NOVA IMS Information Management School

MGI

Mestrado em Gestão de Informação
Master Program in Information Management

Big Data Analytics

Predictive Consumer Behaviour Analysis

António Alberto Legoinha Vilares

Trabalho de Projeto apresentado como requisito parcial para obtenção do grau de Mestre em Gestão de Informação

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

2016

Big Data Analytics

Predictive Consumer Behaviour Analysis

António Alberto Legoinha Vilares

MGI

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

BIG DATA ANALYTICS

PREDICTIVE CONSUMER BEHAVIOUR ANALYSIS

por

António Legoinha Vilares

Trabalho de Projeto apresentado como requisito parcial para a obtenção do grau de Mestre em Gestão de Informação, Especialização em Gestão do Conhecimento e *Business Intelligence*

Orientador: Professor Roberto Henriques, PhD

Co-Orientador: Professor Rui Rosa, MsC

AGRADECIMENTOS

O desafio de terminar uma tese de mestrado reúne contributos de várias pessoas que ao longo destes tempos me foram ajudando ao nível pessoal e profissional. Assim, agradeço:

Ao meu orientador de tese, o Professor Roberto Henriques, pelo apoio, partilha do saber e as valiosas sugestões que me foi dando ao longo do trabalho. Obrigado pelo incentivo!

Ao meu co-orientador, o professor Rui Rosa, que prescindiu do seu tempo pessoal para me acompanhar nos "inúmeros almoços nas Amoreiras" e que em muito me incentivou para o sucesso deste projeto. A sua larga experiência e capacidade de ajuda foram um recurso essencial para decisões que se me apresentavam complexas. Obrigado por ter acreditado em mim!

À Deloitte pela proposta do tema e motivação para a utilização de ferramentas *Big Data* no projeto.

Aos meus grandes amigos David, Hugo, António e Rui pela paciência e por falhar a alguns encontros. Mesmo assim, estiveram sempre presentes e mostraram que temos amizade para a vida!

Ao meu amigo e companheiro de sábados, na faculdade, o professor Diogo Vaz! Sempre a mesma rotina de pequeno-almoço, café, almoço, com muito estudo, diversão, conversa e motivação para ambos conseguirmos alcançar o desafio para o qual nos propusemos o ano passado! Obrigado professor!

Ao meu *project manager* Urbano Freitas que sempre se disponibilizou para me aconselhar e rever o trabalho e que tantas lições me foi dando ao longo deste tempo. Seguramente que vamos continuar as nossas partilhas de experiência e conhecimento durante muito tempo! Obrigado amigo!

À minha irmã e cunhado que eu sei que posso sempre contar e me confortam quando me sinto com mais dúvidas no caminho a seguir! São duas pessoas que eu procuro sempre aconselhar-me!

Aos meus pais… por todo o apoio e aconchego que me dão sempre. Se hoje estou aqui é porque acreditaram em mim e me incentivaram a fazer mais e melhor. São incríveis!

RESUMO

O trabalho realizado visa analisar o desempenho da utilização de ferramentas *Big Data*, para a componente de tratamento de dados e para a implementação de um algoritmo de *Data Mining*, nomeadamente FP-Growth para a extração de regras de associação, aplicadas ao registo de transações de produtos no mercado do retalho.

Os dados extraídos visam analisar as transações realizadas pelos consumidores, de uma cadeia de supermercados, de forma a compreender quais os produtos que são adquiridas em simultâneo, análise denominada como *Market Basket Analysis*. Foram extraídos registos de um ano, com o histórico de compras de cada cliente. Cada registo contém todos os produtos adquiridos num espaço de um ano. Pretende-se utilizar a informação obtida para identificar produtos correlacionados, com vista a determinar quais os produtos que são frequentemente adquiridos em conjunto. Assim, pretende-se analisar os resultados obtidos e implementar novas estratégias de negócio, adaptando a oferta dos supermercados às preferências dos consumidores.

Através de várias ferramentas do ecossistema Hadoop, foram analisados os dados visando eliminar qualquer inconsistência presente na base de dados e gerar novas variáveis para a aplicação de uma segmentação por perfil de consumidor e para a extração de regras de associação. Durante a execução do pré-processamento de dados foram utilizadas as ferramentas de SQL para criar um conjunto de KPIs que permitiu perceber o estado atual do negócio do supermercado. Na análise de *clusters,* foi decidido que seriam definidos 3 grupos. O primeiro cluster foi constituído pelos clientes de necessidades imediatas, o segundo por clientes de contas correntes e o terceiro por consumidores compulsivos. Para cada um dos clusters gerados foram identificadas um conjunto de regras de associação que permitiu entender os hábitos de consumo de cada tipo de cliente. A componente analítica foi implementada em Spark MLlib, em programação Scala.

A utilização de Hadoop em conjunto com Spark permitiu a execução de forma integrada, um conjunto de funcionalidades, sendo possível recorrer a linguagens como SQL, HiveQL, Pig Latin, Python ou Scala numa única plataforma.

PALAVRAS-CHAVE

Big Data, *Data Mining*, Análise de Clusters, Regras de Associação

ABSTRACT

The main goal of this project is the analysis of performance in Big Data tools across multiple business perspectives. During this project, tasks were created for data preprocessing and an algorithm was implemented in order to extract frequent item sets using Big Data tools, such as Apache Hadoop and Spark.

The database represents a list of transactions from a supermarket chain so one can identify which products are frequently bought together. The aim of this project is, by using a scalable Data Mining technique called Market Basket Analysis, generate some association rules that link two or more products together. The detection of strength associations between large quantities of data can then leverage decision-making. In the end, the findings one may encounter in this approach, can be use by retailers to organize the storage of the products in store, as well to design the web catalog and increase their sales.

The data was analyzed using several tools of the Hadoop ecosystem to remove any noise and to generate new variables. With this, it was then possible the creation of a customer profile segmentation and the extraction of association rules. During the execution of data preprocessing, a set of KPIs was generated to understand the current state of the supermarket business.

Using 'Spark Machine Learning Library' some Data Mining techniques, including clustering and association rules, were used to identify patterns over the products purchased together. Within clustering analysis, three clusters were obtained: the first cluster was composed by customers with immediate needs; the second cluster by current account customers and the third cluster with customers with a compulsive behavior. For each cluster a set of association rules was extracted so one can easily understand the supermarket habits of customers.

KEYWORDS

Big Data, Data Mining, Cluster Analysis, Association Rules

ÍNDICE

I. Introdução 1
2. Revisão da Literatura 3
2.1. Big Data 3
2.1.1. Hadoop 4
2.1.2. Spark 9
2.1.3. Diferenças entre MapReduce e Spark 11
2.2. Data Science 12
2.2.1. *Data Mining* 12
3. Metodologia 21
3.1. Carregamento dos dados no HDFS 22
3.2. Extração de uma amostra estratificada 23
3.3. Deteção de outliers e missing values 23
3.4. Criação de um conjunto de KPIs 24
3.5. Tratamento de variáveis 25
3.6. Segmentação por perfil de consumidor 27
3.7. Implementação do algoritmo FP-Growth 28
3.8. Extração de regras de associação 29
4. Resultados e discussão 31
4.1. Análise de Key Performance Indicators (KPIs) 31
4.2. Análise de Clusters 39
4.3. Regras de Associação 49
5. Conclusões 54
6. Limitações e Recomendações para trabalhos futuros 56
7. Bibliografia 57
8. Anexos 60
8.1. Extração dos dados do HDFS para uma tabela em Hive 60
8.2. Extração de uma Amostra estratificada em Hive 61
8.3. Exemplo para a deteção de outliers em Impala 62
8.4. Tratamento de Variáveis em Hive e Pig 63
8.5. Extração do gráfico do Cotovelo em R 67
8.6. Implementação do algorimto K-Means em Spark 68
8.7. Implementação do FP-Growth em Spark para o Cluster 1 71

ÍNDICE DE FIGURAS

Figura 1 – Processamento de dados em MapReduce 5
Figura 2 – Principais projetos do ecossistema Hadoop 6
Figura 3 – Sqoop *workflow* 7
Figura 5 – Processo de descoberta de conhecimento. 12
Figura 6 – *Pseudocode* do algoritmo *FP-Tree* 18
Figura 7 – Construção da FP-Tree 19
Figura 8 – Subconjunto dos *itemsets* com o prefixo e 19
Figura 9 – *FP-Tree* condicional de *e* 20
Figura 10 – Metodologia do Projeto 21
Figura 11 – Número de transações ao longo dos meses 22
Figura 13 – Gráfico do Cotovelo 27
Figura 14 – Tarefas executadas para a implementação do FP-Growth 28
Figura 15 – *Dashboard* executivo 31
Figura 16 – KPI 1 – Evolução mensal do número médio de clientes 32
Figura 17 – KPI 2 – Evolução mensal do número de novos clientes do supermercado 32
Figura 18 – KPI 3 – Quantidade de produtos por transação (%) 33
Figura 19 – KPI 4 – *Top/Bottom* 5 Volume de vendas por cada região 34
Figura 20 – KPI 5 – *Top/Bottom* 5 Volume de negócio por cada região 34
Figura 21 – KPI 6 – Top 10 Volume de vendas por cada loja de supermercado 35
Figura 22 – KPI 7 – Top 10 Volume de negócio por cada loja de supermercado 35
Figura 23 – KPI 8 – Top 10 Produtos mais vendidos 36
Figura 24 – KPI 9 – *Top/Bottom* 5 Lista de produtos por preço 37
Figura 25 – KPI 10 – Evolução mensal do número de fornecedores 37
Figura 26 - KPI 11 – Produtividade por dia de semana 38
Figura 27 – Percentagem do número de clientes por *cluster* 39
Figura 28 – Volume de Vendas por *cluster* 41
Figura 29 – Volume de Negócio por *cluster* 41
Figura 30 – Dia da Semana preferencial por *cluster* 42
Figura 31 – *Cluster* 1 – Frequência de Venda por Região 43
Figura 32 – Cluster 1 – *Top/Bottom* Lista de Produtos 44
Figura 33 – *Cluster* 2 – Frequência de Venda por Loja 45
Figura 34 – *Cluster* 2 – *Top/Bottom* Lista de Produtos 46
Figura 35 – *Cluster* 3 – Frequência de Venda por Loja 47
Figura 36 – *Cluster* 3 – *Top/Bottom* Lista de Produtos 48

Figura 37 – Cluster 1 – Regras de Associação 49
Figura 38 – Cluster 2 – Regras de Associação 50
Figura 39 – Cluster 3 – Regras de Associação 52
Figura 40 – Importação do ficheiro de configuração de Hive para conexão em Pig 64
Figura 41 – Criação do RDD para a implementação do algoritmo K-Means..... 68

ÍNDICE DE TABELAS

Tabela 1 – Diferenças entre MapReduce e Spark 11
Tabela 2 – *Frequent Itemsets (FP-Growth)* 20
Tabela 3 – Lista de *KPIs* 24
Tabela 4 – *Feature Selection* 26
Tabela 5 – *Baskets format* 29
Tabela 6 – Perfis do valor de cliente 40
Tabela 7 – Cluster 1 – Conjunto de *items* frequentes 44
Tabela 8 – Cluster 2 – Conjunto de *items* frequentes 46
Tabela 9 – Cluster 3 – Conjunto de *items* frequentes 48
Tabela 10 – *Cluster* 1 – Regras de Associação – Análise Descritiva 50
Tabela 11 – Cluster 2 – Regras de Associação – Análise Descritiva 51
Tabela 12 – Cluster 3 – Regras de Associação – Análise Descritiva 53

LISTA DE SIGLAS E ABREVIATURAS

BI	Business Intelligence
DM	Data Mining
ETL	Extract, Transform, Load
KPI	Key Performance Indicator
RDBMS	Relational Database Management System
SQL	Structured Query Language
NoSQL	Not Only SQL
Spark MLlib	Spark *Machine Learning* (ML) library
FP-Growth	Frequent Pattern Growth
FP-Tree	Frequent Pattern Tree
IoT	Internet of Things
BDaaS	*Big Data* as a Service
DaaS	Database as a Service
SaaS	Software as a Service
PaaS	Platform as a Service
YARN	Yet Another Resource Negotiator
HDFS	Hadoop Distributed File System
UDF	User Defined Function
MPP	Massive Parallel Processing
CART	Classification and Regression Tree
SVM	Support Vector Machine
RDDs	Resilient Distributed Dataset
JDBC	Java Database Connectivity
RCFile	Record Columnar File
ORC	Optimized Row Columnar

I. INTRODUÇÃO

A evolução tecnológica e o consequente aumento da dependência da sociedade e das organizações pela Internet levaram, nos últimos anos, a um enorme crescimento do volume e variedade de dados existentes. O mundo gera, diariamente, 52 milhões de *terabytes* de dados, provenientes do fenómeno da *Internet of Things (IoT),* isto é, a crescente utilização dos dispositivos móveis, redes sociais ou serviços de *cloud computing*, resulta na geração de bilhões de dados (Shah, 2016). Estas atividades obrigam a uma mudança de paradigma nas organizações, com a competitividade do mercado a exigir o armazenamento de todo o tipo de informação, oriundo de diversas fontes. Estas exigências tornam as soluções tradicionais, como bases de dados relacionais, como inviáveis (Ma, 2016).

O ***Big Data*** é o termo utilizado para descrever este crescimento exponencial de dados sejam estruturados ou não-estruturados e a sua grande disponibilidade e acessibilidade (Leskovec, Rajaraman & Ullman, 2014). As organizações enfrentam, hoje, o desafio de armazenamento e processamento destes grandes volumes de dados. É importante salientar que as organizações são obrigadas a lidar com quantidades de dados que, muitas vezes, excedem a capacidade de processamentos dos tradicionais RDBMS, não só pela dimensão dos mesmos, mas pela velocidade e variabilidade com que estes novos dados são gerados. Estas características acabam por tornar os projetos tradicionais de *Data Warehousing* pouco produtivos. Por norma, as operações de transformação de dados entre os sistemas operacionais e o *Data Warehouse* são realizadas em tabelas temporárias que, inseridas num contexto de *Big Data*, excedem a capacidade de processamento disponível nas máquinas, tornando o processo mais lento e ineficaz para a criação de *queries,* e consequente, *reporting*. A presença de um grande volume de dados em projectos que utilizam as tradicionais ferramentas de BI, como SQL Server, requerem um período mais longo para a sua implementação, não apenas para a modelação de dados, mas também para as alterações nos processos de ETL encontradas no período de implementação (Kimble & Milolidakis, 2015).

Hoje em dia, as organizações são obrigadas a extrair *insights* dos dados, praticamente, em tempo real, pelo que se exige que o pré-processamento dos mesmos seja concluído utilizando ferramentas de elevado desempenho, capazes de aceder e processar grandes volumes de dados que possam ser úteis na criação de valor para o negócio (Leskovec, Rajaraman & Ullman, 2014). A tendência será, cada vez mais, a utilização de uma única plataforma capaz de processar grandes volumes de dados, independentemente da estrutura, a um custo e tempo reduzido (eliminando o problema da existência de múltiplas tabelas para diferentes propósitos), como é o caso do Apache *Hadoop* (White, 2015). A grande capacidade destas ferramentas possibilita a análise dos dados existentes, bem como a exploração de novas possibilidades de atuação no mercado, através da identificação de novos padrões e exploração de questões, que até aí ainda não tinham sido feitas.

É do interesse das organizações aproveitar as oportunidades que o ***Big Data Analytics*** poderá trazer para o negócio, otimizando a eficiência e qualidade dos seus serviços, de forma a obter maiores índices de *performance*. Em primeira instância, as tecnologias de *Big Data* permitem vantagens de custo bastante significativas, uma vez que permitem o armazenamento e processamento distribuído de grandes volumes de dados, bem como a identificação de novas estratégias de negócio mais eficientes e rápidas (Prasad & Sheth, 2013). A velocidade e a agilidade proporcionada pela análise *in-memory* e a capacidade de analisar novos tipos de dados, como não-estruturados, permite às organizações analisar os dados em tempo real de vários formatos, pelo que as decisões são tomadas mais

rapidamente e com maior eficiência. Contudo, as organizações não conseguem acompanhar este crescimento, variedade e variabilidade de dados apenas recorrendo a intervenção-humana. A utilização de métodos de *Artificial Intelligence* permite acrescentar uma camada de inteligência em grandes quantidades de dados de forma a lidar com tarefas analíticas complexas mais rapidamente. Segundo Jurney, R. (2013), a solução passa pelo ***Big Data Science*** que combina um conjunto de abordagens estatísticas com técnicas de matemática e programação, e permite extrair conhecimento sob diversas perspetivas, facilitando o desenvolvimento de novos produtos e serviços disruptivos no mercado.

O mercado do Retalho é cada vez mais competitivo e exigente, obrigando as empresas a atuarem cada vez mais rápido (Gupta & Pathak, 2014). A exigência do mercado é influenciada pelas mudanças disruptivas que atualmente existem, provocadas, pelas inovações tecnológicas. A prioridade das organizações do setor do Retalho consiste na recolha de dados provenientes de variadas fontes de informação e em manter esta quantidade de informação disponível sobre os clientes existentes, conseguindo assim, a adaptar os seus serviços ao comportamento do mesmo. O foco dos retalhistas centra-se, atualmente, na experiência de compra do consumidor (Woo, 2015). Uma das formas de alinhar uma boa estratégia com as necessidades do consumidor é conhecer os seus bens de consumo. Esta informação possibilita identificar quais os produtos mais procurados, mas também, as associações existentes entre estes. Este conhecimento é essencial para os analistas do sector do Retalho, para identificar aspetos de negócio que a organização desconhece e assim ser capaz de, por exemplo, lançar novas campanhas de promoção ou reduzir o espaço ocupado por marcas com pouca procura. No caso do presente estudo, procura-se encontrar padrões de comportamento dos consumidores que sejam totalmente desconhecidos.

O ***Data Mining*** consiste na procura de padrões e relacionamentos que estão implícitos nos dados, permitindo às organizações desenvolver conhecimento sobre as suas atividades de negócio (Hand, Mannila & Smyth, 2001). Uma das técnicas de *Data Mining* para estudar as relações existentes, são as Regras de Associação. Esta tem como principal objetivo mapear relações entre os itens que ocorreram em conjunto num determinado evento/registo. Com esta análise poderemos encontrar padrões e regras relevantes que ajudam no aperfeiçoamento da gestão.

O objetivo principal deste trabalho consiste em criar uma prova de conceito de análise de dados provenientes de um conjunto de lojas de supermercado, utilizando ferramentas de *Big Data,* para encontrar associações entre produtos de supermercado de forma a suportar as campanhas de Marketing das lojas em questão. No capítulo da Revisão da Literatura serão abordados os tópicos mencionados no decorrer do projecto, como Big Data, Data Science e Data Mining, com o objetivo de introduzir todos os conceitos e ferramentas que permitiram a concretização deste projeto. Posteriormente, é abordada a metodologia utilizada no trabalho, explicando todos os passos necessários para a extração, pré-processamento e análise de dados. Através de ferramentas do ecossistema Hadoop foi efetuado o carregamento de dados na *virtual machine* da Cloudera e a implementação do pré-processamento de dados, com vista a eliminar as inconsistências encontradas e a definir as variáveis necessárias para a aplicação dos algoritmos de *clustering* e regras de associação. Esta componente de *analytics* foi implementada com recurso a bibliotecas de Spark, permitindo produzir os resultados para análise. Assim, no capítulo dos "Resultados e Discussão" é realizada uma segmentação por perfil de cliente, e em cada *cluster* serão analisadas as regras de associação extraídas e a proposta de novas campanhas de promoção para cada grupo de clientes.

2. REVISÃO DA LITERATURA

2.1. BIG DATA

O *Big Data* corresponde à quantidade de dados que excede a capacidade de processamento dos tradicionais RDBMS (*relational database management system*). Esta pode ser definida como grandes volumes de dados disponíveis por diferentes escalas de complexidade, concebidos a grande velocidade e que não se ajustam às estruturas de dados das atuais arquiteturas (Jurney, 2013).

Considera-se um cenário de *Big Data* quando se verifica algum ou vários Vs dos 3Vs (Volume, Velocidade e Variedade), afirmando que *Big Data* não se limita apenas à grande quantidade dados (Lydia & Swarup, 2016). O volume de dados já foi abordado, e refere-se à grande quantidade de dados que cresce exponencialmente com a informação a ser medida, atualmente, em *exabytes.* A solução passa pelo uso de infraestruturas baseadas em múltiplos dispositivos de armazenamento (servidores muitas vezes em ambiente nuvem), para diminuir o custo e aumentar a capacidade de armazenamento de grandes quantidades de dados. A frequência com que os dados são gerados *(Batch, Real-Time ou Streaming)* obriga a que o pré-processamento de dados seja concluído no momento, com altos índices de desempenho e mapeamentos dinâmicos. A velocidade refere-se a esta agilidade com que atualmente os dados são produzidos. Esta grande quantidade de dados também tem origem das diversas fontes de informação, não só vindas das tradicionais bases de dados, mas também de esquemas não-estruturados como texto, imagem, áudio ou vídeo. É necessário saber lidar com todos estes diferentes formatos de dados em simultaneamente.

Buyya, *et al* (2016), acrescenta outras duas características que se deve realçar:

- Veracidade – A maior complexidade dos dados obriga a uma avaliação mais rigorosa para garantir a autenticidade dos mesmos;
- Variabilidade – A inconsistência verificada nos fluxos de dados, torna impossível prever os picos de informação;

Prasad*, et al* (2013), ainda adiciona uma outra característica que representa a capacidade das organizações em aproveitar este acesso a grandes quantidades de dados:

- Valor – A oportunidade de obter vantagem competitiva com a implementação de projetos de *Big Data*, pelo seu valor presente e futuro, com o desenvolvimento de análises preditivas e identificação de correlações.

Atualmente, as tecnologias de *Big Data* são na maioria das vezes entendidas como semelhantes às de *Business Intelligence* (Kimble & Milolidakis, 2015). É consensual que ambos os tipos de tecnologias são utilizados para a exploração de dados e extração de conhecimento dos seus dados. Quando nos focamos na evolução de um projeto de *Big Data* Analytics percebemos um alcance de estratégias de negócio muito mais disruptivas, com a inclusão de inteligência nos negócios baseada na interpretação de grandes volumes de dados de diferentes origens (Williams, 2016). No paradigma do *Business Intelligence*, as tarefas são orientadas para a construção de relatórios baseados em dados históricos que permitem compreender o estado atual do negócio, necessitando sempre de intervenção humana para a interpretação dos resultados (Santos & Ramos, 2009). As ferramentas de *Big Data* estão preparadas para o processamento de dados estruturados e não estruturados, permitindo uma melhor

compreensão dos dados e antecipar possíveis comportamentos, gerando *insights* contínuos em tempo real, como define Leskovec, *et al* (2014).

O actual desafio das organizações passa pela combinação de *Big Data* com *Cloud Computing*. Este paradigma representa o armazenamento e processamento de grandes volumes de dados num conjunto de recursos de computação partilhados, a um menor custo de *hardware* (Neves & Bernardino, 2015). O modelo *cloud* assenta em três modelos de serviço, que permitem substituir o modelo tradicional que apresenta custos de implementação e manutenção da tecnologia mais elevados (Buyya, Broberg & Goscinski, 2011):

- IaaS (Infrastructure-as-a-Service) – Corresponde aos recursos de infraestrutura básicos de armazenamento e processamento, como RAM, CPU, Disco ou sistemas operativos;
- PaaS (Plataform-as-a-Service) – Consiste no ambiente de desenvolvimento ou serviço de base de dados. Um dos exemplos é o ecossistema Hadoop;
- SaaS (Software-as-a-Service) – Aplicação disponibilizada numa interface Web normal.

Neste cenário, as ferramentas de armazenamento e processamento de dados serão colocadas na nuvem, oferecendo maior agilidade no serviço, bem como maior *performance* no processamento das máquinas.

2.1.1. Hadoop

O crescente aumento do volume de dados torna impossível o acesso a esse conjunto de dados através de uma única máquina, e a capacidade de processamento de dados acaba por se tornar num processo demasiado lento e pesado tendo em conta a capacidade de armazenamento existente (Jurney, 2013). Como tal, a tradicional solução de armazenamento de dados assente num processo centralizado nos *mainframes,* servidores monolíticos, responsáveis por distribuir dados por diferentes máquinas, excede, facilmente, o limite de capacidade de rede do servidor, por isso foi necessário alterar a estratégia. Assim, a solução mais óbvia acaba por colocar múltiplos discos a processar dados em simultâneo, isto é, colocar várias máquinas a trabalhar em paralelo, por vários *clusters*. É por esta razão que surgiram novas ferramentas de processamento distribuído, como o *Hadoop* (White, 2015).

O Hadoop é um ecossistema que permite o armazenamento e processamento de *Big Data* numa forma distribuída, em grandes *clusters* de hardware. Uma arquitetura como a do Hadoop acaba por trazer maiores vantagens ao nível de *armazenamento* para a organização uma vez que consegue suportar grandes quantidades de dados, distribuindo-os em pequenos blocos, que posteriormente são armazenados em *hardware low cost* (Holmes, 2012). O facto de o processamento de dados utilizar uma distribuição paralela acaba por conseguir obter resultados a uma maior velocidade, e segurança. O Hadoop caracteriza-se como uma solução *fault tolerance*, pois os dados são replicados em diferentes *nodes* do nó mestre, e em caso de falha, o Hadoop automaticamente replica os dados para outro nó (White, 2015)

A plataforma possui três grandes componentes: HDFS, MapReduce e YARN (Holmes, 2012).

HDFS

O *Hadoop Distributed File System* (HDFS) é o sistema de ficheiros *standard* para armazenar os dados em diferentes nós do cluster Hadoop. Para Lydia, E. et al (2016), o HDFS permite a conexão de vários

nós do *cluster* Hadoop, pelos quais os dados são distribuídos de forma contínua. É projetado para ser implementado em *hardware* de custo reduzido e capaz de armazenar, como uma sequência de blocos, arquivos de dados muito volumosos em todos os nós de um *cluster.* Os blocos são, por padrão, replicados três vezes permitindo tolerância a falhas. Os ficheiros já arquivados não podem sofrer alterações.

MapReduce

O MapReduce é uma *framework* de processamento distribuído de dados em *clusters*, baseado em *Java*. Um *job* de MapReduce é responsável por executar paralelamente os dados armazenados no HDFS ao longo do cluster Hadoop (figura 1).

