320	"SANDUICHES ANG PAD LJ LANCH"
R1.3	Bolos	293,	"BOLOS SECOS CONF"
		14097	"BOLOS COM RECHEIO CONF"
R1.4	Cafés	14113	"CAFES"
R1.5	Ovos	412	"OVOS"
R1.6	Pães	14094	"PAES CASEIROS PAD REGIONAL"
		14095	"PAES DO MUNDO"
		14128	"PAES VITRINES PAD LJ BALCAO"
		14126	"PAES RUSTICOS PAD LJ BALCAO"
		14125	"PAES IMPORTACAO TERCEIRO"

Uma vez que os dados estão organizados ao nível do produto transacionado, cada produto vendido num dado momento numa dada loja é um registo singular (secção 3.3.2), logo os dados foram selecionados filtrando-os pela categoria de produtos. Esta seleção faz com que todas as transações que vão ser consideradas só contenham produtos da categoria escolhida mesmo que uma dada transação também contenha produtos de outras categorias. Isto, juntamento com a decisão de remover todas as transações de apenas um produto, vai diminuir a quantidade de transações que não vão ser analisadas pelo algoritmo.

Depois dos dados (Figura 14) estarem carregados, através do operador "*Retrieve*", os dados passam pelo operador "*Select Attributes*". Com este operador, selecionamos apenas os atributos necessários ao problema, para o caso destes modelos, os atributos necessários são o ID da transação, o ID do produto e quantidade de produto comprada numa dada transação.

	A	B	C
1	TRANS_ID	PROD_KEY	PROD_QTY_RUP
2	2638_14_8158_49247	931353	1
3	2638_14_8158_49247	891994	1
4	2638_18_8126_64045	973162	1
5	2638_18_8126_64045	891994	1
6	2644_20_8152_73922	891994	1
7	2644_20_8152_73922	973162	1
8	2650_10_8123_75176	973162	1
9	2650_10_8123_75176	891994	4
10	2653_18_8160_36753	931353	1
11	2653_18_8160_36753	891994	1
12	2659_18_8153_50782	891994	1
13	2659_18_8153_50782	931353	1
14	2667_16_8123_80776	898077	1
15	2667_16_8123_80776	891994	1
16	2676_0_2841_78177	931353	1
17	2676_0_2841_78177	973162	1
18	2694_19_8147_65235	931353	1
19	2694_19_8147_65235	898077	1

Figura 14 - Estrutura dos dados recebidos no RapidMiner

Uma vez que todo o processo de *Market Basket Analysis* do RapidMiner obriga à utilização de dados no formato binário, é necessário fazer algumas transformações aos dados antes de estes poderem ser processados. Tal como já foi descrito, os dados estão organizados ao nível do produto e não da transação, que é o pretendido. Para que os dados fiquem organizados ao nível da transação, transformam-se os dados iniciais numa tabela *pivot*. Este processo é obtido através do operador "Pivot". Este operador vai rodar a tabela dos dados recebidos agrupando-os por transação. Antes da transformação *pivot,* os dados têm três atributos, o ID da transação, o ID do produto e a quantidade vendida desse produto. Uma transação que tenha, por exemplo, quatro produtos, terá também na tabela quatro registos/linhas (Figura 14). Com o operador *Pivot* a tabela passará a estar organizada ao nível da transação. Assim, cada registo será uma transação diferente, os atributos da tabela serão os vários produtos e os valores da tabela serão a quantidade vendida de um produto para cada transação.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	TRANS_ID	p_1082670.0	p_1217972.0	p_1217973.0	p_1222151.0	p_1222152.0	p_1222153.0	p_1222156.0	p_891994.0	p_898077.0	p_908217.0	p_916236.0	p_931353.0	p_973162.0
2	2638_14_8158_49247	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE
3	2638_18_8126_64045	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE
4	2644_20_8152_73922	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE
5	2650_10_8123_75176	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE
6	2653_18_8160_36753	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE
7	2659_18_8153_50782	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE
8	2667_16_8123_80776	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
9	2676_0_2841_78177	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE
10	2694_19_8147_65235	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
11	2699_10_8132_96399	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE
12	2699_10_8162_47080	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE
13	2706_16_8150_40918	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
14	2707_10_8132_97958	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE
15	2746_13_8127_68217	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	FALSE
16	2754_11_8155_82937	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE
17	2755_8_8127_71994	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE
18	2757_14_8158_74637	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
19	2763_15_8127_75583	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE

Figura 15 - Estrutura dos dados depois da transformação Pivot

Antes do operador "*Pivot*" é utilizado o operador "*Rename*" para alterar o nome do atributo de "PROD_QT_RUP" para "p". A utilização deste operador é apenas para facilitar a organização da tabela *pivot* depois do processo de transformação. Tal como pode ser visto na Figura 15, os produtos ficam com a designação "p_#########".

Depois da transformação *pivot* o operador "*Set Role*" é usado para transformar o atributo "TRANS_ID" num id, ou seja, numa chave primária. De seguida, por uma questão de precaução, os dados passam pelo operador "*Replace Missing Values*" para ter a certeza que nenhum valor é "*null*". Caso exista alguma ocorrência de valores "*null*", esses valores serão substituídos por zero. Finalmente, os dados passam ainda pelo operador "*Numerial to Binominal*". É neste operador que todos os valores dos vários atributos/produtos serão transformados em valores binários. Assim, caso um produto exista numa determinada transação, ou seja, a quantidade de produto vendida for maior ou igual a um, esse produto ficará marcado como verdadeiro, caso contrário, marcado como falso (Figura 15).

Depois de todos estes passos, os dados estão finalmente prontos a entrarem no operador "FP-Growth" para realmente iniciar o processo de mineração de regras de associação. Depois de encontrados os *itemsets* mais frequentes, o operador envia a lista de *itemsets* para o operador seguinte mas também a tabela *pivot* que este operador recebe para os resultados finais do RapidMiner para que seja possível, no fim do processo, consultar a tabela pivot. Finalmente, o último operador neste processo é o "*Create Association Rules*" que, recebendo a lista de *itemsets* mais frequentes, vai extrair a lista de regras de associação mais interessantes. Este operador envia para a zona de resultados do RapidMiner a lista de regras obtidas e também a lista de *itemsets* mais frequentes que recebe do operador "FP-Growth".
As características dos dados destes modelos, tal como os valores de suporte e de confiança usados, podem ser consultados na "tabela 6".

Tabela 6 - Características dos modelos R1, R1.1, R1.2, R1.3, R1.4, R1.5, R1.6

Modulo	Categoria	Registos	Transações	Produtos	Suporte	Confiança
R1	Culinários	224	110	13	0.01	0.3
R1.1	Doces	4070	1814	29	0.01	0.3
R1.2	Sanduiches	2972	1477	15	0.01	0.3
R1.3	Bolos	13021	6110	30	0.01	0.3
R1.4	Cafés	16527	8196	10	0.01	0.3
R1.5	Ovos	13027	6418	21	0.01	0.3
R1.6	Pães	96108	43385	47	0.01	0.3

O suporte escolhido foi de 0.01, ou seja, 1% para que todos os *itemsets* encontrados tenham pelo menos 1% de peso nos dados que estão a ser analisados. 1% pode parecer pouco, já que, um *itemset* com um peso nos dados de apenas 1% pode parecer um produto com pouca relevância. Ainda assim, optou-se por usar 1%, já que numa primeira análise dos dados, nenhum produto ou grupo de produtos mostrou ter um peso interessante, isto é, todos os *itemsets* tinham um peso relativamente pequeno. Ainda assim, o baixo valor de suporte não vai prejudicar as regras obtidas ou mesmo a quantidade das regras já que o valor do suporte é um valor de suporte mínimo. Depois de obtida a lista de regras, qualquer regra que se ache desinteressante pode ser desconsiderada. Outro motivo para se utilizar um valor de suporte baixo é também tentar encontrar regras com um valor de confiança alto ainda que com baixo impacto nos dados, ou seja, regras com *itemsets* pouco frequentes. O valor de suporte escolhido foi de 0.3 ou 30%, já que regras com uma confiança inferior a 30% são regras com uma relação entre os produtos baixa e por isso desinteressante.

4.2 Os Modelos R2, R2.1, R2.2, R2.3, R2.4, R2.5

Os modelos R2, R2.1, R2.2, R2.3, R2.4 e R2.5, à semelhança dos modelos anteriores, também foram implementados na ferramenta RapidMiner com recurso ao algoritmo FP-Growth. Estes modelos distinguem-se entre eles pela quantidade de dados usados.

Ainda que o objetivo destes modelos continue a ser descobrir relações de associação entre produtos, estes modelos têm também o objetivo de analisar o desempenho do algoritmo. Cada um dos modelos deste grupo tem quantidades de dados muito diferentes, desde os 250 mil registos até ao último modelo onde foi usado a totalidade de dados disponível. Tal como nos modelos anteriores, todas as transações que só tinham um produto também foram retiradas.

O processo de execução destes modelos na ferramenta RapidMiner é em tudo igual ao processo dos modelos anteriores (Figura 13). As características dos dados destes modelos, tal como os valores de suporte e de confiança usados, podem ser consultados na "tabela 7".

Tabela 7 - Características dos modelos R2, R2.1, R2.2, R2.3, R2.4, R2.5, R2.6

Modulo	Registos	Transações	Produtos	Suporte	Confiança
R2	250000	89514	551	0.005	0.3
R2.1	500000	179087	608	0.005	0.3
R2.2	750000	266968	661	0.005	0.3
R2.3	1000000	356410	681	0.005	0.3
R2.4	1250000	448070	694	0.005	0.3
R2.5	todos	723292	847	0.005	0.3

Em contraste com os modelos anteriores, apenas o valor de suporte foi alterado. O valor do suporte usado foi 0.005 ou 0.5%. A razão para utilizar um valor de suporte ainda baixo deve-se ao facto de agora se estar a trabalhar com quantidades de dados muito maiores, tanto no número de transações como no número de produtos. Com este número alargado de dados, a probabilidade de um *itemset* ter um peso relevante é ainda mais baixa. O valor de confiança foi mantido em 30%, uma vez que regras com uma confiança inferior a 30% pressupõem uma relação entre os produtos baixa e desinteressante.

4.3 Os Modelos K1, K1.1, K1.2, K1.3, K1.4

Os Modelos K1, K1.2, K1.2, K1.3 e K1.4 foram implementados na ferramenta KEEL com o recurso ao algoritmo MOPNAR. Estes modelos, à semelhança do primeiro grupo de modelos (R1), vão distinguir-se pelos dados usados, isto é, categorias de produtos de nível 4 diferentes.

