

Mestrado em Engenharia de Software

Estágio

Relatório Final

Motor de deteção de *outliers*

José Miguel Cruz Alves

jmcalves@student.dei.uc.pt

Orientadores do estágio:

Francisco Valdez | Novabase

Alexandre Miguel Pinto | DEI

Data: 1 de Julho de 2016



FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Mestrado em Engenharia de Software

Estágio

Relatório Final

Motor de deteção de *outliers*

José Miguel Cruz Alves

jmcalves@student.dei.uc.pt

Júri:

Alberto Cardoso

João Paulo Vilela

Data: 1 de Julho de 2016



Resumo

Hoje em dia praticamente todas as empresas, inclusive as de telecomunicações, recorrem a sistemas que auxiliam o processo de tomada de decisão. Estes sistemas oferecem, na sua interface gráfica, *dashboards* contendo um conjunto de gráficos que ilustram indicadores de desempenho da empresa, permitindo aos gestores de topo analisá-los, de forma a retirar conclusões sobre o progresso da mesma. No entanto, o processo de análise dos referidos gráficos pode tornar-se uma tarefa bastante demorada uma vez que é realizada apenas por pessoal com elevadas competências de análise.

Uma empresa de telecomunicações sentiu a necessidade de tornar esta tarefa transversal, ou seja, permitir que todos os funcionários tenham acesso ao seu progresso individual para que eles próprios tirem conclusões sobre a sua atividade diária e consequentemente melhorem o seu desempenho.

Porém, nem todos os funcionários têm capacidades para analisar este tipo de gráficos. Aliado a esse problema, o facto de os funcionários não terem muito tempo disponível torna o processo de análise ainda mais difícil. Desta forma, torna-se fulcral criar um mecanismo que permita aos funcionários compreender os resultados apresentados nos gráficos de uma forma rápida, sem ter de perder muito tempo a analisá-los. No sentido de desenvolver tal mecanismo, esta empresa contactou a Novabase para que esta procedesse a esse desenvolvimento.

A Novabase identificou então esta oportunidade de desenvolver um componente que complementa os tradicionais gráficos com textos — frases simples, às quais damos o nome de *Findings* — que apresentam a informação já analisada evitando que os colaboradores da empresa tenham de perder tempo a interpretar os gráficos. Este mecanismo revela ser uma mais-valia, uma vez que agiliza o processo de análise de informação.

O objetivo do estágio passou por desenvolver o componente enunciado anteriormente, o qual designámos de *Findings Engine*. Este motor foca-se na deteção de eventos/valores/comportamentos que se desviam significativamente do normal, e é por esse motivo se pode dizer que se trata de um *Engine* de deteção de *outliers* produzindo descrições textuais dessas deteções. Este componente visa processar a informação produzida diariamente por uma empresa de telecomunicações e, depois de a analisar, apresenta-a ao utilizador num formato textual com o intuito de fornecer conclusões rápidas. O *Findings Engine* será, numa primeira fase, integrado num Sistema de Suporte à Decisão (SSD) já existente no cliente. Contudo é esperado que no futuro possa vir a ser integrado noutros SSD direcionados à indústria das telecomunicações. Outro desafio que tivemos de ultrapassar foi garantir que o produto fosse o mais parametrizável possível, para que pudesse vir a ser integrado em vários ambientes diferentes.

Palavras-chave: *Findings Engine*, *Findings*, *Outliers*, Telecomunicações

Índice

Resumo	1
Índice.....	2
Índice das Figuras	5
Índice das Tabelas.....	7
Glossário	9
Acrónimos.....	10
1 Introdução	11
1.1 Contexto	11
1.2 Motivação.....	11
1.3 Estágio.....	13
1.4 Objetivos	13
1.5 Estrutura do Documento.....	14
2 Metodologia de desenvolvimento	14
2.1 <i>Drop I</i>	15
2.2 <i>Drop II</i>	15
2.3 <i>Drop III</i>	16
2.4 Ferramentas utilizadas.....	16
3 Estado da Arte.....	18
3.1 Detecção de <i>outliers</i>	18
3.1.1 Estatísticos	19
3.1.2 Profundidade	25
3.1.3 Desvio	26
3.1.4 Distância	27
3.1.5 Densidade.....	28
3.1.6 Ângulo.....	29

3.1.7	<i>Clustering</i>	31
3.2	Classificação dos <i>outliers</i>	32
3.2.1	Classificação decimal.....	32
3.2.2	Classificação binária	32
3.3	Comparação entre métodos	32
3.4	Metodologia	33
4	Análise de Requisitos.....	34
4.1	Requisitos funcionais	35
4.2	Atributos de qualidade	44
4.2.1	Interoperabilidade	46
4.2.2	Parametrização.....	46
4.2.3	Escalabilidade	46
4.3	Restrições	47
5	Arquitetura	47
5.1	Tecnologias	48
5.2	Base de dados	49
5.2.1	<i>Data</i>	51
5.2.2	<i>Combination</i>	52
5.2.3	<i>Output</i>	54
5.2.4	<i>Configuration</i>	55
5.3	<i>Findings Engine</i>	55
5.4	Fluxo dos dados.....	57
6	Implementação	57
6.1	Dados operacionais	58
6.1.1	Métricas.....	58
6.1.2	Atributos	59
6.2	Extração.....	60

6.2.1	Combinações.....	62
6.2.2	Leitura dos dados	65
6.3	Deteção.....	65
6.3.1	Deteção de <i>outliers</i>	66
6.3.2	Estabelecer a <i>baseline</i>	68
6.3.3	Priorização	69
6.4	Interpretação.....	71
6.4.1	Classificação	72
6.4.2	Tradução	73
6.5	<i>Output</i>	74
6.5.1	Base de dados.....	74
6.5.2	<i>Web service</i>	75
7	Testes	76
7.1	Performance	77
7.2	Unitários.....	81
7.3	Aceitação.....	81
8	Conclusões e Trabalho futuro	85
8.1	Análise do trabalho realizado.....	85
8.2	Trabalho futuro.....	85
9	Referências.....	87

Índice das Figuras

Figura 1. (a) Número de vendas feitas por um colaborador de uma determinada loja ao longo de um dia; (b) Finding produzido para ser mostrado na sexta-feira.	12
Figura 2. Diagrama de Gantt detalhado descrevendo o trabalho realizado no primeiro semestre (drop I)	15
Figura 3. Diagrama de Gantt detalhado descrevendo o trabalho realizado no segundo semestre (drop II).....	16
Figura 4. Diagrama de Gantt detalhado descrevendo o trabalho realizado no segundo semestre (drop III).....	16
Figura 5. Google Drive - Ferramenta utilizada para o armazenamento e partilha de ficheiros.	17
Figura 6. Trello – Ferramenta utilizada para o auxílio no cumprimento das metas propostas.	17
Figura 7. (a) Conjunto de dados sem presença de outliers e com os respetivos limites da média e desvio padrão e do MAD; (b) Conjunto de dados com presença de outliers e com os respetivos limites da média e desvio padrão e do MAD.....	24
Figura 8. Comportamento de alto-nível dos métodos baseados em profundidade. Cada linha representa uma curva batimétrica. Os números no gráfico representam a profundidade de cada camada.	25
Figura 9. Comparação entre a densidade à volta do ponto A e a densidade à volta dos seus vizinhos.	28
Figura 10. Ângulo α formado entre o vetor de distância AB e AC	30
Figura 11. (a) (b) (c) Vetores de distância formados pelo ponto A e respetivo ângulo entre eles; (d) (e) (f) Vetores de distância formados pelo ponto B e respetivo ângulo entre eles. ...	31
Figura 12. Ecossistema BI contendo o componente Findings Engine.....	34
Figura 13. Diagrama de casos de uso identificados para o Findings Engine. Apenas serão desenvolvidas as funcionalidades assinaladas a azul.....	35
Figura 14. Diagrama de camadas que representa a arquitetura de alto-nível do Findings Engine.	48
Figura 15. Comparação da performance entre Oracle e MySQL	49
Figura 16. Diagrama físico da base de dados. (PK) – Primary Key; (FK) – Foreign Key;	50
Figura 17. Componentes que integram a camada do Findings Engine.....	56
Figura 18. Interação entre os utilizadores/sistema operacional e o Findings Engine.	57

Figura 19. Amostra dos dados operacionais de uma empresa de telecomunicações.	58
Figura 20. Diferença entre a análise de uma métrica e duas métricas, com vários atributos combinados. (a) Análise de uma métrica (vermelho) com quatro atributos (azul) combinados. (b) Análise de duas métricas (vermelho) com quatro atributos (azul) combinados.	59
Figura 21. Conjunto de dados de análise – vendas feitas pelo John – e de comparação – vendas feitas pelos colegas; Linha a tracejado: conjunto de comparação; Linha contínua: conjunto de análise.....	61
Figura 22. Número médio de vendas diárias feitas pelo John e pelos seus colegas; Azul - vendas do John; Cinzento - vendas dos colegas	62
Figura 23. Diagrama de fluxo do algoritmo de deteção de outliers - MAD.	66
Figura 24. Resultado da aplicação do MAD ao conjunto de dados de comparação – vendas feitas pelos colegas do John; Pontos vermelhos: outliers.	67
Figura 25. Comparação entre a aplicação do MAD e do teste de quartis ao número de vendas feitas pelos colegas do John. (a) Limites resultantes do MAD; (b) Limites resultantes do teste de quartis.	68
Figura 26. Desvio percentual entre as vendas irregulares feitas pelo John – outliers – e a baseline.	69
Figura 27. Matriz urgência/importância.	70
Figura 28. (a) Prioridade com que cada venda feita pelo John deve ser tratada. Ponto vermelho: prioridade 4. Pontos amarelos: prioridade 2; (b) Matriz urgência/importância com os respetivos níveis de prioridade. Nível 4: maior prioridade. Nível 1: menor prioridade.....	71
Figura 29. Classificação de cada venda feita pelo John.	72
Figura 30. (a) Vendas irregulares feitas pelo John com o respetivo desvio percentual face à baseline; (b) Findings produzidos relativamente às vendas irregulares feitas pelo John	74
Figura 31. Findings devolvidos em formato JSON através do web service.	76
Figura 32. Tempo de execução de cada componente do sistema.	78
Figura 33. Tempo de execução de cada componente do sistema.	80