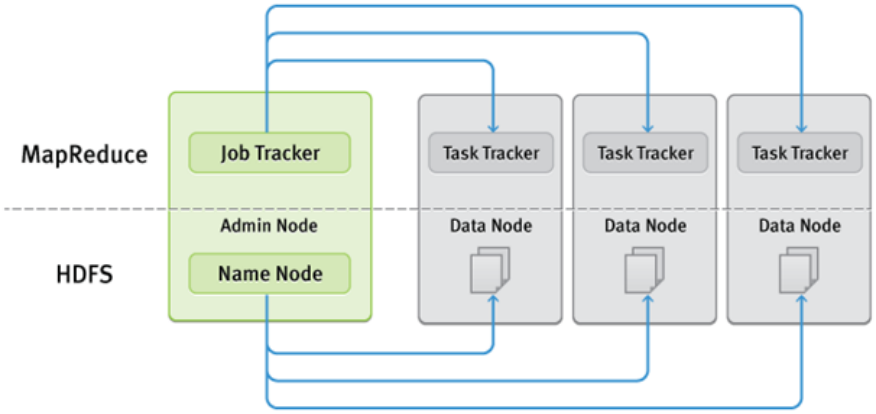

Figura 1 – Processamento de dados em MapReduce

(Fonte: Kimble, C., & Milolidakis, G. (2015). *Big Data* and Business Intelligence: Debunking the Myths. Global Business and Organizational Excellence, 35(1), 23-34. doi:10.1002/joe.21642)

Assim, o MapReduce permite que os dados, distribuídos por diversos nós, estejam acessíveis no HDFS. Como White, T. E. (2015) estrutura este processo contém três fases:

1. *Map;*
2. *Shuffle & Sort;*
3. *Reduce.*

A fase de *Map* é responsável por recolher toda a informação ao longo de todos os nós e monitoriza-la para diferentes máquinas, que terão a missão de converter noutros conjuntos de dados decompostos em pares chave/valor (*k-values*). Por exemplo, analisando o fluxo de trabalho de um processo de MapReduce para contar o número de produtos comuns num conjunto de supermercado, este é dividido em três ações:

1) Recolher a lista de produtos distintos e, em seguida, associa-los por diferentes chaves (*key value*).
2) A fase de *Shuffle* permite agrupar os produtos por cada chave gerada e ordenar os conjuntos por cada chave.
3) Por fim, a cada um dos conjuntos é aplicado a função *Reduce*, que procura cruzar os produtos por cada conjunto de dados e contar o número de produtos em comum.

YARN

Durante os *jobs*, o MapReduce utiliza todos os recursos do *cluster* (CPU e Memória), assim como outras *fameworks* de processamento presentes no *cluster Hadoop*, o que pode originar um problema de sobreposição de recursos. É neste contexto que surge o YARN *(Yet Another Resource Negotiator)*, que consiste em alocar, de forma dinâmica, os recursos para cada uma das *frameworks*, permitindo assim a partilha das capacidades computacionais. Através do YARN é possível que todos os *softwares* consigam aceder à máquina em simultâneo.

O *Hadoop* caracteriza-se por ser uma *framework open source* e por possuir ferramentas para diversas áreas do negócio, como o ETL ou BI. A integração deste leque de ferramentas distintas permite aos programadores forcarem-se, cada vez mais, na exploração de dados, o *business analytics* (Leskovec, Rajaraman & Ullman, 2014). Na figura 2 estão representados os principais projetos do ecossistema Hadoop:

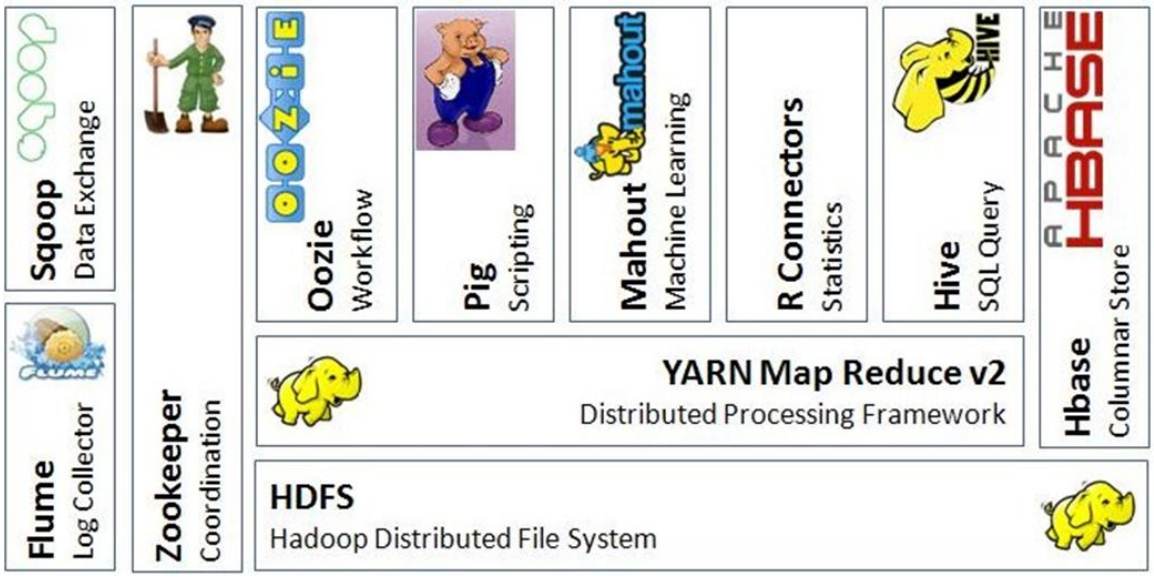

Figura 2 – Principais projetos do ecossistema Hadoop

(Fonte: Bengfort, B., & Kim, J. (2016). Data analytics with Hadoop: An introduction for data scientists. Sebastopol, CA: O'Reilly.)

Ao longo deste projeto foram as seguintes plataformas:

- **Sqoop** – Ferramenta responsável por transferir dados em massa, de forma otimizada, entre as bases de dados relacionais e o Hadoop;
- **Hive** – Ferramenta criada para consulta de dados, com uma *syntax* semelhante ao SQL. Funciona com base em MapReduce e cada *query* é executada em todos os clusters;
- **Impala** – Como o Hive, o Impala é um mecanismo de consulta SQL, contudo apenas para dados armazenados num cluster Hadoop;
- **Pig** – Ferramenta que permite a análise de grandes volumes de dados e de diferentes estruturas com uma linguagem de programação high-level, não obrigando os utilizadores a escrever programas de MapReduce.

Em seguida, é feita uma análise mais detalhada sobre cada uma das ferramentas utilizadas ao longo do projeto:

Apache Sqoop

O Sqoop (SQL-to-Hadoop) como a ferramenta utilizada para transferir os dados entre as bases de dados relacionais e Hadoop (Grover, *et al*., 2015). Em situações que os dados de *input* se encontram num formato estruturado, por norma armazenados em base de dados relacionais, o Sqoop é utilizado para carregar os dados no Hadoop de uma forma mais eficiente que a manual. O Sqoop foi projetado para transferir dados de um RDBMS, como MySQL ou Oracle, para uma plataforma de armazenamento como o HDFS, Hive e Hbase, com funções de MapReduce (White, 2015). Este automatiza a maioria do processo de transformação de dados, replicando o *schema* de dados inserido no RDBMS. Para o mesmo autor, a maior vantagem do Sqoop é fornecer flexibilidade para manter os atuais dados em produção, em simultâneo que se replica os dados para o cluster Hadoop para análises mais avançadas, sem comprometer a base de dados em produção. A figura 3 demonstra a forma como os dados são distribuídos paralelamente e inseridos no HDFS via Sqoop:

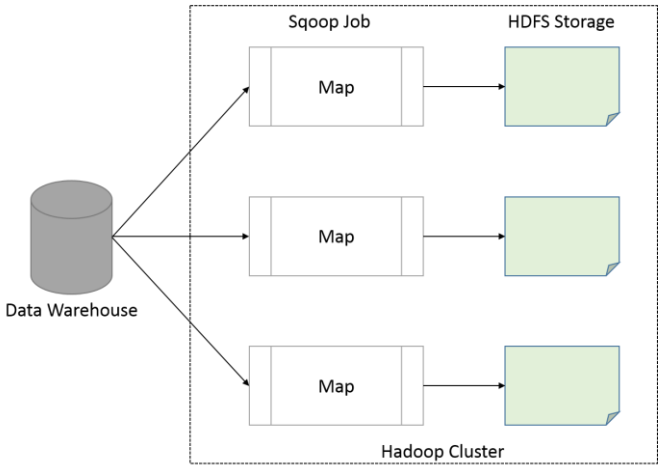

Figura 3 – Sqoop *workflow*

(Fonte: Lydia, E., & Swarup, M. (2016). Analysis of *Big Data* through Hadoop Ecosystem Components like Flume, MapReduce, Pig and Hive. Ijcse.Net, 5(1), 21–29.)

Ao importar os dados a partir das bases de dados, o Sqoop procura aceder às bases de dados originais de modo a reunir os metadados necessários para a importação do conjunto de dados (Grover, *et al*., 2015). Posteriormente, é criado uma tarefa Sqoop que é transferido para Hadoop, com vista a transferir o conjunto de dados baseado nos metadados capturados no passo anterior. Este processo gere um conjunto de arquivos serializados que são armazenados como CSVs no HDFS com o nome da tabela original. A *metadata* é arquivada numa classe Java que regista o *schema* de cada linha da tabela importada. Esta classe Java é utilizada durante o processo de importação, mas também pode ser utilizada no processamento subsequente dos dados com MapReduce.

Hive & Impala

O ecossistema Hadoop contempla dois projetos *open-source* para a análise de dados através da linguagem SQL, denominados Hive e Impala. Ambas as plataformas partilham a mesma interface de

programação, *HiveQL*, contudo, o tempo de resposta visto em cada um é completamente distinto (White, 2015).

O Hive é uma ferramenta de processamento de dados em grande escala e que permite o carregamento de dados estruturados no HDFS (Kornacker, *et al*., 2015). Como referido anteriormente, o processamento de dados no Hadoop é efetuado via *MapReduce*, e no caso do Hive, cada comando e consulta HiveQL corresponde a um plano de execução de um conjunto de tarefas de MapReduce, executados no *cluster* Hadoop. White, T. E. (2015) defende que a principal vantagem do Hive é permitir que os utilizadores sem grandes conhecimentos de Java e/ou *MapReduce* consigam realizar uma série de análises *ad-hoc* através de scripts *HiveQL*, que apresenta um formato semelhante ao *Transact-SQL*. No entanto, os tempos de resposta do Hive são muito superiores aos valores registado em Impala. Como Lydia, *et al* (2016), explica, estes tempos devem-se à enorme latência que é gerada para compilar os *jobs* de MapReduce no cluster. Este plano de execução torna o Hive como uma solução *fault-tolerance,* isto é, qualquer problema que ocorra no *worker node* durante a consulta de dados em *Hive*, o *MapReduce* replica os dados para outro nó do cluster Hadoop. Esta vantagem torna o Hive como a solução ideal para tarefas de exploração e processamento de dados estruturados em grande escalabilidade (White, 2015). O mesmo autor apresenta a implementação de cubos OLAP (Online Analytical Processing) na escala do *terabyte* e *petabyte*, como um dos use-case mais frequente em Hive, possibilitando o acesso, visualização e análise dos dados com enorme flexibilidade e performance.

No caso do Apache Impala, as consultas de dados são executadas diretamente no HDFS, obtendo tempos de processamento muito mais rápidos. O Impala caracteriza-se por ser um mecanismo MPP (*Massively Parallel Processing*) que oferece, ao contrário do Hive, índices de alto desempenho e baixa latência, pois não é baseado em tarefas MapReduce, mas sim numa arquitetura distribuída com base em processos de *daemon*[1] (Lydia, *et al*., 2016). Em cada processo, o Impala cria um processo de paralelização e distribuição dos dados, inseridos no HDFS, por múltiplos nós do cluster Hadoop, não exigindo o processamento dos mesmos e, consequentemente, a utilização de tarefas de MapReduce. Desta feita, o Impala não é uma solução *fault-tolerance*, ou seja, qualquer problema no *worker node* durante a *query* impossibilita a execução da mesma.

O Impala e Hive partilham a mesma filosofia, pois ambos permitem o acesso e análise de dados através de SQL, contudo os índices verificados de desempenho e latência são diferentes (Kornacker, *et al*., 2015). O Impala é, por norma, adequado para análises *ad-hoc*, e em situações que exijam que múltiplos utilizadores tenham acesso aos dados em simultâneo. O Hive é recorrentemente utilizado em tarefas de ETL e processamento em massa (White, 2015).

Apache Pig

A plataforma Pig é utilizada para a análise de grandes quantidades de dados através do paradigma de *dataflow programming*, muito utilizada para a *hardware* de computação paralela, como é o caso de Pig Latin (White, 2015). Esta é uma linguagem de programação procedimental utilizada para a manipulação de fluxos de dados e é executada em modo local ou *MapReduce* e inclui a interface de

[1] Processos que correm continuamente, em background, num sistema. Estes não requerem a interação do utilizador para a sua execução, iniciando-se no momento do boot do sistema com o objetivo de responder aos requisitos de rede, hardware, entre outras tarefas.

linha de comandos *Grunt*. Cada operação script em Pig Latin é convertida para uma sequência de jobs em *MapReduce*, não obrigando os utilizadores a possuírem conhecimento de Java. Esta é a principal vantagem da ferramenta, pois foi projetada para permitir que uma série de tarefas sejam implementadas com menor dificuldade (Ryza, *et al*., 2015). Os scripts de Pig Latin permitem integrar código personalizado com UDFs (*User-Defined Functions*), que pode ser escrito em Java, Python ou JavaScript. O Apache Pig foi concebido para processo de ETL em contextos de *Big Data*, facilitando a construção de MapReduce pipelines (Grover, *et al*., 2015). No entanto, o mesmo autor refere que a programação Pig Latin poderá uma excelente alternativa para realizar análises ad-hoc e criar modelos preditivos em grandes conjuntos de dados.

Para executar um Script de Pig, é necessário recorrer a um dos ambientes de execução (por exemplo, Grunt Shell) para que o script seja submetido a quatro fase de transformações, de modo a produzir a o output desejado. A fase de análise é responsável por verificar a *syntax* do código executado, bem como outras verificações. Posteriormente, são aplicadas otimizações lógicas como projeção e *pushdown*, por forma a compilar o script em diversos jobs de MapReduce. Estes jobs são implementados, de forma ordenada, no *cluster* Hadoop, e assim são produzidos os resultados desejados.

Apache HCatalog

O Apache HCatalog é uma ferramenta de gestão de armazenamento com vista a partilhar estruturas de dados, disponibilizados em Hive, para ferramentas de processamento do Hadoop, como Pig e MapReduce (White, 2015). O HCtalog cria uma tabela abstrata sobre os metadados do Hive, não exigindo aos utilizadores o conhecimento relativo ao formato dos dados (RCFile – estruturas de dados para armazenado de tabelas relacionais, Parquet – permite o armazenado de dados em coluna num formato comprimido, ORC – formato de dados altamente vantajoso para o processamento de tarefas MapReduce, como Hive, ou ficheiros de texto).

2.1.2. Spark

O Apache Spark é uma ferramenta de processamento paralelo de dados que permite aceder, transformar e carregar grandes volumes de dados de uma forma mais otimizada que a ferramenta MapReduce, utilizada no Hadoop (Ryza, *et al*., 2015). Desenvolvido em 2009, a Apache tinha como objetivo a implementação de uma *framework* capaz de processar uma variedade de conjuntos de dados de diversos formatos (vídeo, texto, imagem, etc.), bem como diversas origens em *real-time*.

Todo o processamento computacional de Spark reside em torno da sua estrutura de dados distribuída, os RDDs (*Resilient Distribute Dataset*). Estes correspondem a uma abstração de dados resiliente a falhas, ou seja, em caso de falha num *worker node* do cluster, o Spark tem a capacidade de recolher a partição RDD a partir da última transformação de dados e, assim, recuperar a informação (Karau, *et al*., 2015). Esta método de processamento torna o Spark como uma solução *lazy* onde os dados são mantidos em memória e em caso de falha são executadas as últimas operações necessárias para recuperar os dados perdidos. Este processamento representado como um Grafo Direcionado e Acíclico (DAG) de tarefas que são executadas em blocos de memória distribuídos e permite uma maior performance de processamento face ao MapReduce, embora mantendo a garantia de tolerância a falhas. É através dos RDDs que Spark implementa as bibliotecas necessárias para a disponibilização das tarefas de *data science* que interagem com o Hadoop (White, 2015). Cada operação de Spark é

executada independentemente num *cluster* e o seu fluxo de trabalho de é coordenado pelo objeto *SparkContext*. É através deste objeto que o YARN consegue alocar um conjunto de recursos de Spark para o processamento de dados, armazenados no HDFS (Ryza, *et al*., 2015).

Assim, o Apache Spark apresenta quatros grandes vantagens para as organizações (Ryza, *et al*., 2015). A primeira direciona-se para o tempo de processamento e velocidade com que é possível gerar os *outputs*. O Spark utiliza um processamento paralelo *in-memory* e consegue produzir muito mais rapidamente que ferramentas em disco. Como tal, o acesso aos resultados produzidos é muito mais rápido, o que permite que a tomada de decisão seja feita mais previamente. A segunda vantagem consiste na acessibilidade e flexibilidade de exploração dos dados. O Spark, foi criado para qualquer utilizador com o mínimo de conhecimento de base de dados e programação *high-level* (como Python ou Scala), não exigindo conhecimentos de MapReduce. A utilização do Spark permite o acesso a uma *framework* construída para aplicação de métodos de analise avançada no negócio, incluindo visualização de dados, análise em *streaming*, optimização de *queries* ou aprendizagem de máquina.

Bibliotecas de Spark

O ecossistema do Spark é composto por várias bibliotecas que permitem um conjunto de capacidades adicionais para a análise de dados e implementação de algoritmos de *Machine Learning* (White, 2015). As principais bibliotecas do Spark são:

- Spark Streaming – Permite o processamento de dados de *streaming* em *real-time;*
- Spark SQL – Permite a execução de *queries* SQL, substituindo as implementações em Hive. Cada *query* SQL é transformada numa operação de Spark;
- Spark MLlib – Biblioteca de aprendizagem máquina Spark, que fornece um conjunto de algoritmos.
- Spark GraphX – API do Spark orientada para a visualização de dados.

Neste projeto foram utilizadas as bibliotecas **Spark SQL** e **Spark MLlib**.

O Spark SQL é uma biblioteca da *framework* Apache Spark criada para o processamento de dados estruturados através de instruções SQL. Esta biblioteca permite a consulta e transformação de dados estruturados alocados, em memória, num *DataFrame* sem exigir o conhecimento do modelo de programação do Spark (Karau, *et al.,* 2015). Um *DataFrame* corresponde a uma coleção de dados organizados em colunas nomeadas sendo conceptualmente equivalente a uma tabela inserida numa base de dados relacional (Ryza, *et al*., 2015). Estes podem ser criados através dos RDDs existentes, sendo possível, assim, consultar os dados num formato estruturado. As funcionalidades do Spark SQL são importadas com a classe SQLContex, presente no *package* spark.sql.

A utilização do Spark SQL permite combinar os benefícios da consulta de dados relacionais com SQL com a flexibilidade de processamento do Spark e a capacidade de análise das bibliotecas de Scala, num único ambiente de programação (Ryza, *et al*., 2015).

O Spark MLib é a biblioteca de Apache Spark utilizada para soluções de *Machine Learning*, oferecendo um conjunto de algoritmos de aprendizagem automática, como classificação, regressão, *collaborative filtering* e *reduction dimensionality,* bem como uma *framework* "ML-pipeline" que permite o acesso a um conjunto de APIs de alto nível para a implementação dos algoritmos. Este conjunto de algoritmos são implementados de forma paralela entre os diferentes nós do *cluster*, recorrendo às operações RDD

do Spark, isto é, a utilização dos algoritmos do Spark MLlib, ou uma API do Spark, exige a criação de estruturas de dados de forma distribuída e paralelizada (Ryza *et al*., 2015). Além dos algoritmos e das operações de pré-processamento de dados, esta biblioteca também oferece ferramentas para avaliar a qualidade e desempenho dos algoritmos implementados. Escrita em Scala[2], esta biblioteca resulta em grandes benefícios para o negócio, pois permite, de forma eficiente, a implementação de algoritmos de aprendizagem em grande escala (Bengfort, *et al.,* 2016).

Os programadores dispensam a maioria do seu tempo a dar suporte às infraestruturas ao invés de desenvolverem modelos capazes de extrair conhecimento dos dados da organização (Ryza, *et al*., 2015). A grande utilidade de Spark MLlib é o grande leque de algoritmos e tipos de dados que esta biblioteca fornece para diferentes casos de negócio (White, 2015).

2.1.3. Diferenças entre MapReduce e Spark

Como analisado, o Spark e o Hadoop permitem a criação de aplicações para o processamento de grandes volumes de dados (White, 2015). Ambas as plataformas podem ser utilizadas isoladamente, pois o Hadoop apresenta uma componente de processamento de dados como o MapReduce, não exigindo o Spark. O autor Amirghodsi, S. (2016) afirma que o Spark foi criado pela necessidade de alcançar um melhor desempenho que o MapReduce neste tipo de tarefas.

A velocidade verificada no processamento de dados em Spark é muito superior aos tempos de uma solução de MapReduce, uma vez que o fluxo de trabalho é diferente (Ryza, *et al.,* 2015). MapReduce cria vários segmentos sobre o conjunto de dados, estruturado por várias iterações, enquanto o Spark processa todo o conjunto de dados de uma só vez. Na tabela 1, verifica-se algumas diferenças chave entre Apache Spark e MapReduce (Ryza *et al.,* 2015):

Diferenças	**MapReduce**	**Spark**
Armazenamento	Armazenamento dos dados é feito em disco	Armazenamento dos dados é feito em memória
Distribuição de Dados	Utiliza o HDFS	Utiliza RDDs (*Resilient Distributed Dataset*)
Processamento de Dados	Tem dificuldades em lidar com modelos complexos (requer muitas iterações)	Capacidade para processar modelos complexos
Programação	Java	R, Python, Scala e SQL

Tabela 1 – Diferenças entre MapReduce e Spark

(Fonte: Ryza, S., Laserson, U., Owen, S., & Wills, J. (2015). Advanced analytics with Spark. Beijing: O'Reilly.)

Analisando a tabela acima, é possível concluir que o Apache Spark corresponde à evolução do MapReduce e à *framework* que permite o processamento em paralelo de grandes quantidades de dados num *cluster Big Data* (Bengfort, *et al.,* 2016). Embora mantendo a escalabilidade e tolerância a

[2] Scala é uma linguagem de programação multiparadigma – orientada a objectos e funcional – de alta escalabilidade.

falhas do MapReduce, o Spark, derivado do modelo *in-memory*, apresenta um tempo de processamento muito superior, ao contrário do MapReduce que faz vários acessos ao disco.

Contudo, o Spark não contém nenhum sistema de ficheiros capaz de fazer armazenamento distribuído de dados, ao contrário do Hadoop com o HDFS. Como tal, a utilização do HDFS para o armazenamento de dados e de Spark para o processamento dos mesmos, ao contrário do MapReduce, torna-se a solução ideal para as organizações obterem resultados mais rapidamente (Ryza, *et al.,* 2015). A alocação dos recursos de processamento e memória de Spark no cluster Hadoop é via YARN que utiliza o *ResourceManager* para a distribuição de capacidade.

2.2. DATA SCIENCE

Data Science é o termo utilizado para a ciência que tem como objetivo a análise de dados e extração de conhecimento, por meio de conceitos de estatística, técnicas de *Data Mining* e algoritmos de *Machine Learning* (Jurney, 2013). Para Donoho, D. (2015), *Data Science* equivale a um superconjunto de disciplinas como a estatística, *Data Mining* e *machine learning*, que integrado com tecnologias de alta escalabilidade e grandes volumes de dados, procura extrair conhecimentos valiosos. A combinação deste conjunto de competências permite explicar a grande quantidade de dados em diferentes perspetivas através de tarefas de limpeza, preparação e visualização de dados (Horvitz, 2016).

As áreas de pesquisa do *Data Mining* e *Machine Learning* representam uma forte componente para os *Data Scientists (*Donoho, 2015). A crescente utilização de métodos capazes de extrair conhecimento num curto espaço de tempo para responder rapidamente às necessidades de mercado, obriga as organizações a possuir conhecimentos de técnicas *Data Mining* para uma otimização dos modelos de Machine Learning (Horvitz, 2016).

2.2.1. *Data Mining*

O *Data Mining* consiste na procura de padrões e modelos que estão implícitos nos dados, com o objetivo de encontrar relações desconhecidas que possam trazer valor para o negócio, através de um conjunto de algoritmos (Feng, *et al*., 2015). O *Data Mining* como uma das fases do processo de "Descoberta de Conhecimento em Bases de Dados", que inclui as componentes de recolha e pré-processamento de dados, bem como a interpretação dos resultados (Figura 5) (Hand, *et al*., 2001). As técnicas de *Data Mining* distinguem-se dos tradicionais métodos estatísticos, pela capacidade de usar métodos analíticos mais avançados, como inteligência artificial.

Figura 4 – Processo de descoberta de conhecimento.

(Fonte: Hand, D., Mannila, H., & Smyth, P. (2001). Principles of *Data Mining* Cambridge. MIT Press (Vol. 2001). https://doi.org/10.1007/978-1-4471-4884-5)

Tendo em conta que o volume de dados cresce exponencialmente ao longo do tempo e estimando que os dados produzidos em 2020 serão 44 vezes maior que em 2020, pelo que é imprescindível que

os modelos de *Data Mining* sejam capazes de interpretar e extrair conhecimento a partir destes grandes conjuntos de dados (Leskovec & Rajaraman & Ullman, 2014). O crescente volume e complexidade dos dados tornam os tradicionais métodos baseados em conhecimento ou pressupostos como impraticáveis, uma vez que exigem o conhecimento heurístico dos dados por parte dos analistas de negócio. A solução passa pela utilização de métodos baseados em dados, modelos *Data-Driven*, na qual se assume que uma observação que ocorre de forma consistente repetir-se-á em acontecimentos futuros (Dean, 2014). É facilmente percetível que quanto maior o volume de dados, melhor será a precisão nos resultados obtidos, pois o espaço de procura será maior e mais observações serão analisadas. Assim, é possível generalizar o conhecimento extraído de modo a que seja possível identificar relações desconhecidas no problema negócio (Hand, Mannila & Smyth, 2001).