Ao contrário dos modelos anteriores que só iam analisar regras de associação positivas, o algoritmo MOPNAR é capaz de capturar tanto regras positivas como negativas. Como os algoritmos para conseguir capturar tanto as regras positivas como negativas têm de analisar tanto quando um produto está presente numa transação como quando um produto não está presente numa transação, toda a informação é relevante. Assim, para estes modelos, nenhuma das transações que só tenham um só produto foram desconsideradas.

Tabela 8 - Características dos modelos K1, K1.1, K1.2,K1.3, K1.4

Modulo	Merch_L4	Registos	Transações	Produtos
K1	Culinários	6723	6609	13
K1.1	Doces	23118	20462	52
K1.2	Sanduiches	43302	41806	25
K1.3	Bolos	93553	86642	35
K1.4	Cafés	127744	119412	10

Estes modelos distinguem-se pelas categorias de produtos de nível 4 que foram escolhidas. Para poder ser efetuado um estudo que pudesse, no fim, ser de algum modo comparável com os modelos usados no RapidMiner (R1) as categorias de produtos escolhidas foram as mesmas - "tabela 8". Se formos comparar a quantidade de dados, nomeadamente a quantidade de transações que estes modelos contêm comparativamente aos modelos R1, "tabela 6", rapidamente verificamos que o facto de as transações de um só produto não serem removidas, vai tornar estes modelos bastante mais volumosos. Anteriormente, o modelo R1, com as transações da categoria "culinários", só tinha 110 transações, enquanto que agora o modelo K1, também com as transações da categoria "culinários", já tem 6609 transações.

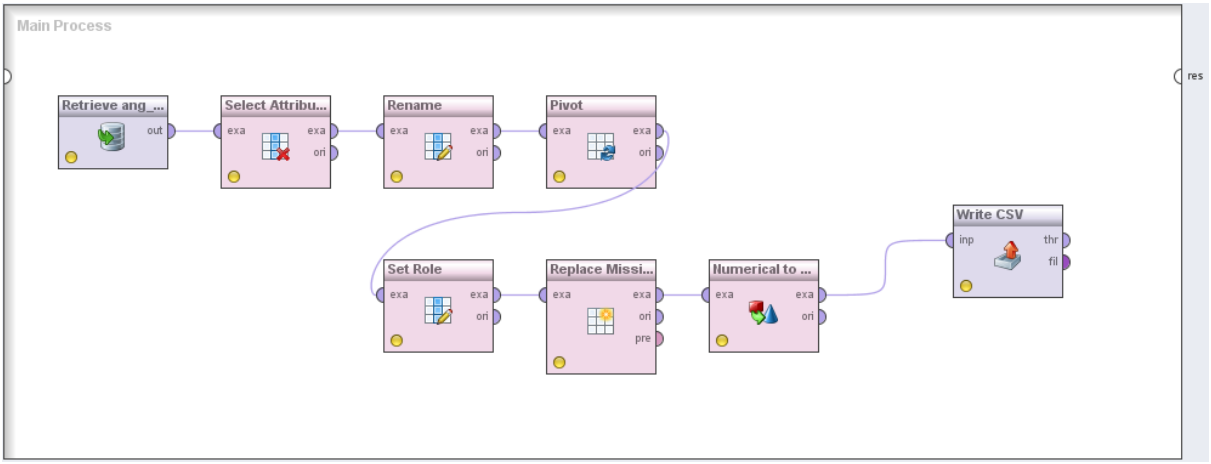


Figura 16 - Processo RapidMiner - transformação Pivot

Quanto ao processo de mineração na ferramenta KEEL com o algoritmo MOPNAR, como a ferramenta não apresenta nenhuma solução para a transformação dos dados numa estrutura ao nível da transação tal como o método de transformação *pivot* do RapidMiner, esta parte do processo foi também realizada no RapidMiner. Para isso utilizou-se um processo idêntico ao da Figura 13, com uma distinção no final do processo já que agora os dados, depois da transformação *pivot*, serão guardados num ficheiro *csv* para serem, de seguida, transportados para a ferramenta KEEL (Figura 16).

Uma vez que o KEEL utiliza o seu próprio sistema de organização de dados, antes dos dados poderem ser processados pelo algoritmo, têm de ser carregados para a ferramenta para que esta os guarde segundo a sua própria estrutura. Uma das particularidades do algoritmo MOPNAR é que não precisa de um atributo identificador do registo, neste caso, transação. Assim, quando os dados são carregados no KEEL, o atributo "TRANS_ID" é removido, ficando a tabela apenas com os atributos identificativos dos vários produtos. O processo de construção do modelo na ferramenta KEEL pode ser consultado na secção 3.4.4.

Outro diferenciador deste algoritmo em relação ao FP-Growth é o facto de este não precisar que o utilizador defina os valores de suporte e confiança. O algoritmo analisa os dados e decide quais os melhores valores para que fiquem sempre ajustados da melhor maneira possível com base nos dados a ser analisados.

4.4 Os Modelos K2, K2.1, K2.2, K2.3, K2.4

Os modelos K2, K2.1, K2.2, K2.3 e K2.4, à semelhança dos modelos anteriores, também foram implementados na ferramenta KEEL, com recurso ao algoritmo MOPNAR. Estes modelos distinguem-se dos anteriores pelos atributos selecionados.

As características dos modelos K2* são idênticas às dos modelos K1* com a diferença que estes também têm informações da data a que a transação foi efetuada - "tabela 8". Uma vez que o algoritmo MOPNAR já não tem a restrição de que todos os dados precisam de estar no formato binário, outro tipo de informações pode ser usado no processo de mineração. Aproveitando esta característica, decidiu-se fazer um estudo idêntico ao anterior, ou seja, com os mesmos grupos de dados, mas agora com um atributo indicativo no dia da semana e outro com a hora que a transação foi realizada. O dia da semana é um número inteiro de domínio [1, 7], sendo que o valor 1 simboliza sábado, 2 domingo, etc. A hora é também um inteiro de domínio [0, 23], ou seja, segue o sistema de 24h, onde o valor zero é a meia-noite, etc. Justifica-se a utilização das 24 horas, já que em algumas lojas e circunstâncias, o retalhista está aberto 24 horas por dia.

Com estes novos atributos, o objetivo do estudo é alargado, para além de descobrir relações de compra entre produtos, tenta descobrir também relações entre a compra de produtos e momentos temporais. Seria interessante, por exemplo, conseguir descobrir a que dias da semana a venda de um produto está mais associada.

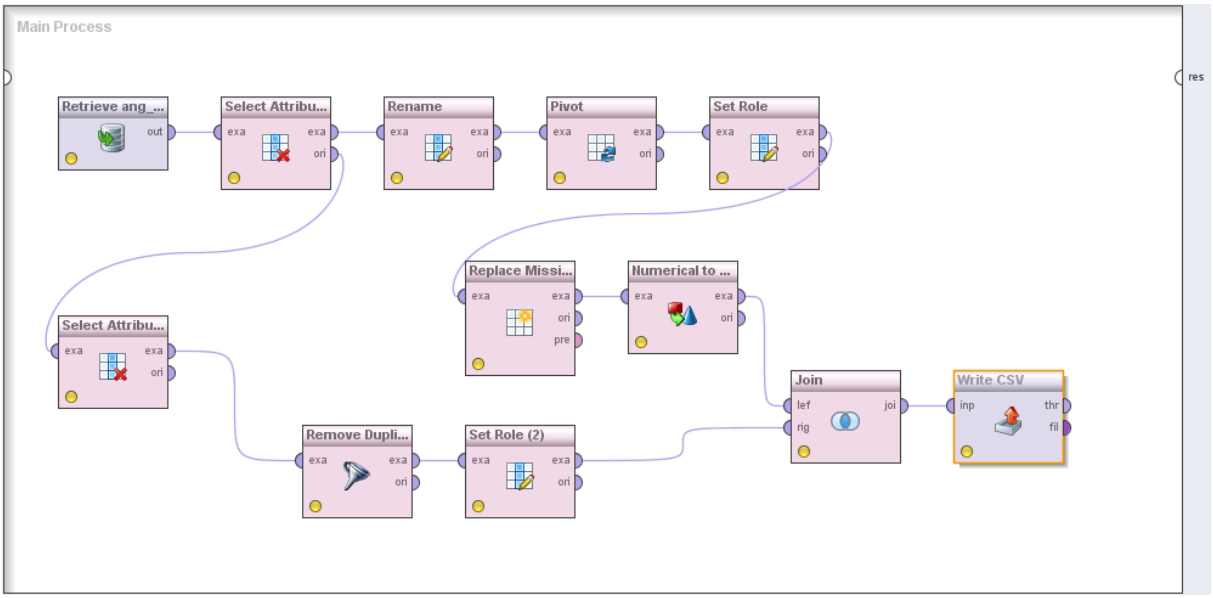


Figura 17 - Processo RapidMiner - transformação Pivot, com atributos de data

Para que os dados contenham os atributos do dia da semana e da hora da transação, é necessário fazer algumas alterações ao processo de transformação da tabela *pivot* (Figura 17). Assim, no início do processo, no RapidMiner, os atributos de data são separados dos dados referentes aos produtos. A transformação em tabela pivot decorre, tal como anteriormente, apenas com as informações dos produtos transacionados. Despois da transformação, os atributos de data são ligados aos dados da tabela pivot segundo o atributo identificativo da transação. Por fim, os dados são guardados para depois serem carregados para a ferramenta KEEL.

Todo o processo de mineração destes modelos é igual ao utilizado nos modelos anteriores K1 (Figura 16). Também à semelhança dos modelos anteriores, todas as outras características dos dados e parâmetros do algoritmo destes modelos não foram alterados - “tabela 9”.

4.5 Os Modelos K3, K3.1, K3.2, K3.3, K4, K4.1, K4.2, K4.3

Finalmente, os modelos K3, K3.1, K3.2, K3.3, K4, K4.1, K4.2 e K4.3, à semelhança dos modelos anteriores, também foram implementados na ferramenta KEEL com recurso ao algoritmo MOPNAR. Estes modelos foram implementados para realizar uma análise mais aprofundada ao desempenho do algoritmo MOPNAR.