Índice das Tabelas

Tabela 1. Relação entre o desvio máximo aceitável e o desvio padrão, para diferentes números de amostras [8].	20
Tabela 2. Valor de R para diferentes números de amostras e diferentes números de observações suspeitas. [9].	21
Tabela 3. Valores <i>Qtable</i> para diferentes números de amostras e diferentes intervalos de confiança. [11].	22
Tabela 4. Comparação entre os vários métodos de deteção de outliers.	33
Tabela 5. Descrição do ator do sistema – Assistente.	36
Tabela 6. Descrição do ator do sistema – Supervisor.	36
Tabela 7. Descrição do ator do sistema – Gestor.	36
Tabela 8. Descrição do ator do sistema – Administrador.	37
Tabela 9. Descrição do ator do sistema - Findings Engine.	37
Tabela 10. Descrição do ator do sistema - Sistema Operacional.	37
Tabela 11. Cenário do caso de uso - UC1.	38
Tabela 12. Cenário do caso de uso - UC2.	39
Tabela 13. Cenário do caso de uso – UC3.	39
Tabela 14. Cenário do caso de uso - UC4.	40
Tabela 15. Cenário do caso de uso - UC5.	41
Tabela 16. Cenário do caso de uso - UC6.	41
Tabela 17. Cenário do caso de uso - UC7.	42
Tabela 18. Cenário do caso de uso - UC8.	43
Tabela 19. Cenário do atributo de qualidade - QA1.	44
Tabela 20. Cenário do atributo de qualidade - QA2.	45
Tabela 21. Cenário do atributo de qualidade - QA3.	45
Tabela 22. Restrições do Findings Engine.	47
Tabela 23. Descrição das colunas da tabela data.	51
Tabela 24. Descrição das colunas da tabela combination.	52

Tabela 25. Descrição das colunas da tabela output.....	54
Tabela 26. Descrição das colunas da tabela translation.	55
Tabela 27. Tabela combination populada com dados que irão dar origem ao Finding.	63
Tabela 28. Queries criadas pelo mecanismo da extração.....	63
Tabela 29. Tabela configuration populada com o nome das tabelas com que o Findings Engine irá interagir.	64
Tabela 30. Tabela de factos SALES_PER_WEEK criada para responder ao Finding.....	64
Tabela 31. Dados parametrizados nas colunas urgency e importance da tabela combination.	70
Tabela 32. Dados parametrizados nas colunas above e below da tabela combination.	72
Tabela 33. Frase textual, parametrizada na coluna translation, em que o outlier será transformado.	73
Tabela 34. Tabela de output com os Findings produzidos pelo sistema.....	75
Tabela 35. Informação sobre lojas/assistentes utilizados para efeitos de teste.....	77
Tabela 36. Queries criadas para responder ao Finding. Dados hospedados numa tabela de factos indexada.....	78
Tabela 37. Queries criadas para responder ao Finding. Dados hospedados numa vista materializada.	80
Tabela 38. Resultados dos testes unitários.....	81
Tabela 39. Teste de aceitação - T_01.....	82
Tabela 40. Teste de aceitação - T_02.....	82
Tabela 41. Teste de aceitação - T_03.....	83
Tabela 42. Teste de aceitação - T_04.....	83
Tabela 43. Teste de aceitação - T_05.....	83
Tabela 44. Teste de aceitação - T_06.....	84
Tabela 45. Teste de aceitação - T_07.....	84
Tabela 46. Teste de aceitação - T_08.....	84

Glossário

Curva batimétrica	Curva que une todos os pontos de um conjunto de dados com a mesma densidade.
Data Warehouse	Repositório de dados que contém o historial dos dados de um sistema.
D_{max}	Desvio máximo aceitável.
ETL	<i>Extraction Transformation and Loading</i> é o processo de extração, transformação e carregamento dos dados, de uma base de dados operacional para um <i>data warehouse</i> .
Finding	Frase simples que apresenta informação já analisada.
MVC	<i>Model-view-controller</i> é uma arquitetura padrão amplamente utilizada. Esta arquitetura separa em três camadas distintas a representação da informação (<i>view</i>), as regras de negócio (<i>controller</i>) e o armazenamento dos dados (<i>model</i>).
n	Número de amostras do conjunto de dados x .
Q_{table}	Tabela que contém o valor Q _{table} para um determinado número de amostras e intervalo de confiança.
x_i	Valor na posição i do conjunto de dados x .
\bar{x}	Média do conjunto de dados x .
σ	Desvio padrão do conjunto de dados x .

Acrónimos

ABOD	<i>Angle Based Outlier Detection</i>
BD	Base de dados
BI	<i>Business Intelligence</i>
CPU	<i>Central Processing Unit</i>
GB	<i>GigaByte</i>
GHz	<i>GigaHertz</i>
IP	<i>Internet Protocol</i>
IT	<i>Information Technology</i>
JSON	<i>JavaScrip Object Notation</i>
MB	<i>MegaByte</i>
K-NN	<i>K-Nearest Neighbor</i>
LOF	<i>Local Outlier Factor</i>
OLAP	<i>OnLine Analytical Processing</i>
RAM	<i>Random-Access Memory</i>
SIM	<i>Subscriber Identification Module</i>
SO	Sistema Operativo
SQL	<i>Structured Query Language</i>
SSD	Sistema de Suporte à Decisão
XML	<i>EXtensible Markup Language</i>

1 Introdução

O presente documento é o resultado do estágio realizado na Novabase, sob a supervisão de Alexandre Miguel Pinto, Professor Auxiliar Convidado no Departamento de Engenharia Informática da Universidade de Coimbra, e de Francisco Valdez, consultor BI na Novabase.

O primeiro capítulo está subdividido em cinco secções. A primeira contextualiza o problema abordado durante o estágio. A segunda secção descreve as razões que deram origem ao estágio bem como a sua importância e contributo, apresentando também as maiores dificuldades que se encontraram e ultrapassaram. A terceira secção descreve informação sobre a empresa onde se realizou o estágio. Os principais objetivos do estágio bem como as características do produto estão descritos na quarta secção. A descrição de cada capítulo seguinte do documento é feita na quinta secção.

1.1 Contexto

O termo “sistemas de suporte à decisão” (SSD) foi introduzido por Peter Keen e Scott Morton durante os anos 70. Nessa altura os dois investigadores publicaram um livro [1] onde definiram o termo como “sistemas computacionais que têm impacto nas decisões” [2]. Esta definição revolucionou a forma como os computadores passaram a ser usados e deu origem a diversas ideias que foram a base da criação desses sistemas. Ao longo dos anos, diferentes definições foram aparecendo. Em 1984 Mann e Watson descreveram um SSD como “um sistema interativo que fornece ao utilizador modelos de decisão e conjuntos de dados, a fim de apoiar na tomada de decisão” [2].

Um grande passo foi dado em 1987 quando a *Texas Instruments* completou o desenvolvimento do *Gate Assignment Display System*, um SSD desenvolvido para a *United Airlines* que auxilia a tomada de decisões no terreno.

Durante os anos seguintes, a definição e o âmbito dos SSD sofreram várias alterações. Nos anos 90, com o aparecimento dos conceitos *data warehouse* e OLAP, Srague e Watson definiram os SSD como sendo “sistemas computacionais que ajudam as pessoas que têm tomar decisões, através da apresentação de dados e modelos analíticos” [2]. Já Sauter definiu os SSD como “sistemas computacionais que aglomeram informação de várias fontes de dados e apoiam as empresas no processo de análise de informação” [2].

Apesar das diferentes alterações, a definição de um SSD convergiu para um *software* interativo que ajuda os utilizadores a compilar informação que pode ser útil na tomada de decisão. Esta informação é normalmente recolhida dos *data warehouses* e é apresentada ao utilizador final através de gráficos.

1.2 Motivação

Hoje em dia os SSD são amplamente usados no seio das empresas e têm um papel fundamental no sucesso das mesmas [3]. Muitas delas integraram estes sistemas com o objetivo de monitorizar a atividade diária. Um SSD pode ajudar os gestores a perceberem melhor o negócio da empresa através de análises de custos, comparações de desempenho, relatórios da atividade diária, entre outros. Estes indicadores, ao serem analisados minuciosamente, podem contribuir com uma grande ajuda no processo de tomada de decisão.

A principal limitação destes sistemas, tal como foi referido na secção anterior, é que os resultados são apresentados ao utilizador final em forma de gráfico. Desta forma as empresas optam por ter pessoas com capacidades, nomeadamente gestores de topo, para fazer a análise desses gráficos a fim de retirar resultados conclusivos. Uma vez que uma empresa gera grandes quantidades de informação diariamente o processo de análise torna-se uma tarefa demorada.

De forma a colmatar os problemas enunciados anteriormente, uma empresa de telecomunicações sentiu a necessidade de reformular o processo de análise de informação, tornando-o numa tarefa transversal no seio da empresa. Para além dos gestores de topo, a empresa de telecomunicações decidiu disponibilizar a informação ao pessoal operacional que trabalha no terreno para eles próprios fazerem a análise do seu desempenho. Porém, eles nem sempre têm tempo ou conhecimentos para analisar os gráficos apresentados no SSD, pelo que é fulcral arranjar um mecanismo alternativo.

Como resposta ao problema identificado, a Novabase viu uma oportunidade de desenvolver um componente, denominado *Findings Engine*, capaz de analisar e interpretar conjuntos de dados para serem apresentados ao utilizador de uma forma digerida, ou seja, ao contrário dos SSD tradicionais a informação é disponibilizada em formato textual. A esta informação textual já digerida e analisada damos o nome de *Findings*. Para perceber melhor este conceito a figura 1 (a) mostra um gráfico relativamente ao número de vendas de um determinado produto ao longo de uma semana numa loja e a figura 1 (b) apresenta um possível *Finding* que poderia ser produzido para reportar a um colaborador dessa loja.



Figura 1. (a) Número de vendas feitas por um colaborador de uma determinada loja ao longo de um dia; (b) *Finding* produzido para ser mostrado na sexta-feira.