Existem dois tipos de modelação que são utilizados em *Data Mining*: a modelação descritiva e modelação preditiva. A modelação descritiva tem como objetivo a identificação de propriedades intrínsecas aos dados, de modo a resumir grandes quantidades de informação, de forma a servir de auxílio às tomadas de decisão. Este método é utilizado para encontrar padrões ou tendências que auxiliem na compreensão dos dados (Jiawei, *et al*., 2012). Existem quatro tarefas utilizadas para descrever e resumir um determinado conjunto de dados, a segmentação, as regras de associação, o *link analysis* e a visualização (Ester, *et al*., 1996). A modelação preditiva permite prever um atributo num conjunto de dados, baseado nos outros atributos dos dados. No âmbito da modelação preditiva consiste em utilizar dados históricos para desenvolver um modelo que permita prever resultados de um espaço de interesse (Feng, et al., 2015). É necessário que cada objeto do conjunto de treino possua atributos de *input* e *output* (Santos et al., 2009). A variável a prever, denominada como *variable target,* é dada por cada observação.

Cada uma das tarefas de *Data Mining* engloba um conjunto de algoritmos de *Machine Learning* utilizados para extrair conhecimento a partir do conjunto de dados.

Nos últimos anos, o paradigma mudou, e a interação entre homem e computador alterou-se, pois, a crescente complexidade dos dados e as constantes evoluções disruptivas vistas no mercado, exigem uma resposta mais rápida nas tomadas de decisão (Jurney, 2013). Esta tendência do mercado veio reforçar a necessidade de tornar as máquinas de processamento mais autónomas e que permitissem reduzir a intervenção humana, por forma a acelerar o negócio a um menor custo. Cada vez mais se verifica a existência de ferramentas computacionais capazes de simular a capacidade de um ser humano para resolver os problemas. Atividades como memorizar, observar e explorar registos passados são essenciais para o sistema computacional aprender/otimizar as suas habilidades cognitivas e, assim, atingir a solução ótima do problema. Estes são resolvidos mediante algoritmos, métodos computacionais, que permitem ao sistema a capacidade de aprendizagem de uma forma autónoma. Esta capacidade de aprendizagem é o principal fator para um comportamento inteligente (Dean, 2014).

Existem diferentes métodos de aprendizagem, utilizados para a prospeção e extração de conhecimento a partir dos dados, como é o caso da aprendizagem supervisionada e não-supervisionada (Bengfort, et al., 2016). Cada uma destas técnicas oferece um conjunto de algoritmos que permitem a construção de modelos capazes de acrescentar grande valor ao negócio. Se os atributos e as classes que vão conduzir o processo de classificação dos dados são conhecidos à partida, utiliza-se aprendizagem supervisionada. Classifica-se como aprendizagem supervisionada pois os

dados são pré-classificados, de modo a que seja possível supervisionar a aprendizagem do modelo, sendo assim possível avaliar a capacidade da hipótese induzida para calcular novos valores futuros. Na aprendizagem não-supervisionada, a grande diferença para aprendizagem supervisionada é que em tarefas não-supervisionadas, ignora-se o atributo de *output* (Dean, 2014). Neste tipo de aprendizagem, desconhece-se a classificação a cada observação do conjunto de dados e pretende-se descobrir padrões de semelhança desconhecidos entre os dados. O objetivo passa por agrupar os dados através de uma determinada medida de semelhança (Bengfort, et al., 2016).

Cada tipo de aprendizagem é constituído por vários algoritmos, seguindo-se uma apresentação mais detalhada da aprendizagem supervisionada e não-supervisionada.

2.2.1.1. Aprendizagem Supervisionada

Refere-se aprendizagem supervisionada quando os dados, denominado conjunto de treino, são previamente conhecidos e são inseridos num algoritmo com o objetivo de prever um novo atributo para futuras instâncias (Hand, et al., 2001). Esta técnica pode ser definida como um conjunto de observações D = $\{(x_i, f(x_i)), i = 1, \dots, n\}$ em que f representa uma função desconhecida, um algoritmo supervisionado aprende uma aproximação *f* da função desconhecida (Fayyad, et al., 1996). Essa função aproximada f permite estimar o valor de f para novas observações x. De acordo com a natureza de *f*, existem duas situações possíveis: classificação e regressão. A classificação é uma função que assume valores discretos, tipicamente uma classe, não ordenada, como cores (verde, vermelho ou azul). A função de regressão assume valores contínuos, infinitos, como a idade ou tempos, e tem como principal objetivo mapear um conjunto de dados a uma variável de valor real. Existem diversos algoritmos que podem ser utilizados em aprendizagem supervisionada, como:

Árvores de Decisão

Árvores de Decisão são representações simples do conhecimento e um meio eficiente de construir classificadores que preveem informações úteis baseadas nos valores de atributos de um conjunto de dados. A estrutura de uma árvore de decisão é composta por nós, arcos e folhas (Fayyad, et al., 1996). Cada nó da árvore representa uma interrogação a uma variável do conjunto de dados. Os arcos correspondem a cada uma das respostas possíveis à interrogação criada, o que permite separar o conjunto de dados de acordo com alguns critérios de resposta. As folhas correspondem aos nós finais, nos quais já não existem mais interrogações possíveis. A partir das variáveis de *input*, são criadas um conjunto de regras que permitem isolar um subconjunto de observações que possuem valores idênticos para a variável de output. Os algoritmos de árvores de decisão mais utilizados são o C4.5 e o CART.

Support Vector Machine (SVM)

O algoritmo o *Support Vector Machine* (SVM) procura encontrar a melhor superfície de decisão, um hiperplano, por forma a conseguir separar os pontos tendo em conta a sua classe (previamente conhecida) (Jiawei, et al., 2012). o SVM procura mapear os pontos, representantes dos nossos atributos de dados no espaço, para que sejam divididos por um espaço livre que seja o mais amplo possível (Hand, et al., 2001). Assim, é possível prever que os novos dados pertençam a cada um dos conjuntos definidos. O mesmo autor indica que esta tarefa de classificação é baseada na Teoria de

Aprendizagem Estatística: sejam h um classificador e H o conjunto de dados. Durante o processo de aprendizagem, o algoritmo utiliza um conjunto de treino X, composto por n pares (x_i, y_i), para gerar um classificador particular $a \in H$. Os pontos que se encontram mais próximos ao hiperplano são denominados *Support Vectors* e a função de decisão que maximiza a separação dos dados é denominada de ótima.

Redes Neuronais

As redes neuronais como sistemas de classificação modelados segundo os princípios do sistema nervoso humano (Hand, et al., 2001). A unidade básica é o neurónio artificial, o perceptrão, responsável por simular o comportamento de um neurónio real como uma combinação linear de um conjunto de *inputs* e uma função não linear (função de ativação), reproduzindo um *output* final. As unidades, denominadas nós, encontram-se conectadas através de ligações, as quais têm associado um determinado peso. As redes neuronais são constituídas por diferentes níveis. A primeira camada apresenta os *inputs* para a rede sem qualquer ação de pré-processamento de dados. As camadas intermédias recebem os valores de entrada e executam a classificação das características. E a camada de *output* é responsável por transmitir os dados de saída da rede. Durante o processo de treino, os pesos vão sendo ajustados por forma a obter o output mais próximo do desejado. Em cada iteração do processo de treino, o *output* obtido é comparado com o *output* desejado, com base nesta análise os pesos são ajustados e ao longo do processo de aprendizagem, a rede apresenta uma melhor precisão na replicação dos resultados (Feng, et al., 2015).

2.2.1.2. Aprendizagem Não-Supervisionada

A aprendizagem não-supervisionada tem como objetivo descobrir características intrínsecas nos dados (Dean, 2014). Na aprendizagem não-supervisionada, não existe necessidade de que o conjunto de dados seja pré-classificado, não existindo processo de treino, ajustando-se o modelo aos dados de *input*. Ao contrário da aprendizagem supervisionada, este tipo de aprendizagem não requer o conhecimento prévio sobre as suas classes ou categorias.

Um caso particular de aprendizagem não-supervisionada são o *clustering*, e as regras de associação.

Clustering

A análise de *clusters,* ou segmentação*,* lida com a identificação de grupos nos dados de acordo com a semelhança entre os indivíduos (Fayyad, et al., 1996). Cada algoritmo de *clustering* é baseado num determinado critério de agrupamento e utiliza uma medida de distância, tipicamente a distância Euclidiana ou Manhattan, e uma medida de busca para encontrar a solução ótima que descreve os dados, de acordo com o critério de agrupamento selecionado (Bhattacharyya, et al., 2013). Quanto maior a similaridade dos dados *intra-clusters* e maior a diferença *inter-clusters*, mais fácil se torna a sua análise.

Os métodos *clustering* são classificados em duas categorias, métodos hierárquicos e métodos de partição (Fayyad, *et al.,* 1996). O método hierárquico subdivide os dados em múltiplos subgrupos, com uma certa decomposição hierárquica, utilizando um determinado critério, por forma a construir vários níveis de hierarquia e assim tornar-se mais simples a tarefa de escolher a partição mais adequada ao problema. Este divide-se em métodos aglomerativos e divisivos. Os métodos hierárquicos aglomerativos são construídos de forma ascendente, começando por agrupar os diferentes indivíduos

individualmente até que se obtenha um único *cluster* no final. Nos métodos hierárquicos as duas principais regras de agrupamento, que permitem medir as distâncias entre *clusters* são o *Single Linkage* e a *Complete Linkage* (Berkhin, 2002).

No *Single Linkage a distância* entre *clusters* é determinada pela distância dos indivíduos mais próximos de cada cluster. No caso do *Complete Linkage* a distância é calculada pela distância dos indivíduos mais afastados de cada *cluster*. Nos métodos hierárquicos divisivos, a decomposição hierárquica ocorre de uma forma ascendente, com os indivíduos a serem agrupados num único *cluster* inicialmente e a cada iteração vão sendo divididos, até que cada indivíduo representa um *cluster*.

Nos métodos de partição é criada uma partição inicial e os indivíduos vão sendo progressivamente realocados aos novos *clusters* (Dean, 2014). Um dos algoritmos mais utilizado para realizar a partição é o algoritmo *K-Means*. Este algoritmo de *clustering* procura, de forma iterativa, agrupar os dados semelhantes com base num determinado critério. Inicialmente, o algoritmo recolhe todos os atributos de *input*, por forma atribuir a cada valor uma classe. O utilizador é responsável por introduzir o número inicial de *clusters* que pretende, o valor k. De modo a conseguir gerar as classificações, o algoritmo compara a proximidade de cada indivíduo à média do *cluster* onde foi alocado inicialmente. A proximidade é calculada através da soma dos quadrados das diferenças de cada indivíduo à média dos *clusters*. A cada iteração do algoritmo, o indivíduo fica associado ao *cluster*, cujo *centroide* está mais próximo e no fim, é calculado o ponto médio de cada cluster. Este processo é repetido até que a função objetivo convergir, construindo os k *clusters* os mais compactos e separados possíveis.

Regras de Associação

As Regras de Associação como padrões descritivos que procuram identificar regras que relacionem uma determinada conclusão com um conjunto de condições (Jiawei, 2012). Considerando que $I = \{i1, i2, \dots, in\}$ é um conjunto de atributos chamados produtos, S é uma base de dados de transações $S = \{t1, t2, \dots, tm\}$ onde t é uma vetor de produtos. Uma regra de associação é definida pela implicação $X \Rightarrow Y$ em que $X, Y \subseteq I$ e $X \cap Y = \emptyset$ (Woo, 2015). 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. Segundo este formato gera-se uma grande quantidade de regras candidatas, pelo que existem dois grupos de medidas de avaliação de conhecimento que auxiliam na avaliação das regras obtidas: medidas de interesse objetivas e subjetivas (Tan, *et al*., 2005). As medidas subjetivas requerem o conhecimento do analista sobre o problema, o que implica sempre uma subjetividade neste tipo de classificação. As métricas adjacentes nesta medida são a *Unexpectedness* e *Actionability* (Jiawei, 2012). Na métrica *Unexpectedness* as regras são consideradas interessantes se o analista do negócio desconhecer as relações obtidas. No caso da métrica *Actionability*, considera-se uma regra de associação interessante se for possível de aplicar no negócio e consiga trazer vantagem competitiva. Nas medidas objetivas, o grau de interesse das regras é representado pela confiança e pelo suporte.

O suporte de uma regra indica o número de transações que integram a parte antecedente X e consequente Y da regra:

$$\textbf{Suporte}(X \rightarrow Y) = P(X \cup Y)$$

Esta medida representa a frequência com que os *itemset*s ocorrem no conjunto de dados. Pelo que se um *itemset* apresentar uma frequência muito baixa, algumas regras potencialmente interessantes poderão não ser conhecidas (Jiawei, 2012). O valor de suporte varia entre o e 1.

A confiança da regra consiste na probabilidade de ocorrer uma transação que contém o produto X, mas que também contém o produto Y:

$$\textbf{Confiança}(\mathrm{X} \rightarrow \mathrm{Y}) = \frac{\mathrm{P}(\mathrm{X} \rightarrow \mathrm{Y})}{\mathrm{P}(\mathrm{X})} = \frac{\text{Suporte } (\mathrm{X} \cup \mathrm{Y})}{\text{Suporte}(\mathrm{X})}$$

Através destas duas medidas de interesse, é possível conhecer a força de uma regra (confiança) e a sua significância estatística (suporte). Ambas as medidas devem ser tidas em conta em simultâneo, pois o suporte poder ser um valor elevado (a percentagem de registos que satisfazem a regra) mas a confiança ser reduzida, ou seja, o número de registos em que a ocorrência acontece ser baixo (Woo, 2015).

Existem vários algoritmos criados para a extração de *itemsets* frequentes e regras de associação, que utilizam estas duas medidas de interesse para a validação das regras. Os mais utilizados são o algoritmo *Apriori* e *FP-Growth*.

O algoritmo *Apriori* baseia-se no princípio que qualquer subconjunto de *itemsets* frequentes deve ser um *itemset* frequente, isto é, se uma transação contém {cerveja, fralda, nozes} também contém {cerveja, fralda} (Kaur, *et al*., 2016). De acordo com um suporte e uma confiança mínima, o algoritmo inicia-se com a identificação de todos os *itemsets* individuais frequentes, de modo a que, cada item seja um membro do conjunto de *itemsets* candidatos e que sejam eliminados todos os *itemsets* com medidas inferiores às definidas inicialmente. Na segunda iteração, os *items*, cujos parentes não tenham sido eliminados anteriormente, são organizados em *itemsets* de tamanho 2. O processo iterativo continua até que não existam mais *itemsets* válidos. O algoritmo segue uma abordagem *breadth-first*, pois percorre todo o conjunto de dados à procura de todas as combinações possíveis e a calcular a frequência de cada combinação encontrada, e é representado através de uma estrutura em árvore.

Abordagem *breadth-first* utilizada pelo algoritmo *Apriori* para encontrar todos os *itemsets* possíveis, pode tornar-se um processo lento e pesado (Jiawei, *et al*., 2012). Este algoritmo apresenta muitas limitações quando confrontado com muitos itens distintos no conjunto de dados, pois necessita de percorrer diversas vezes todo o conjunto de dados para calcular o suporte dos *itemsets* frequentes candidatos. Pelo que foi necessário criar novas alternativas, sem a necessidade de percorrer repetidamente todos os *itemsets* como o algoritmo *FP-Growth* (Han, *et al*., 2000).

O algoritmo FP-Growth segue uma abordagem *Depth-first search*, ou seja, realiza o espaço de procura em profundidade (Han, *et al*., 2000). Este algoritmo é assente em dois pontos fundamentais: em primeiro constrói uma estrutura de dados, a FP-Tree, o que permite gerar um menor número de *itemsets* candidatos, comparativamente com o *Apriori* e, em segundo, ocorre a extração dos *itemsets* mais frequentes.

A construção da *FP-Tree* ocorre em duas fases, como verificado pela figura 6:

1) O algoritmo percorre todo o conjunto de dados e identifica os *itemsets* de tamanho 1 com suporte superior ao mínimo definido, removendo os restantes. Após isto, os *itemsets* são organizados, por ordem decrescente de acordo com o seu valor de suporte.

2) É construída uma árvore de padrões frequentes. De modo a organizar os *itemsets* de forma compacta, os nós da árvore de padrões frequentes correspondem a *items* de tamanho 1. É com base nesta estrutura que são extraídos todos os conjuntos de *items* frequentes

Input: Dada uma base de dados B;
Definindo um suporte mínimo σ;
Output: Todos os *itemsets* com suporte maior que σ;
Percorrer o conjunto de dados B uma vez;
Calcular F, o conjunto de itens frequentes e o suporte de cada;
Ordenar F em ordem decrescente de suporte F list;
Criar a raiz da FP-Tree, T, e atribuir o valor *null*;
Para cada transação t ∈ B:
  Selecionar os itens frequentes em t;
  Ordená-los de acordo com F list;
  Seja ([i\Its],T) os itens frequentes ordenados em t;
  Chamar *insere_arvore*([i\Its],T)
Fim
Função *insere_arvore*([i\Its],T) ;
Se T *tem um filho* N *rotulado i* **então**
  Incrementar N com 1;
Fim
Caso contrário
  Cria um novo nó, N, com valor 1, com o pai ligado a T, e o nó ligado aos restantes nós com o mesmo rótulo i;
  Se Its é não vazio **então**
    Chamar *insere_arvore*(Its,N);
  Fim
Fim

Figura 5 – *Pseudocode* do algoritmo *FP-Tree*

Analisando a seguinte lista de transações, verificam-se 7 transações distribuídas por 5 *items*:

1. {a, b}
2. {b, c, d}
3. {a, c, d, e}
4. {a, d, e}
5. {a, b, c}
6. {a, b, c, d}
7. {b, c, e}

A construção da *FP-Tree* inicia-se com a criação da raiz da árvore com o valor *null* e para cada transação, os itens são ordenados segundo o seu valor de suporte. Verificando a primeira transação {a, b}, a raiz atribuída será o valor *null,* gerando *null* $\rightarrow$ a $\rightarrow$ b. Na segunda transação, embora não partilhe o mesmo

prefixo que a primeira transação, o item {b} coexiste em ambas as transações, pelo que é estabelecida uma ligação que possibilita o cálculo da frequência (suporte) de b. No caso da terceira transação, esta partilha o mesmo prefixo que a primeira transação, o item {a}, pelo que o *itemset null* $\rightarrow$ a $\rightarrow$ c $\rightarrow$ d sobrepõe-se ao registo da primeira transação e a frequência de a é incrementada em 1 valor. O algoritmo termina quando todas as transações forem processadas, extraindo a *FP-Tree*, como representada na figura 7:

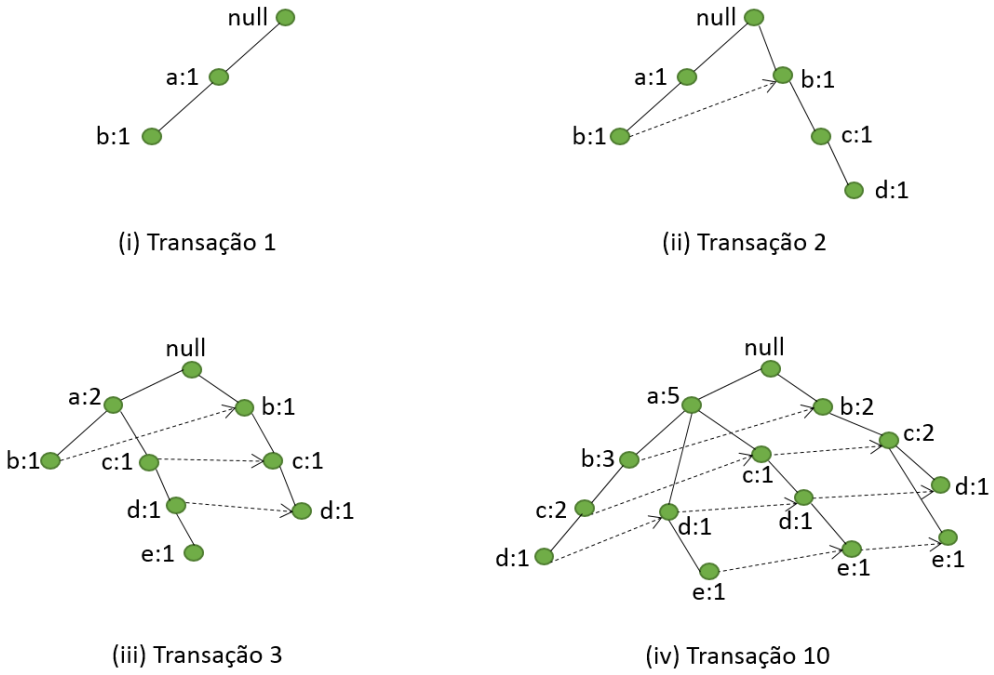

Figura 6 – Construção da FP-Tree

Para a extração dos *itemsets* frequentes, o algoritmo constrói novos subconjuntos para cada um dos prefixos de cada item (**a, b, c, d, e**). Tratando-se de um algoritmo *bottom-up,* os *items* com maior frequência encontram-se na base da árvore, pelo que serão analisados primeiro. Na figura 8, está representada a árvore que mostra o conjunto de *items* associado ao prefixo **e**:

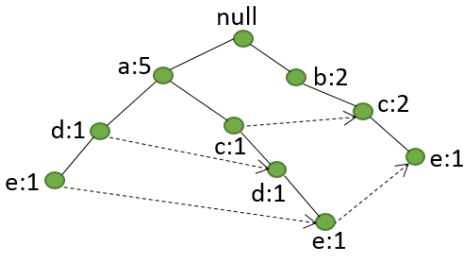

Figura 7 – Subconjunto dos *itemsets* com o prefixo e

É a partir destas estruturas que o algoritmo *FP-Growth* gere os *itemsets* frequentes. Inicialmente, é construída uma nova estrutura, denominada *FP-Tree* condicional, com o objetivo de atualizar o suporte do *itemset* associado a cada prefixo e eliminar os itens pouco frequentes, como demonstra figura 9:

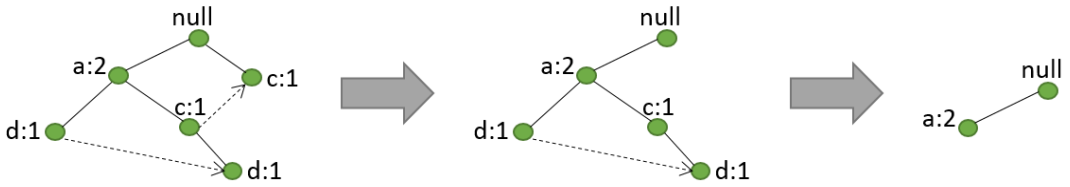

Figura 8 – *FP-Tree* condicional de *e*

Definindo um suporte mínimo de 2, verifica-se que o item {**b,e**} não é um item frequente, pelo que b é removido da árvore. Na transação 2 o prefixo **e** não é considerado, pelo que na segunda iteração para a construção do *FP-Tree Condicional*, esta transação não é considerada para a análise. Deste modo, existem dois *itemsets* em análise, o {**e,a,d**} e {**e,a,c,d**}, contudo, este último *itemset,* como visível na iteração 2 da construção da *FP-Tree,* não tem o suporte mínimo de 2, pelo que não é frequente. Este processo deve ser repetido para todos os prefixos correspondentes a *items* frequentes. Pela análise da tabela 2 verificam-se os conjuntos de *itemsets* frequentes extraídos:

Prefixos	***Itemsets* Frequentes**
{e}	{e}; {d,e}; {a,d,e}; {c,e}; {a,e}
{d}	{d,}; {c,d}; {b,c,d}; {a,c,d}; {b,d}; {b,c,d}; {a,d}
{c}	{c}; {b,c}; {a,b,c}; {a,c}
{b}	{b}; {a,b}
{a}	{a}

Tabela 2 – *Frequent Itemsets (FP-Growth)*

O algoritmo *FP-Growth* apresenta uma maior complexidade na sua implementação face ao algoritmo *Apriori*, contudo, apresenta tempos de processamento superiores e desempenhos mais eficazes (Kaur, et al., 2016). O algoritmo *Apriori* que face ao elevado número de *itemsets* candidatos gerados e à necessidade de percorrer todo o conjunto de dados várias vezes, apresenta processos computacionais mais pesados face ao *FP-Growth*.

As regras de associação são uma técnica aplicável em muitas indústrias, embora seja no mercado retalho que a sua prática seja mais comum. Por exemplo, é recorrente utilizar-se algoritmos de Regras de Associação para se criar recomendações em determinadas cadeias de supermercados, de forma a concluir quais os produtos que são comprados em conjunto. A partir do momento em que foi possível às organizações do retalho reunirem dados referentes ao histórico de compras realizadas pelos seus clientes, a estratégia de marketing nesta indústria mudou de paradigma, com as organizações a promoverem recomendações e a alocarem estrategicamente os seus produtos, quer na web quer em loja (Silvers, 2012).

3. METODOLOGIA

Primeiramente procedeu-se à recolha de dados relativos ao histórico de transações de várias lojas de supermercado (https://www.kaggle.com/c/acquire-valued-shoppers-challenge) e ao carregamento do ficheiro no HDFS da máquina virtual da Cloudera CDH 5.8. Para esta etapa utilizou-se a ferramenta Sqoop e MySQL.

Não foi possível a utilização de um cluster Hadoop com *hardware* capacitado para o processamento de um grande volume de dados. Como tal, foi utilizada a versão demo Cloudera QuickStart VM para a realização do projeto, com todas as limitações inerentes a esta opção. Desta feita, foi extraída uma amostra estratificada dos dados, de acordo com algumas dimensões, para uma tabela em Hive, permitindo implementar o projeto com um menor volume de dados na mesma infraestrutura.

Utilizando Impala, foram analisados os dados da amostra, com vista a identificar as inconsistências dos mesmos e, consequentemente definir as regras de qualidade de dados. Com a mesma ferramenta foram criados um conjunto de *KPIs* onde foi possível verificar alguns detalhes dos dados em análise, através de um *dashboard*.

Posteriormente, procedeu-se à definição de variáveis. Algumas variáveis foram excluídas da amostra pois foram consideradas irrelevantes para o negócio, ao passo que foram geradas novas variáveis, com o objetivo de enriquecer o conjunto de dados para aplicação de técnicas de *clustering*.