Tabela 9 - Características dos modelos K1.3, K3, K3.1,K3.2, K3.3

Modelo	Transações	Produtos
K1.3	86642	35
K3	86642	15
K3.1	86642	20
K3.2	86642	25
K3.3	86642	30

À semelhança dos modelos R2, que tinham como parte do objetivo analisar o desempenho do algoritmo FP-Growth, estes modelos vão analisar o desempenho do MOPNAR. Para isso, escolheu-se o modelo K1.3 que contém as transações referentes à categoria de produtos "Bolos", sem atributos de data, e criou-se dois grupos de modelos. O primeiro grupo de modelos K3 vai analisar o desempenho do algoritmo quanto à quantidade de produtos analisados - "tabela 9". O segundo grupo de modelos K4 vai fazer uma análise do desempenho do algoritmo, mas agora quanto à quantidade de transações analisadas - "tabela 10". Estes dois grupos de modelos vão ser sempre comparados ao modelo K1.3. O objetivo destes modelos será então tentar analisar o desempenho do algoritmo quando o número de atributos (produtos) ou o número de registos (transações) varia.

Tabela 10 - Características dos modelos K2.3, K4, K4.1,K4.2, K4.3

Modulo	Transações	Produtos
K1.3	86642	35
K4	20%	35
K4.1	40%	35
K4.2	60%	35
K4.3	80%	35

Todo o processo de tratamento dos dados e transformação na tabela pivot é idêntico aos modelos anteriores, com a pequena diferença da quantidade de produtos ou transações usadas (Figura 17).

Capítulo 5

5. Análise dos Resultados

Como foi referido no capítulo anterior, foram utilizados vários modelos para testar os dois algoritmos de várias maneiras distintas.

Este capítulo divide-se em duas partes. Na primeira parte, serão apresentadas e analisadas as regras obtidas nos vários modelos implementados. As regras foram avaliadas tendo em conta os medidores de interesse que apresentam. Na segunda parte, os vários modelos serão analisados quanto ao desempenho apresentado.

5.1 Analise aos resultados dos Modelos

5.1.1 Os Modelos R1, R1.1, R1.2, R1.3, R1.4, R1.5, R1.6

Os modelos R1, R1.1, R1.2, R1.3, R1.4, R1.5, R1.6, que foram implementados na ferramenta Rapidminer, com o recurso ao algoritmo FP-Growth (secção 4.1), foram realizados sobre os dados de venda com um intervalo de tempo de aproximadamente dois anos. Seguindo os parâmetros já discutidos na secção 4.1, são apresentados na "tabela 11" os resultados gerais obtidos, nomeadamente o número de *items* frequentes, o número de regras obtidas e ainda o tempo que o algoritmo demorou a executar, em segundos. Esta duração refere-se a todo o processo do RapidMiner, tanto a execução do algoritmo como também todo o processo anterior ao algoritmo onde os dados são transformados na tabela *pivot*.

Tabela 11 - Resultados gerais dos modelos R1

Modelos	Categoria	*Items* Frequentes	Regras Obtidas	Tempo (s)
R1	Culinários	42	19	0
R1.1	Doces	148	2	0
R1.2	Sanduiches	50	22	0
R1.3	Bolos	104	5	0
R1.4	Cafés	28	11	0
R1.5	Ovos	102	6	0
R1.6	Pães	111	6	2

Analisando a "tabela 11", conseguimos ver que em algumas das categorias escolhidas foi bastante complicado obter regras, mesmo com um suporte baixo de 1%, nomeadamente os modelos R1.1, R1.3, R1.5 e R1.6. Isto indica-nos que poderá não haver regras de associação entre produtos destas categorias ou então que os dados analisados não serão suficientes para conseguir encontrar regras com peso nos dados.

Ainda assim, mesmo que em alguns dos modelos não tenha sido encontrado um número de regras significativo, algumas das regras encontradas mostram ter um valor de confiança elevado. A "tabela 12" mostra-nos as cinco regras com maior suporte de cada modelo, e dentro desse conjunto de regras, conseguimos encontrar algumas regras com um valor de confiança alto. O modelo que apresentou melhores valores de suporte foi o modelo R1.4, com os produtos da categoria de cafés, já que também esta é a categoria com menos produtos diferentes, apenas 10 produtos. Já o modelo R1.6 é um dos modelos que apresenta regras com valor de suporte baixo mas, em contrapartida é o modelo que tem mais produtos diferentes, 47 produtos. Isto faz com que os valores de suporte apresentados, ainda que algo baixos em alguns casos, não são necessariamente maus, apenas refletem as categorias dos modelos.

Tabela 12 - As 5 regras com maior suporte de cada modelo R1

Modelo	Regra	Suporte	Confiança	*Lift*
R1	[891994]⟹ [931353]	0.19	0.30	1.154e16
	[931353] ⟹[891994]	0.19	0.72	1.154e16
	[898077] ⟹[891994]	0.17	0.67	1.082e16
	[1082670] ⟹[891994]	0.1	0.69	1.096e16
	[1217972] ⟹[1217973]	0.1	0.73	6.722e15
R1.1	[1206293] ⟹[1226564]	0.013	0.308	5.695e15
	[1226582] ⟹[1226564]	0.012	0.328	6.073e15
R1.2	[1167459] ⟹[927862]	0.178	0.33	0.862
	[927862] ⟹[1167459]	0.178	0.464	0.862
	[1167459] ⟹[1167460]	0.166	0.308	1.230e15
	[1167460] ⟹[1167459]	0.154	0.662	1.230e15
	[1193364] ⟹[1164348]	0.154	0.726	3.841e15
R1.3	[960039] ⟹[1170035]	0.103	0.364	1.478e16
	[1170035] ⟹[960039]	0.103	0.42	1.478e16
	[1127288] ⟹[960039]	0.042	0.318	1.12e16
	[943554] ⟹[896503]	0.040	0.351	1.434e16
	[1226733] ⟹[1226737]	0.008	0.359	9.364e15
R1.4	[1077274] ⟹[1055229]	0.235	0.327	1.108e15
	[1055229] ⟹[1077274]	0.235	0.797	1.108e15
	[1077274] ⟹[1077275]	0.231	0.322	0.696
	[1077275] ⟹[1077274]	0.231	0.500	0.696
	[1128959] ⟹[1055228]	0.202	0.437	1.08e16
R1.5	[1134137] ⟹[1131537]	0.043	0.365	1.037e15
	[920286] ⟹[892027]	0.042	0.404	1.077e16
	[907786] ⟹[892027]	0.032	0.357	0.952
	[1011793] ⟹[892027]	0.024	0.300	0.800
	[1131480] ⟹[1035174]	0.011	0.397	1.648e14
R1.6	[1156672] ⟹[1156380]	0.056	0.405	1.76e16
	[1156205] ⟹[1156203]	0.044	0.306	1.710e15
	[1170160] ⟹[1156380]	0.043	0.310	1.131e16
	[1193028] ⟹[1136577]	0.037	0.369	1.613e16
	[1140107] ⟹[1179628]	0.012	0.361	1.249e16

Na "tabela 13" apresenta-se a tradução das regras com maior valor de confiança, das regras apresentadas na "tabela 12". Através da tradução das regras é possível observar a que produtos a associação se refere. Estas são as regras onde a relação entre os produtos é mais forte e como tal, onde é mais provável que haja uma relação de compra de produtos segundo os hábitos de compra dos consumidores.

Tabela 13 - Regras com maior confiança de cada modelo R1

Modelo	Regras	Suporte	Tradução
R1	[1217972] ⟹[1217973]	0.73	"FILE ALICHE ZAROTTI LT 48G" ⟹"FILE ALICHE ZAROTTI 90G"
R1.1	[1226582] ⟹[1226564]	0.328	"DOCE FIGO BOM DOCE KG" ⟹"DOCE ABOBORA C/COCO BOM DOCE KG"
R1.2	[1193364] ⟹[1164348]	0.726	"SANDUICHE ANG CROISSANT MINI KG" ⟹"HAMBURGUER ANG KG"
R1.3	[1170035] ⟹[960039]	0.42	"BOLO ANG NEGA MALUCA KG" ⟹"BOLO ANG CENOURA C/COBERTURA KG"
R1.4	[1055229] ⟹[1077274]	0.797	"CAFE EXPRESSO MEDIO (LANCH)" ⟹"CAFE EXPRESSO C/LEITE MEDIO (LANCH)"
R1.5	[920286] ⟹[892027]	0.404	OVO FRIOLAR BCO C/12" ⟹"OVO FRIOLAR VERM C/12"
R1.6	[1156672] ⟹[1156380]	0.405	"PAO PAGNIFIQUE CIABATTA RUSTICA IMPORTAD" ⟹"PAO PAGNIFIQUE MINI BAGUETE RUSTICO IMPO"

Analisando os valores de suporte como os de confiança, podemos verificar que o modelo R1.4 é o modelo que apresenta os melhores resultados. É este modelo que tem as regras com mais impacto nos dados, tal como as regras onde a associação entre os produtos é mais forte.

5.1.2 Os Modelos R2, R2.1, R2.2, R2.3, R2.4, R2.5

Os modelos R2, R2.1, R2.2, R2.3, R2.4, R2.5 foram implementados à semelhança dos modelos anteriores. Os modelos foram executados segundo os parâmetros apresentados na secção 4.2. Na "tabela 14" são apresentados os resultados gerais dessas execuções.

Tabela 14 - Resultados gerais dos modelos R2

Modelos	Nº de Registos	*Items* Frequentes	Regras Obtidas	Tempo (s)
R2	250 mil	171	26	34
R2.1	500 mil	168	29	81
R2.2	750 mil	163	24	143
R2.3	1 milhão	164	22	200
R2.4	1.25 milhões	159	21	269
R2.5	tudo	148	19	697

Analisando a tabela, podemos ver que o número de *items* frequentes encontrados são sempre números, sempre acima dos 150 *items*. É de notar que, para estes modelos, o valor de suporte usado foi 0.5%, e nestes casos já não houve situações onde o número de regras obtidas era baixo. O número de regras obtidas foi quase sempre superior a 20.