Todos nós estamos familiarizados com texto, uma vez que lemos ou escrevemos todos os dias, portanto o objetivo deste componente é acelerar todo o processo de análise de informação, uma vez que não há necessidade de interpretar gráficos. Através da figura 1 (b) o processo de análise torna-se muito mais rápido, sendo mais fácil tirar conclusões comparativamente com o gráfico da figura 1 (a). Para além disso, a análise de dados pode passar a ser uma tarefa transversal na empresa, estando cada empregado responsável pela monitorização dos seus próprios níveis de desempenho.

O objetivo deste estágio é desenvolver o componente que foi referido anteriormente. Um dos desafios do estágio passa por perceber a informação relevante que deve ser apresentada ao utilizador final. Outro desafio será a tradução da análise em texto. Por fim, um dos maiores

desafios é desenvolver o componente de tal forma flexível que, numa primeira fase possa ser integrado com um SSD específico mas mais tarde possa vir a ser integrado em qualquer SSD.

1.3 Estágio

O estágio decorreu durante um período de 9 meses na sede da Novabase em Lisboa. A Novabase é uma empresa de IT focada essencialmente na implementação de soluções de negócio. A atividade da empresa abrange as seguintes indústrias:

- Governo
- Transportes
- Energia
- Serviços financeiros
- Telecomunicações

O *Findings Engine* será integrado num SSD direcionado às empresas na indústria das telecomunicações. Hoje em dia, estas empresas possuem sistemas de suporte à decisão que lhes permitem monitorizar a atividade diária e consequentemente ajudar os seus gestores no processo de tomada de decisão. Estes sistemas guardam informação como a quantidade de produtos vendidos diariamente, o tempo que cada colaborador demora a registar um processo de venda, entre outros indicadores de desempenho.

Por exemplo, quando nos dirigimos a uma loja para comprar um telemóvel, o empregado do balcão regista a venda do produto numa aplicação que, por sua vez guarda esses dados numa base de dados. Estes dados, registados diariamente por todas as lojas de uma empresa, são mais tarde processados e mostrados aos gestores através das *dashboards* ou dos relatórios do SSD da empresa. Através da análise desta informação o gestor consegue, por exemplo, perceber as lojas que vendem mais produtos, os dias em que se faturou mais, os colaboradores que venderam menos produtos. O *Findings Engine* será integrado num destes SSD e, ao complementar as tradicionais *dashboards* com frases textuais que apresentam informação já digerida, irá agilizar o processo de análise de informação.

1.4 Objetivos

Como referido anteriormente, o objetivo principal deste estágio foi desenvolver um componente de *software* que respondesse às necessidades de uma empresa de telecomunicações, tornando o resultado do processo de análise de informação de compreensão mais fácil e direto possível para o humano. Em vez de recolher dados de um conjunto de sistemas operacionais e depois compilá-los para serem apresentados em formato gráfico, pretende-se revolucionar este processo analisando e interpretando comportamentos que fujam de um padrão normal para que sejam apresentados em formato textual. Esta abordagem permite ao utilizador final tirar conclusões mais rapidamente sem ter necessariamente de analisar os tradicionais gráficos.

De forma a satisfazer as necessidades, o componente deve ter as seguintes funcionalidades implementadas e testadas:

- **Deteção de comportamentos anormais:** um mecanismo, baseado na deteção de *outliers*, que identifica os dados que se desviam do padrão esperado para serem reportados ao utilizador final em formato de *Finding*;

- **Interpretação de comportamentos anormais:** um processo de interpretação do significado do comportamento desviante, bem como a sua importância para o utilizador;
- **Tradução de comportamentos anormais:** um mecanismo que traduz os comportamentos desviantes em frases conclusivas, permitindo ao utilizador interpretá-las rapidamente;

O processo de desenvolvimento do *Findings Engine* inclui a análise de requisitos, para que todos os requisitos sejam identificados e documentados; o desenho da arquitetura tendo em conta os requisitos identificados; a implementação do componente e os testes do mesmo de forma a garantir a qualidade do produto final.

1.5 Estrutura do Documento

Após este capítulo introdutório, que explica os aspetos básicos do estágio, o documento está dividido nos seguintes capítulos:

2. Metodologia de desenvolvimento: descreve a metodologia, bem como as ferramentas utilizadas no desenvolvimento do componente;
3. Estado da Arte: apresenta uma breve descrição sobre a deteção de *outliers* e os algoritmos da literatura desenvolvidos para esse propósito, fazendo uma análise comparativa entre eles;
4. Análise de requisitos: apresenta os requisitos que devem ser assegurados de forma a garantir a qualidade do produto final;
5. Arquitetura: descreve as decisões arquiteturais tomadas nesta fase preliminar;
6. Implementação: resume as funcionalidades que compõem o motor;
7. Testes: capítulo dedicado aos testes de requisitos funcionais e não funcionais;
8. Conclusões e Trabalho Futuro: apresenta conclusões relativamente ao trabalho executado e ainda o trabalho a realizar no futuro;

2 Metodologia de desenvolvimento

Este capítulo é dedicado à descrição da metodologia utilizada no processo de desenvolvimento do *Findings Engine*. Sendo este um projeto de desenvolvimento de *software*, diversas abordagens podem ser seguidas de forma a obter um produto com maior qualidade e que cumpra os requisitos propostos.

O processo de desenvolvimento escolhido divide-se em três *drops* distintas, culminando cada uma delas com a entrega de um produto funcional. Uma *drop* é um ciclo de desenvolvimento que dura, em média, três meses. Trata-se portanto de uma metodologia iterativa permitindo assim identificar antecipadamente os problemas que surgem para serem corrigidos nas *drops* seguintes.

2.1 Drop I

A *drop I*, que decorreu durante o primeiro semestre, representa o primeiro ciclo do desenvolvimento do *Findings Engine* e como tal, para além das tarefas comuns às *drops* futuras, a primeira fase é dedicada à análise do estado da arte de forma a estudar o trabalho já desenvolvido na área em que o produto será integrado. Esta *drop* serve essencialmente para definir um conjunto de propriedades base sobre as quais o produto será desenvolvido. Dado que se trata do primeiro ciclo de desenvolvimento, as decisões tomadas serão as mais simples possíveis, uma vez que as *drops* futuras servem para aperfeiçoar o trabalho desenvolvido.

Esta abordagem revela ser bastante vantajosa uma vez que o final da primeira *drop* culmina com a entrega de um produto funcional já testado, permitindo perceber se as decisões tomadas vão de acordo com as necessidades do cliente, ou se porventura o projeto deve tomar outro rumo. A próxima figura ilustra o planeamento detalhado da primeira *drop*.



Figura 2. Diagrama de Gantt detalhado descrevendo o trabalho realizado no primeiro semestre (*drop I*)

2.2 Drop II

A *drop II* representa o segundo e maior ciclo de desenvolvimento do produto. Esta *drop* começou com uma fase de análise que permitiu perceber melhor as necessidades do cliente e ajustar o planeamento consoante essas mesmas necessidades.

Inicialmente estava previsto que o *Findings Engine* detetasse comportamentos anormais em dados bidimensionais, porém a prioridade do cliente foi focar a análise em conjuntos de dados unidimensionais. Para além desta surgiram outros desvios face ao planeamento inicial:

- Algoritmo da deteção de *outliers*
- Tecnologia para hospedar os dados operacionais

Para além destas alterações, o maior foco desta *drop* foi desenvolver novas funcionalidades que irão ser explicadas na íntegra nos capítulos 5 e 6. A próxima figura (figura 3) ilustra na íntegra o planeamento da *drop II*.

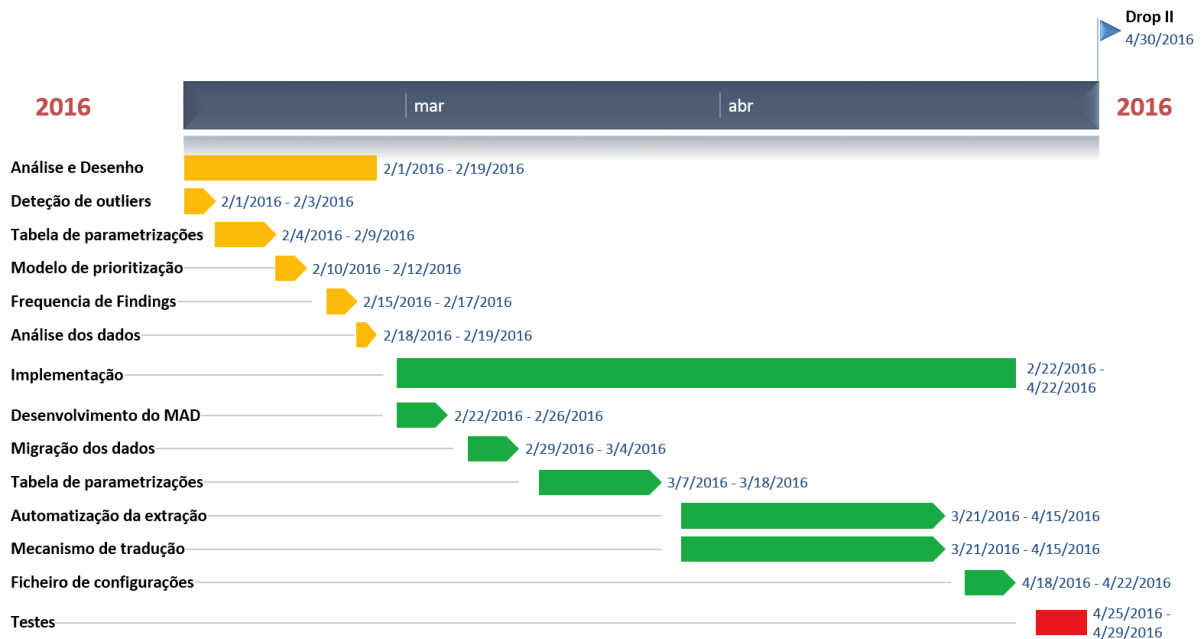


Figura 3. Diagrama de Gantt detalhado descrevendo o trabalho realizado no segundo semestre (*drop II*).

2.3 Drop III

A *drop III* é o último ciclo de desenvolvimento do *Findings Engine* e tem como foco principal afinar as funcionalidades desenvolvidas nas *drops* anteriores. A figura seguinte ilustra o planeamento da *drop III*.



Figura 4. Diagrama de Gantt detalhado descrevendo o trabalho realizado no segundo semestre (*drop III*).