Com vista a uma melhor compreensão dos hábitos de consumo do cliente, foi aplicada uma segmentação por perfil de consumidor. Em seguida, iniciou-se a implementação do algoritmo de *Frequent Pattern Mining* com os dados de cada *cluster*, a serem formatados para um registo de um carrinho de compras, na qual, cada linha do conjunto de dados representava a sequência de produtos comprados no dia por cada cliente, isto é, cada linha da tabela representa uma transação e cada coluna um produto adquirido. Com base no formato do *dataset* gerado, foi possível aplicar o *Market Basket Analysis*, com a implementação do algoritmo *FP-Growth*. Por fim, extraíram-se os resultados gerados, de modo a identificar quais os produtos que são comprados simultaneamente. O processo descrito encontra-se ilustrado na figura 10:

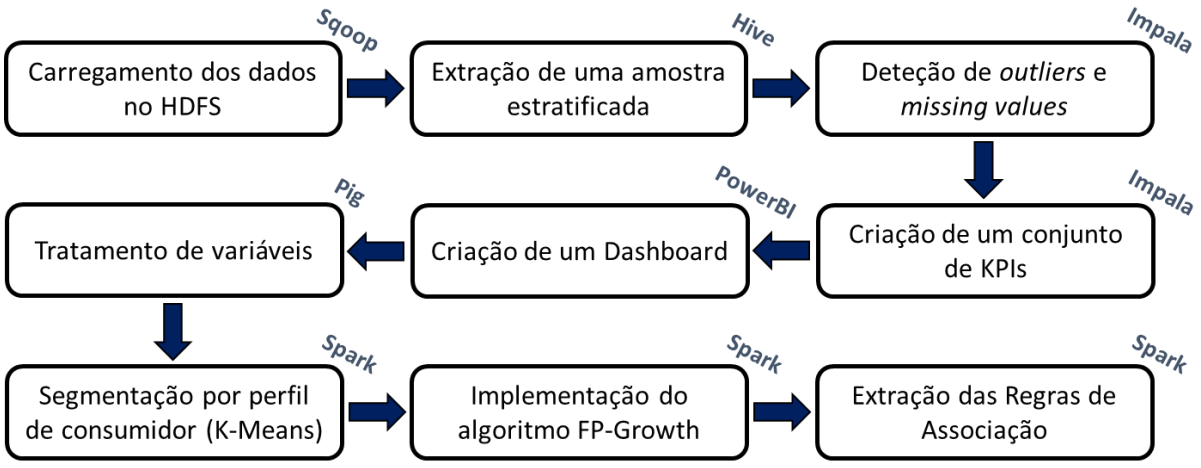

Figura 9 – Metodologia do Projeto

3.1. CARREGAMENTO DOS DADOS NO HDFS

Foram recolhidos dados de várias lojas de supermercado com informação relativa ao histórico de transações dos consumidores. Para o estudo em causa, foram recolhidos cerca de 15349956 registos divididos por 134 cadeias de lojas entre o mês de março de 2012 e fevereiro de 2013. Na figura 11 está representado a variação do número de transações ao longo dos meses, na qual se verifica que o número se manteve consistente ao longo do período em análise.

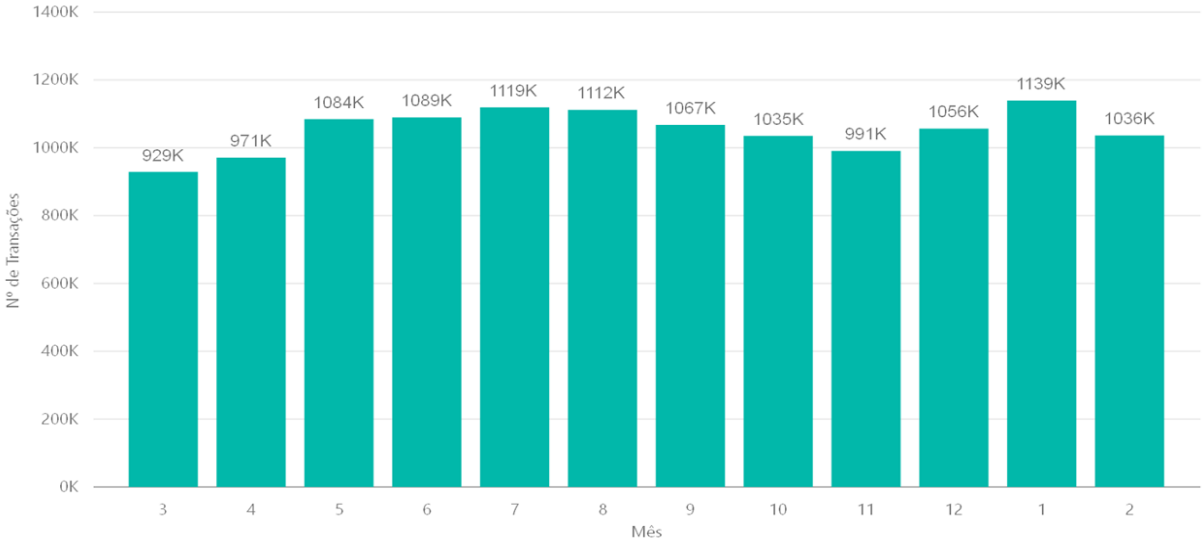

Figura 10 – Número de transações ao longo dos meses

Para o carregamento dos dados no HDFS, foi utilizada a ferramenta Sqoop para transferir o conjunto de dados para o Hadoop. Inicialmente foi criada um *schema* de dados no MySQL, e em seguida realizou-se uma importação de dados usando Sqoop para carregar os dados no HDFS. A conexão com o sistema de armazenamento do Hadoop foi feita com recurso a uma conexão JDBC.

Caso existisse a necessidade de lidar com dados em *streaming*, seriam utilizadas as ferramentas Apache Kafka e Flume que possibilitariam o carregamento de dados em *streaming*, em *real-time*, no Hadoop.

A base de dados original era constituída por 12 variáveis, referentes a quatro perspetivas: Cliente, Loja, Produto e Fornecedor. Em seguida serão apresentados os campos da base de dados:

1. **ID** – Valor único representativo do cliente;
2. **Store** – Valor único representativo de uma loja;
3. **Market** – Valor único representativo de uma região;
4. **Industry** – Valor único representativo da indústria de produto;
5. **Category** – Valor único representativo da categoria de produto;
6. **Company** – Valor representativo do fornecedor;
7. **Brand** – Valor único representativo da marca do produto;
8. **Date** – Data da transação;
9. **Product_Size** – Unidade de transação;
10. **Product**_Measure – Unidade de medição do produto;
11. **Purchase_Quantity** – Quantidade de unidades compradas por produto;
12. **Purchase_Amount** – Valor de venda por produto.

3.2. EXTRAÇÃO DE UMA AMOSTRA ESTRATIFICADA

Como referido anteriormente, por questões de limitação do cluster Hadoop, foi extraída uma amostra com vista a otimizar a performance na criação do algoritmo. Recorrendo ao *Apache Hive* (código SQL está disponível no anexo 8.2) a abordagem utilizada consistiu numa amostra estratificada aplicada nas seguintes variáveis:

- *Date* – Foram extraídos dados a partir do mês de outubro de 2012;
- *Store* – Aleatoriamente foram selecionadas 40 cadeias de lojas, condizentes a 30% do número total de cadeias de lojas (134).

Assim, foi assumida uma amostra de 4340560 de registos. O código SQL utilizado para a extração da amostra encontra-se disponível no anexo 8.2.

3.3. DETEÇÃO DE OUTLIERS E MISSING VALUES

A fase de exploração dos dados visa conhecer o conjunto de dados e identificar possível ruído intrínseco. Para este efeito, a pré-análise e o tratamento dos dados foi implementado com base nas ferramentas *SQL* do *Hadoop,* como *Apache Impala e Hive.* Inicialmente a consulta de dados foi direcionada para identificação de problemas, como dados inconsistentes, *outliers* ou *missing values.* Utilizando a amostra extraída anteriormente, foram criadas um conjunto de *queries ad-hoc* em Impala, para a definição das regras de *Data Quality*.

No caso do presente estudo, foi utlizado o método estatístico do desvio padrão sobre a frequência de cada valor para a identificação de *outliers*. Esta técnica consiste em definir um *threshold* (limite) máximo e mínimo baseado na média e desvio-padrão para cada variável. Os valores fora do limite de interesse apresentam uma materialidade muito baixa e encontram-se fora da distribuição normal, pelo que são removidos dos conjuntos de dados. Para a deteção dos *missing values*, com exceção da variável *Date*, foi decidido que todos os valores iguais ou inferiores a 0 seriam considerados como *missing values.*

Tendo em conta todas as variáveis, foram excluídas 57729 observações (1,33%) resultando num total de 4282831 registos. Estas observações são assim consideradas como dispensáveis, para a análise pretendida, pois foi considerado que não se iria perder informação relevante. Desta feita, é possível desenvolver modelos mais fiéis à distribuição real dos dados, reduzindo a sua complexidade computacional, e facilitar a interpretação dos padrões extraídos pelo modelo. Esta fase torna-se vital no projeto pois o desempenho do modelo criado, posteriormente, é geralmente afetado pelo estado dos dados.

3.4. CRIAÇÃO DE UM CONJUNTO DE KPIS

Foram definidos indicadores de desempenho, listados na tabela 3, que incluem a análise às diferentes dimensões do conjunto de dados, como clientes, lojas, produtos e tempo. Esta tarefa tem como objetivo a sintetização de um conjunto de informação complexa e que permita ganhar mais conhecimento para uma melhor tomada de decisão, fornecendo informação e feedback útil à instituição para uma gestão mais adequada visando a melhoria contínua da mesma.

Dimensão	Key Performance Indicator
Cliente	Evolução mensal do número médio de clientes por dia
Cliente	Número de novos clientes por mês
Cliente	Quantidade de produtos por transação (%)
Região	*Top/Bottom* 5 Volume de vendas por cada região
Região	*Top/Bottom* 5 Volume de negócio por cada região
Loja	*Top 10* Volume de vendas por cada loja de supermercado
Loja	*Top 10* Volume de negócio por cada loja de supermercado
Produto	Top 10 Produtos mais vendidos
Produto	*Top/Bottom* 5 Lista de produtos por preço
Fornecedores	Evolução mensal do número de fornecedores
Tempo	Produtividade por dia de semana

Tabela 3 – Lista de *KPIs*

A análise dos indicadores referidos tem como objetivo compreender alguns dos resultados obtidos durante o período em análise. Referente à dimensão cliente, é importante perceber se o número de clientes tem vindo a variar durante os meses e, em caso afirmativo avaliar as causas. A evolução do valor médio gasto por cliente tem como intuito demonstrar se existe algum período do ano mais propício para um maior consumo por parte do cliente. Relaltivamente à dimensão lojas de supermercado, procurou-se identificar quais as lojas com melhores resultados ao nível de vendas e de receita financeira. Considerando que existem 40 lojas, os gráficos são filtrados para as 10 organizações mais valiosas. A análise de produtos permite perceber quais os bens preferenciais dos consumidores, e se o seu valor de mercado tem influência no volume de vendas de cada produto. Posteriormente à análise do produto, verifica-se a produtividade de cada fornecedor, com vista a concluir se o número de fornecedores tem vindo a diminuir ou aumentar. Por último, foi criada uma análise temporal com o fim de identificar se existe um dia da semana preferencial dos clientes.

Para a elaboração do *dashboard* foi utilizado a ferramenta PowerBI da Microsoft que através de um conector ODBC foi possível aceder aos KPIs, criados no Impala. No capítulo dos resultados e discussão será apresentado o *dashboard* criado para a análise dos indicadores de desempenho.

3.5. TRATAMENTO DE VARIÁVEIS

Nesta etapa pretende-se melhorar a qualidade dos dados com o objetivo de preparar os dados para o processo de extração de conhecimento, com a adição de novas variáveis que possibilitam maximizar a extração de informação relevante dos dados.

Através das ferramentas Hive, HCatalog e Pig foram adicionadas doze novas variáveis com vista a enriquecer a análise dos dados. De forma a conectar a base de dados em Hive para a transformação de dados em Pig foi necessário importar um conjunto de ficheiros do tipo Jar e importar o ficheiro de configuração XML do Hive (anexo 8.4 na Figura 12).

- **Visit_Freq** – Número de visitas do cliente a uma das lojas;
- **Monetary** – Valor de compra em produtos;
- **Total_Articles** – Número total de produtos adquiridos;
- **Recency_Weeks** – Número de dias desde última visita;
- **Year** – Valor do ano;
- **Quarter** – Valor do trimestre do ano;
- **Month** – Valor do mês do ano;
- **Day** – Valor do dia do mês;
- **WeekDay** – Valor do dia da semana;
- **Transaction_ID** – Valor único de transação de um cliente, por dia, numa determinada loja;
- **Transaction_Quantity** – Número de itens comprados por cada transação;
- **Transaction_Amount** – Valor total pago por cada transação.

As variáveis “Visit_Freq”, “Monetary”, “Total_Articles” e “Recency_Weeks” foram criadas com vista a realizar uma segmentação por perfil de consumidor.

Neste momento, a amostra contém um total de 24 variáveis, sendo que algumas poderão não ser interessantes para incluir no algoritmo. É necessário selecionar um subconjunto de variáveis mais relevantes para a construção do modelo, de acordo com um determinado critério de avaliação. O objetivo passa por remover variáveis irrelevantes e redundantes, aumentando a eficiência, desempenho e precisão nos resultados durante a fase de extração de conhecimento.

Nem todas as variáveis são uteis para a identificação de padrões nos bens de consumo dos clientes, pois poderão não apresentar mais valias para a análise do problema. As variáveis “Brand” e “Company” foram retiradas pois não seria valioso identificar alguma marca ou consumidor preferencial nos produtos transacionados em conjunto. A variável “Date” foi decomposta nas variáveis “Year”, “Quarter”, “Month”, “Day” e “Weekday”, permitindo um maior detalhe da variável. Relativamente às variáveis de produto “Product_Size” e “Product_Measure”, estas foram consideradas como irrelevantes face ao problema em questão. No caso das variáveis “Purchase_Quantity” e “Purchase_Amount”, estas foram agrupadas por “Transaction_ID”, resultando na quantidade de produtos transacionados e valor total por cada compra de cliente.

Na tabela 4 estão representadas as variáveis utilizadas para a implementação dos algoritmos finais:

Variável	Tipo de Variável
Customer_ID	Nominal
Visit_Freq	Ordinal
Monetary	Ordinal
Total_Articles	Ordinal
Recency_Weeks	Ordinal
Transaction_ID	Nominal
Store	Nominal
Market	Nominal
Year	Ordinal
Quarter	Ordinal
Month	Ordinal
Day	Ordinal
WeekDay	Ordinal
Dept	Nominal
Category	Nominal
Product_Amount	Contínua
Transaction_Quantity	Discreta
Transaction_Amount	Contínua

Tabela 4 – *Feature Selection*

3.6. SEGMENTAÇÃO POR PERFIL DE CONSUMIDOR

Após o pré-processamento dos dados, inicia-se o processo de segmentação. Para completar o processo, foi implementado o algoritmo *k-Means* em *Spark* usando a biblioteca *MLlib*. A implementação do algoritmo *k-Means* exige a seguinte parametrização:

- *k* – número de *clusters*;
- *maxIterations* – número máximo de iterações;
- *featuresCol – feature vector* (as variáveis selecionadas).

No caso do número máximo de iterações foi utilizado o valor *default*, 30 iterações com vista à convergência do algoritmo. Este parâmetro corresponde a uma das medidas de paragem do algoritmo, pois a interrupção ocorre quando a oscilação dos centroides deixa de ser significativa ou quando se atinge o valor máximo de iterações.

A análise de *clusters* relativa ao perfil do consumidor tinha como objetivo compreender quais os clientes mais fiéis à cadeia de supermercado e identificar os produtos mais procurados por cada grupo, para no fim, ser possível criar uma recomendação de produtos para cada cluster. Para tal foram selecionadas as seguintes variáveis: "Visit_Freq", "Monetary", "Total_Articles" e "Recency_Weeks". As restantes variáveis não foram incluídas pois não acrescentariam valor ao processo de segmentação, pois não caracterizam o perfil do cliente.

Antes de se implementar o algoritmo *k-Means* é necessário determinar o número correto de *clusters.* Por questões de limitação de tempo e considerando o volume de dados da amostra, o número ideal de clusters não foi calculado no cluster Hadoop, tendo sido utilizado código R para se criar o gráfico do cotovelo, este método compara a soma do erro quadrático (SSE) para o número ideal de *clusters.* Uma opção para determinar o número ótimo de clusters para o algoritmo K-Means através do Spark MLlib é a implementação da função WSSE (Unyelioglu, 2016). A SSE corresponde à soma das distâncias ao quadrado entre cada individuo e o centróide do *cluster*. Por norma, à medida que o número de *clusters* aumenta, verifica-se um SSE cada vez menor pois os *clusters* têm uma menor dimensão. Esta visualização oferece um método simples para determinar a variável *k* para o algoritmo *K-Means*. O número ideal de *clusters* obtém-se quando o ponto a partir do qual o declive passa a ser menos acentuado. Criou-se o gráfico para 1 a 10 *clusters*, gerando o seguinte gráfico do cotovelo:

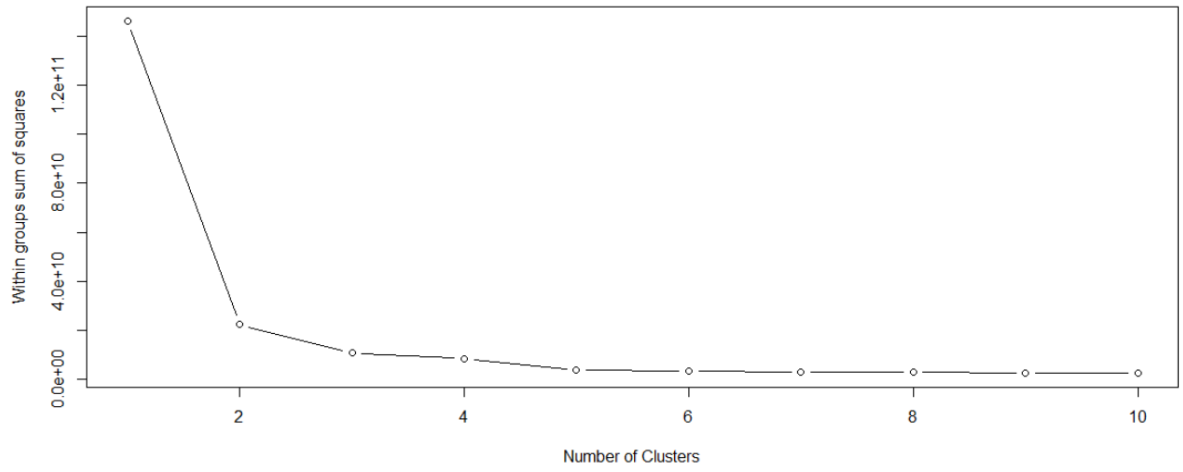

Figura 11 – Gráfico do Cotovelo

Através do gráfico do cotovelo, observa-se que o maior decréscimo ocorre no valor 4, pelo que o número ideal de *clusters* para a realização da segmentação por perfil de consumidor é 3. Assim, foi adicionada uma nova variável ao conjunto de dados, com o número do *cluster* a que cada cliente pertence.

3.7. IMPLEMENTAÇÃO DO ALGORITMO FP-GROWTH

A componente do *"Market Basket Analysis"* tem como intuito compreender os diferentes produtos que os clientes costumam comprar numa única transação. Esta engloba um conjunto de técnicas que permitem prever o carrinho de compras de cada cliente, e assim perceber quais os produtos que, normalmente, são comprados em conjunto. Através da extração deste conhecimento é possível a criação de várias estratégias de marketing, como:

- Organização do espaço de venda;
- Criação de campanhas de promoção de produtos;
- Criação de cartões de fidelidade de clientes.

Para tal, recorreu-se à implementação do algoritmo *Frequent Pattern Growth* para identificar quais os produtos que são normalmente transacionados em conjunto (o código Scala foi disponibilizado no anexo 8.6). Para cada grupo de clientes identificado no passo anterior foi aplicado o algoritmo referido, através das tarefas ilustradas na figura 14:

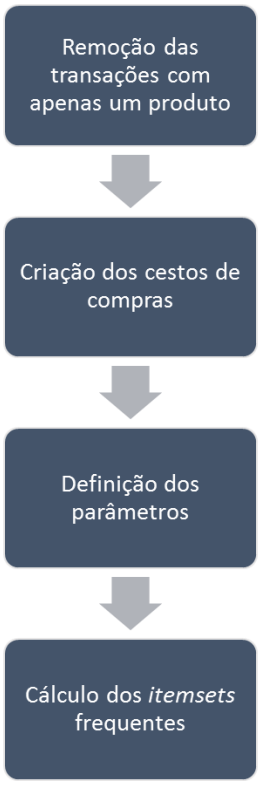

Figura 12 – Tarefas executadas para a implementação do FP-Growth

Inicialmente, foi necessário remover do conjunto de dados as transações com apenas um produto, pois foi entendido que não faria sentido incluir produtos, habitualmente, transacionados individualmente e assim não correr o risco de enviesar os resultados. Posteriormente, o *schema*

original dos dados foi transformado para o formato desejado, com os produtos a serem agrupados com base na chave da transação (criação dos *baskets*).

Por fim, foi possível definir as métricas de interesse para gerar os *itemsets* frequentes. Como já referido, o algoritmo foi validando as regras utilizando métricas objetivas:

- Suporte
- Confiança

Para cada cluster foi definido um valor mínimo para cada métrica, sendo assim possível determinar quais os produtos que são frequentemente transacionados em conjunto e extrair as regas de associação.

3.8. EXTRAÇÃO DE REGRAS DE ASSOCIAÇÃO

Por fim, foram extraídas as regras de associação através do algoritmo *FP-Growth*, em Spark MLlib. Para a extração das regras de associação foram utilizadas três variáveis: "Transaction_ID", "Product_ID" e "Customer_Cluster". Posto isto, todo o conjunto de dados foi transformado, com os dados relativos aos produtos a serem agrupados por cada "Transaction_ID", como exemplificado pela tabela 5, e assim criar os cestos de compra:

Transaction_ID	Lista de Produtos
1	9909, 3504
2	5616, 9909, 3504
3	9909
4	5616, 7076
5	9909, 5616

Tabela 5 – *Baskets format*

Em seguida, todas as transações com apenas um único produto, foram removidas da análise. Analisando a tabela acima, a transação número 3 foi eliminada, pois a presença de transações com produtos sem qualquer correlação com outros, poderia enviesar o modelo das regras de associação.

De forma, a analisar a qualidade das regras extraídas, foram definidos dois parâmetros para garantir a qualidade das regras extraídas. Os parâmetros utilizados foram: valor mínimo de suporte e o valor mínimo de confiança. Os valores utilizados no algoritmo *FP-Growth* foram:

- Suporte Mínimo: 10%
- Confiança Mínimo: 50%

De acordo com as métricas definidas, será possível concluir que as regras extraídas ocorrem em 10% das transações e que em 50% das transações que contêm o produto X, também contém o produto Y.

Como visto no capítulo anterior, as medidas de interesse são calculadas com base na frequência dos *itemsets.* Pegando no exemplo da tabela de transações representada na tabela 8, e por forma a avaliar a regra "Quem compra 9909, também compra 3504", as medidas de interesse, seriam calculadas da seguinte forma:

- Total de Transações: 4
- Frequência do Antecedente: 4
- Frequência do Consequente: 2
- Frequência do *itemset*: 2
- Suporte: 2/5
- Confiança: 2/4

Foram apenas consideradas 4 transações, pois os carrinhos de compras com apenas um único produto, não foram considerados no algoritmo. Assim, de acordo com os parâmetros definidos, são extraídas as seguintes regras:

{9909} → {3504} (0,66)
{3504} → {9909} (1)
{9909} → {5616} (0,5)
{5616} → {9909} (0,66)
{3504} → {5616} (0,5)
{9909, 5616} → {3504} (0,5)
{3504, 5616} → {9909} (1)
{3504, 9909} → {5616} (0,5)

A regra [5616 -> 3504] embora satisfaça o valor mínimo de suporte, apresenta um nível de confiança de 30%, pelo que não respeita a força desejada.

4. RESULTADOS E DISCUSSÃO

No capítulo anterior abordou-se a metodologia para a construção de um sistema de apoio à decisão na indústria do retalho com tecnologias *Big Data*. Foram definidos um conjunto de indicadores de modo a avaliar os atuais resultados da cadeia de supermercado, bem como criada uma segmentação por perfil de consumidor com o objetivo de compreender as correlações existentes entre os produtos transacionados em conjunto, por cada *cluster*.

4.1. ANÁLISE DE KEY PERFORMANCE INDICATORS (KPIS)

Primeiramente foram analisados um conjunto de KPIs, com vista a monitorizar e otimizar de forma contínua a qualidade de negócio do supermercado, em estudo. Estes indicadores poderão ser valiosas métricas para comparar o desempenho da organização em diferentes perspetivas de análise.

Na figura 15 está ilustrado o *dashboard* criado para a análise dos indicadores de desempenho. O dashboard é composto por 11 reports distribuídos pelas perspetivas de cliente, região, loja, fornecedor e tempo.

Figura 13 – *Dashboard* executivo

Inicialmente, analisaram-se os indicadores de acordo com a dimensão cliente. A primeira análise, demonstrada na figura 16, procura relacionar a variação do número médio de clientes com a média do valor pago por cada cliente numa única transação. Na figura seguinte os meses estão organizados consoante o ano do mês (outubro de 2012 para fevereiro de 2013).

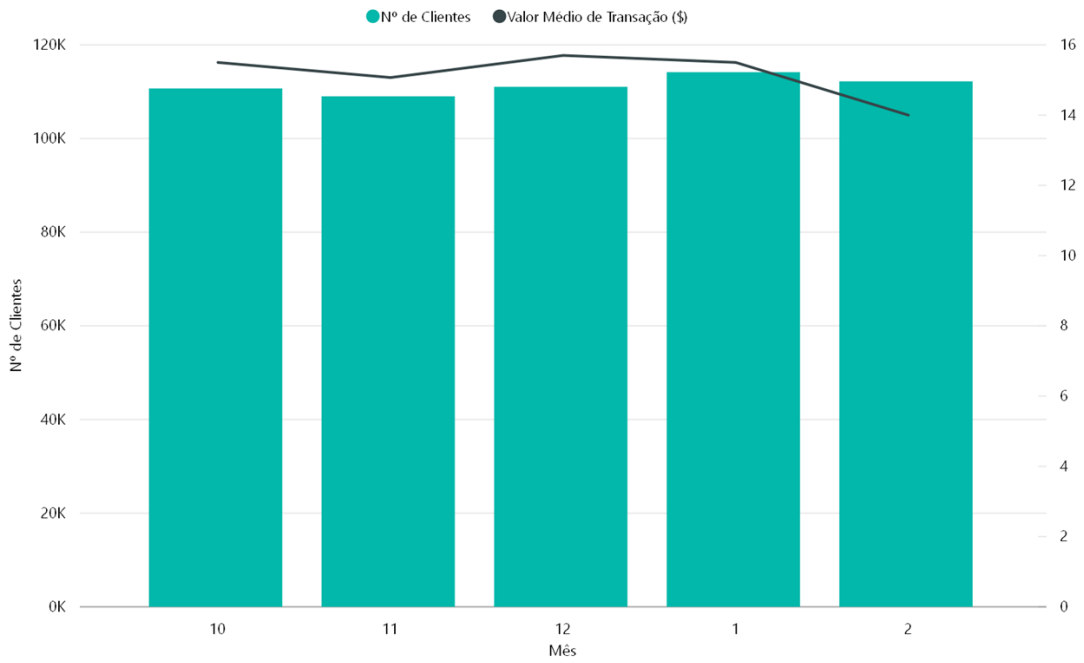

Figura 14 – KPI 1 – Evolução mensal do número médio de clientes

Analisando o gráfico, verifica-se que o número de clientes, foi sempre superior aos 100 mil clientes. Já no valor médio pago por cliente em cada transação, este atingiu o valor máximo no mês de dezembro, talvez devido às necessidades existentes para a época natalícia, e o valor mínimo em fevereiro. No mês de novembro registou-se uma redução no número de clientes e no valor médio pago por transação. A figura 17 procura demonstrar se a angariação de novos clientes se tem mantido constante ao longo dos meses e, assim, avaliar se a estratégia atual tem apresentado os resultados desejados.