Tabela 15 - As 5 regras com maior suporte de cada modelo R2

Modelo	Regra	Suporte	Confiança	Lift
R2	[1077274] ⟹ [927862]	0.024	0.392	1.172e16
	[927862] ⟹ [1077274]	0.024	0.711	1.172e16
	[1156203] ⟹ [885335]	0.019	0.582	1.674e16
	[918887] ⟹ [885335]	0.018	0.428	1.23e16
	[922819] ⟹ [885335]	0.018	0.428	1.23e16
R2.1	[1077274] ⟹ [927862]	0.019	0.312	1.141e16
	[927862] ⟹ [1077274]	0.019	0.699	1.141e16
	[896673] ⟹ [885335]	0.019	0.448	1.297e16
	[1156203] ⟹ [885335]	0.019	0.587	1.7e15
	[918887] ⟹ [885335]	0.018	0.416	1.205e15
R2.2	[896673] ⟹ [885335]	0.023	0.439	1.295e16
	[1156203] ⟹ [885335]	0.019	0.585	1.726e16
	[1136577] ⟹ [885335]	0.018	0.436	1.286e16
	[918887] ⟹ [885335]	0.017	0.4	1.179e16
	[922819] ⟹ [885335]	0.016	0.423	1.248e16
R2.3	[896673] ⟹ [885335]	0.025	0.442	1.315e16
	[1156203] ⟹ [885335]	0.018	0.578	1.721e16
	[1136577] ⟹ [885335]	0.018	0.436	1.297e16
	[918887] ⟹ [885335]	0.016	0.395	1.176e16
	[922819] ⟹ [885335]	0.016	0.42	1.249e16
R2.4	[896673] ⟹ [885335]	0.026	0.443	1.311e16
	[1136577] ⟹ [885335]	0.019	0.440	1.301e16
	[1156203] ⟹ [885335]	0.019	0.578	1.710e16
	[918887] ⟹ [885335]	0.016	0.395	1.169e16
	[922819] ⟹ [885335]	0.016	0.418	1.238e15
R2.5	[896673] ⟹ [885335]	0.022	0.436	1.301e16
	[1136577] ⟹ [885335]	0.020	0.442	1.321e16
	[1156203] ⟹ [885335]	0.019	0.574	1.712e16
	[922819] ⟹ [885335]	0.015	0.412	1.23e16
	[918887] ⟹ [885335]	0.014	0.385	1.149e16

Tal como na secção anterior, foram selecionadas as 5 regras com maior suporte de todos os modelos da "tabela 15". Analisando a tabela é possível ver que os valores de suporte nunca são muito altos e vão sempre diminuindo ao longo dos modelos. Isto é facilmente explicável já que todos estes modelos

têm um grande número de transações e produtos. Conforme vamos avançando nos modelos, o número de transações vai aumentando e por isso é também normal que os valores de suporte diminuam. Com o aumento do número de transações e produtos, a variedade de produtos e o peso que eles vão ter nos dados, vai diminuindo. Os valores de confiança são relativamente constantes, sempre com valores a rondar os 0.5.

Tabela 16 - Regras com maior confiança de cada modelo R2

Modelo	Regras	Suporte	Tradução
R2	[927862] ⟹[1077274]	0.711	"SANDUICHE ANG COMUM KG" ⟹ "CAFE EXPRESSO C/LEITE MEDIO (LANCH)"
R2.1	[927862] ⟹[1077274]	0.699	"SANDUICHE ANG COMUM KG" ⟹ "CAFE EXPRESSO C/LEITE MEDIO (LANCH)"
R2.2	[1156203] ⟹[885335]	0.585	"PAO ANG C/FAROFA MINI BALCAO" ⟹"PAO ANG FRANCES KG"
R2.3	[1156203] ⟹[885335]	0.578	"PAO ANG C/FAROFA MINI BALCAO" ⟹"PAO ANG FRANCES KG"
R2.4	[1156203] ⟹[885335]	0.578	"PAO ANG C/FAROFA MINI BALCAO" ⟹"PAO ANG FRANCES KG"
R2.5	[1156203] ⟹[885335]	0.574	"PAO ANG C/FAROFA MINI BALCAO" ⟹"PAO ANG FRANCES KG"

Na "tabela 16" estão as regras com maior valor de suporte das regras da "tabela 15". Analisando a tabela, é fácil verificar que, ao longo dos vários modelos, duas regras sobressaem das outras. Estas duas regras são sempre as regras com maior suporte em todos os modelos. Se considerarmos que no modelo R2.5 é analisado a totalidade dos dados podemos considerar que a regra '"PAO ANG C/FAROFA MINI BALCAO" ⟹"PAO ANG FRANCES KG"' é uma regra com imensa relevância para o negócio desta empresa e como tal, contém uma relação de produtos com muito valor comercial.

5.1.3 Os Modelos K1, K1.1, K1.2, K1.3, K1.4

Os modelos K1, K1.1, K1.2, K1.3 e K1.4 foram implementados na ferramenta KEEL com o algoritmo MOPNAR (secção 4.3). Estes modelos foram implementados com os parâmetros já descritos na secção 4.3 e podemos ver na "tabela 17" os resultados gerais que o algoritmo nos devolve no fim da execução. Ao contrário com o que acontecia com os modelos anteriores, o tempo apresentado nesta tabela apenas diz respeito ao tempo de execução dos modelos na ferramenta e todo o processo de transformação dos dados não está incluído.

Tabela 17 - Resultados gerais dos modelos K1

Modelos	Categoria	Regras Obtidas	Suporte (m)	Confiança (m)	Tempo (s)
K1	Culinários	36	0.29	0.54	10
K1.1	Doces	39	0.12	0.49	95
K1.2	Sanduiches	31	0.25	0.76	104
K1.3	Bolos	30	0.14	0.64	372
K1.4	Cafés	43	0.28	0.91	222

Analisando a "tabela 17", conseguimos ver que o algoritmo conseguiu encontrar uma boa quantidade de regras de associação, para os vários modelos. Se compararmos estes resultados, nomeadamente os valores médios de suporte e confiança, com os valores dos modelos R1*, é possível observar que os valores são equiparáveis. Considerando que estes valores são de média, não é fácil fazer uma comparação direta mas é o suficiente para vermos que os valores de suporte nunca vão ser muito altos, tal como acontecia nos modelos do FP-Growth. Tal como aconteceu no modelo R1.4, com os produtos da categoria de cafés, foi também no modelo K1.4 que conseguimos extrair os melhores resultados do algoritmo MOPNAR. É de notar, que estes valores são valores médios e por isso as regras com os melhores valores de suporte e confiança podem não ter sido encontradas no modelo K1.4.

À semelhança dos modelos anteriores, na "tabela 18", podemos ver as cinco regras com o melhor valor de suporte. Ao contrários do algoritmo FP-Growth, como agora já estamos a trabalhar com a totalidade dos dados para cada característica e como estamos tanto a trabalhar com regras tanto positivas como negativas, isto vai exponenciar o número de relações e regras possíveis de obter. É graças a estes fatores que este algoritmo já é capaz de encontrar regras com um valor de suporte muito mais elevado. Analisando a tabela, é fácil de verificar que o modelo K1, com os produtos culinários, foi o que obteve as melhores regras tanto no valor de suporte como no de confiança. Podemos também ver que, com este algoritmo, conseguiu-se obter um maior número de regras em que o antecedente da regra é composto por mais do que um produto, ao contrário do que acontecia com o FP-Growth.

Tabela 18 - As 5 regras com maior suporte de cada modelo K1

Modelo	Regras				Suporte	Confiança	Lift
	Antecedentes	Valor	Conclusão	Valor			
K1	1217973 1222151	False Not True	1222153	Not True	0,98	1	1.01
	1217973 1222151 1222156	Not True False Not True	1222152	False	0,98	1	1.01
	1217972 1222152 1222153 1222156	False False Not True Not True	1222151	Not True	0,96	1	1.01
	1082670 1222153 1222156	Not True Not True Not True	1222151	Not True	0,94	1	1.01
	1082670 1222152 1222153 1222156 916236	Not True Not True Not True Not True Not True	1222151	Not True	0,9	1	1.01
K1.1	1015789	Not False	1221967	False	0,45	1	1.03
	1015789	Not False	1221960	Not True	0,45	1	1.04
	1015789	Not False	1226587	False	0,45	1	1.01
	1015789	Not False	1221977	False	0,45	1	1.05
	1015789 1221963	Not False False	1221960	Not True	0,45	1	1.04
K1.2	927862	Not False	1234467	False	0,37	1	1.01
	927862	Not False	1193364	Not True	0,37	1	1.13
	927862	Not False	1187188	False	0,37	1	1.01
	1189547 927862	False Not False	1234467	False	0,37	1	1.01
	1187188 927862	Not True Not False	1234467	False	0,37	1	1.01
K1.3	906752	False	1147056	False	0,96	1	1.01
	896503	True	960039	False	0,16	0.99	1.18
	960039	Not False	1220223	Not True	0,16	1	1.01
	960039	Not False	906752	False	0,16	1	1.05
	960039	Not False	881153	False	0,16	1	1.07
K1.4	1077274	Not False	1167928	False	0,66	1	1.01
	1077274	True	1055228	Not True	0,65	0.99	1.15
	1077274 1128959	Not False False	1055228	Not True	0,65	0.99	1.15
	1077274 1128848 1128959	Not False False Not True	1055228	False	0,65	0.99	1.15
	1077274	Not False	1077275	False	0,64	0.99	1.18

A partir da "tabela 18" encontramos vários casos onde a regra nega o valor de *True* ou *False*. Isto pode parecer estranho, já que o contrário de *True* é obviamente *False* e vice-versa. Depois de analisar os dados, verificou-se que o resultado estar "*True*" ou "*Not False*" tem o mesmo valor nos dados. A maneira como o algoritmo FP-Growth gera regras negativas é através da negação dos valores que estão nos dados. Logo, se os dados que o algoritmo recebe utilizam apenas valores binários, o processo do algoritmo criar regras negativas é negando esses valores, daí o algoritmo devolver regras com os valores de "*Not True*" ou "*Not False*". Da perspetiva do algoritmo, regras que contenham valores com negação, são regras negativas. Ainda que a regra "X Not False ⟹ Y Not False" para o algoritmo seja uma regra negativa é na prática uma regra positiva onde "X ⟹ Y".

A "tabela 19" contém a tradução das regras com maior valor de confiança das regras apresentadas na "tabela 18". A regra traduzida do modelo K1 é uma regra composta por mais do que um produto e diz-nos que quando o produto "FILE ALICHE ZAROTTI 90G" não está no carrinho, isto é, tem o valor "*False*", e quando o produto "MOLHO PARMESAO RANA PT 190G" também não está no carrinho ("*Not True*") então o produto "MOLHO FUNGHI RANA PT 190G" também nunca vai ser comprado.