2.4 Ferramentas utilizadas

Durante as três *drops* foram utilizadas ferramentas auxiliares no processo de desenvolvimento. Mais especificamente foi utilizado o *Google Drive* para a partilha de ficheiros e o *Trello* para garantir o cumprimento das datas propostas. As figuras 5 e 6 ilustram o papel que estas ferramentas tiveram no projeto.

The screenshot shows the Google Drive interface. On the left is a sidebar with navigation options: My Drive, Shared with me, Google Photos, Recent, Starred, and Trash. Below this, it indicates '2 GB of 15 GB used' and an 'Upgrade storage' button. The main area shows a breadcrumb path: 'My Drive > Celfocus > 13. Engines > Findings Engine'. Below the path is a table listing files and folders.

Name ↑	Owner	Last modified
00. Preparação	Francisco Valdez	Oct 1, 2015 Francisco Valdez
01. Estado da Arte	Francisco Valdez	Oct 1, 2015 Francisco Valdez
02. Análise	Francisco Valdez	Oct 1, 2015 Francisco Valdez
03. Desenho e Implementação	Francisco Valdez	Oct 1, 2015 Francisco Valdez
04. Operacionalização	Francisco Valdez	Oct 1, 2015 Francisco Valdez
Notas	me	Oct 16, 2015 me
Relatório	me	Oct 16, 2015 me
Sprints	me	Oct 16, 2015 me

Figura 5. Google Drive - Ferramenta utilizada para o armazenamento e partilha de ficheiros.

The screenshot shows a Trello board titled 'Outliers Detection' with a 'Team Visible' icon. The board is organized into three columns: 'To Do', 'Doing', and 'Done'. Each column contains several task cards with progress bars, checkmarks, and due dates.

- To Do:**
 - Investigação - Técnicas (0/4, Oct 16, 2015)
 - Use Case (Drop I) (1/2, Oct 23, 2015)
 - Requisitos (Drop I) (0/1, Oct 30, 2015)
 - Desenho (Drop I) (0/1, Nov 6, 2015)
 - Protótipo (Drop I) - Mecanismo Detecção Outliers (0/2, Nov 16, 2015)
 - Protótipo (Drop I) - Mecanismo de Tradução em Findings (Dec 11, 2015)
 - Apresentação Protótipo (Drop I) (0/1, Dec 11, 2015)
 - Relatório Intermédio - Entrega do
- Doing:**
 - Relatório Intermédio - Estrutura e Plano de Comunicação (2/2, Oct 9, 2015)
 - Investigação - Conceitos (1/4, Oct 9, 2015)
- Done:**
 - Enquadramento (1/2, Oct 2, 2015)

Figura 6. Trello – Ferramenta utilizada para o auxílio no cumprimento das metas propostas.

As reuniões semanais com o orientador Francisco Valdez e as reuniões mensais com o orientador Alexandre Miguel Pinto permitiram um acompanhamento contínuo do progresso do projeto, garantindo assim a máxima qualidade do produto e o cumprimento dos objetivos propostos.

3 Estado da Arte

Neste capítulo é explicado o significado de *outlier*, bem como os algoritmos da literatura que são utilizados na detecção dos mesmos.

3.1 Detecção de *outliers*

Um *outlier* [4] é uma observação que se desvia fortemente das outras observações num conjunto de dados. Em 1980, Hawkins [5] definiu um *outlier* como “uma observação que se desvia tanto das outras observações que é suspeitável que tenha sido gerada por um mecanismo diferente”. Este desvio pode ser natural ou pode ser reflexo de que algo errado existe no conjunto de dados, como por exemplo um erro de medição. Porém, a detecção de *outliers* num conjunto de dados pode ser determinante para ajudar a perceber se existe algum padrão anormal e é já hoje utilizada nas seguintes áreas:

- Detecção de fraude: normalmente quando um cartão de crédito é roubado, o comportamento relativamente às compras efetuadas é significativamente diferente do anterior ao roubo. Este novo padrão pode ser encontrado através da detecção de *outliers*. Também num sistema podem ser detetados intrusos através desta técnica;
- Medicina: a detecção de *outliers* é usada para determinar comportamentos anormais em dados médicos que podem indicar condições de doença. (e.g. alterações do batimento cardíaco, valores anormais nas análises sanguíneas);
- Desporto: no desporto as técnicas de detecção de *outliers* são utilizadas para acompanhar o desempenho de um atleta. O desvio do mesmo pode revelar ser bastante conclusivo quando se trata de analisar a condição do atleta;
- Mercado: a detecção de *outliers* é utilizada para analisar as tendências num mercado. A presença de padrões anormais podem indicar oportunidades de compra/venda;

Ao longo dos anos a detecção de *outliers* foi objeto de muitas investigações. Os primeiros passos foram dados pela comunidade estatística [6] que se baseou sobretudo em modelos matemáticos para a detecção. Mais recentemente, investigadores associados à área computacional desenvolveram vários estudos [6] que contribuíram para o avanço da detecção de *outliers*, sendo eles os responsáveis pelo aparecimento de diversos algoritmos. Esses algoritmos podem ser divididos nas seguintes categorias:

- Estatísticos
- Baseados em profundidade
- Baseados em desvio
- Baseados em distância
- Baseados em densidade

- Baseados no ângulo
- *Clustering*

As próximas secções descrevem cada uma das categorias supramencionadas dando exemplos de algoritmos que fazem parte de cada uma. Os algoritmos estatísticos que são apresentados no presente documento foram desenvolvidos para detetar e eliminar *outliers*, no entanto, apenas será descrito o processo de deteção.

3.1.1 Estatísticos

Como foi referido anteriormente, os métodos estatísticos foram os primeiros métodos usados na deteção de *outliers* [7]. Estes métodos são os mais simples de implementar, uma vez que se baseiam na aplicação de fórmulas matemáticas ao conjunto de dados de forma a detetar os pontos que se desviam da restante maioria.

Apesar destes métodos serem matematicamente mais precisos que os restantes, não permitem detetar *outliers* em conjuntos de dados com mais que uma dimensão. Para além disso, alguns métodos estatísticos assumem que os dados seguem uma determinada distribuição (e.g. distribuição normal/exponencial), o que pode tornar a sua aplicação bastante limitada. Contudo, uma vantagem desta abordagem é que não requer nenhum parâmetro fornecido pelo utilizador.

Na próxima secção serão apresentados quatro algoritmos que são geralmente usados para detetar *outliers*.

3.1.1.1 Critério de Chauvenet

O critério de Chauvenet, desenvolvido por William Chauvenet, é um método estatístico que é usado para detetar e eliminar conjuntos de pontos que sejam considerados *outliers*. Este algoritmo assume que os dados seguem uma distribuição normal e os *outliers* são identificados através da média e do desvio padrão. A maior desvantagem deste algoritmo é o facto de se basear na média e desvio padrão, que são duas medidas clássicas fortemente afetadas pela presença de *outliers*. Outra desvantagem é o facto de detetar apenas um *outlier* de cada vez, ou seja, no caso de existirem dois *outliers* no conjunto de dados o algoritmo só determina o segundo *outlier* depois de eliminar o primeiro [8].

Os próximos passos descrevem o funcionamento do algoritmo:

- 1) Calcula a média e o desvio padrão do conjunto de dados.
- 2) Obtém a relação entre o desvio máximo aceitável e o desvio padrão através da tabela (tabela 1).
- 3) Calcula o valor absoluto da diferença entre cada ponto e a média do conjunto de dados, e divide esse valor pelo desvio padrão. A próxima fórmula representa este cálculo:

$$\frac{|x_i - \bar{x}|}{\sigma}$$

- 4) Se o resultado da fórmula em 3) for maior que o valor obtido em 2), o ponto é considerado um *outlier* e pode ser eliminado do conjunto de dados.

Tabela 1. Relação entre o desvio máximo aceitável e o desvio padrão, para diferentes números de amostras [8].

Número de amostras	$\left(\frac{D_{max}}{\sigma}\right)$
3	1.38
4	1.54
5	1.65
6	1.73
7	1.80
8	1.87
9	1.91
10	1.96
15	2.13
20	2.24
25	2.33
50	2.57
100	2.81
300	3.14
500	3.29
1000	3.48

Apesar do algoritmo ser simples de implementar, a desvantagem é que os *outliers* são identificados iterativamente, portanto em vez de retornar todos os *outliers* no conjunto de dados, o algoritmo deteta um *outlier* e só depois de o eliminar é que consegue identificar um novo. No caso do conjunto de dados ter muitas amostras e conter uma quantidade significativa de *outliers* o algoritmo tem um mau desempenho.

3.1.1.2 Critério de Peirce

O critério de Peirce, desenvolvido por Benjamin Peirce, é um algoritmo semelhante ao critério de Chauvenet, uma vez que também foi desenhado para detetar e eliminar *outliers*. Este algoritmo também assume que os dados seguem uma distribuição normal e utiliza a média e o desvio padrão para identificar os pontos que se desviam do padrão esperado. No entanto, a maior diferença entre os dois algoritmos reside no número de *outliers* que cada um determina.

O critério de Peirce, ao contrário do critério de Chauvenet, suporta a detecção de múltiplos *outliers* de uma só vez.

O algoritmo funciona da seguinte forma:

- 1) Calcula a média e o desvio padrão.
- 2) Obtém o valor R da tabela (tabela 2).
- 3) Calcula o desvio máximo aceitável através da seguinte fórmula:

$$D_{\max} = \sigma * R$$

- 4) Calcula o valor absoluto da diferença entre cada ponto e a média do conjunto de dados, através da seguinte fórmula:

$$|x_i - \bar{x}|$$

- 5) Se o resultado da equação descrita em 4) for maior que o valor obtido em 3), o ponto é considerado um *outlier* e pode ser eliminado.

Tabela 2. Valor de R para diferentes números de amostras e diferentes números de observações suspeitas. [9].

Número de amostras	Número de observações suspeitas				
	1	2	3	4	5
3	1.19				
4	1.38	1.07			
5	1.50	1.20			
6	1.61	1.29	1.09		
7	1.69	1.38	1.18	1.02	
8	1.76	1.45	1.26	1.10	
9	1.82	1.515	1.324	1.178	1.045

Para obter o valor de R da tabela em cima representada, é necessário saber quantas observações existem no conjunto de dados que são suspeitas de ser *outliers*. Desta forma, o utilizador tem de conhecer completamente o conjunto de dados mesmo antes de o submeter ao algoritmo de detecção de *outliers*, sendo esta a maior desvantagem do critério de Peirce.