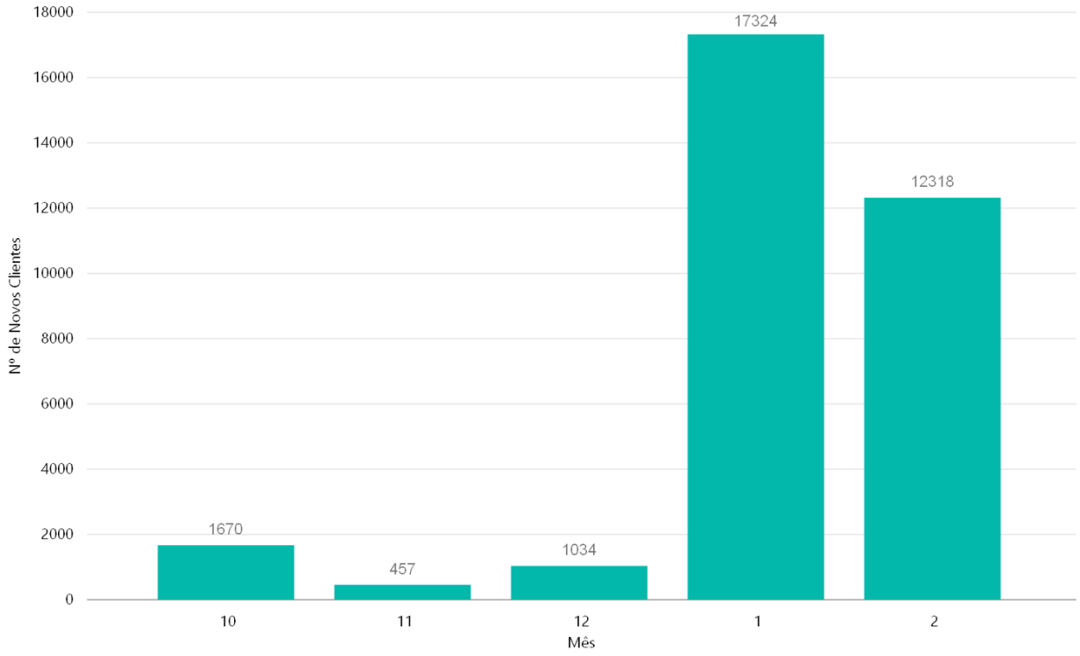

Figura 15 – KPI 2 – Evolução mensal do número de novos clientes do supermercado

No início do ano de 2013, verificou-se um aumento considerável do número novos clientes face ao mês anterior. Como foi visível na figura 15, a empresa tem conseguido manter a maioria dos clientes e o objetivo é conseguir angariar mais clientes e continuar a satisfazer os clientes já existentes, proporcionando experiências de consumo mais atrativas e que vão ao encontro do seu perfil. Em média por dia, cada cliente na loja gasta entre 14€ a 15€ na compra dos seus produtos.

Por forma a extrair alguns *insights* para a futura análise de *clustering*, especialmente na fase das regras de associação, pretendeu-se verificar a quantidade de produtos que os consumidores transacionam por compra, como é visível na figura 18:

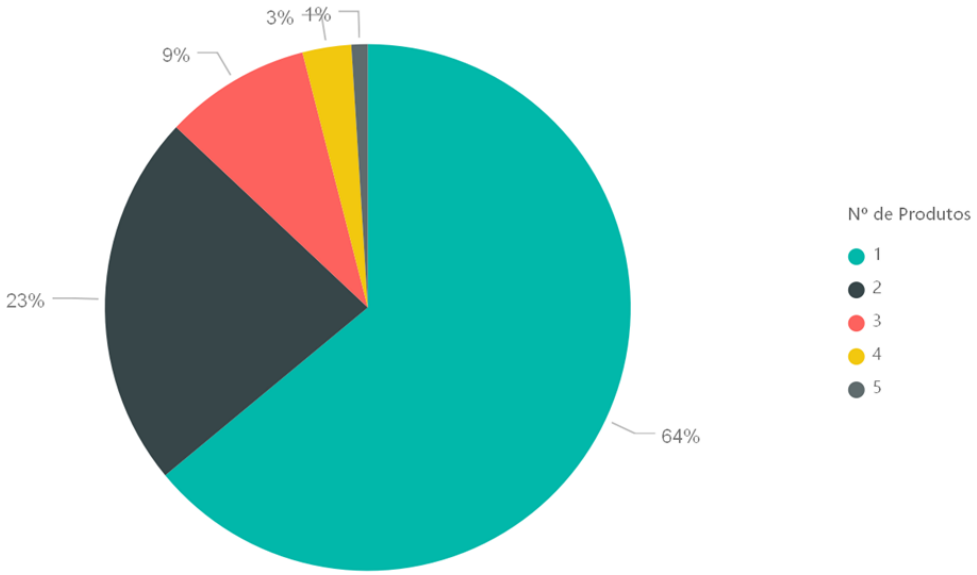

Figura 16 – KPI 3 – Quantidade de produtos por transação (%)

Como esperado, as compras com apenas uma unidade, representam a maioria das transações, apresentando uma proporção de 40%. As transações com dois ou três produtos, também apresentam uma relevância considerável, cerca de 14% e 5% consecutivamente.

Também foi feita uma análise à dimensão região, pois é importante perceber a produtividade de cada região, e, assim, ser possível tomar medidas para cada um dos locais.

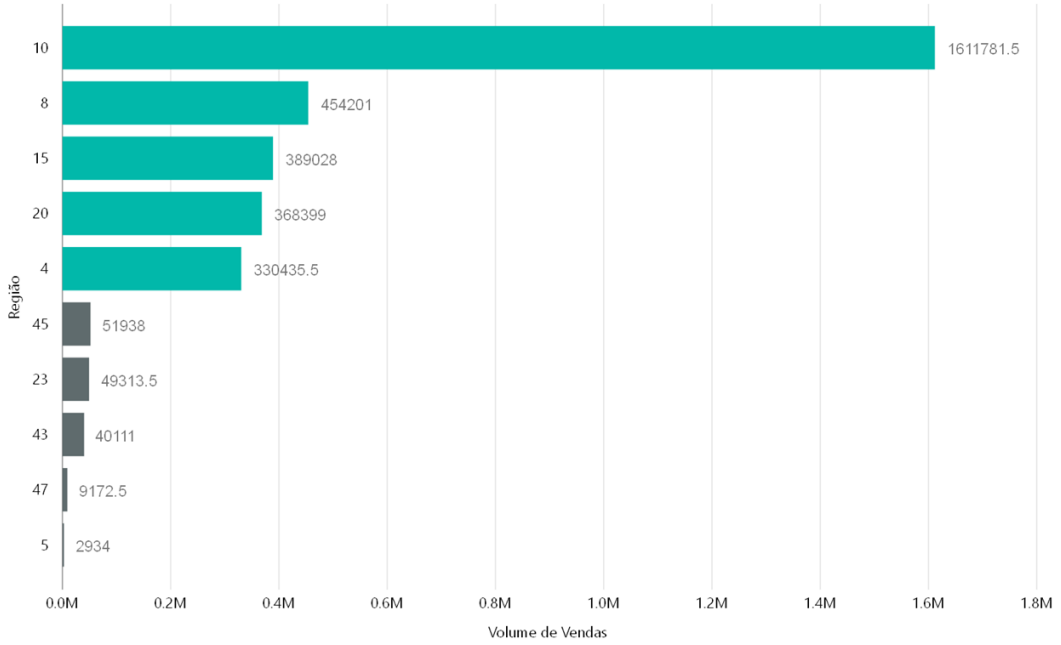

Figura 17 – KPI 4 – *Top/Bottom* 5 Volume de vendas por cada região

Pela análise do gráfico 19, verifica-se que a região 10 contempla a maioria das transações da cadeia de supermercado. No caso das regiões 5 e 47, estas deverão ser analisadas em maior detalhe, por forma a proporcionar melhor serviço aos clientes, visto que estas apresentam resultados muito abaixo da média das restantes regiões. Já em termos do valor total de receitas, são as duas regiões com piores resultados, como verificado pela figura 20:

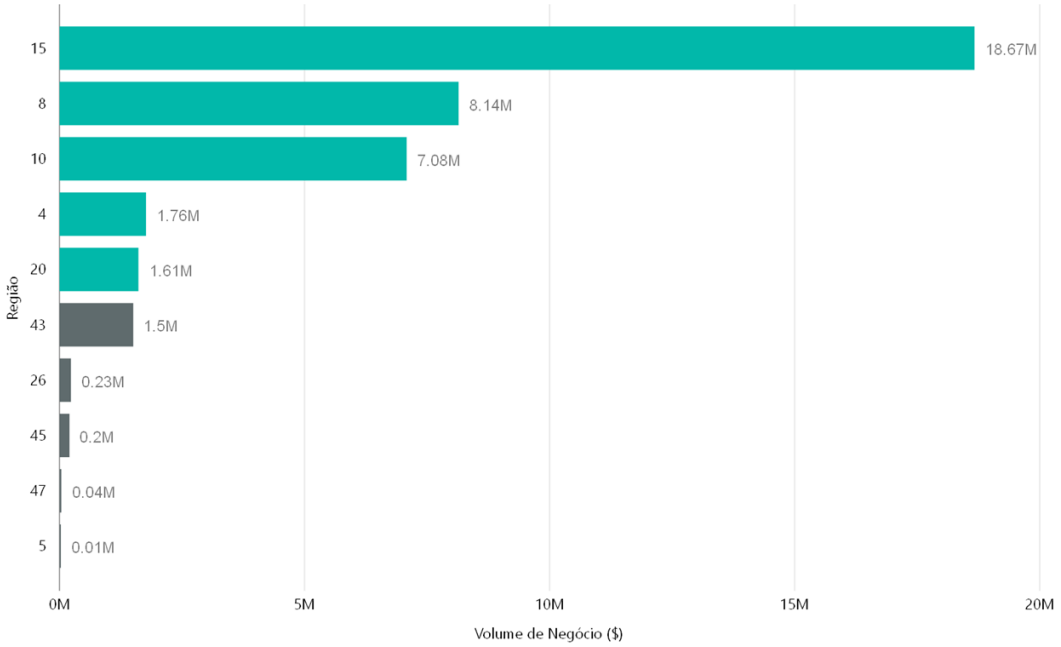

Figura 18 – KPI 5 – *Top/Bottom* 5 Volume de negócio por cada região

Em seguida, procedeu-se à análise da dimensão loja, com vista a entender quais as lojas com maior produtividade da cadeia do supermercado. Na figura 21, analisa-se sob o ponto de vista do volume de vendas, as 10 lojas com melhores resultados.

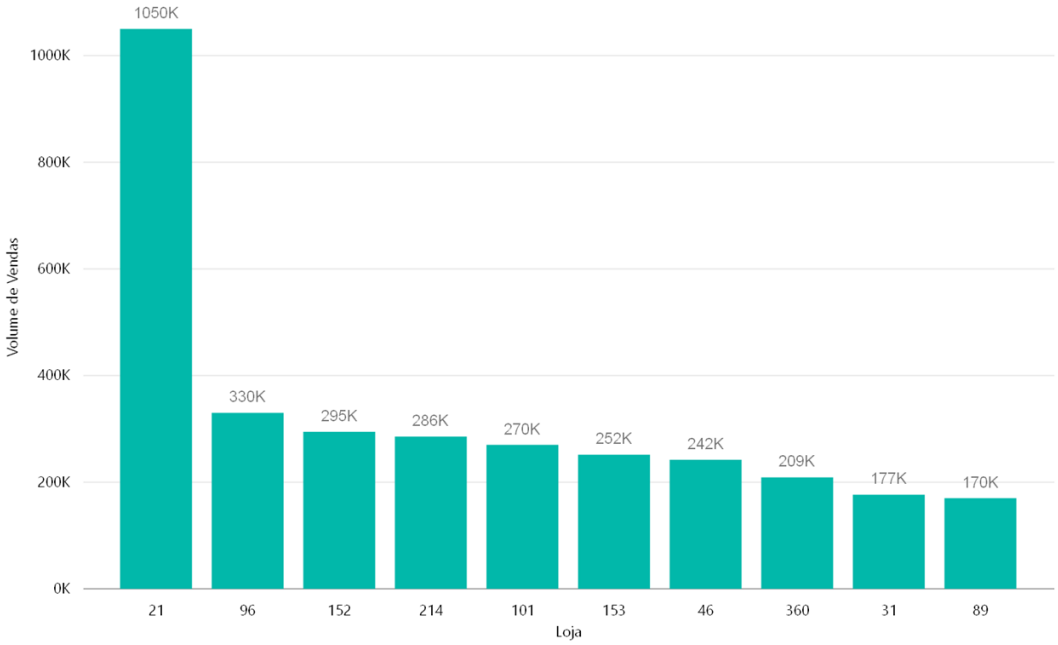

Figura 19 – KPI 6 – Top 10 Volume de vendas por cada loja de supermercado

Pelo gráfico anterior, é facilmente percetível que existe uma loja que se destaca no número total de transações, a loja 21, que representam num total de 14% do total do volume de vendas da cadeia de supermercado. A tendência confirma-se se a análise for centrada no volume de negócios, ou seja, no total de receita obtida na venda dos produtos, como é verificado na figura 22:

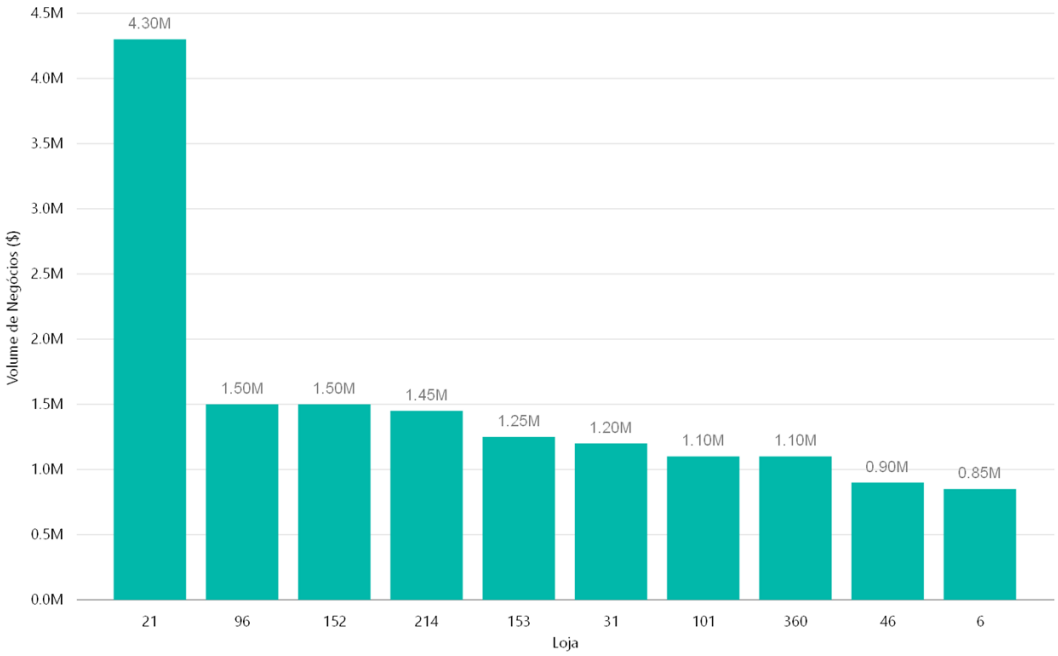

Figura 20 – KPI 7 – Top 10 Volume de negócio por cada loja de supermercado

Neste caso, a loja 21 apresenta uma receita muito superior, agrupando cerca de 9% do total de receita da cadeia de supermercado.

Deste modo também é importante perceber os produtos mais desejados e identificar se existem produtos com maior relevância junto dos consumidores. Esta análise é ilustrada na figura 23:

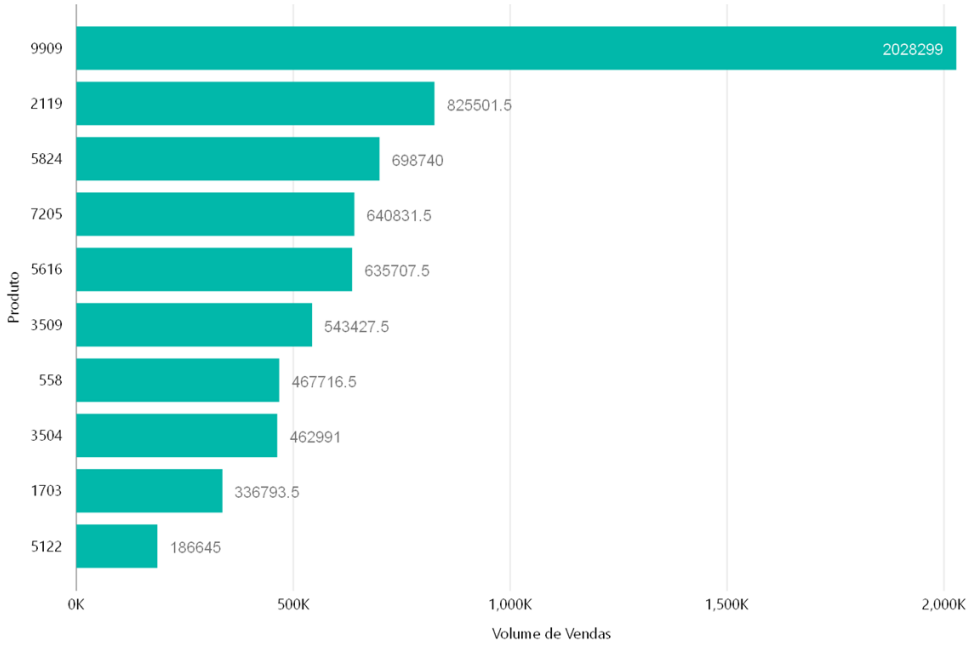

Figura 21 – KPI 8 – Top 10 Produtos mais vendidos

O produto 9909 é, claramente, o mais desejado pelos consumidores desta cadeia de supermercado. Cerca de 40% das transações existentes, incluem o produto 9909, pelo que é muito provável que nas regras a extrair, posteriormente, o produto 9909 seja um importante fator no cálculo das mesmas e seja associado à compra de outro produto em simultâneo. Este produto aparenta ser um bem essencial dos consumidores, pois, como demonstra a figura 24, inclui-se na lista dos produtos com preço mais baixo e excesso de procura. No gráfico abaixo, estão listados os cinco produtos mais caros, representados azul, e os cinco produtos mais baratos, representados a cinzento.

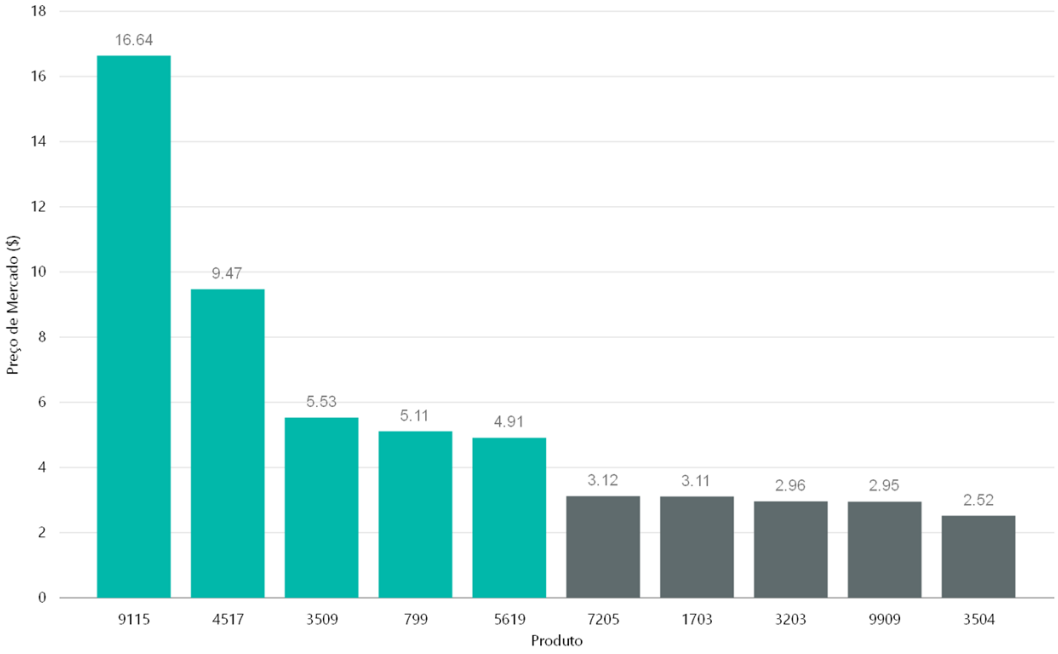

Figura 22 – KPI 9 – *Top/Bottom* 5 Lista de produtos por preço

Do conjunto dos dez produtos mais vendidos em loja, apenas o produto 3509 faz parte do universo dos produtos mais vendidos e mais caros. Como é natural, o produto 9115 representa apenas 2% do total de transações, valor explicado pelo elevado preço de mercado, cerca do dobro da média de preços. Já o produto 3203 é o terceiro com valor de mercado mais baixo, e não consta nos produtos mais vendidos em loja.

No que concerne à avaliação do desempenho dos fornecedores, verificou-se que o número de fornecedores ativos da cadeia de supermercado tem vindo a decrescer, como consta na figura 25:

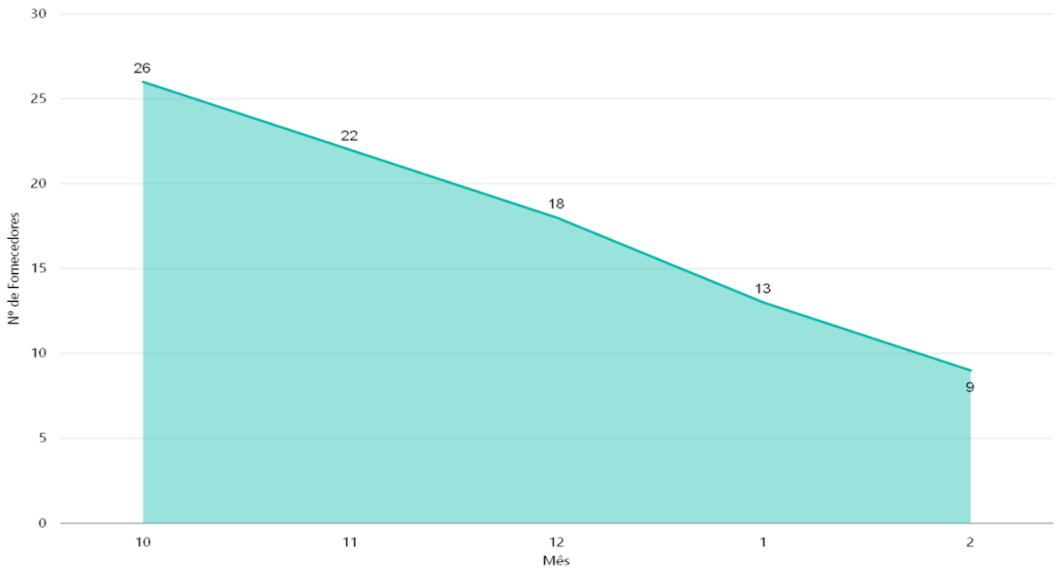

Figura 23 – KPI 10 – Evolução mensal do número de fornecedores

No inico do mês de outubro o supermercado apresentava 26 empresas, enquanto no último mês apenas 9 empresa forneciam produtos às lojas.

A última análise foi relativa ao dia da semana preferencial dos consumidores. Na figura 26, estão representados os 7 dias da semana e o número total de vendas alcançado em cada dia. Analisando o gráfico, conclui-se que o número de transações, e consequentemente o valor de receita, atingem o seu valor máximo nos últimos dias da semana (Sexta-Feira, Sábado e Domingo). Verifica-se que a meio da semana, a cadeia de supermercado regista os resultados mais baixos em ambas as variáveis, embora se verifique uma redução bastante acentuada na receita obtida. Perspetiva-se que a meio da semana, as transações sejam direcionadas para a compra de bens bastante específicos, com as compras dos bens essenciais a decorrerem no fim/início da semana.