Tabela 19 - Regras com maior confiança de cada modelo K1

Modelo	**Regra**		**Sup**	**Conf**
	Antecedente	Consequente		
K1	"FILE ALICHE ZAROTTI 90G" == False & "MOLHO PARMESAO RANA PT 190G" == Not True	"MOLHO FUNGHI RANA PT 190G" == Not True	0.98	1
K1.1	"GOIABADA PREDILECTA CASCAO FRAC KG" == Not False	"DOCE COLONIAL FORNO VELHO LARANJ PT 250G" == False	0.45	1
K1.2	"SANDUICHE ANG COMUM KG" == Not False	"MISTO QUENTE+CAFÉ C/LEITE+MUFFIN NUTELLA" == False	0.37	1
K1.3	"BOLO ANG CENOURA KG" == False	"BOLO ANG COCO F KG" == False	0.96	1
K1.4	"CAFE EXPRESSO C/LEITE MEDIO (LANCH)" == Not False	"CAFE ANG ORGANICO UN" == False	0.66	1

5.1.4 Os Modelos K2, K2.1, K2.2, K2.3, K2.4

Os modelos K2, K2.1, K2.2, K2.3 e K2.4 também foram implementados na ferramenta KEEL com o algoritmo MOPNAR (secção 4.4). Estes modelos foram implementados com os parâmetros já descritos anteriormente e podemos ver na "tabela 20" os resultados gerais que o algoritmo nos devolve, no fim da execução.

Tabela 20 - Resultados gerais dos modelos K2

Modelos	Categoria	Regras Obtidas	Suporte (m)	Confiança (m)	Tempo (s)
K2	Culinários	45	0.23	0.61	11
K2.1	Doces	29	0.2	0.71	102
K2.2	Sanduiches	40	0.23	0.78	106
K2.3	Bolos	35	0.18	0.71	374
K2.4	Cafés	45	0.29	0.91	239

Tal como nos outros modelos, nestes também foi possível encontrar uma boa quantidade de regras, agora também com as informações temporais do carrinho, nomeadamente o dia da semana e a hora da transação. Se compararmos estes valores obtidos com os anteriores verificamos que são muito equivalentes, apenas com umas pequenas variações justificadas pelo acréscimo dos atributos de data.

Na "tabela 21" estão as cinco regras com os melhores valores de suporte de cada modelo. É possível ver na tabela, a existência de regras que contêm informação não só de produtos, mas também com informações do dia de semana (*Week Day*) ou informações da hora (*Hours*).

Tabela 21 - As 5 regras com maior suporte de cada modelo K2

Modelo	**Regras**				**Suporte**	**Confiança**	***Lift***
	Antecedentes	Valor	Conclusão	Valor			
K2	1222151 973162	False Not True	1222153	False	0,98	1	1.01
	1217972 1222151 1222156	False False False	1222152	FALSE	0,96	1	1.01
	Hours	Not [19, 23]	1222152	False	0,78	1	1.01
	1222153 Hours	False Not [18, 23]	1222152	False	0,7	1	1.01
	1222151 931353 Hours	Not True False Not [19, 23]	1222152	False	0,7	1	1.01
K2.1	1184024 1188307 1221965 Hours	False False False Not [2, 13]	1188304	False	0,76	1	1.01
	1184000 1188305 1336591 Hours	False False False Not [19, 16]	1188311	False	0,59	1	1.01
	1015789	True	1206293	False	0,45	1	1.03
	1015789	True	1226564	False	0,45	1	1.02
	1015789	True	1221960	False	0,45	1	1.04
K2.2	927862	Not False	1235781	False	0,72	1	1.01
	927862	Not False	1187188	False	0,37	1	1.01
	Hours	Not [15, 18]	1228159	False	0,37	1	1.01
	927862	Not False	1193364	False	0,37	1	1.13
	1133102 927862	False Not False	1234467	False	0,37	1	1.01
K2.3	Week Day	Not [3, 3]	1150234	False	0,88	1	1.01
	Week Day	Not [3, 4]	1220225	False	0,77	1	1.01
	896501 Week Day	False Not [4, 6]	1220225	False	0,6	1	1.01
	896503	Not False	1226688	False	0,16	1	1.01
	960039	Not False	999181	False	0,16	1	1.01
K2.4	1077274	Not False	1167928	False	0,66	1	1.01
	1077274	Not False	1055228	Not True	0,65	0.99	1.15
	1077274 1128959	Not False False	1055228	False	0,65	0.99	1.15
	1077274 1129948 1128959	Not False False Not False	1055228	False	0,65	0.99	1.15
	1077274	TRUE	1077275	False	0,64	0.99	1.18

Na “tabela 22” é possível ver a tradução das regras com o maior valor de confiança das regras apresentadas na “tabela 21”. Um exemplo de uma regra com informações de data, é a regra do modelo K2.3, onde, quando o dia da semana não é igual a 3, isto é, não é segunda-feira, o produto “BOLO ANG ABOBORA C/COCO KG”, o produto nunca está presente na transação, já que confiança é 100%.

Tabela 22 - Regras com maior confiança de cada modelo K2

Modelo	**Regra**		**Sup**	**Conf**
	Antecedente	Consequente		
K2	“MOLHO PARMESAO RANA PT 190G” == False & “FILE ALICHE MELLILA OLEO KG” == Not True	“MOLHO FUNGHI RANA PT 190G” == Not True	0.98	1
K2.1	“DOCE RESERVA MINAS LARANJA CALDA KG” == False & “DOCE LEITE PURO DOCE ROCA KG” == False & “DOCE COLONIAL FORNO VELHO MORANG PT 250G” == False & “Hours” == Not [2, 13]	“DOCE ABOBORA C/COCO DOCE ROCA KG” == False	0.76	1
K2.2	“SANDUICHE ANG COMUM KG” == Not False	“MISTO QUENTE+CAFE C/LEITE+MUFIN GOT CHOC” == False	0.72	1
K2.3	“Week Day” == Not [3, 3]	“BOLO ANG ABOBORA C/COCO KG” == False	0.88	1
K2.4	“CAFE EXPRESSO C/LEITE MEDIO (LANCH)” == Not False	“CAFE ANG ORGANICO UN” == False	0.65	1

5.2 Análise de Desempenho

Para a execução de todos os modelos deste caso de estudo foi utilizada uma máquina com as seguintes características: processador Intel Core i5-4670 3.4GHz 6MB, memória RAM 2x8GB DDR3-1600Mhz e sistema operativo Windows 7 com o *Service Pack* 1.

Os Modelos R1, K1 e K2

Os modelos R1, K1 e K2 são comparáveis entre si já que todos eles analisam os dados das mesmas categorias de produtos. Todos estes modelos analisam as categorias “Culinários”, “Doces”, “Sanduiches”, “Bolos” e “Cafés”. A “tabela 6” e a “tabela 8”, na secção 4.1 e 4.3, mostram as características dos modelos R1, e K1 e K2, respetivamente. Os modelos R1, uma vez que não têm transações com apenas um produto, são modelos com uma quantidade de dados consideravelmente mais pequenos.

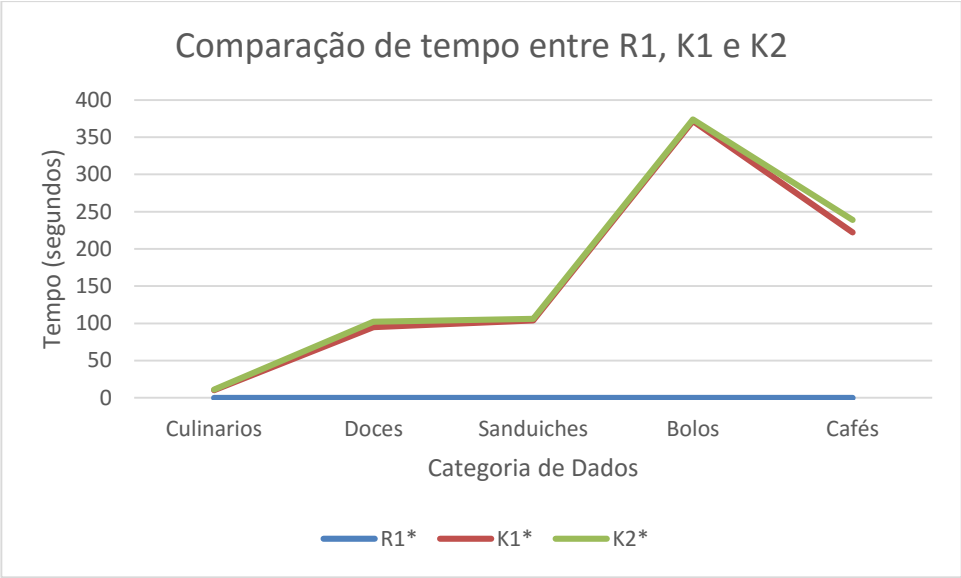


Figura 18 - Comparação de tempo entre R1, K1 e K2

Analisando o gráfico da Figura 18, podemos facilmente ver que os modelos R1, isto é, os modelos do algoritmo FP-Growth que utilizam a ferramenta RapidMiner são sempre mais rápidos do que os modelos K1 e K2 que usam o algoritmo MOPNAR na ferramenta KEEL. O facto dos modelos R1 serem substancialmente mais rápidos deve-se ao facto de que o algoritmo dos modelos R1 analisar apenas regras positivas, um processo consideravelmente menos complexo e menos exigente do que analisar regras negativas e por isso mais rápido de executar. O facto dos dados só possuírem transações com mais de um produto não é um fator importante nesta comparação, já que, caso fosse utilizada a totalidade dos dados, à semelhança dos modelos K1 e K2, o tempo de execução manter-se-ia baixo. No caso onde o processo é mais demorado, (produtos da categoria Bolos) mesmo que seja executado com a totalidade das transações, tem uma duração de 4 segundos, bastante inferior comparativamente aos outros modelos.

Entre os modelos K1 e K2, existe sempre uma diferença muito baixa entre todos os casos. Os modelos K2 são sempre mais lentos mas apenas alguns segundos. Uma vez que a diferença é tão pequena, podemos concluir que, acrescentar os atributos de data nos modelos K2, não cria um impacto significativo no tempo de execução dos modelos.