3.1.1.3 Teste Q de Dixon

O teste Q de Dixon, tal como os algoritmos anteriores, foi desenvolvido para detetar e eliminar *outliers*, exceto que um *outlier* é determinado através da mediana. A vantagem de utilizar a

mediana – uma medida robusta – é que esta não é afetada pela presença de *outliers*. Este teste é apropriado para detetar *outliers* em conjuntos de dados com poucas amostras [10].

O algoritmo é descrito nos passos em baixo:

- 1) Ordena o conjunto de dados de forma crescente.
- 2) Obtém o valor Q_{table} através da tabela 3.
- 3) Calcula o valor Q para cada ponto do conjunto de dados através da seguinte fórmula:

$$Q = \frac{x_i - x_{i-1}}{x_n - x_1}$$

- 4) Se o resultado da equação 3) for maior que o valor obtido em 2), o ponto é considerado um *outlier*.

Tabela 3. Valores Q_{table} para diferentes números de amostras e diferentes intervalos de confiança. [11].

Número de amostras	$Q_{90\%}$	$Q_{95\%}$	$Q_{99\%}$
3	0.941	0.970	0.994
4	0.765	0.829	0.926
5	0.642	0.710	0.821
6	0.560	0.625	0.740
7	0.507	0.568	0.680
8	0.468	0.526	0.634
9	0.437	0.493	0.598

A tabela 3 contém os valores Q_{table} para cada tamanho do conjunto de dados e diferentes intervalos de confiança. Para um conjunto de dados com cinco amostras, se o ponto tiver um valor Q superior a 0.821 pode ser considerado *outlier* com 99% de confiança.

O problema deste algoritmo, tal como os dois enunciados anteriormente, é que é necessário o auxílio de uma tabela externa que, no caso do conjunto de dados ter muitos *outliers*, poderá tomar grandes proporções exigindo uma grande quantidade de recursos disponíveis. Este algoritmo, tal como os que foram descritos até então, são boas escolhas para detetar *outliers* em conjuntos de dados com poucas amostras.

3.1.1.4 Teste de quartis

O teste de quartis é um algoritmo estatístico que tem a particularidade de não depender de uma tabela para a deteção de *outliers*. Para além disso, o algoritmo foi desenvolvido apenas para

detetar *outliers* e não para os eliminar do conjunto de dados. Os quartis são o conjunto de três pontos que dividem o conjunto de dados em quatro partes iguais.

Eis o funcionamento do algoritmo:

- 1) Ordena o conjunto de dados de forma ascendente
- 2) Calcula o quartil Q1 e Q3 através das fórmulas:

$$Q1 = \frac{1}{4} * (n + 1)$$

$$Q3 = \frac{3}{4} * (n + 1)$$

- 3) Calcula a *inter quartil range* (IQR) através da seguinte fórmula:

$$IQR = Q3 - Q1$$

- 4) Um ponto é considerado um *outlier* se for maior que:

$$Q3 + 1,5 * IQR$$

Ou menor que:

$$Q1 - 1,5 * IQR$$

Este algoritmo revela ser uma boa escolha para conjuntos de dados com muitas amostras. Outra vantagem do teste de quartis é que pode ser facilmente implementado e ter por base a mediana para a deteção de *outliers*. Porém este algoritmo obtém resultados pouco satisfatórios para conjuntos de dados com poucas amostras. Aliado a esta desvantagem existe ainda o problema do teste de quartis detetar *outliers* apenas em dados unidimensionais.

3.1.1.5 Median Absolute Deviation

O *Median Absolute Deviation* (MAD) é um algoritmo estatístico robusto que permite detetar *outliers* [12]. Um método bastante simples de detetar *outliers* é recorrer à média e ao desvio padrão. Tipicamente, considera-se que um ponto é um *outlier* se for maior/menor que $\bar{x} \pm 1.96 * \sigma$. Porém, estas duas medidas são fortemente influenciadas pela presença de *outliers*, como já foi referido anteriormente.

Uma alternativa ao modelo clássico (média e desvio padrão) é o uso de um método robusto (mediana e MAD). Desta forma, o MAD está para a mediana como o desvio padrão está para a média e considera-se que um ponto é *outlier* se for maior/menor que $mediana \pm 1.96 * MAD$. A grande vantagem de utilizar a mediana e o MAD na deteção de *outliers* é o facto destas medidas não serem afetadas pela presença de medidas anormais.

A figura seguinte demonstra a comparação entre a deteção de *outliers* com o auxílio do MAD e do desvio padrão.

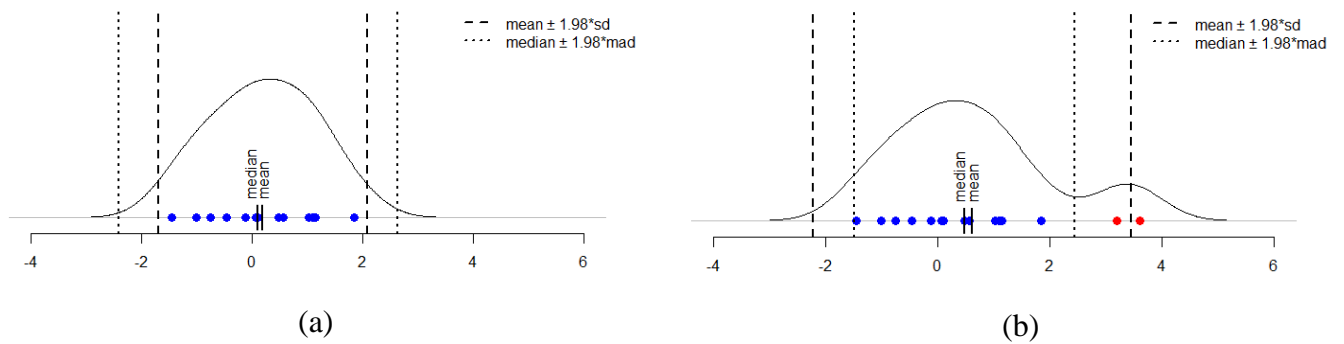


Figura 7. (a) Conjunto de dados sem presença de *outliers* e com os respetivos limites da média e desvio padrão e do MAD; (b) Conjunto de dados com presença de *outliers* e com os respetivos limites da média e desvio padrão e do MAD.

Na figura 7 (a) está representado um conjunto de dados sem qualquer presença de medidas anormais. Neste cenário os dois algoritmos (MAD e desvio padrão) tem resultados idênticos. No entanto, no caso do conjunto de dados ter medidas anormais, como mostra a figura 7 (b), os limites da média e desvio padrão são fortemente influenciados enquanto os do MAD se mantêm idênticos.

Os próximos passos indicam como se calcula o MAD:

- 1) Calcular a mediana (**M1**) do conjunto de dados.
- 2) Calcula o valor absoluto da diferença entre cada ponto e a mediana do conjunto de dados, através da seguinte fórmula:

$$|x_i - M1|$$

- 3) Calcular a mediana (**M2**) do novo conjunto de dados, resultante do ponto 2).
- 4) Calcular o MAD através da seguinte fórmula:

$$MAD = 1,4826 * M2$$

- 5) Um ponto é *outlier* se for maior que:

$$M1 + 1,96 * MAD$$

Ou menor que:

$$M1 - 1,96 * MAD$$

Para além de este método obter melhores resultados que a média e desvio padrão num cenário em que haja *outliers*, tem também a vantagem de obter bons resultados em conjuntos de dados pequenos, o que revela ser uma boa escolha quando se trata de detetar comportamentos anormais. A desvantagem do algoritmo é o facto de, à semelhança dos restantes apresentados até agora, não detetar comportamentos anormais em conjuntos de dados multidimensionais.

3.1.2 Profundidade

Os métodos baseados em profundidade são uma abordagem que tem em conta os princípios da geometria durante a deteção de *outliers*. Esta abordagem é baseada na definição de Tukey, em 1975, de curvas batimétricas [13]. Basicamente, a ideia principal por trás deste método é organizar o conjunto de dados em camadas de forma a detetar os *outliers*. Os pontos que se encontram nas camadas mais interiores tem menor probabilidade de ser *outliers*, comparativamente com os pontos que se encontram nas camadas exteriores.

Ao contrário dos métodos estatísticos, os métodos baseados em profundidade requerem parâmetros fornecidos pelo utilizador. Quando se trata do tipo de dados, estes métodos não assumem que os dados seguem um determinado tipo de distribuição. Simplesmente, os dados a serem analisados devem ser multidimensionais, porque só assim é possível atribuir uma profundidade a um ponto.

A figura seguinte (figura 8) ilustra como os métodos baseados em profundidade funcionam. É importante referir que os pontos com maior profundidade são os que tem menor probabilidade de ser *outliers*.