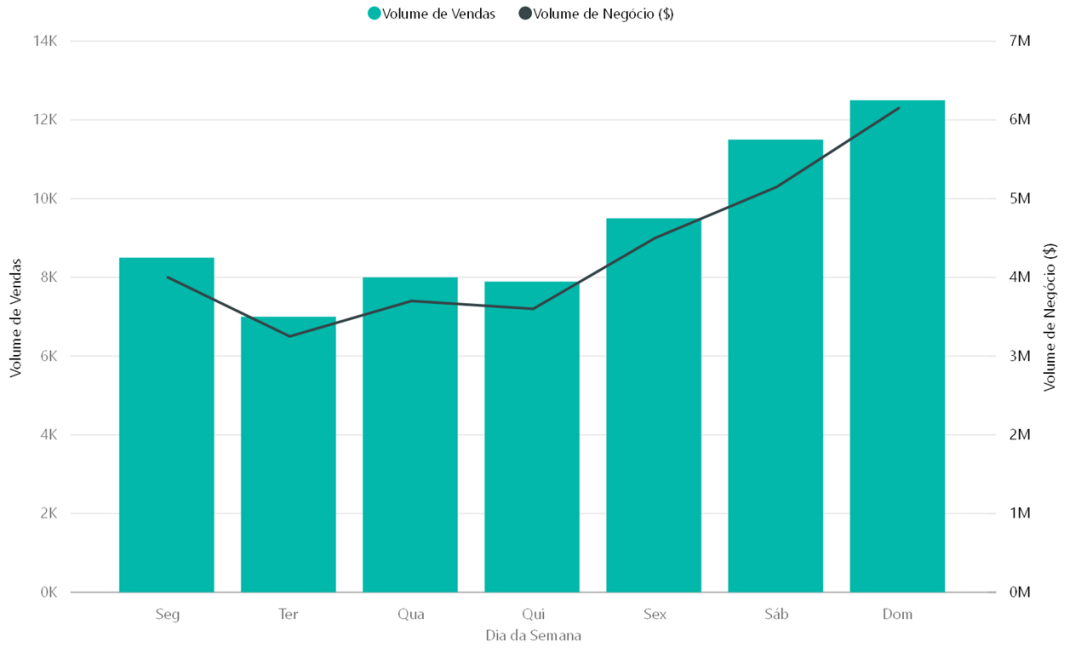

Figura 24 - KPI 11 – Produtividade por dia de semana

Após análise dos indicadores é possível concluir:

- O número de clientes tem vindo a manter-se constante, tendo atingido o valor mais alto no início do ano de 2013. O valor médio pago por cliente em cada transação atingiu o valor máximo em dezembro perto dos 16€;
- A maioria dos clientes apenas se desloca ao supermercado para a compra de um único produto;
- A loja 21 representa o local com melhores resultados ao nível do volume de vendas e do total da receita gerada;
- O produto 9909 é o produto mais procurado pelos clientes, e um dos com valor de mercado mais baixo. Ao contrário do produto 9115 que possuí um preço mais alto e um nível de consumo muito baixo. O produto 3203, é um dos produtos com o preço mais barato e com níveis de procura mais reduzido;
- O número de fornecedores tem vindo a decrescer consideravelmente com a empresa a contar apenas com 9 fornecedores no momento, quando em outubro disponha de 26;
- Os dias preferências dos clientes para compras ocorre no final da semana, Sábado e Domingo.

4.2. ANÁLISE DE CLUSTERS

Relativamente à análise de *clusters*, analisaram-se os dados perante os grupos formados, com o objetivo de perceber quais os clientes com maior fidelidade ao supermercado, ou seja, se visitam algumas das lojas muito frequentemente, ou se já não as visitam há algum tempo. Também é importante concluir que tipo de produtos habitualmente consumem. Para aumentar o lucro do supermercado, as conclusões retiradas podem ser aplicadas em códigos de desconto. Estes devem ser constituídos por produtos habitualmente transacionados por este grupo de clientes, mas também por produtos que habitualmente não são do seu interesse. Assim, é possível que os consumidores aproveitem a campanha do desconto para comprar em conjunto produtos habituais que, por norma, não consomem. Por exemplo, um cliente que por hábito compra pão, também compra manteiga. Um exemplo de uma campanha de desconto, poderá ser "Na compra de pão e manteiga, ganhe 25% de desconto na compra de detergentes para roupa".

Como verificado na Metodologia, foi concluído que, tendo em conta as variáveis selecionadas, o número correto de *clusters* seria 3. Na figura 27, está representado a frequência relativa de cada *cluster* para com o número de clientes:

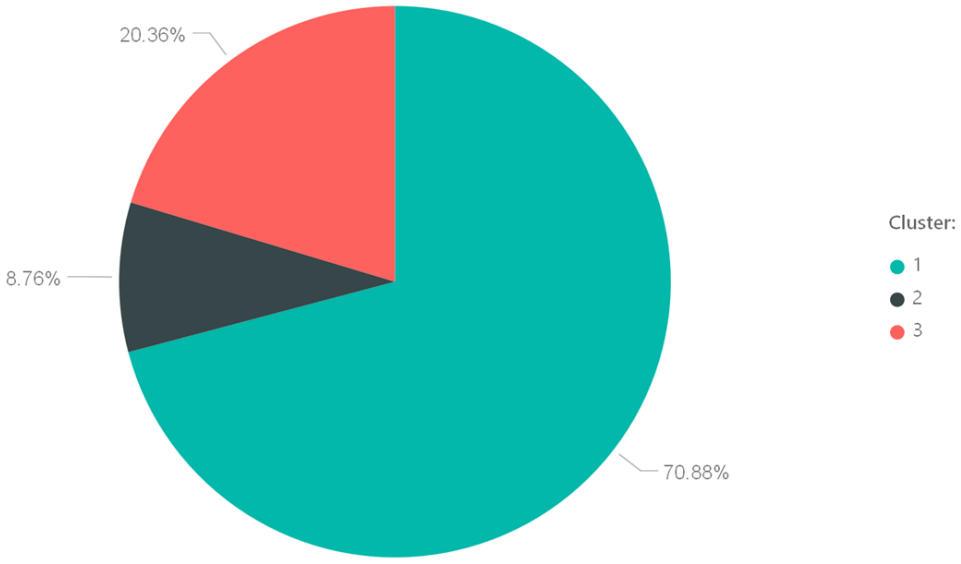

Figura 25 – Percentagem do número de clientes por *cluster*

As variáveis utilizadas para a segmentação em 3 *clusters* foram relativas ao:

- Número de semanas que o cliente percorre à cadeia de supermercado;
- Número de semanas que o cliente não visita na cadeia de supermercado;
- Montante total despendido em produtos;
- Total de produtos adquiridos na cadeia de supermercado.

Na tabela 6 estão apresentados os centróides, isto é, as médias verificadas em cada um dos *clusters* no que diz respeito às quatro variáveis listadas anteriormente.

Cluster	Nº de semanas	Nº de semanas ausente	Total de investimento	Total de produtos
1	20	2	€428	90
2	13	4	€23596	368
3	19	0	€45978	696

Tabela 6 – Perfis do valor de cliente

O *cluster* 1 é formado pelos clientes que recorrem com maior frequência às lojas do supermercado, mas que investem menos dinheiro em produtos do supermercado. Não tendo conhecimento relativo a informações do contexto pessoal sobre os clientes, estes poderão ser denominados como clientes de idade mais jovem, e que procuram as lojas em situações pontuais para a compra de produtos específicos para necessidades do momento, ou seja, procuram comprar os produtos necessários para o próprio dia. Este grupo será denominado como o *cluster* de **clientes de necessidade imediata**.

O cluster 2 é constituído por clientes com um índice de frequência de visita à loja menor, mas que compram uma quantidade considerável de produtos da empresa. Apresentam uma média de visita ao supermercado de 1 vez por mês, o que leva acreditar que são clientes que procuram a empresa por forma a comprar os produtos que satisfaçam as suas necessidades mensais, provavelmente com alguns elementos dependentes desta transação (famílias, por exemplo). Estes são caracterizados pela compra de produtos necessários para período prolongado de semanas/meses. São clientes que se espera que continuem a comparecer com esta regularidade e, pelos quais, o supermercado deve criar estratégias de continuidade. Estes clientes são denominados como **clientes de conta corrente**.

O *cluster* 3 agrupa os clientes de maior valia para a cadeia de supermercado, pois são clientes que apresentam uma média de frequência de visita à loja muito alta, assim como, são os clientes que proporcionam maior valor de receita à loja com o consumo de um grande leque de produtos. Estes clientes serão denominados como os **consumidores compulsivos**.

Verificando a figura 28, conclui-se que o *cluster* 1 é responsável pela maioria das transações na cadeia de supermercado, o que acaba por ir ao encontro das proporções demonstradas na figura 27. Em média, os clientes pertencentes ao *cluster* dos clientes de baixo valor realizam 615 mil transações por mês, sendo que o valor do número de vendas tem-se mantido constante ao longo dos meses, registando-se, em novembro, o valor mais baixo da amostra. Contudo, são o grupo de clientes que apresenta o menor valor de investimento ao longo dos meses da amostra, embora no mês de fevereiro tenha apresentado o maior valor de receita dos três *clusters*. As duas figuras confirmam os valores de centróide verificadas na tabela 9, ou seja, o *cluster* 1 é formado por clientes que frequentam muito as lojas do supermercado, mas para compras de poucos produtos a um baixo custo.

O *cluster* 2 agrupa os clientes com o menor volume de vendas. No caso do cluster 2, caracteriza-se por clientes que frequentam poucas vezes por mês as lojas de supermercado, apresentando uma média de 76 mil transações por mês. O mês de novembro foi o mês no qual, este grupo de clientes menos recorreu ao supermercado, e, consequentemente, menos receita gerada por destes clientes. Embora no último e primeiro mês do ano se tenha verificado um aumento considerável no valor de investimento, provavelmente, relacionado com a época festiva vivida nessa altura do ano. Considerando o perfil de consumidor é uma época especial para estes.

No *cluster* 3, verifica-se a maior disparidade em termos do valor de investimento efetuado nos produtos da empresa. Apenas no mês de outubro e fevereiro, o *cluster* 3 apresentou valores monetários inferiores face aos restantes grupos. Pelo que se confirma que são grupo de clientes com grande poder de compra para adquirir produtos do supermercado.

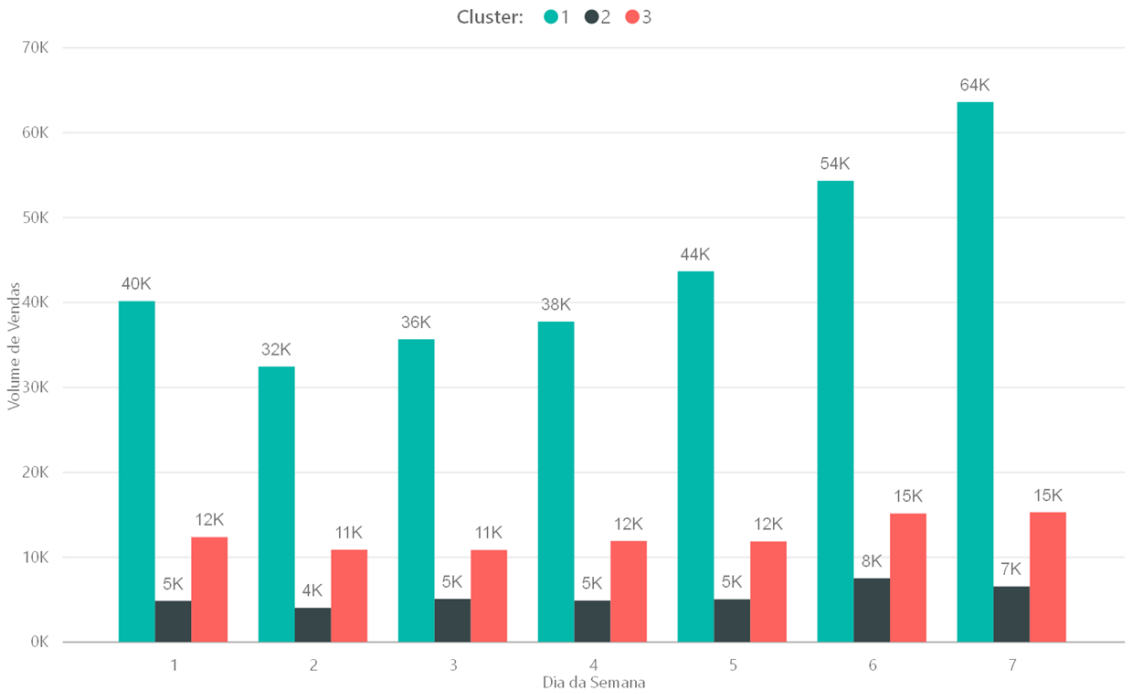

Figura 26 – Volume de Vendas por *cluster*

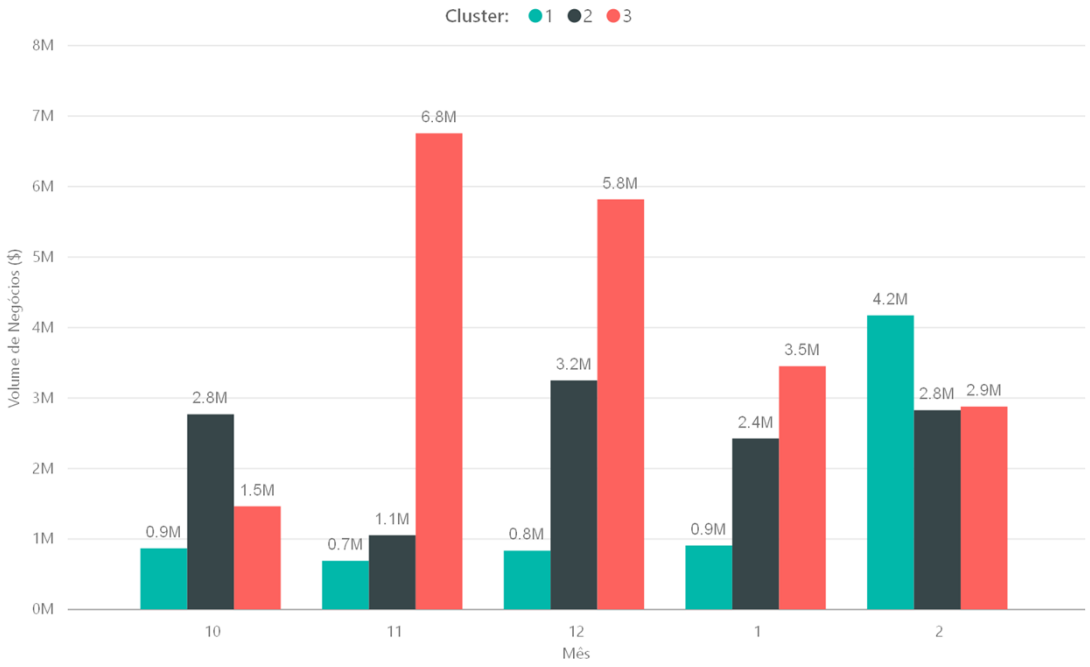

Figura 27 – Volume de Negócio por *cluster*

Na figura 30 ilustra a afluência de cada *cluster* ao dia da semana e fim-de-semana:

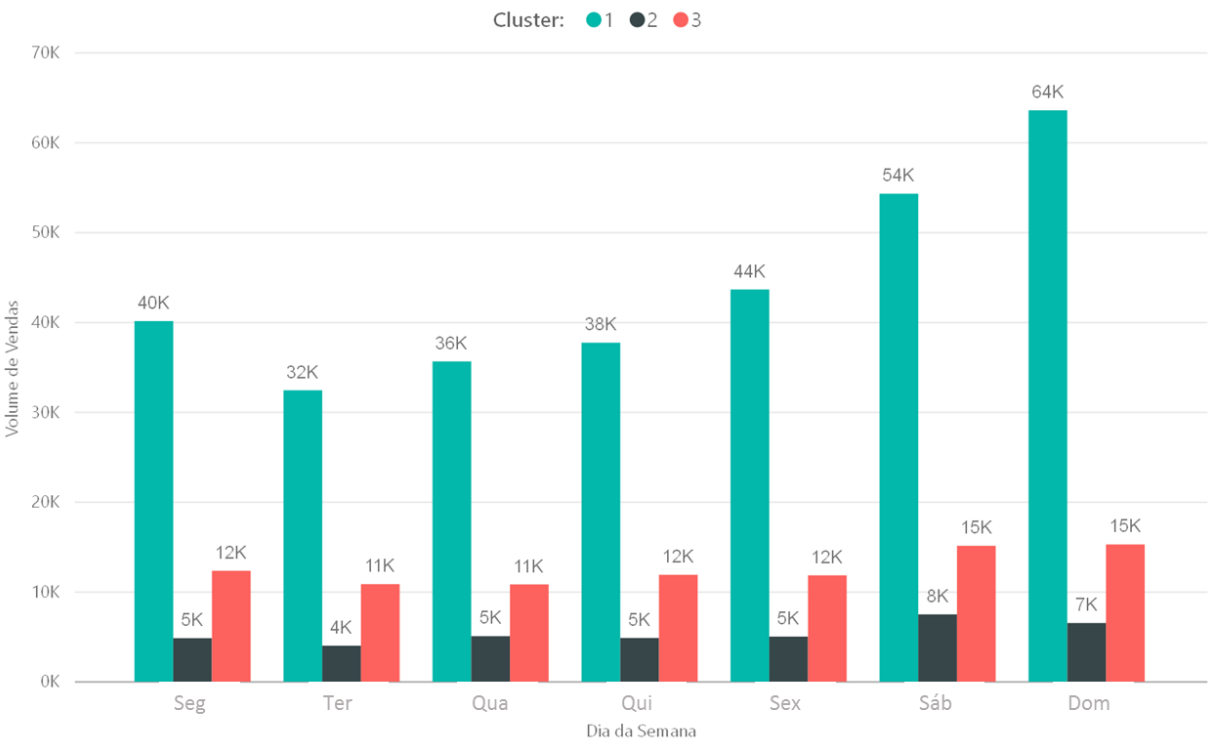

Figura 28 – Dia da Semana preferencial por *cluster*

Contrariamente aos clientes de necessidade imediata, os clientes de conta corrente e os consumidores compulsivos apresentam um volume de vendas regular durante todos os dias de semana, não apresentando valores muito distintos, apesar de nos dias de fim-de-semana, o número de vendas crescer ligeiramente. No caso do *cluster* 1, o número de transações cresce exponencialmente no fim-de-semana, apresentando uma média de 59 mil transações, sendo a média dos dias de semana de 30 mil transações.

Através da análise de *clusters* é possível concluir sobre o valor de cada cluster para o negócio. Existem duas visões possíveis, a definição de estratégias que permitam segurar os clientes de grande valor para a cadeia de supermercado e a criação de estratégias que permitam atrair, com maior frequência, os clientes regulares e pontuais. A criação de estratégias de Marketing que vão ao encontro da necessidade dos clientes é a melhor solução para o negócio. Um dos exemplos poderá ser a criação de pacotes de produtos que engloba os produtos mais consumidos por cada conjunto de clientes. Assim, será possível criar um serviço que satisfaça as necessidades dos clientes, gerando um relacionamento que é essencial para a atração e fidelização de clientes, sendo possível maximizar o número de clientes fidelizados.

De seguida será feita uma análise para cada grupo de clientes, com o objetivo de compreender o seu comportamento no espaço da loja, como, por exemplo, os seus hábitos de consumo, por forma a ser possível criar estratégias de negócio eficazes, que permitam à empresa melhorar os resultados no futuro. Desta forma, a organização da loja, a logística, a gestão do espaço comercial e a gestão de recursos são estrategicamente escolhidos e adaptados tendo em conta o perfil do consumidor.

- **Clientes de necessidade imediata:**

Inicialmente, foi realizado um estudo sobre afluência de clientes por região, podendo assim identificar quais as regiões com maior potencial para cada grupo de clientes, pois os hábitos, o poder de compra, assim como a frequência com que cada tipo de cliente se dirige à loja difere para cada região de supermercado. Na figura 31 é demonstrado a região mais representativa para os clientes de necessidade imediata:

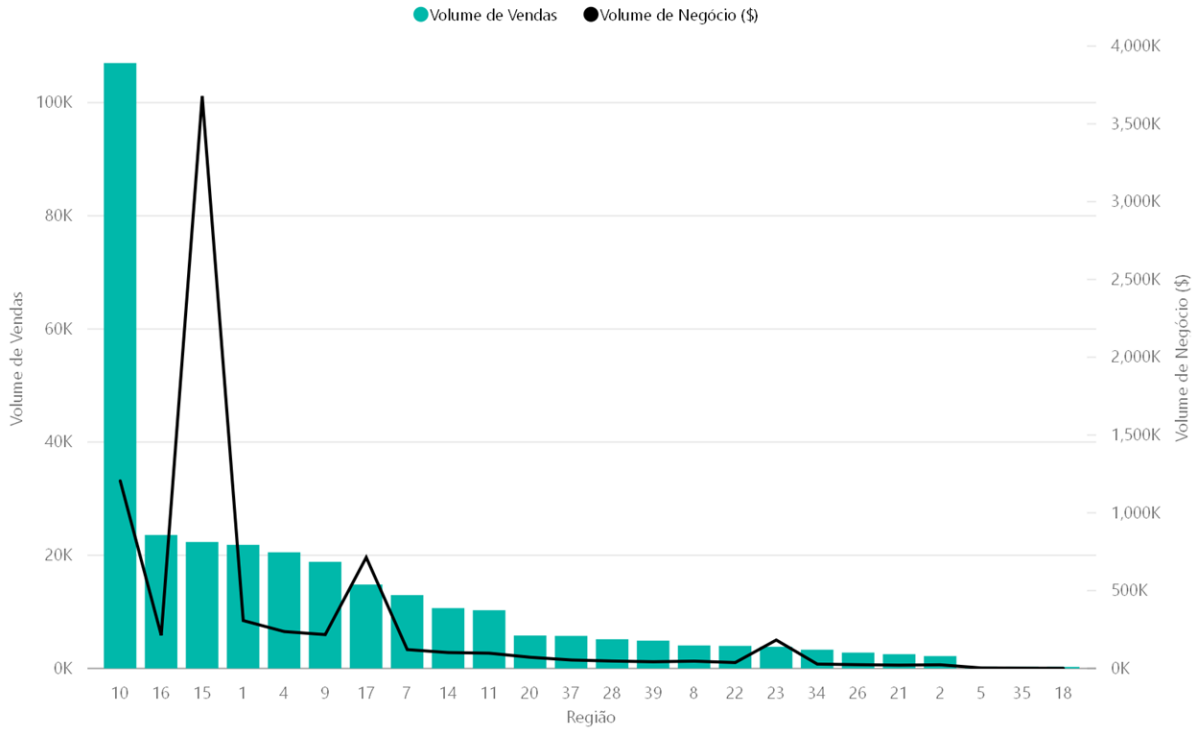

Figura 29 – *Cluster* 1 – Frequência de Venda por Região

Analisando o gráfico, é possível verificar que a região 10 é, claramente, a mais representativa do *cluster* 1. Este indicador acaba por ir ao encontro dos resultados verificados no KPI 4, que indicava a região 10 como a mais importante para o supermercado e, sendo o *cluster* 1 a maioria dos clientes do conjunto de dados, este resultado acaba por ser esperado. As regiões 16 e 1 embora não pertencem ao grupo de regiões com maiores resultados de vendas ao nível geral do supermercado, são duas regiões nas quais se deve apostar para a fidelização dos clientes de necessidades imediatas.

Em seguida realiza-se uma análise à dimensão produto, não só para avaliar os produtos mais procurados por este grupo de clientes, mas também para avaliar os produtos menos desejados, por forma a criar estratégias de negócio capazes de aumentar o interesse nestes bens de consumo e, assim, aumentar a receita do supermercado. A indústria 99 apresenta valores, em ambas as partições, muito mais elevados face aos restantes. Valores que podem ser explicados pelo sucesso de vendas do produto 9909, como verificado na figura 32, pertencente à indústria mencionada. É importante também registar o elevado volume de negócio gerado pela indústria 72, que contempla um conjunto de produto de baixo valor, como o produto 7205, e apresenta resultados bastante significativos.

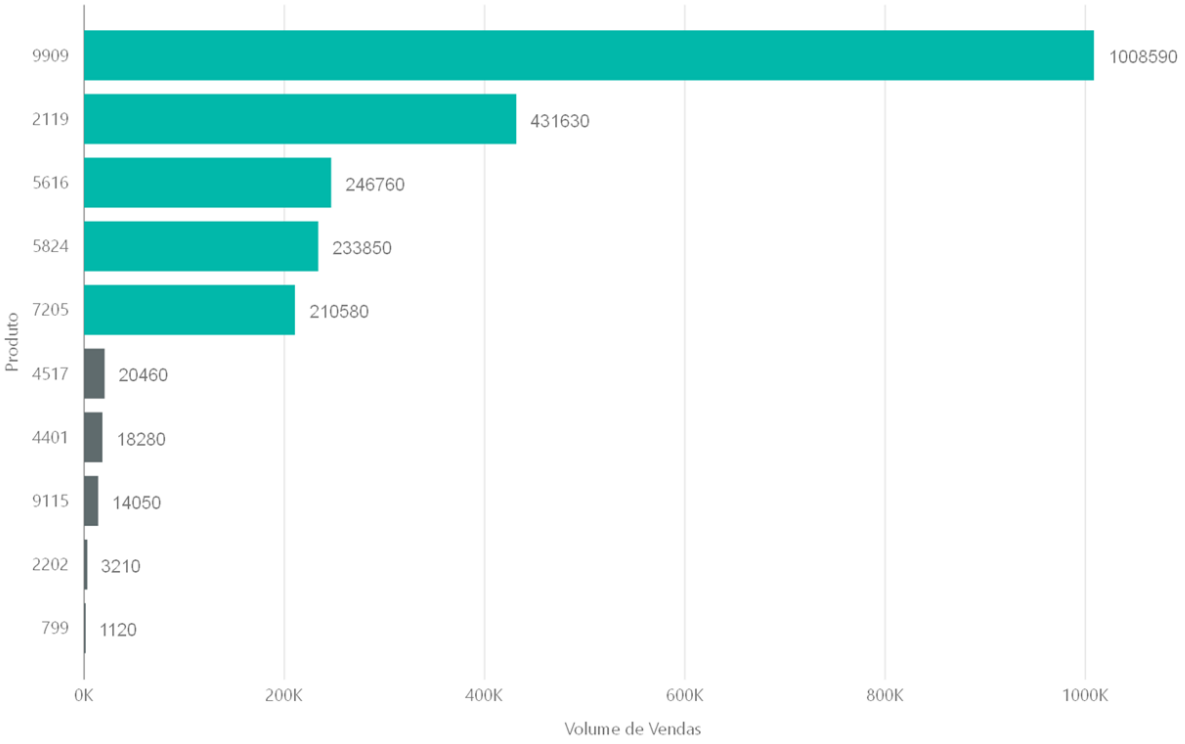

Figura 30 – Cluster 1 – *Top/Bottom* Lista de Produtos

Em seguida, analisa-se os *items* mais frequentes do conjunto de transações dos clientes de necessidades imediatas, listados na tabela 7:

Item	Suporte
9909	33%
2119	14%
5616	8%

Tabela 7 – Cluster 1 – Conjunto de *items* frequentes

O item 9909 apresenta um suporte muito superior face aos restantes *items* do supermercado, pelo que é expectável que as regras mais credíveis, a extrair, possuam o produto 9909, como recomendação. Este surge em 1008590 das transações, representante de um suporte de 33%, enquanto os produtos 2119 e 5616 apresentam um valor de 14% e 8%, respetivamente.