Os Modelos R2

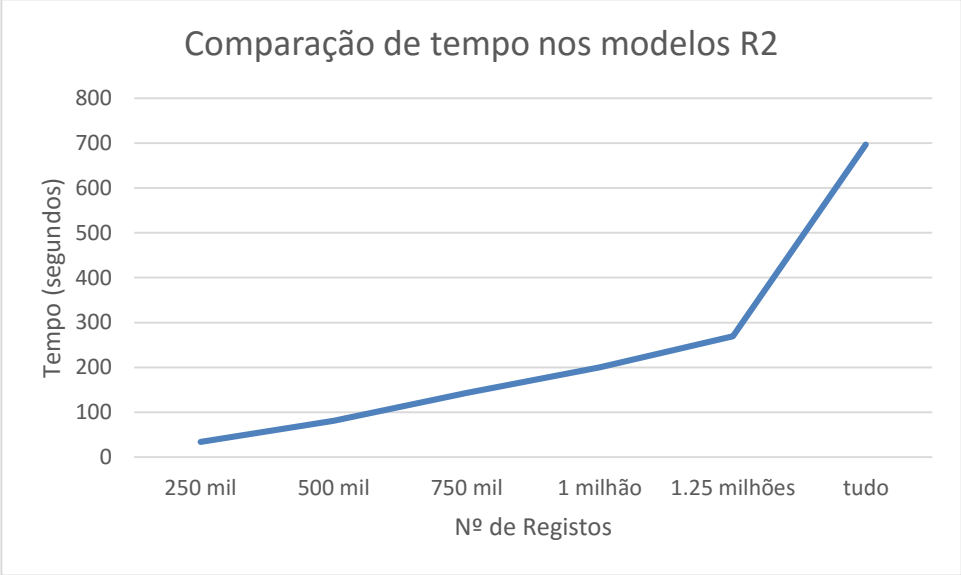


Figura 19 - Comparação de tempo nos modelos R2

Analisando o gráfico da Figura 19, que comprara os tempos de execução dos seis modelos R2, podemos ver que a evolução do tempo de execução dos modelos é linear com o aumento do número de registos. O último modelo tem um aumento considerável no tempo de execução já que contém a totalidade dos dados da base de dados.

A "tabela 7", da secção 4.2, tem o número de transações e número de produtos que cada um destes modelos contém, podendo ver-se que estes modelos têm quantidades de transações e de produtos muito superiores aos modelos anteriores. Mesmo com o grande aumento na quantidade de dados processados, o algoritmo apresenta tempos de execução bastante satisfatórios. É de notar que estes modelos foram bastante exigentes em termos de *hardware*. Nos modelos com um milhão de registos ou mais, a utilização de memória RAM ultrapassou os 12GB, fazendo deste grupo de modelos os mais exigentes na máquina.

Os Modelos K3 e K4

Tal como referi na secção 4.5, os modelos K3 e K4 foram criados para testar a influência do número de produtos processados (K3) e o número de registos (K4). Todos estes modelos foram criados a partir do modelo K1.3, que contém os produtos da categoria "Bolos", e por isso foram também comparados com este modelo.

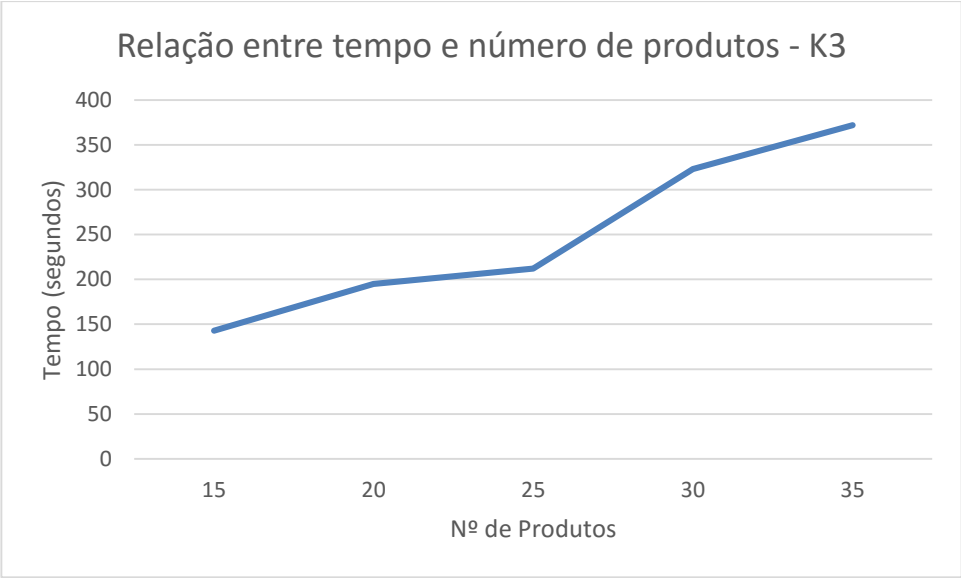


Figura 20 - Relação entre tempo e número de produtos - K3

O gráfico da Figura 20 mostra o tempo de execução do modelo K1.3, originalmente com 35 produtos diferentes, quando é reduzido o número de produtos analisados. O último modelo apresentado no gráfico (35 produtos) representa o tempo de execução obtido do modelo K1.3. Os restantes modelos foram criados a partir do modelo K1.3, mas com as diferentes reduções no número de produtos processados. Analisando o gráfico, é possível ver que os vários modelos mostram uma evolução linear no tempo de execução. O algoritmo não aparenta ter um aumento temporal, além de um aumento linear, para execução de modelos com as mesmas características mas com um aumento no número de produtos/atributos.

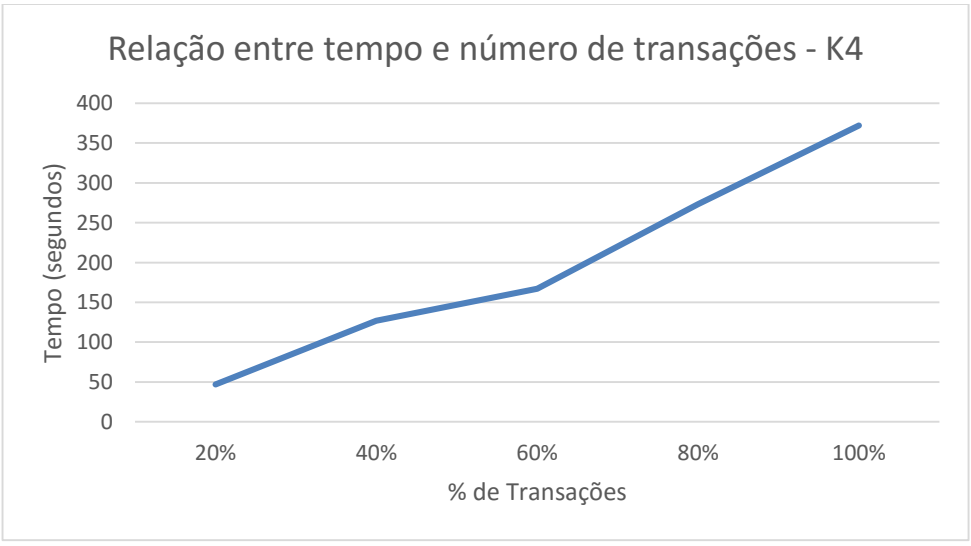


Figura 21 - Relação entre tempo e número de registos - K4

O gráfico da Figura 21 mostra-nos o tempo de execução do modelo K1.3 quando se reduz o número de transações processadas. O modelo K1.3 contém 86642 transações, enquanto que os restantes modelos K4 contêm apenas uma percentagem dessas transações ("tabela 11", secção 4.5). Tal como na situação anterior, o algoritmo não mostra ter nenhum grande aumento ou diminuição no tempo de execução, sendo que os vários modelos mostram uma evolução linear quanto ao tempo necessário para analisar os dados.

Ainda que muitos dos modelos testados precisassem de alguns minutos para conseguir analisar os dados, problemas deste tipo não são geralmente problemas onde é necessário obter resultados instantaneamente. Normalmente, situações onde é necessário descobrir relações entre produtos para a criação, por exemplo, de uma campanha promocional, são processos onde são analisados vários fatores, além do relacionamento entre produtos, e por isso, necessitam de algum tempo até estarem prontos. Em situações como estas, é normal algoritmos desta natureza serem executados durante períodos noturnos ou durante o fim-de-semana, por terem de analisar grandes quantidades de dados e por terem grandes exigências computacionais e temporais. Por estas razões, os tempos de execução apresentados são bastante satisfatórios.

Capítulo 6

6. Conclusões e Trabalho Futuro

6.1 Conclusões

A existência atual de grandes repositórios de informação, largamente utilizados pela grande maioria dos ramos de negócios, leva à necessidade da criação de mecanismos sofisticados e eficientes que consigam analisar e extrair conhecimento desses repositórios de informação. Com o grande desenvolvimento tecnológico que temos vindo a presenciar nas últimas décadas, é cada vez mais fácil, para qualquer negócio, armazenar grandes volumes de informação. Estes grandes volumes de informação levam-nos à necessidade da implementação de técnicas que tornem possível a extração de conhecimento, que de outro modo seria humanamente impossível. Para isso, nos últimos anos, temos assistido a um grande aumento no número de técnicas de *data mining*, que vêm solucionar o problema da extração de conhecimento de grandes bases de dados.

Isto é muito real na área do retalho onde, por exemplo, os registos das transações efetuadas por clientes são armazenadas e têm um grande valor comercial. Conseguir descobrir padrões relativos aos hábitos de compra dos clientes, tem um grande valor de negócio, já que com esse conhecimento, é possível adaptar o método de venda dos produtos de modo a alinhar a venda dos mesmos com as tendências de compra dos consumidores.

Neste trabalho, o que se pretendia, era encontrar dois algoritmos e testá-los numa base de dados de registos de compras reais de um retalhista, para conseguir extrair tanto regras de associação positivas como negativas. Os algoritmos escolhidos foram o FP-Growth, através da ferramenta RapidMiner para extrair apenas regras positivas e o algoritmo MOPNAR através da ferramenta KEEL para extrair tanto regras positivas como negativas.