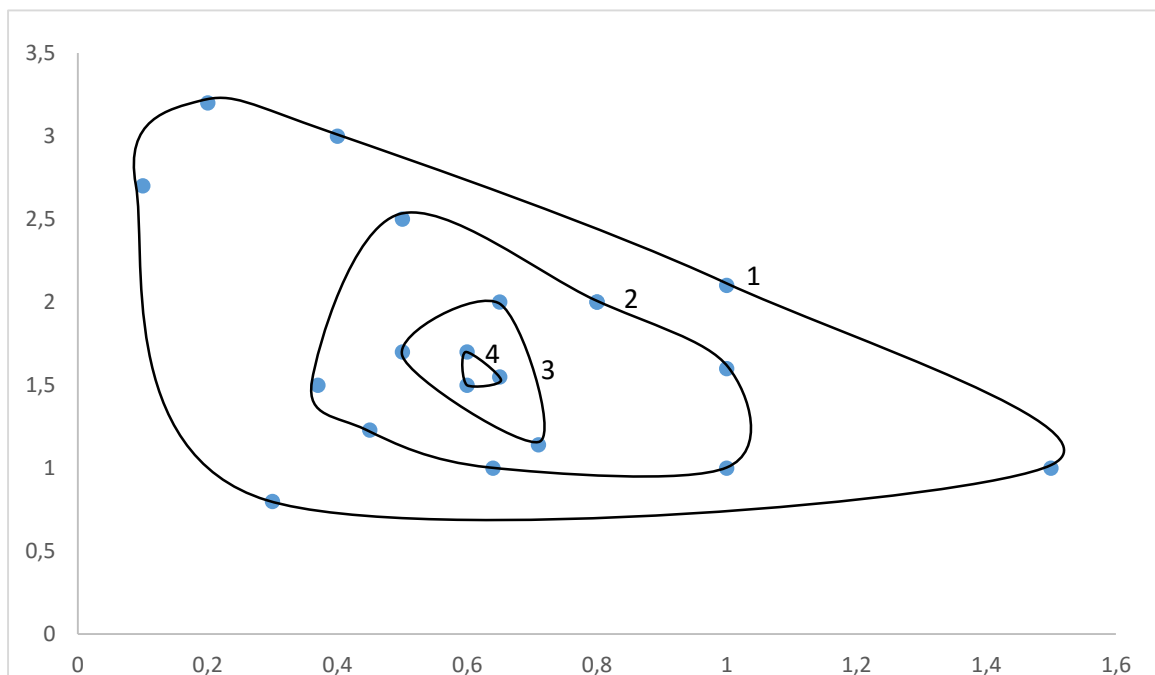


Figura 8. Comportamento de alto-nível dos métodos baseados em profundidade. Cada linha representa uma curva batimétrica. Os números no gráfico representam a profundidade de cada camada.

3.1.2.1 ISODEPTH

O algoritmo ISODEPTH foi projetado por Rousseeuw and Ruts [14] para calcular as K curvas batimétricas num conjunto de dados. O algoritmo tem por base o conceito de semiplano. Os próximos passos descrevem o comportamento de alto nível do algoritmo para um conjunto de dados com n pontos:

- 1) Para todos os pontos do conjunto de dados verifica se dois pontos não estão na mesma posição e se três pontos não são colineares.

- 2) Deteta todos os semiplanos que contêm $n-K$ pontos.
- 3) Intersecta os semiplanos, obtendo uma curva batimétrica.
- 4) Atribui aos pontos que estão contidos na curva barimétrica o valor $n-K$, sendo essa a profundidade dos pontos.

É importante referir que o algoritmo para quando estiver atribuída uma profundidade a todos os pontos. Posto isto, os pontos que têm menor profundidade são os que têm maior probabilidade de ser *outliers*. Apesar de ser um algoritmo que identifica os *outliers* através de um processo mais complexo, o cálculo das curvas barimétricas é uma tarefa computacionalmente dispendiosa, pelo que não é aconselhado aplicar o algoritmo em conjuntos de dados com muitas amostras. Porém, ao contrário dos algoritmos descritos até então, este tem a particularidade de detetar *outliers* em conjuntos de dados multidimensionais.

3.1.3 Desvio

Os métodos baseados em desvio detetam *outliers* com base nas características principais do conjunto de dados [15]. Tipicamente, os pontos que se desviam mais das características principais do conjunto de dados tendem a ser *outliers*. Para além disso, esta abordagem não assume que os dados seguem uma determinada distribuição.

3.1.3.1 Teste Sequential Exception

O teste *sequential exception* é um algoritmo que simula a forma como os seres humanos distinguem objetos anormais dentro de um conjunto de objetos com um determinado padrão. O algoritmo seleciona subconjuntos dos dados originais e calcula o fator de suavização que irá determinar os pontos que são *outliers*. Os próximos passos demonstram o funcionamento do algoritmo:

- 1) Num conjunto de dados A , o subconjunto A_i é selecionado.
- 2) A cardinalidade, C , do subconjunto $A-A_i$ é calculada. Neste contexto a cardinalidade é o número de amostras do conjunto de dados.
- 3) A função de dissemelhança, aplicada ao subconjunto $A-A_i$, é calculada através da fórmula:

$$D(A - A_i) = \frac{1}{n} * \sum_{i=1}^n (x_i - \bar{x})^2$$

- 4) O fator de suavização do subconjunto A_i é calculado através da fórmula:

$$S(A_i) = C(A - A_i) * (D(A) - D(A - A_i))$$

- 5) Quando o fator de suavização de A_i é máximo, os pontos contidos nesse subconjunto são considerados *outliers*.

Quando se trata de conjuntos de dados com muitas amostras deve ser evitada a utilização deste algoritmo, uma vez que este calcula todos os subconjuntos possíveis dentro de

um conjunto de dados, sendo esta operação bastante pesada em termos de desempenho. Contudo, uma vez que é fácil de implementar, o algoritmo pode ser uma boa escolha quando se trata de conjuntos de dados com poucas amostras.

3.1.4 Distância

Os métodos baseados em distância utilizam a distância entre os pontos do conjunto de dados para determinar a presença de *outliers*. Em conjuntos de dados multidimensionais, a distância entre um ponto e um conjunto de pontos pode ser um indicador da presença de *outliers*. Esta abordagem foi proposta por Ng e Knorr [16] e emergiu como uma alternativa aos tradicionais métodos de detecção.

A ideia base desta abordagem é que os pontos considerados normais têm uma vizinhança mais densa que os pontos considerados *outliers*. Ao contrário dos métodos estatísticos, esta abordagem requer parâmetros inseridos pelo utilizador, o que pode ser uma desvantagem.

Ao longo dos anos os investigadores têm estudado estes métodos e vários algoritmos foram propostos. A próxima secção descreve um algoritmo baseado em distância utilizado na detecção de *outliers*.

3.1.4.1 *K-Nearest Neighbor*

K-Nearest Neighbor é um algoritmo de classificação amplamente utilizado na área de análise preditiva. Geralmente é utilizado para classificar um determinado ponto de um conjunto de dados, tendo em conta as características desse mesmo conjunto.

Em 2002, Yihua and Vemuri [17] publicaram um estudo onde demonstram as vantagens computacionais do algoritmo na detecção de invasões. O estudo provou que o algoritmo deteta eficientemente padrões anormais dentro de um conjunto de dados.

Este algoritmo é denominado *K-Nearest Neighbor* uma vez que classifica um ponto com base na distância aos seus vizinhos mais próximos.

Os passos seguintes descrevem o funcionamento do algoritmo:

- 1) Para todos os pontos no conjunto de dados calcula a distância entre eles e os seus K vizinhos mais próximos através da distância euclidiana:

$$\sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

- 2) Quanto maior for a distância de um ponto aos seus K vizinhos mais próximos, maior é a sua probabilidade de ser um *outlier*.

É de notar que este algoritmo atribui um grau de *outlier* aos pontos no conjunto de dados, sendo o que tem maior classificação o ponto que se desvia mais do padrão normal. Neste caso, essa classificação é baseada na distância entre o ponto e os seus K vizinhos mais próximos. O *K-NN* tem a desvantagem de depender de parâmetros fornecidos pelo utilizador e

de ser pouco eficiente para conjuntos de dados com muitas amostras. Porém, a vantagem deste algoritmo é que pode ser aplicado em dados multidimensionais.

3.1.5 Densidade

Os algoritmos baseados em densidade determinam a presença de *outliers* num conjunto de dados através da comparação entre a densidade à volta de um ponto e a densidade à volta dos seus vizinhos. Com base neste pressuposto, um ponto é classificado com maior grau de *outlier* se a densidade dos seus vizinhos for superior à sua. Através da figura 9 é perceptível que existe menor densidade à volta do ponto A do que à volta dos seus vizinhos, fazendo deste um *outlier*.

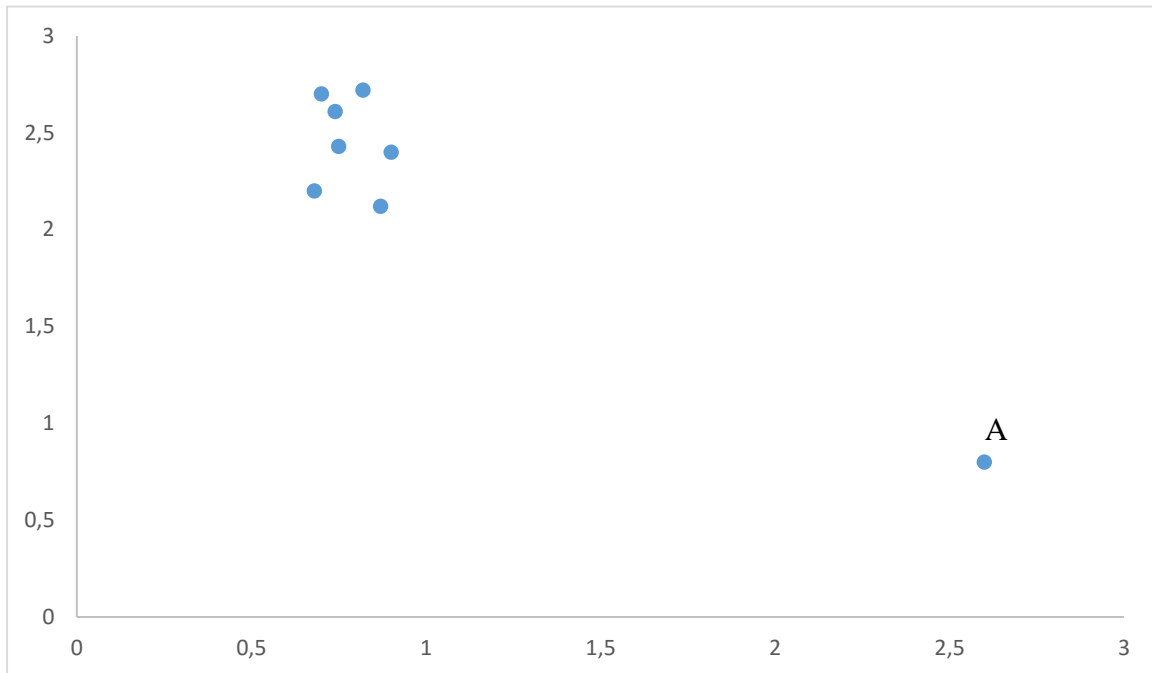


Figura 9. Comparação entre a densidade à volta do ponto A e a densidade à volta dos seus vizinhos.