- **Clientes de conta corrente:**

Analisando as regiões com maior relevância para os clientes de conta corrente, apenas duas localizações apresentam materialidade considerável para análise, como é verificado pela figura 33:

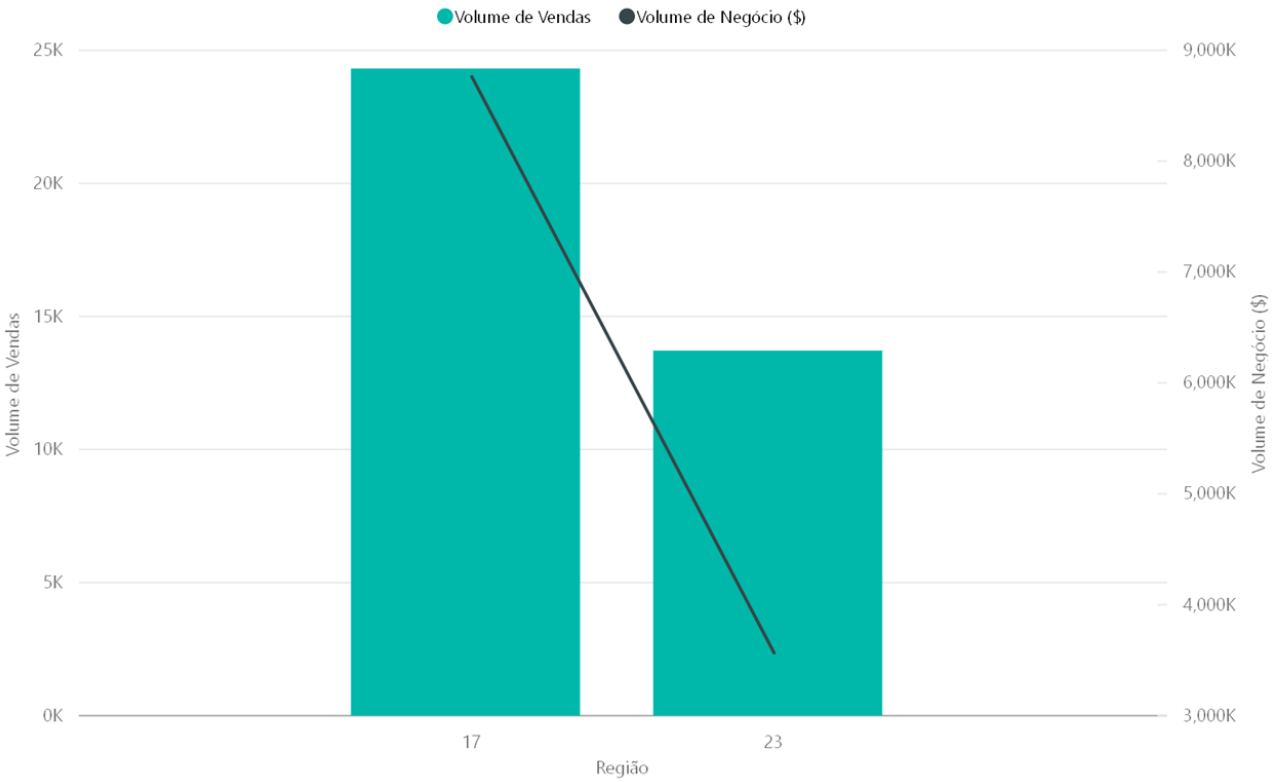

Figura 31 – *Cluster* 2 – Frequência de Venda por Loja

Como era expectável, as duas regiões apresentam um número total de transações muito inferior ao verificado nos clientes de necessidade imediata e um volume de negócios superior, pois são clientes que embora com menor frequência, apresentam uma regularidade de compra grande. Ambas as regiões não pertencem ao leque de locais com maior volume de dados, o que leva acreditar que não são regiões centrais, situadas em grandes cidades.

Relativamente aos produtos mais procurados por este tipo de clientes, analisando a figura 34, os produtos pertencentes à categoria 99, assim como no *cluster* anterior, são os mais desejados e com uma taxa de consumo superior, valor confirmado pelo volume de vendas do produto 9909. No caso dos produtos 9115, 2202, 799 e 4401, estes apresentam uma materialidade muito baixa, pelo que é necessário avaliar se a situação destes produtos, por forma a incentivar o consumo por parte dos clientes de conta corrente. Os produtos 2119, 3509 e 5824 são bons candidatos para integrar promoções com os produtos mais procurados, pois apresentam níveis de consumo muito altos.

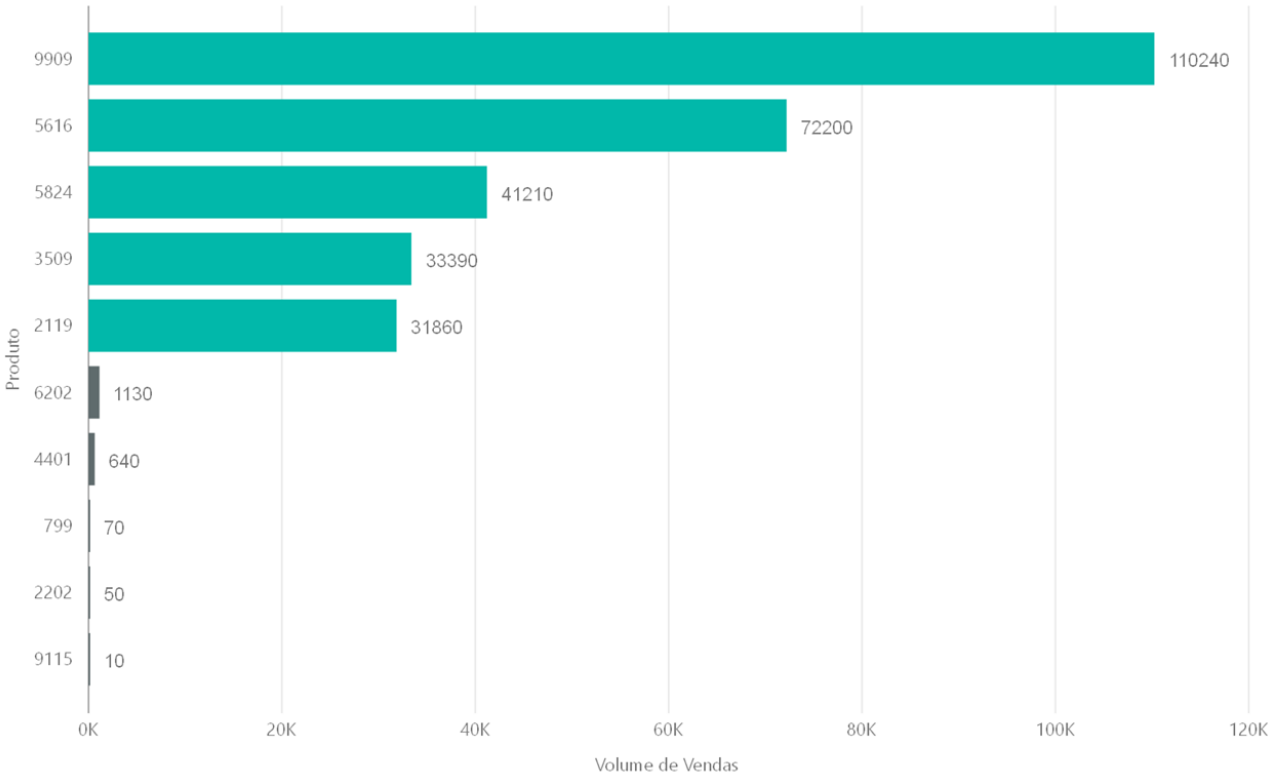

Figura 32 – *Cluster* 2 – *Top/Bottom* Lista de Produtos

Os *items* mais frequentes do conjunto de transações dos clientes de conta corrente, listados na tabela 8:

Item	Suporte
9909	22%
5616	14%
5824	8%

Tabela 8 – Cluster 2 – Conjunto de *items* frequentes

Assim, como no cluster 1, o item 9909 apresenta um suporte superior face aos restantes *items* mais frequentes do supermercado, como os produtos 5616 e 5824.

- **Consumidores compulsivos:**

Analisando as regiões com maior relevância para os consumidores compulsivos, ilustrado na figura 35, o elevado volume de negócios e o baixo volume de vendas da região 4, sobressai logo à primeira vista.

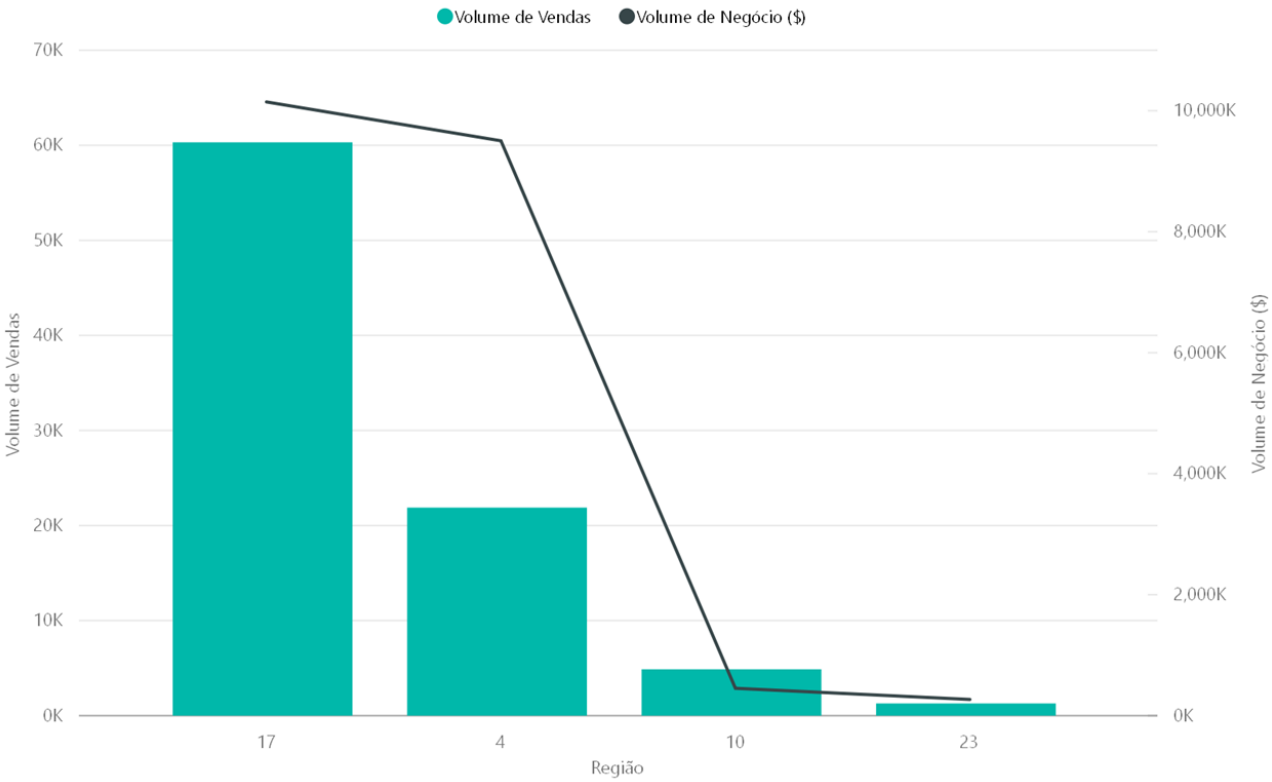

Figura 33 – *Cluster* 3 – Frequência de Venda por Loja

Uma das razões para a elevada receita gerada na região 4, apesar do reduzido número de vendas, é a elevada taxa de consumo de dois dos produtos mais caros do mercado, nomeadamente os produtos 9115 e 5824, como é visível pela figura 36.

Como verificado na análise dos indicadores de desempenho, o produto 9909 representava a maior receita para a cadeia de supermercado. Como tal, era expectável que nas três análises de *clusters* o produto 9909 fosse o mais representativo, como concluído pela figura 36 e as análises anteriores.

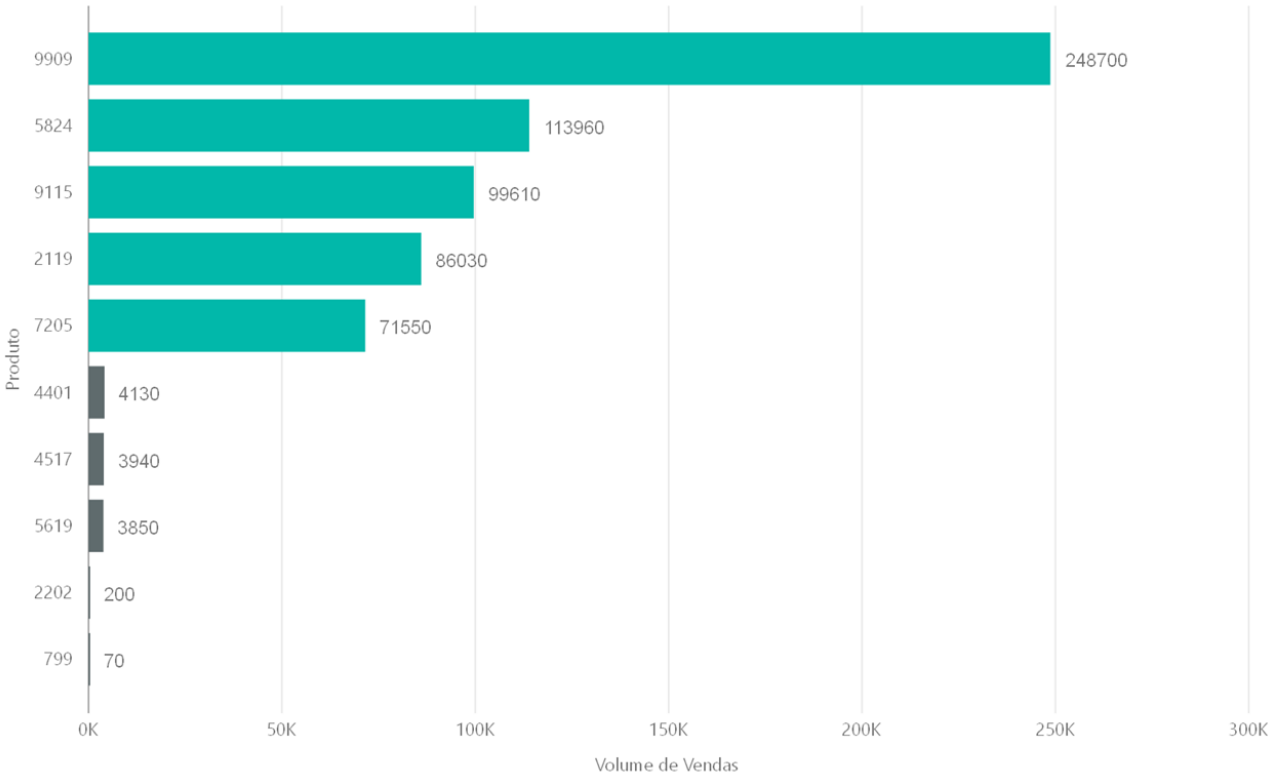

Figura 34 – *Cluster* 3 – *Top/Bottom* Lista de Produtos

O elevado número de vendas do produto 9115 permite distinguir este grupo de clientes dos restantes, pois é um produto com um valor de mercado alto, pelo que seria expetável que apresentasse uma baixa materialidade. Assim, como nos restantes *clusters,* os produtos 9909, 5824 e 2119 correspondem à lista de produto com maior taxa de consumo, o que vai ao encontro do KPI 8, que concluía que eram os três produtos com maior procura.

Os *items* mais frequentes do conjunto de transações dos consumidores compulsivos, listados na tabela 9:

Item	Suporte
9909	28%
5824	13%
9115	11%

Tabela 9 – Cluster 3 – Conjunto de *items* frequentes

O produto 9115, sendo um dos produtos com menor procura no supermercado, apresenta um suporte para este conjunto de clientes bastante razoável, pois em 868112 transações, o produto 9115 é comprado em 99610 das mesmas. O produto 9909, como era natural, apresenta o suporte com valor mais elevado.

4.3. REGRAS DE ASSOCIAÇÃO

Posteriormente ao tratamento de dados e segmentação por perfil de consumidor, extraíram-se as regras de associação, com vista analisar as associações existentes entre os artigos do supermercado.

Como foi referido na Metodologia, para cada cluster foram definidos diferentes valores mínimos de parâmetros, de forma alcançar o objetivo pretendido.

- **Clientes de necessidade imediata:**

Para o cluster 1, o algoritmo *FP-Growth*, com um suporte mínimo de 1%, apenas gerou cinco regras, distribuídas por cinco *items*, como é verificável pela figura 37:

[6202 → 9909, 0.5555555555555556]

[3509 → 9909, 0.5555555555555556]

[7205 → 9909, 0.53125]

[3203 → 9909, 0.5]

Figura 35 – Cluster 1 – Regras de Associação

Como pode ser observado, todas as regras extraídas apresentam um valor de confiança na casa dos 50%. Pegando nas regras com as regras com maior Confiança podemos concluir que:

- Em 1% do total de transações verificadas no cluster 1, em 55% das transações, quem compra o produto 6202, também compra o produto 9909;
- Em 1% do total de transações verificadas no cluster 1, em 55% das transações, quem compra o produto 3509, também compra o produto 9909.
- Em 1% do total de transações verificadas no cluster 1, em 53% das transações, quem compra o produto 7205, também compra o produto 9909.

Tendo em conta o baixo valor de suporte, a frequência de ocorrência das combinações de produtos é no mínimo de 43405 transações e a percentagem de transações onde um produto aparece dada a ocorrência de outro produto nunca é superior a 55%. Mesmo com este reduzido valor de parametrização, apenas foram geradas 5 regras de associação. Deste modo, pode-se concluir que existe uma baixa correlação entre os produtos transacionados pelos clientes de necessidade imediata. As regras obtidas devem ser analisadas em detalhe podendo representar apenas associações derivadas de uma época de curta duração que incentivou à compra destes artigos em conjunto. Na tabela 10, está representado uma análise descritiva às regras criadas para uma melhor proposta de recomendação:

Regra	Valor Médio	Dia da Semana mais Popular
6202 → 9909	7$	Domingo
3509 → 9909	9$	Quinta-Feira
7205 → 9909	7$	Domingo

Tabela 10 – *Cluster* 1 – Regras de Associação – Análise Descritiva

A primeira regra conclui que, em média, quando um cliente compra os produtos 6202 e 9909 em simultâneo, gasta 7$ na sua transação. Na maioria das ocasiões, estes produtos são frequentemente transacionados aos Domingos. A criação de um programa de promocional na compra dos produtos 6202 e 9909, com um valor de desconto aos fins-de-semana, poderá aumentar o número de vendas e, consequentemente, o lucro gerado pelo supermercado. Verifica-se a mesma tendência na terceira regra gerada. No caso da regra "quem compra o produto 3509, também compra o produto 9909", esta é recorrentemente verificada às Quintas-Feiras.

- **Clientes de conta corrente:**

Por fim, foram extraídas as regras de associação mais relevantes para este conjunto de dados. Para tal foi considerado um suporte mínimo de 10% e de confiança de 70%. Atendendo aos parâmetros definidos para a extração de regras de associação, foram geradas 28 regras que permitem compreender alguns hábitos de consumo deste grupo de clientes, como listadas pela figura 38:

```
[5616,5824 => 9909,0.98]
[5616,5824,9909 => 2119,0.91]
[7205,5824,9909 => 2119,0.90]
[5616,9909 => 2119,0.89]
[5616,2119,9909 => 5824,0.88]
[7205,2119 => 5824,0.87]
[3504 => 2119,0.87]
[7205,9909 => 5824,0.85]
[5616,9909 => 7205,0.80]
[7205,9909 => 3504,0.78]
```

Figura 36 – Cluster 2 – Regras de Associação

Como pode ser observado pela figura 35, foram extraídas três regras com índice de confiança acima dos 90%:

- Em 20% do total de transações no cluster 2, em 98% das transações dos produtos 5616 e 5824, o produto 9909 também é transacionado;

- Em 20% do total de transações no cluster 2, em 91% das transações dos produtos 5616, 5824 e 9909, o produto 2119 também é transacionado;
- Em 20% do total de transações no cluster 2, em 90% das transações dos produtos 7205, 5824 e 9909, o produto 2119 também é transacionado.

As regras geradas podem ser exploradas para novas estratégias promocionais, bem como para recomendações em web-catálogo. Por exemplo, quando o cliente compra o produto 5616 e 5824, automaticamente, na próxima visita do cliente, o produto 9909 seria recomendado. Ao nível de loja, é recomendável colocar os produtos 5626, 2824, 9909 e 2119 em espaços próximos, por forma a incentivar o cliente a comprar estes produtos.

Era expectável extrair regras de associação com um valor de suporte superior, visto que este cluster é caracterizado por clientes que procuram o supermercado uma vez por mês e por transacionar uma quantidade considerável de produtos por cada compra. Como tal, seria de esperar um maior grau de associação entre as compras de supermercado neste *cluster*.

Estes clientes são de grande valia para o supermercado, visto que são clientes fieis, que procuram o supermercado com regularidade. Através do conhecimento gerado, o objetivo passa pelo aumento da frequência de visita destes clientes, procurando incentivar o seu interesse através de programas de fidelização, com especial foco para a compra dos produtos, acima mencionados. Pretende-se assim induzir o aumento do consumo dos clientes proporcionando-lhes programas de fidelização com base em descontos/promoções.

Na tabela 11 encontra-se a análise descritiva para as regras mais credíveis:

Regra	Valor Médio	Dia da Semana mais Popular
5616,5824 →9909	68$	Sábado
5616,5824,9909 → 2119	124$	Sábado
7205,5824,9909 → 2119	98$	Domingo

Tabela 11 – Cluster 2 – Regras de Associação – Análise Descritiva

Analisando a tabela acima, é possível extrair mais conhecimento para uma tomada de decisão mais otimizada. No caso dos clientes de conta corrente, estes por norma realizam a maioria das suas transações durante o fim-de-semana. A criação de programas de desconto ao fim-de-semana que engloba as regras acima, poderia incentivar estes clientes a recorrerem mais frequentemente ao supermercado e assim, aumentar a frequência média de visita ao supermercado.

- **Consumidores compulsivos:**

Tendo em conta os elevados níveis de consumo apresentados por estes clientes, para a implementação do algoritmo *FP-Growth* foi definido um valor de suporte mínimo de 50% e um nível de confiança de 80%. As regras de associação mais frequentes são:

[1703,5558,3504 => 5616,0.99]

[1703,5558, 9909 => 5616,0.98]

[5558,3509,7205,5616 => 5824,0.97]

[1703, 2119,9909 => 5616,0.94]

[3203,5558,5616 => 3509,0.92]

[3203,7205,2119 => 5558,0.92]

[7205,2119 => 3509,0.92]

[3203,5558 => 3509,0.92]

[706 => 3509,0.91]

[3509,9909 => 7205,0.87]

Figura 37 – Cluster 3 – Regras de Associação

Como pode ser observado pela figura 39, foram extraídas três regras com índice de confiança acima dos 90%

- Em 50% do total de transações no cluster 3, em 99% das transações dos produtos 1703, 5558, 3504, o produto 5616 também é transacionado;
- Em 50% do total de transações no cluster 3, em 98% das transações dos produtos 1703,5558 e 9909, o produto 5616 também é transacionado;
- Em 50% do total de transações no cluster 3, em 97% das transações dos produtos 5558,3509,7205 e 5616, o produto 5824 também é transacionado.

Os elevados níveis de consumo praticados por estes conjuntos de clientes e quantidade de regras credíveis geradas, permitem boas perspetivas para a recomendações de novas propostas promocionais. Analisando os resultados do algoritmo, conclui-se que existe uma forte correlação entre os produtos de supermercado transacionados pelos clientes compulsivos. O grande leque de regras com confiança acima dos 90% e suporte de 50% comprova que estes clientes apresentam hábitos de consumo que constituem uma grande oportunidade para o supermercado conseguir aumentar o seu valor de receita total no fim de cada ano, aumentando o nível de consumo destes clientes.

Na tabela 15 foi feita uma análise às três regras com maior força para concluir qual o preço médio de transação, contemplando, os *items* da regra extraída, bem como o dia da semana na qual se verifica um maior número de transações. Através da análise seguinte, a empresa poderá definir novas estratégias de negócio para os clientes identificados como compulsivos.

Regra	Valor Médio	Dia da Semana mais Popular
1703,5558,3504 → 5616	499$	Segunda-Feira
1703,5558, 9909 → 5616	402$	Domingo
5558,3509,7205,5616 → 5824	512$	Sábado

Tabela 12 – Cluster 3 – Regras de Associação – Análise Descritiva

Os consumidores compulsivos, como já tinha sido analisado, apresentam valores médios de transação muito elevados, e que por norma, realizam a maioria das suas transações aos fins-de-semana ou início da semana. A fixação de comprar mais do que é necessário, poderá ser uma vantagem para o supermercado para incentivar ao consumo de uma maior variedade de produtos, oferecendo um cartão do supermercado pelo qual os clientes poderiam ter acesso a diversas ofertas e promoções ao longo das semanas, provocando um maior desejo dos clientes em frequentar as lojas da cadeia.

5. CONCLUSÕES

Este projeto tinha como intuito utilizar e analisar o desempenho de ferramentas *Big Data*, de modo a concluir a relevância e utilidade destas para as atuais arquiteturas das organizações. Ao longo do projeto foram utilizados vários projetos do ecossistema Hadoop, podendo concluir que este está preparado para atuar em várias perspetivas do negócio. Ao nível do armazenamento de dados recorreu-se ao Sqoop, sendo possível a extração de dados relacionais para o *cluster* Hadoop, de um modo otimizado. Estes ficam armazenados no HDFS - *Hadoop Distributed File System* – que é utlizado pelo Hadoop como *file system* distribuído. Para aproveitar as potencialidades do SQL em ambientes distribuídos, foram utilizadas as ferramentas Hive e Impala, sendo possível explorar e ganhar conhecimento sobre os dados com uma linguagem de alto nível como o HiveQL e SQL. Para a transformação de dados recorreu-se à *framework* de processamento distribuído MapReduce, responsável pelo processamento de dados ao longo dos *nodes* do *cluster* Hadoop. A transformação de variáveis foi criada usando a ferramenta Apache Pig que, através de *jobs* MapReduce, consegue processar grandes volumes de dados. Ainda foram utilizadas as ferramentas Apache Oozie, para o *scheduling* das tarefas de pré-processamento, e o Apache Solr, para a visualização de alguns dados armazenados no Hive. Como é possível verificar, através do Hadoop é possível recorrer a vários tipos de processamento utilizados pelas organizações.