Antes dos algoritmos serem aplicados foi necessário fazer uma limpeza de tratamento dos dados disponíveis. Foram selecionados vários grupos de produtos com características diferentes de modo a analisar situações diferentes pelos dois algoritmos. Os dois algoritmos foram assim aplicados a vários modelos que diferiam entre si pela quantidade de dados ou pelo tipo de produtos. Ambos os algoritmos conseguiram devolver resultados satisfatórios e interessantes. O algoritmo FP-Growth mostrou que, em certos casos, quando se analisam os dados referentes a um grupo específico de produtos, pode não ser possível encontrar-se um grande número de regras. Isto já não acontecia com o algoritmo MOPNAR. O facto de conseguir extrair tanto regras positivas como negativas, aliadas ao facto do algoritmo ser capaz de, analisando os dados, definir autonomamente os melhores valores de suporte e confiança para o problema, permite ao algoritmo conseguir encontrar um maior número de regras para as situações onde um grupo mais restrito dos dados é analisado. Em compensação, o algoritmo FP-Growth por ser largamente menos exigente em termos computacionais, permitindo fazer análises a quantidades de dados largamente superiores àquelas que o MOPNAR consegue analisar.

Depois de analisados os resultados dos dois algoritmos, tanto das regras obtidas como dos tempos de execução, conseguiu-se concluir que os dois algoritmos aplicam-se melhor em situações distintas. O algoritmo MOPNAR, por conseguir extrair regras positivas assim como negativas e por necessitar de um maior poder computacional, aplica-se melhor em análises a grupos mais restritos de dados. Análises como as que foram efetuadas, com o objetivo de descobrir relações entre produtos numa categoria específica de produtos ou para descobrir relações entre produtos de duas categorias de produtos diferentes, é onde o algoritmo mostra melhores resultados e desempenhos. Situações onde o volume de dados era maior mostraram-se demasiado exigentes para o algoritmo.

O algoritmo FP-Growth, principalmente por ser consideravelmente menos exigente do que o MOPNAR, aplica-se melhor em análises de grande escala. Tal como foi visto, o FP-Growth consegue analisar cerca de 450 mil transações com perto de 700 produtos, sensivelmente no mesmo tempo que o

MOPNAR analisa cerca de 86 mil transações com 35 produtos. Ainda assim, isto não é em nada impeditivo de aplicar este algoritmo em casos mais pequenos, como análises a categorias específicas de produtos.

Concluindo ainda que estes dois algoritmos embora bastante diferentes, tanto nos resultados como nas exigências, os dois complementam-se um ao outro. Casos onde um não mostra resultados tão bons é onde o outro se mostra superior. Concluiu-se que estes dois algoritmos podem perfeitamente ser usados em paralelo, para assim conseguir analisar e encontrar regras de associação em casos diferentes do mesmo negócio.

6.2 Trabalho Futuro

Depois de terminado este projeto, alguns aspetos podem ser considerados para serem trabalhados no futuro. Relativamente ao algoritmo MOPNAR seria interessante alargar o estudo feito e considerar um estudo onde outras características da transação são consideradas em vez de uma análise apenas aos produtos transacionados. Um estudo com isto em mente poderia analisar características da transação e também do próprio cliente.

Sobre todo o processo, desde a extração dos dados passando pela fase de mineração e por fim a análise de resultados, constatou-se que não é um ato único mas sim um processo com várias etapas distintas. Seria interessante a criação de um sistema onde o utilizador pudesse selecionar os dados pretendidos, escolher os parâmetros para o algoritmo e por fim visualizar os resultados, criando assim um processo automático e mais direto.

BIBLIOGRAFIA

Abbas, W.F., Ahmad, N.D. & Zaini, N.B., 2013. Discovering Purchasing Pattern of Sport Items Using Market Basket Analysis. *2013 International Conference on Advanced Computer Science Applications and Technologies*, pp.120–125. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6836560 [Accessed November 26, 2014].

Agrawal, J. et al., 2014. SET-PSO-based approach for mining positive and negative association rules. *Knowledge and Information Systems*. Available at: http://link.springer.com/10.1007/s10115-014-0795-2 [Accessed November 26, 2014].

Agrawal, R., Imielinski, T. & Swami, A., 1993. Database mining: A performance perspective. *IEEE Transactions on, 1993*, pp.1–22. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=250074.

Agrawal, R., Imieliński, T. & Swami, A., 1993. Mining association rules between sets of items in large databases. In *1993 ACM SIGMOD International Conference on Management of Data*. pp. 207–216. Available at: http://dl.acm.org/citation.cfm?id=170072 [Accessed December 3, 2014].

Agrawal, R. & Srikant, R., 1994. Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 487–499. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.7506.

Alataş, B. & Akin, E., 2006. An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules. *Soft Computing*, 10, pp.230–237.

Alcalá-Fdez, J. et al., 2008. KEEL : a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3), pp.307–318.

Alcalá-Fdez, J. et al., 2011. KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2-3), pp.255–287. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-79951829331&partnerID=tZOtx3y1.

Alguliev, R.M., Aliguliyev, R.M. & Nazirova, S.A., 2011. Classification of Textual E-Mail Spam Using Data Mining Techniques. *Applied Computational Intelligence and Soft Computing*, 2011, pp.1–8. Available at: http://www.hindawi.com/journals/acisc/2011/416308/.

Azevedo, P.J. & Jorge, A.M., 2007. Comparing Rule Measures for Predictive Association Rules. In *Proceedings of the 18th European Conference on Machine Learning*. ECML '07. Berlin, Heidelberg: Springer-Verlag, pp. 510–517. Available at: http://dx.doi.org/10.1007/978-3-540-74958-5_47.

Basuchowdhuri, P., Shekhawat, M.K. & Saha, S.K., 2014. Analysis of Product Purchase Patterns in a Co-Purchase Network. *Emerging Applications of Information Technology (EAIT), 2014 Fourth International Conference of*, pp.355–360.

Bayardo, R.J., 1998. Efficiently mining long patterns from databases. *ACM SIGMOD Record*, 27(2), pp.85–93. Available at: http://cs.sungshin.ac.kr/~de/Seminar/ps_files/sigmod98_max.pdf.

Brin, S., Motwani, R., Ullman, J.D., et al., 1997. Dynamic itemset counting and implication rules for market basket data. *ACM SIGMOD Record*, 26(2), pp.255–264.

Brin, S., Motwani, R. & Silverstein, C., 1997. Beyond Market Baskets: Generalizing Association Rules to Correlations. *SIGMOD Rec.*, 26(2), pp.265–276. Available at: http://doi.acm.org/10.1145/253262.253327.

Burton A. Leland et al., 1997. Managing the Combinatorial Explosion. *Journal of Chemical Information and Computer Sciences*, 37(1), pp.62–70. Available at: http://dx.doi.org/10.1021/ci960088t.

ButleAnalytics, 2015. RapidMiner Review. *butleranalytics.com*. Available at: http://butleranalytics.com/rapidminer-6-review/.

Chandola, V., Banerjee, A. & Kumar, V., 2009. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(September), pp.1–58. Available at: http://portal.acm.org/citation.cfm?id=1541882\nhttp://dl.acm.org/citation.cfm?id=1541882\nhttp://portal.acm.org/citation.cfm?doid=1541880.1541882.

Chandola, V. & Kumar, V., 2005. Summarization - Compressing data into an informative representation. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 12, pp.98–105.

Chapman, P. et al., 2000. CRISP-DM 1.0 Step-by-step data mining guide. Available at: ftp://ftp.software.ibm.com/software/analytics/spss/support/Modeler/Documentation/14/UserManual/CRISP-DM.pdf [Accessed September 4, 2015].

Chapman, P., 1999. The CRISP-DM User Guide. Available at: http://lyle.smu.edu/~mhd/8331f03/crisp.pdf [Accessed September 4, 2015].

Chen, Y.-L. et al., 2005. Market basket analysis in a multiple store environment. *Decision Support Systems*, 40(2), pp.339–354. Available at:

http://linkinghub.elsevier.com/retrieve/pii/S0167923604000685.

Crone, S.F., Lessmann, S. & Stahlbock, R., 2006. The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing. *European Journal of Operational Research*, 173(3), pp.781–800. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0377221705006739.

Dalton, L., Ballarin, V. & Brun, M., 2009. Clustering algorithms: on learning, validation, performance, and applications to genomics. *Current genomics*, 10(6), pp.430–445.

Deutsch, G., 2010. RapidMiner from Rapid-I at CeBIT 2010. *Data Mining Blog*. Available at: http://www.data-mining-blog.com/cloud-mining/rapidminer-cebit-2010/ [Accessed November 12, 2015].

Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P., 1996. From data mining to knowledge discovery in databases. *AI magazine*, 17(3), pp.37–54. Available at: http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/1230 [Accessed December 10, 2014].

Fidelis, M. V, Lopes, H.S. & Freitas, A.A., 2000. Discovering comprehensible classification rules with a genetic algorithm. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*. pp. 805–810 vol.1.

Gu, H. & Lin, C., 2012. Application of data warehouse techniques in retail trade. *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, (Fskd), pp.2594–2597. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6233966.

Gupta, S., Kumar, D. & Sharma, A., 2011. Data Mining Classification Techniques Applied for Breast Cancer Diagnosis and Prognosis. *Journal of Computer Science*, 2(May), pp.188–195.

Gutierrez, N., 2006. Demystifying Market Basket Analysis. *Information Management*. Available at: http://www.information-management.com/specialreports/20061031/1067598-1.html [Accessed September 15, 2015].

Hahsler, M., 2015. A Probabilistic Comparison of Commonly Used Interest Measures for Association Rules.

Hájek, P., Havel, I. & Chytil, M., 1966. The GUHA method of automatic hypotheses determination. *Computing*, 1(4), pp.293–308.

Hájek, P., Holeňa, M. & Rauch, J., 2010. The GUHA method and its meaning for data mining. *Journal of Computer and System Sciences*, 76(1), pp.34–48.

Han, J. & Kamber, M., 2001. *Data mining: concepts and techniques*, Morgan Kaufmann Publishers.

Han, J., Pei, J. & Yin, Y., 2000. Mining Frequent Patterns Without Candidate Generation. *SIGMOD Rec.*, 29(2), pp.1–12. Available at: http://doi.acm.org/10.1145/335191.335372.

Hassan, S. et al., 2013. Bioprocess data mining using regularized regression and random forests. *BMC systems biology*, 7 Suppl 1. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-84902252614&partnerID=tZOtx3y1.

Haughton, D. et al., 2003. A Review of Software Packages for Data Mining. *The American Statistician*, 57(4), pp.290–309. Available at: http://www.tandfonline.com/doi/abs/10.1198/0003130032486.

Hodge, V.J. & Austin, J., 2004. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22(2004), pp.85–126.

Hofmann, M. & Klinkenberg, R., 2013. *RapidMiner: Data Mining Use Cases and Business Analytics Applications*, Chapman & Hall/CRC.