Entre os vários algoritmos existentes para a deteção de *outliers* através da densidade, o que difere entre eles é a forma como determinam o grau de densidade entre os pontos. A secção seguinte descreve um algoritmo desenvolvido para esse efeito, o *Local Outlier Factor*.

3.1.5.1 LOF

Local Outlier Factor [18] foi um algoritmo proposto por Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng e Jörg Sander, em 2000. Os autores assumem que num conjunto de dados um *outlier* é um ponto cuja densidade de pontos à sua volta é inferior à densidade de pontos à volta dos seus vizinhos mais próximos.

Ao contrário de alguns algoritmos a cima mencionados que determinam se um ponto é ou não um *outlier*, o *LOF* atribui um grau de *outlier* a um ponto, ou seja, os pontos que se desviam mais do padrão normal têm consequentemente um grau de *outlier* maior. Esta abordagem, à semelhança do *K-NN*, revela ser uma mais-valia quando se pretende estabelecer um *ranking* entre os *outliers*.

Para o cálculo da densidade à volta dos pontos, o algoritmo tem por base o K -NN descrito na secção 3.1.4.1. Para tal necessita necessariamente de um parâmetro fornecido pelo utilizador.

Os passos que se seguem descrevem o funcionamento do algoritmo:

- 1) Para cada ponto calcula os K vizinhos mais próximos através do algoritmo descrito na secção 3.1.4.1
- 2) Determina a distância entre o ponto e os seus K vizinhos.
- 3) Calcula a *local reachability density* do ponto através da fórmula:

$$LRD(P) = 1 / \left(\frac{\sum_{i=0}^k reach-distance(P, P_i)}{|K|} \right)$$

- 4) Para cada um dos K vizinhos, faz o processo de 1) a 3).
- 5) Calcula o *local outlier factor* do ponto através da fórmula:

$$LOF(P) = \left(\frac{\sum_{i=0}^k \frac{LRD(P_i)}{LRD(P)}}{|K|} \right)$$

- 6) O ponto é considerado um *outlier* se o resultado obtido em 5) for muito superior a 1

É de notar que a *reach-distance*(P, P_i), descrita na fórmula em 3), é a distância entre o ponto e os seus K vizinhos, obtida no ponto 2). O algoritmo é bastante poderoso e destaca-se dos restantes por identificar os *outliers* através da densidade à volta dos pontos. Este algoritmo foi concebido para determinar *outliers* em conjuntos de dados bidimensionais, mas no caso de estes terem muitas amostras o algoritmo revela ser menos eficiente. Outra desvantagem, e à semelhança com o ISODEPTH (secção 3.1.2.1), é que este algoritmo não é tão simples de implementar comparativamente com os métodos estatísticos ou os métodos baseados em distâncias.

3.1.6 Ângulo

A deteção de *outliers* pode ser uma tarefa computacionalmente dispendiosa quando se tratam de dados multidimensionais. Até à presente secção foram descritos algoritmos que estão otimizados para detetar *outliers* num espaço unidimensional ou no melhor dos casos num espaço bidimensional. Por outro lado os algoritmos baseados em ângulos foram propostos para detetar *outliers* eficientemente num espaço multidimensional.

A próxima secção descreve o *Angle-Based Outlier Detection*, um algoritmo baseado em ângulos.

3.1.6.1 ABOD

O *Angle-Based Outlier Detection* [19] foi um algoritmo proposto por Hans-Peter Kriegel, Matthias Schubert e Arthur Zimek que não depende de nenhum parâmetro fornecido pelo utilizador e tem como ideia base analisar os ângulos formados por dois vetores de distância. É de notar que um vetor de distância é um vetor que conecta dois pontos do conjunto de dados. Na figura 10 podemos verificar dois vetores de distância num determinado conjunto de dados, bem como o ângulo formado entre eles.

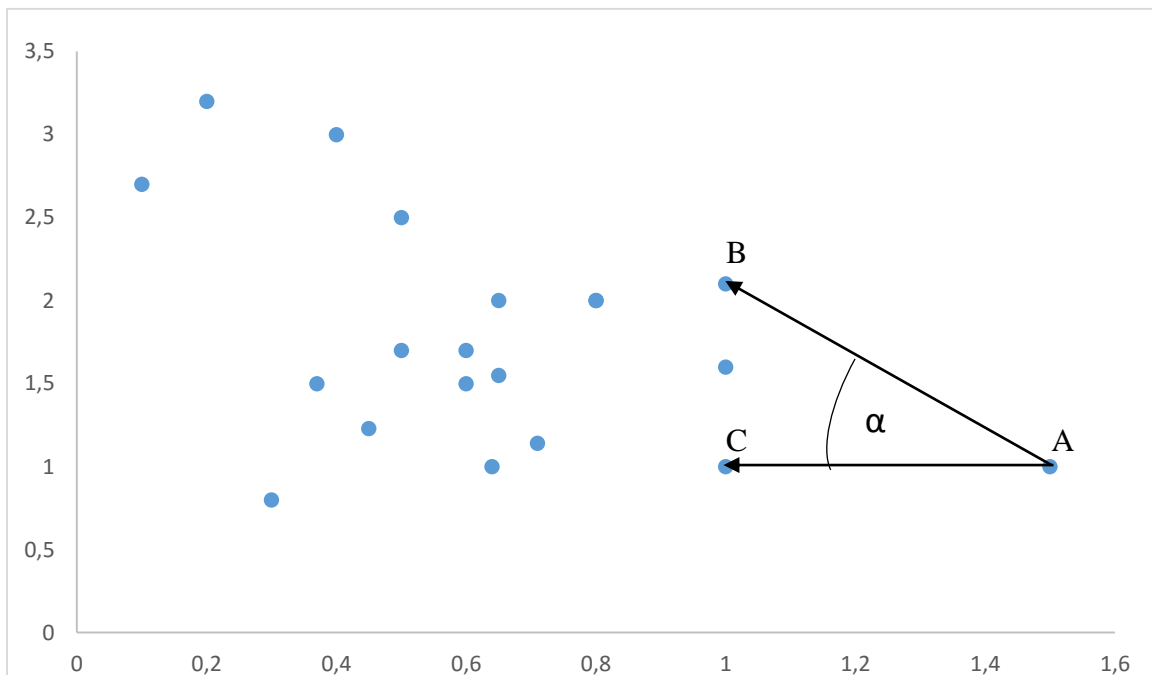


Figura 10. Ângulo α formado entre o vetor de distância \overline{AB} e \overline{AC} .

O pressuposto base do algoritmo é que um ponto A é considerado um *outlier* se a variância dos ângulos entre todos os vetores de distância formados por A for mínima. No contexto do algoritmo, a variância entre os ângulos é denominada por *Angle-Based Outlier Factor* (ABOF). Portanto quanto menor for o ABOF de um ponto, maior é a probabilidade de ele ser um *outlier*. Os próximos passos descrevem o funcionamento do algoritmo.

- 1) Para cada ponto do conjunto de dados o algoritmo calcula o ABOF através da seguinte fórmula:

$$VAR_{B,C} \frac{\langle \overline{AB}, \overline{AC} \rangle}{\|\overline{AB}\|^2 \cdot \|\overline{AC}\|^2}$$

- 2) Consideram-se os pontos com maior grau de *outlier* aqueles que tem menor ABOF.

É importante referir que para um determinado ponto do conjunto de dados, o algoritmo calcula os todos os vetores de distância que esse ponto pode formar com os restantes pontos. A figura 11 mostra um exemplo de como determinar o ABOF para o ponto A e B.

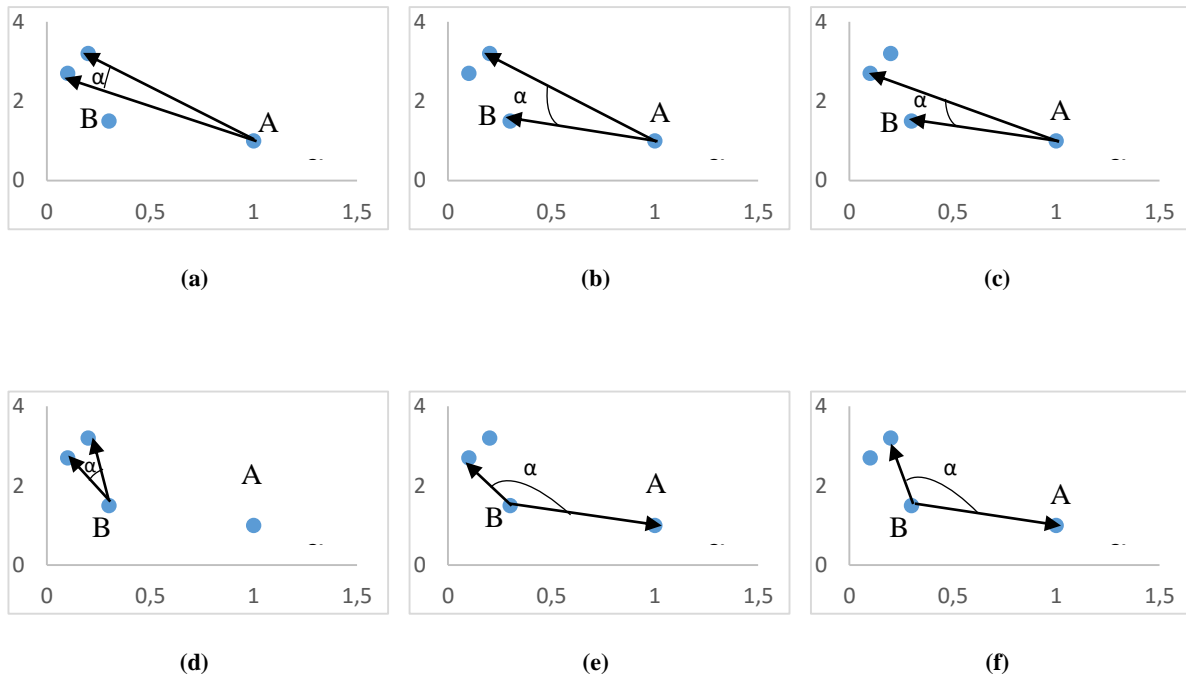


Figura 11. (a) (b) (c) Vetores de distância formados pelo ponto A e respetivo ângulo entre eles; (d) (e) (f) Vetores de distância formados pelo ponto B e respetivo ângulo entre eles.