A componente de *analytics* foi via Apache Spark com a implementação dos algoritmos *K-Means*, para *clustering*, e *FP-Growth*, para a extração das regras de associação. Caso a opção fosse a implementação destes dois algoritmos via Hadoop/MapReduce, estes teriam de ser parametrizados com Java, além de que os tempos de processamento seriam muito superior, como foi verificado durante a fase de pré-processamento em, por exemplo, Apache Pig que utiliza como base MapReduce. Ao longo da execução dos scripts de Hive e Pig verifica-se um plano de execução distribuído por várias etapas, aumentando o tempo de processamento. Com o uso do Spark, através da criação de clusters e extração de *itemsets* frequentes, foi possível concluir que a capacidade de processamento do Spark é muito superior face ao Hadoop, que continua a ser essencial para o armazenamento dos ficheiros em ambientes distribuídos. Ao longo do projeto foi essencial a utilização do HDFS que permitiu armazenar os outputs derivados das várias ferramentas de processamento MapReduce e Spark.

A utilização das bibliotecas de Spark permitiu recorrer na mesma aplicação, de forma integrada, a *syntaxes* de SQL, Scala e componentes de *advanced analytics*, para a descoberta de conhecimento em grandes volumes de dados. Pelo que a componente de *Data Mining* neste projeto Spark, obrigaria a uso de ferramentas MapReduce, que constituiria um problema durante o projeto, pela dificuldade de implementação e baixos índices de *performance*. Relativamente à escolha da linguagem para a vertente analítica, os algoritmos foram escritos em Scala, pois é a linguagem da infraestrutura Spark, pelo que existe maior suporte para novas funcionalidades e maior velocidade de processamento.

Este trabalho permitiu assim construir um sistema de apoio à decisão com tecnologias *Big Data*, no qual, foram verificados diversos benefícios que não seriam possíveis com as tradicionais ferramentas. A grande escalabilidade alcançada para grandes volumes de dados seria um *handicap* com a utilização dos tradicionais RDBMS, como um SQL Server, pois os tempos de processamento seriam impraticáveis. No caso do Hadoop e Spark, os dados são distribuídos em blocos por vários *clusters* e processados paralelamente, pelo que os tempos de processamento de grandes volumes de dados são muito reduzidos. Além de que a utilização do ecossistema Hadoop com Spark permitiu, através de um

conjunto integrado de ferramentas, o tratamento de dados e extração de conhecimento num ambiente centralizado no HDFS. Esta arquitetura tem a vantagem de não exigir um processo de migração de dados ao longo de várias plataformas, que constitui um problema nos projetos de diversas organizações.

O Hadoop e o Spark, além dos benefícios apontados, tornam-se uma solução muito relevante para as organizações ao nível do custo de investimento, uma vez que se tratam-se de ferramentas *open source,* ao contrário das tecnologias tradicionais, como SAP ou Microsoft, que exigem uma licença de utilização. Assim, é possível concluir que a integração das ferramentas Hadoop e Spark nas atuais arquiteturas de sistemas de informação permitiriam um conjunto de benefícios de curto prazo pois, tecnologias *open source.* Funcionam de forma distribuída, atingindo maior capacidade de processamento. Oferecem grande escalabilidade e proteção de dados, pois constituem uma solução *fault-tolerant*, com capacidade de autorrecuperação.

6. LIMITAÇÕES E RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Ao longo do trabalho foram detetadas algumas limitações relacionadas com o volume de dados. A utilização da máquina virtual *Cloudera Distribution Incluidng Apache Hadoop* (CDH 5.8), apesar do considerável número de registos da base de dados, não permitiu efetuar este projeto com volumes de dados superiores. Outra limitação foi para a extração do gráfico do cotovelo que, após o teste de uma solução de pySpark com Jupyter Notebook, teve de ser extraído em RStudio com dados em disco, logo não armazenados no HDFS. A simplicidade da base de dados também foi uma limitação deste projeto, visto que não permitiu o teste de alguns temas de pré-processamento de dados, como a análise de componentes principais ou do tratamento de variáveis categóricas. É recomendável que se utilize uma base de dados mais completa para o teste destas funcionalidades.

Algumas sugestões/recomendações são dadas na medida em que se pensa que existe trabalho a complementar todo o estudo desenvolvido e que, como tal, podem trazer mais benefícios para o projeto. A implementação do algoritmo *Random Forest* para a criação de um *time series forecasting,* que permitisse prever os valores de transação praticados ao longo dos dias de semana, funcionaria como caso de teste para aplicação de algoritmos supervisionados através de *Spark MLlib.* Por último, recomenda-se a instalação do *plugin* de Scala IDE no Eclipse, instalado na máquina virtual, de modo a facilitar os desenvolvimentos dos algoritmos. O projeto foi desenvolvido via Spark Shell, dificultando alguns testes das funcionalidades de Spark.

7. BIBLIOGRAFIA

Kimble, C., & Milolidakis, G. (2015). *Big Data* and Business Intelligence: Debunking the Myths. Global Business and Organizational Excellence, 35(1), 23-34. doi:10.1002/joe.21642

Neves, P. C., & Bernardino, J. (2015). *Big Data* in the cloud. Open Journal of *Big Data* (OJBD), 1(2), 3--20. Retrieved from http://www.slideshare.net/AmazonWebServices/big-data-the-cloud-13953902

Lydia, E., & Swarup, M. (2016). Analysis of *Big Data* through Hadoop Ecosystem Components like Flume, MapReduce, Pig and Hive. Ijcse.Net, 5(1), 21–29.

Mark Grover, Ted Malaska, Jonathan Seidman, Gwen Shapira. (2015). Hadoop Application Architectures. Retrieved January 12, 2017, from http://shop.oreilly.com/product/0636920033196.do

Prasad, T. K., & Sheth, A. P. (2013). Semantics-Empowered Approaches to *Big Data* Processing for Physical-Cyber-Social Applications (pp. 68–75). Presented at the 2013 AAAI Fall Symposium Series. Retrieved from http://works.bepress.com/tk_prasad/24

Leskovec, J., Rajaraman, A., & Ullman, J. (2014). Link Analysis. Mining of Massive Datasets, 163–200. Retrieved from http://i.stanford.edu/~ullman/mmds.html

Holmes, A. (2012). Hadoop in practice. Shelter Island, NY: Manning.

Kornacker, M., Behm, A., Bittorf, V., Bobrovytsky, T., Ching, C., Choi, A., Yoder, M. (2015). Impala: A Modern, Open-Source SQL Engine for Hadoop. Cidr.

Ryza, S., Laserson, U., Owen, S., & Wills, J. (2015). Advanced analytics with Spark. Beijing: O'Reilly.

Karau, H., Konwinski, A., Wendell, P., & Zaharia, M. (2015). Learning Spark: Lightning-fast data analysis. Sebastopol (CA): O'Reilly.

Zaharia, M., Chowdhury, M., Das, T., & Dave, A. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. NSDI'12 Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, 2–2. https://doi.org/10.1111/j.1095-8649.2005.00662.x

Dean, J. (2014). *Big Data*, *Data Mining*, and *Machine Learning*: Value Creation for Business Leaders and Practitioners, 265. https://doi.org/10.1126/science.1247727

Jurney, R. (2013). Agile data science. Sebastopol, CA: O'Reilly Media.

Donoho, D. (2015). 50 years of Data Science, 1–41.

Horvitz, E. (2016). ARTIFICIAL INTELLIGENCE AND LIFE IN 2030.

Gupta, R., & Pathak, C. (2014). A *Machine Learning* framework for predicting purchase by online customers based on dynamic pricing. Procedia Computer Science, 36(C), 599–605. https://doi.org/10.1016/j.procs.2014.09.060

Woo, J. (2015). Market Basket Analysis using Spark, 5(4), 207–209.

Amirghodsi, S. (2016). Apache spark *Machine Learning* cookbook. Place of publication not identified: Packt Publishing Limited.

Jiawei, H., Kamber, M., Han, J., Kamber, M., & Pei, J. (2012). *Data Mining*: Concepts and Techniques. San Francisco, CA, itd: Morgan Kaufmann.

Feng, C., Pan, D., Wan, J., Zhang, D., Athanasios, V., & Xiaohui, R. (2015). *Data Mining* for the internet of things: Literature review and challenges. International Journal of Distributed Sensor Networks, 2015(i). https://doi.org/10.1155/2015/431047

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From *Data Mining* to knowledge discovery in databases. AI Magazine, 37–54. https://doi.org/10.1145/240455.240463

Hand, D., Mannila, H., & Smyth, P. (2001). Principles of *Data Mining* Cambridge. MIT Press (Vol. 2001). https://doi.org/10.1007/978-1-4471-4884-5

M. Ester, H.-p. Kriegel, J. S e X. Xu. "A density-based algorithm for discovering clusters in large spatial databases with noise". Em: In 2nd Int. Conf. on Knowledge Discovery and *Data Mining* (KDD 96). Portland, USA, 1996, pp. 226–231.

Ma, Z. (2016). Managing big data in cloud computing environments. Hershey, PA: Information Science Reference.

Bengfort, B., & Kim, J. (2016). Data analytics with Hadoop: An introduction for data scientists. Sebastopol, CA: O'Reilly.

Bhattacharyya, S., & Dutta, P. (2013). Handbook of research on computational intelligence for engineering, science, and business. Hershey, PA: Information Science Reference.

Williams, S. (2016). Business intelligence strategy and big data analytics: a general management perspective. Cambridge, MA: Morgan Kaufmann.

Berkhin, P., 2002, "Survey of Clustering *Data Mining* Techniques", Technical Report, Accrue Software.

Silvers, F. (2012). Data warehouse designs: Achieving ROI with market basket analysis and time variance. Boca Raton, FL: CRC Press.

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). Association Analysis: Basic Concepts and Algorithms. Introduction to *Data Mining*, 327–414. https://doi.org/10.1111/j.1600-0765.2011.01426.x

Kaur, J., Singh, R., & Gurm, R. K. (2016). Performance evaluation of Apriori algorithm using association rule mining technique, 2(5).

Han, J., Pei, J., & Yin, Y. (2000). Frequent Pattern Tree : Design and Construction. Networks, 1–12. https://doi.org/10.1145/342009.335372

Santos, M., & Ramos, I. (2009). Business Intelligence: Tecnologias da Informação na Gestão de Conhecimento. FCA-Editora de Informática, 25. Retrieved from http://repositorium.sdum.uminho.pt/bitstream/1822/6198/1/Resumo_Livro_BI_MYS_IR.pdf

Buyya, R., Broberg, J. & Goscinski, A. (2011). Cloud Computing, Principles and Paradigms. New Jersey: John Wiley & Sons, Inc.

Unyelioglu, K. (2016, July 22). Data Clustering Using Apache Spark - DZone *Big Data*. Retrieved January 27, 2017, from https://dzone.com/articles/cluster-analysis-using-apache-spark-exploring-colo

8. ANEXOS

8.1. EXTRAÇÃO DOS DADOS DO HDFS PARA UMA TABELA EM HIVE

```
set hive.mapred.supports.subdirectories=true;
set hive.supports.subdirectories=true;
set mapreduce.input.fileinputformat.input.dir.recursive=true;
CREATE EXTERNAL TABLE staging_area(
 ID BIGINT,
 Store INT,
 Market INT,
 Dept INT,
 Category BIGINT,
 Company BIGINT,
 Brand BIGINT,
 Date_ID STRING ,
 ProductSize INT,
 ProductMeasure CHAR(2),
 PurchaseQuantity INT,
 PurchaseAmount DOUBLE)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
  location '/user/cloudera/Input'
        tblproperties ("skip.header.line.count"="1");
```

Código utilizado para importar os dados a partir do HDFS para Hive

8.2. EXTRAÇÃO DE UMA AMOSTRA ESTRATIFICADA EM HIVE

```
CREATE TABLE sample AS
select A.ID as Customer_ID,
      A.Store,
      A.Market,
      A.Dept,
      A.Category,
      A.Company,
      A.Brand,
      from_unixtime(unix_timestamp(A.DATE_ID , 'yyyy-MM-dd')) AS Date_ID,
      YEAR(A.DATE_ID) as Year_ID,
      (INT((MONTH(A.DATE_ID)-1)/3)+1) as Quarter,
      MONTH(A.DATE_ID) as Month_ID,
      weekofyear(A.DATE_ID) AS Week_ID,
      DAY(A.DATE_ID) as Day_ID,
      from_unixtime(unix_timestamp(A.DATE_ID , 'yyyy-MM-dd'),'u') as WeekDay,
      A.ProductSize,
      A.ProductMeasure,
      A.PurchaseQuantity,
      A.PurchaseAmount
from (select e.*,
          tb1.seqnum
          from (select Distinct d. Store as Store_rank,DENSE_RANK() over (order by d.Store, RAND())
as seqnum
                    from staging_area d) tb1
                   join staging_area e
              on tb1. Store _rank = e.Store
    ) A
where seqnum < 41 and Date_ID >= '2012-10-01';
```

Código utilizado para a extração da amostra estratificada

8.3. EXEMPLO PARA A DETEÇÃO DE OUTLIERS EM IMPALA

```
SELECT company
   FROM (
    SELECT company
      , Freq
      , 1 AS flg
      , AVG(Freq)        OVER () AS avg_Freq
      , STDDEV_SAMP(Freq) OVER () AS std_Freq
    FROM (
      SELECT company
        , COUNT(*) AS Freq
      FROM sample
      GROUP BY company ) TB_Freq
    ) outlier
WHERE Freq > avg_Freq + std_Freq AND Freq < avg_Freq - std_Freq
```

Código utilizado para deteção de outliers da variáveis “company”

8.4. TRATAMENTO DE VARIÁVEIS EM HIVE E PIG

```
create table dim_customer as
SELECT customer_id,
        count(DISTINCT week_id) as freq_visit,
           cast(sum(purchaseamount) as int) as monetary,
              count(category) as total_articles,
                    cast(datediff(to_date("2013-06-30 00:00:00"), to_date(max(date_id)))/7 as int) as
Recency_Weeks
FROM sample
GROUP BY customer_id
```

Código utilizado para criação das variáveis representativas do valor de consumidor

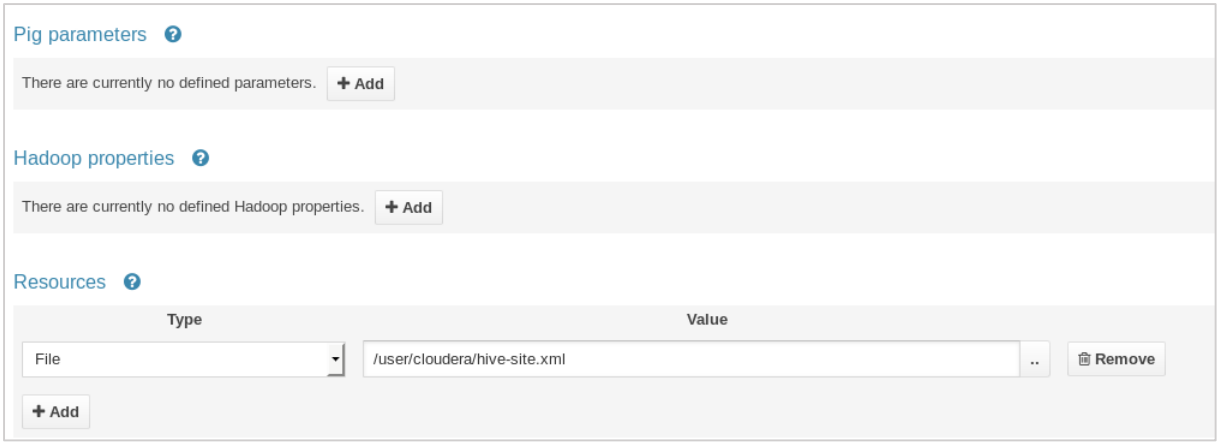

Figura 38 – Importação do ficheiro de configuração de Hive para conexão em Pig

```
data = load 'sample' using org.apache.hive.hcatalog.pig.HCatLoader()
AS
        (Customer_ID:Int,
          Freq_Visit:Int,
          Monetary:Double,
          Total_Articles:Int,
          Recency_Weeks:Int
          Store:Int,
          Market:Int,
          Dept:Int,
          Category:Int,
          Company:Int,
          Brand:Int,
          Date_ID:Chararray,
          Year_ID:Int,
          Quarter:Int,
          Month_ID:Int,
          Week_ID:Int,
          Day_ID:Int,
          Weekday:Int,
          Product_Size:Int,
          Product_Measure:Int,
          Purchase_Quantity:Double,
```

```
        Purchase_Amount:Double);
DATA_TRANSFORMATION = FOREACH data GENERATE
                              Customer_ID,
                              Store,
                              Market,
                              Dept,
                              Category,
                              Company,
                              Brand,
                              Date_ID,
                              Year_ID,
                              Quarter,
                              Month_ID,
                              Week_ID,
                              Day_ID,
                              Weekday,
                              Product_Size,
                              Product_Measure,
                              Purchase_Amount/Purchase_Quantity AS Product_Value,
                              Purchase_Quantity,
                              Purchase_Amount,
                              Freq_Visit,
                              Monetary,
                              Recency_Weeks;

RANKING = rank DATA_TRANSFORMATION BY Customer_ID,Date_ID,Store DENSE;
GRP = GROUP RANKING BY $0;
SUMMED = foreach GRP {
    Purchase_Summed = SUM(RANKING.Purchase_Amount);
          Purchase_Quantity = SUM(RANKING.Purchase_Quantity);
    generate $0, Purchase_Quantity AS TRANSACTION_QUANTITY, Purchase_Summed AS
TRANSACTION_AMOUNT;
}
JOINED = join RANKING by $0, SUMMED by $0;
```

```
FINAL = foreach JOINED generate Customer_ID, Freq_Visit, Monetary, Total_Articles,
Recency_Weeks, Store, Market, Year_ID,Quarter,Month_ID, Week_ID, Day_ID, Weekday ,Dept,
Category AS Product_ID,Product_Value,$0 AS TRANSACTION_ID,
TRANSACTION_QUANTITY,TRANSACTION_AMOUNT;
DISTINCT_SET = DISTINCT FINAL;
ORDER_BY_TRANSACTION = ORDER DISTINCT_SET BY TRANSACTION_ID;
STORE ORDER_BY_TRANSACTION INTO '/user/cloudera/Staging_Area' USING PigStorage(',');
```

Código utilizado para criação de variáveis em Pig

8.5. EXTRAÇÃO DO GRÁFICO DO COTOVELO EM R

```
library(NbClust)
source_data=read.csv("transactions.csv",header = FALSE, sep = ",")
names(source_data) <- c("Customer_ID",
"Freq_Visit",
"Monetary",
"Total_Articles",
"Recency_Weeks",
"Store",
“Market”,
“Year_ID”,
"Quarter",
"Month_ID",
"Week_ID",
"Day_ID",
"Weekday",
"Dept",
"Product_ID",
"Product_Value",
"TRANSACTION_ID",
"TRANSACTION_QUANTITY",
"TRANSACTION_AMOUNT")
data_sample <- source_data[sample(nrow(source_data), 10000), ]
data_sample
myData <- data_sample [,2:5]
myData
wss <- (nrow(myData)-1)*sum(apply(myData,2,var))
for (i in 1:10) {
wss[i] <- sum(kmeans(myData,
centers=i)$withinss) }
plot(1:10, wss, type="b", xlab="Number of Clusters",
ylab="Within groups sum of squares")
k <- NbClust(dataset,distance="euclidean",method="kmeans",min.nc=2,max.nc=5)
```

Código R para definir o número ótimo de outliers

8.6. IMPLEMENTAÇÃO DO ALGORIMTO K-MEANS EM SPARK

```
scala> import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.linalg.Vectors

scala> import org.apache.spark.mllib.clustering.KMeans
import org.apache.spark.mllib.clustering.KMeans

scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._

scala> val data = sc.textFile("/user/cloudera/Staging_Area")
data: org.apache.spark.rdd.RDD[String] = /user/cloudera/Staging_Area MapPartitio
nsRDD[1] at textFile at <console>:32
```

Figura 39 – Criação do RDD para a implementação do algoritmo K-Means

```
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.clustering.KMeans
import org.apache.spark.sql.functions._
val data = sc.textFile("/user/cloudera/Staging_Area")
val allSplit = data.map(line => line.split(","))
case class Transactions(Customer_ID:Int,
                        Freq_Visit:Double,
                        Monetary:Double,
                        Total_Articles:Double,
                        Recency_Weeks:Double,
                        Store:Int,
                       Market:Int,
                        Year_ID:Int,
                        Quarter:Int,
                        Month_ID:Int,
                        Week_ID:Int,
                        Day_ID:Int,
                        Weekday:Int,
                        Dept:Int,
                        Product_ID:Int,
                        Product_Value:Double,
                        TRANSACTION_ID:Int,
                        TRANSACTION_QUANTITY:Double,
```

```
                    TRANSACTION_AMOUNT:Double)
val allData = allSplit.map( p => Transactions(
                    p(0).toInt,
                    p(1).toDouble,
                    p(2).toDouble,
                    p(3).toDouble,
                    p(4).toDouble,
                    p(5).toInt,
                    p(6).toInt,
                    p(7).toInt,
                    p(8).toInt,
                    p(9).toInt,
                    p(10).toInt,
                    p(11).toInt,
                    p(12).toInt,
                    p(13).toInt,
                    p(14).toInt,
                    p(15).toDouble,
                    p(16).toInt,
                    p(17).toDouble,
                    p(18).toDouble))
val allDF = allData.toDF()
val rowsRDD = allDF.rdd.map(r => (
                    r.getInt(0),
                    r.getDouble(1),
                    r.getDouble(2),
                    r.getDouble(3),
                    r.getDouble(4),
                    r.getInt(5),
                    r.getInt(6),
                    r.getInt(7),
                    r.getInt(8),
                    r.getInt(9),
                    r.getInt(10),
```

```
                r.getInt(11),
                r.getInt(12),
                r.getInt(13),
                r.getInt(14),
                r.getDouble(15),
                r.getInt(16),
                r.getDouble(17),
                r.getDouble(18)))
rowsRDD.cache()
val vectors_transactions = allDF.rdd.map(r => Vectors.dense(r.getDouble(1),
r.getDouble(2),r.getDouble(3), r.getDouble(4)))
val kMeansModel_transactions = KMeans.train(vectors_transactions, 3, 20)
vectors_transactions.cache()
kMeansModel_transactions.clusterCenters.foreach(println)
val predictions_transaction = rowsRDD.map{r => (r._1,
kMeansModel_transactions.predict(Vectors.dense(r._2,r._3, r._4, r._5) ))}
val predDF_transaction = predictions_transaction.toDF("Customer_ID", "Cluster_Customer")
val t = allDF.join(predDF_transaction, "Customer_ID")
t.rdd.saveAsTextFile("/user/cloudera/Clustering")
val input = sc.textFile("/user/cloudera/Clustering").flatMap { line => line.split("\n") }.distinct()
input.map { vertex => vertex.replace("[","").replace("]","")
}.saveAsTextFile("/user/cloudera/Clustering/Regex")
```

Código Scala para a implementação do algoritmo K-Means

8.7. IMPLEMENTAÇÃO DO FP-GROWTH EM SPARK PARA O CLUSTER 1

```
import org.apache.spark.sql.functions._;
import org.apache.spark.sql.DataFrame;
import org.apache.spark.sql.SQLContext;
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.functions.col;
import org.apache.spark.mllib.fpm.FPGrowth
import org.apache.spark.rdd.RDD
import org.apache.spark.mllib.clustering.{KMeans, KMeansModel}
import org.apache.spark.mllib.linalg.Vectors
val data = sc.textFile("/user/cloudera/Clustering/Regex")
case class Transactions(Customer_ID:String, Freq_Visit:String,
Monetary:String,Total_Articles:String,Recency_Weeks:String,
Store:String,Market:String,Date_ID:String,Quarter:String,Month_ID:String,Week_ID:String,
Year_ID:String,Weekday:String,Dept:String,Product_ID:String,Product_Value:String,TRANSACTION
_ID:String,TRANSACTION_QUANTITY:String,TRANSACTION_AMOUNT:String,Customer_Cluster:Stri
ng)
def csvToMyClass(line: String) = {
val p = line.split(',');
Transactions(p(0),p(1),p(2),p(3),p(4), p(5),p(6),p(7),
p(8),p(9),p(10),p(11),p(12),p(13),p(14),p(15),p(16),p(17),p(18),p(19)}
val df = data.map(csvToMyClass).toDF("Customer_ID", "Freq_Visit",
"Monetary","Total_Articles","Recency_Weeks","Store","Market",
"Year_ID","Quarter","Month_ID","Week_ID","Day_ID","Weekday","Dept","Product_ID","Product_
Value","TRANSACTION_ID","TRANSACTION_QUANTITY","TRANSACTION_AMOUNT",
"Customer_Cluster")
df.registerTempTable("Transactions")
val i = 0;
val df = sqlContext.sql("SELECT TRANSACTION_ID,Product_ID FROM Transactions WHERE
Customer_Cluster = 1")
val rawDict = df.select('Product_ID).sort('Product_ID)
val dictCounts = rawDict.groupBy('Product_ID).count()
val valid = dictCounts.filter('count > 1).drop('count)
val singleRemoved=df.join(valid,Seq("Product_ID")).distinct()
```

```
val baskets =
singleRemoved.groupBy('Transaction_ID).agg(collect_list('Product_ID).as('StockCodes)).cache()
import org.apache.spark.mllib.fpm.FPGrowth
import org.apache.spark.rdd.RDD
val basketsRDD = baskets.select('StockCodes).as[Array[String]].rdd
val fpg = new FPGrowth().setMinSupport(0.01).setNumPartitions(20)
val model = fpg.run(basketsRDD)
val numBaskets = baskets.count()
val freqItemsets = model.freqItemsets.map(itemset => (itemset.freq, itemset.items)).toDF("freq",
"items")
val freqItems = freqItemsets.filter(size('items) === 1).select('items(0).as('item), 'freq)
val numFreqItems = freqItems.count()
val freqDict = freqItems.join(singleRemoved, 'item === 'Product_ID, "left").select('item,
'Product_ID).as[(String, String)].collect().toMap
val toDescs = udf { items: Seq[String] => items.map(item => freqDict(item)) }
val freqItemDescs = freqItemsets.select('freq, toDescs('items).as('itemDescs))
val minConfidence = 0.5
val associationRules = model.generateAssociationRules(minConfidence)
val associationRuleNames = associationRules.map { rule => (rule.antecedent.map(item =>
freqDict(item)).mkString(",") + " => " + rule.consequent.map(item => freqDict(item)).mkString(","),
rule.confidence) }.toDF("rule", "confidence")
associationRuleNames.show();
associationRuleNames.rdd.coalesce(1, shuffle =
true).saveAsTextFile("/user/cloudera/Market_Basket_Analysis/Cluster_1")
```

Código Scala para a implementação do algoritmo FP-Growth e extração das Regra de Associação