Hornick, M.F., Marcadé, E. & Venkayala, S., 2007. *Java Data Mining: Strategy, Standard, and Practice: A Practical Guide for Architecture, Design, and Implementation*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hu, M. & Liu, B., 2004. Mining and summarizing customer reviews. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining KDD 04*, 04, p.168. Available at: http://portal.acm.org/citation.cfm?doid=1014052.1014073.

Hu, X., 2003. DB-HReduction: A data preprocessing algorithm for data mining applications. *Applied Mathematics Letters*, 16(6), pp.889–895. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0893965903900139.

IBM, CRISP DM 1.0 - Step-by-step data mining guide. Available at: http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=SA&subtype=WH&htmlfid=YTW03084USEN [Accessed September 1, 2015].

Jiang, H., Luan, X. & Dong, X., 2012. Mining Weighted Negative Association Rules from Infrequent Itemsets Based on Multiple Supports. *2012 International Conference on Industrial Control and Electronics Engineering*, pp.89–92. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6322321.

Kanungo, T. et al., 2002. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7), pp.881–892. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-0036647190&partnerID=tZOtx3y1 [Accessed August 24, 2015].

KDnuggets, 2014. CRISP-DM, still the top methodology for analytics, data mining, or data science projects. Available at: http://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html [Accessed September 3, 2015].

KDnuggets, 2007. Free Data Mining Software: RapidMiner 4.0 (formerly YALE). Available at: http://www.kdnuggets.com/news/2007/n15/8i.html [Accessed November 12, 2015].

KDnuggets, 2010. Interview with RapidMiner's Ingo Mierswa, Ralf Klinkenberg. Available at: http://www.kdnuggets.com/2010/02/f-interview-rapid-i-founders.html [Accessed November 12, 2015].

Kesavaraj, G. & Sukumaran, S., 2013. A study on classification techniques in data mining. *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pp.1–7. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6726842.

Leskovec, J., Adamic, L.A. & Huberman, B.A., 2007. The dynamics of viral marketing. *ACM Transactions on the Web*, 1(1), p.5–es. Available at: http://portal.acm.org/citation.cfm?doid=1232722.1232727.

Li, H.L.H. & Zhang, Q.Z.Q., 2009. Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2), pp.284–302.

Martin, D. et al., 2014. A New Multi-Objective Evolutionary Algorithm for Mining a Reduced Set of Interesting Positive and Negative Quantitative Association Rules. *IEEE Transactions on Evolutionary Computation*, 18(1), pp.54–69. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6626591 [Accessed December 17, 2014].

Oestreicher-Singer, G. & Sundararajan, A., 2006. Linking Network Structure to Ecommerce Demand: Theory and Evidence from Amazon.Com's Copurchase Network. *TPRC 2006. Available at SSRN*.

Piatetsky-Shapiro, G., 1991. Discovery, Analysis and Presentation of Strong Rules. In G. Piatetsky-Shapiro & W. J. Frawley, eds. *Knowledge Discovery in Databases*. Menlo Park, California: AAAI Press / The MIT Press, pp. 229–248.

Poundekar, M. et al., 2014. Mining Strong Valid Association Rule from Frequent Pattern and Infrequent Pattern Based on Min-Max Sinc Constraints. *2014 Fourth International Conference on Communication Systems and Network Technologies*, pp.450–453. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6821436 [Accessed December 11, 2014].

Power, D.J., 2002. Newsletter 66. Available at: http://www.dssresources.com/newsletters/66.php

[Accessed September 5, 2015].

Rai, N.S., Jain, S. & Jain, A., 2014. Mining Interesting Positive and Negative Association Rule Based on Improved Genetic Algorithm (MIPNAR_GA). *International Journal of Advanced Computer Science and Applications (IJACSA)*, 5(1), pp.160–165.

Rajak, A. & Gupta, M.K., 2008. Association Rule Mining: Applications in Various Areas. *Proceedings of International Conference on Data Management February 25-26*, (November 2015), pp.3–7.

RapidMiner, 2015a. Create Association Rules. Available at: http://docs.rapidminer.com/studio/operators/modeling/associations/create_association_rules.html [Accessed November 1, 2015].

RapidMiner, 2015b. FP-Growth. Available at: http://docs.rapidminer.com/studio/operators/modeling/associations/fp_growth.html [Accessed November 1, 2015].

RapidMiner, The core of RapidMiner is open source. Available at: https://rapidminer.com/the-core-of-rapidminer-is-open-source/ [Accessed October 12, 2015].

Shearer, C., 2006. First CRISP-DM 2.0 Workshop Held. Available at: http://www.kdnuggets.com/news/2006/n19/4i.html [Accessed September 4, 2015].

Shortliffe, E.H. & Buchanan, B.G., 1990. A model of inexact reasoning in medicine. In G. Shafer & J. Pearl, eds. *Readings in uncertain reasoning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 259–275. Available at: http://dl.acm.org/citation.cfm?id=84628.85330.

Silverstein, C., Brin, S. & Motwani, R., 1998. Beyond Market Baskets: Generalizing Association Rules to Dependence Rules. *Data Min. Knowl. Discov.*, 2(1), pp.39–68. Available at: http://dx.doi.org/10.1023/A:1009713703947.

Skiena, S.S., 2008. *The Algorithm Design Manual*, Springer London.

Tan, P.-N., Steinbach, M. & Kumar, V., 2005. *Introduction to Data Mining, (First Edition)*, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Taniar, D. et al., 2012. Mining Hierarchical Negative Association Rules. *International Journal of Computational Intelligence Systems*, 6891(November 2014), pp.434–451.

Videla-Cavieres, I.F. & Ríos, S. a., 2014. Extending market basket analysis with graph mining techniques: A real case. *Expert Systems with Applications*, 41(4), pp.1928–1936. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0957417413007094 [Accessed November 20, 2014].

Vindevogel, B., Poel, D. VanDen & Wets, G., 2005. Why promotion strategies based on market basket

analysis do not work. *Expert Systems with Applications*, 28(3)(3), pp.583–590.

Wu, X., Zhang, C. & Zhang, S., 2004. Efficient mining of both positive and negative association rules. *ACM Transactions on Information Systems*, 22(3), pp.381–405.

Zaki, M.J. et al., 1997. New algorithms for fast discovery of association rules. *Kdd*, 7, pp.283–286. Available at: http://www.aaai.org/Papers/KDD/1997/KDD97-060.pdf.

Zaki, M.J., Parthasarathy, S. & Li, W., 1997. A Localized Algorithm for Parallel Association Mining. In *Proceedings of the Ninth Annual ACM Symposium on Parallel Algorithms and Architectures*. SPAA '97. New York, NY, USA: ACM, pp. 321–330. Available at: http://doi.acm.org/10.1145/258492.258524.

ANEXO I – LISTA DE ATRIBUTOS DA BASE DE DADOS

Atributo	**Descrisão**
SALES_DATE_KEY	Identificador único da Dia
SALES_WEEK_KEY	Identificador único da Semana
SALES_HOUR_KEY	Identificador único da Hora
PROD_KEY	Identificador único do Produto
PROD_DESC	Descrição do Produto
MERCH_L1_KEY	Identificador único do Nível 1 da Estrutura Mercadológica
MERCH_L1_DESC	Descrição do Nível 1 da Estrutura Mercadológica
MERCH_L2_KEY	Identificador único do Nível 2 da Estrutura Mercadológica
MERCH_L2_DESC	Descrição do Nível 2 da Estrutura Mercadológica
MERCH_L3_KEY	Identificador único do Nível 3 da Estrutura Mercadológica
MERCH_L3_DESC	Descrição do Nível 3 da Estrutura Mercadológica
MERCH_L4_KEY	Identificador único do Nível 4 da Estrutura Mercadológica
MERCH_L4_DESC	Descrição do Nível 4 da Estrutura Mercadológica
MERCH_L5_KEY	Identificador único do Nível 5 da Estrutura Mercadológica
MERCH_L5_DESC	Descrição do Nível 5 da Estrutura Mercadológica
MERCH_L6_KEY	Identificador único do Nível 6 da Estrutura Mercadológica
MERCH_L6_DESC	Descrição do Nível 6 da Estrutura Mercadológica
MERCH_L7_KEY	Identificador único do Nível 7 da Estrutura Mercadológica
MERCH_L7_DESC	Descrição do Nível 7 da Estrutura Mercadológica
LOC_KEY	Identificador único da Loja
ORIGIN_KEY	Origem (sempre 1)
CURRENCY_KEY	Identificador único da Moeda
OFFER_KEY	Identificador único da Promoção (se -1, não existia Promoção)
AD_MODE_TYPE_KEY	Identificador único do tipo de publicidade utilizada para divulgar a Promoção
PROMO_MOTIVE_KEY	Identificador único do motivo da Promoção
TRAN_TYPE_KEY	N/A
CUST_KEY	Identificador único do Cliente
REGISTER_KEY	Identificador único do POS onde foi registada a Transação
EMPLOYEE_KEY	Identificador único do Funcionário
CHANNEL_KEY	Identificador único do Canal de Venda
COUPON_ID	Identificador do Coupon
TRAN_QTY	Quantidade Vendida
TRAN_VAL	Quantidade Vendida em Valor
VAT_01_ID	Informações de Impostos
VAT_01_VAL	Informações de Impostos
VAT_02_ID	Informações de Impostos
VAT_02_VAL	Informações de Impostos
VAT_03_ID	Informações de Impostos
VAT_03_VAL	Informações de Impostos
VAT_04_ID	Informações de Impostos

VAT_04_VAL	Informações de Impostos
VAT_05_ID	Informações de Impostos
VAT_05_VAL	Informações de Impostos
VAT_06_ID	Informações de Impostos
VAT_06_VAL	Informações de Impostos
ATTR_01_CHAR	Informações de Impostos
ATTR_02_CHAR	Informações de Impostos
ATTR_03_CHAR	Informações de Impostos
ATTR_04_CHAR	Informações de Impostos
ATTR_01_NO	Informações de Impostos
ATTR_02_NO	Informações de Impostos
ATTR_03_NO	Informações de Impostos
ATTR_04_NO	Informações de Impostos
RCD_LAST_UPD_DATETIME	N/A
RCD_LAST_INTEGRATION_DATETIME	N/A
COUPON_SEQ_ID	N/A
OFFER_TYPE_KEY	Identificador único do Tipo de Oferta (%Off, Value OF, Buy One – Get One Free, ...)
CALENDARIO	Dia
WEEK_DAY	Dia da semana