Pode-se verificar através da figura 11 (a) (b) (c) que a variância dos ângulos entre os vetores de distância formados pelo ponto A é inferior à variância para o ponto B, o que significa que A tem maior probabilidade de ser um *outlier*.

Este algoritmo foi desenhado para obter bons resultados em dados multidimensionais, mais precisamente com duas e três dimensões. Porém, uma vez que para cada ponto é calculado o ângulo formado por ele e os seus vizinhos, no caso do conjunto de dados conter muitas amostras é fácil de perceber que este processo pode ter um impacto negativo no desempenho do algoritmo, portanto deve ser evitada a sua utilização nestas condições.

3.1.7 Clustering

Clustering é uma técnica amplamente utilizada para classificar um conjunto de pontos e é aplicada em análise de imagens, reconhecimento de padrões, entre outras áreas [20]. A ideia desta técnica é formar grupos de pontos que sejam semelhantes entre si. A próxima secção descreve o *K-Means*, um algoritmo de *clustering*.

3.1.7.1 K-Means

Este algoritmo, à semelhança de outros algoritmos já descritos, recebe um parâmetro do utilizador que indica o número de *clusters* que irão ser formados. Apesar de ser maioritariamente utilizado na área de *data mining* [21] pode ser também uma opção quando se trata de determinar desvios anormais no conjunto de dados.

O pressuposto base deste algoritmo passa por calcular e atualizar os K centroides, que são os pontos que estão no centro do *cluster*, conforme a sua semelhança comparativamente aos restantes pontos do *cluster*. Eis o funcionamento do algoritmo:

- 1) Escolhe K centroides c_j ao acaso.
- 2) Para cada ponto x_i :
 - a. Encontra o centroide c_j que esteja mais perto.
 - b. Atribui o ponto x_i ao cluster c que tem centroide c_j
 - c. Para cada *cluster* c :
 - i. Recalcula o centroide através da média dos pontos do *cluster*
 - ii. Se o centroide não alterar, o algoritmo para. Caso contrário, volta para 2)

O desempenho do algoritmo depende do número de *clusters* que irão ser formados, bem como os centroides que inicialmente são escolhidos ao acaso. O algoritmo foi desenvolvido para classificar os pontos de um conjunto de dados, agrupando pontos semelhantes no mesmo *cluster*. Através destas características é então possível utilizar o algoritmo para detetar *outliers*, que neste caso são todos os pontos que não se assemelham com a restante maioria. Porém, dado que o número de *clusters* é um parâmetro inserido pelo utilizador e que tem tanto impacto no algoritmo, este pode não obter resultados tão fiáveis comparativamente com os restantes algoritmos apresentados.

3.2 Classificação dos *outliers*

A classificação dos *outliers* é o processo que acontece depois da deteção e serve para determinar quais os pontos que são mais desviantes. Ao longo dos últimos anos, os investigadores [22] identificaram duas abordagens principais para classificar os *outliers*. As próximas secções descrevem essas abordagens.

3.2.1 Classificação decimal

Alguns algoritmos de deteção de *outliers*, principalmente algoritmos de *data mining*, classificam os pontos através de números decimais. Esta abordagem baseia-se em atribuir um *ranking* a um *outlier*. Os pontos que têm maior classificação são os que têm maior probabilidade de ser mais desviantes. Este método é bastante útil quando se pretende determinar os *top-n outliers* num conjunto de dados.

3.2.2 Classificação binária

Os algoritmos estatísticos normalmente classificam os pontos de uma forma binária. Esta abordagem diverge da anterior na medida em que o objetivo é determinar os pontos que são *outliers* e os que não são. A classificação binária baseia-se em classificar um ponto com “1” ou “0” que significa que é um *outlier* ou não, respetivamente. Ao contrário da classificação decimal, não existe qualquer prioridade entre os vários *outliers*, o que torna difícil o processo de determinar os *top-n outliers*.

3.3 Comparação entre métodos

A tabela 4 compara algumas características entre cada método de deteção de *outliers* descrito no presente documento. Para cada método é descrito o tipo de classificação que é

aplicado aos *outliers*, se necessita de parâmetros por parte do utilizador, se assume que os dados seguem alguma distribuição e se o método é eficiente.

Tabela 4. Comparação entre os vários métodos de deteção de *outliers*.

	Estatístico	Profundidade	Desvio	Distância	Densidade	Ângulo	Clustering
Classificação	Binária	Decimal	Decimal	Decimal	Decimal	Decimal	Decimal
Parâmetros	×	×	×	✓	✓	×	✓
Distribuição	✓	×	×	×	×	×	×
Eficiência	Eficiente em dados unidimensionais	Eficiente em dados bidimensionais	NP-hard	Complexidade exponencial	Eficiente para dados bidimensionais e tridimensionais	Eficiente para dados com mais de 3 dimensões	Mais eficiente a encontrar <i>clusters</i> que <i>outliers</i>

Através da tabela conseguimos perceber algumas diferenças entre os métodos da literatura. Existem métodos que foram desenvolvidos especificamente para detetar *outliers* em dados unidimensionais e outros em dados multidimensionais. Existem também métodos que não foram desenvolvidos com o propósito de detetar *outliers* pelo que a sua eficiência pode ser menor, quando comparada com a eficiência de outros métodos.

No entanto, uma característica comum a todos os métodos é que o conjunto de dados que se pretende analisar tem de conter uma grande quantidade de observações. Se porventura algum dos algoritmos descritos for aplicado a um conjunto de dados com apenas duas observações, nenhum *outlier* será detetado.

3.4 Metodologia

No processo de deteção de *outliers* não existe uma metodologia que deva ser seguida para seleccionar o melhor algoritmo a utilizar na deteção de *outliers* [21]. O processo de seleção do algoritmo depende sim do tipo de dados que se pretende analisar. Existem portanto alguns aspetos que se devem ter em conta durante a seleção do algoritmo a utilizar, como por exemplo:

- Remover ou preservar os *outliers*
- Ordenar os *outliers* ou aplicar-lhes uma classificação binária
- Tamanho do conjunto de dados

- Quantidade de dimensões a analisar
- Tipo dos dados (e.g. texto, números)

Um dos objetivos do estágio passa por detetar *outliers* num conjunto com uma grande quantidade de amostras. É importante referir que normalmente um *outlier* é sinónimo de um ponto que deve ser descartado do conjunto de dados. Porém no âmbito deste estágio, pretende-se detetar e preservar os *outliers*, a fim de interpretar o seu significado uma vez que estes podem contribuir com informação importante sobre o negócio.

Durante a primeira *drop* o algoritmo utilizado para a deteção de *outliers* foi o teste de quartis, por se tratar de um algoritmo eficaz para conjuntos de dados com muitas amostras e ser um algoritmo fácil de implementar. Porém, durante a segunda *drop* foi feita uma comparação de resultados entre este algoritmo e o MAD. Após a análise desses mesmos resultados, que estão descritos na íntegra no capítulo 6, decidiu-se utilizar o algoritmo MAD.

Uma vez que não se espera detetar *outliers* em conjuntos de dados com duas métricas combinadas, os algoritmos estatísticos revelam ser a melhor escolha uma vez que são mais eficientes. Apesar do grande leque de escolha de algoritmos estatísticos, a opção recaiu sobre um que não necessitasse de receber parâmetros por parte do utilizador e que não fizesse uso de tabelas auxiliares, de forma a evitar maior *overhead* do nosso motor.

4 Análise de Requisitos

Como foi referido anteriormente, o objetivo do estágio passa por desenvolver o *Findings Engine*, um componente capaz de analisar conjuntos de dados e traduzi-los em formato textual de forma a facilitar o processo de análise de informação. Esta solução será incluída num sistema de suporte à decisão direcionado para as empresas de telecomunicações. A figura 1 demonstra onde se integra o *Findings Engine* dentro do ecossistema de BI.

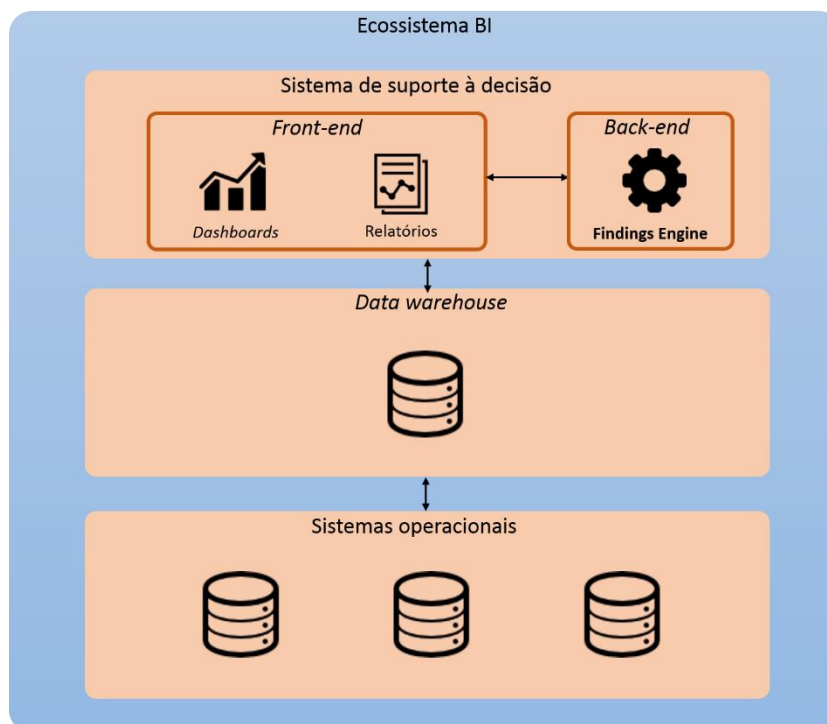


Figura 12. Ecossistema BI contendo o componente *Findings Engine*.

É perceptível através da figura 12 que o objetivo do *Findings Engine*, que funciona no *back-end* da aplicação, é processar conjuntos de dados e produzir os *Findings* que irão ser reportados através do *front-end*. Desta forma, o motor que irá ser desenvolvido funciona como uma ponte entre as fontes de dados e o componente que trata da visualização. O sistema operacional é responsável por invocar o *Findings Engine* que irá analisar a informação e retornar os *Findings* produzidos, para que sejam reportados ao utilizador final através do *front-end* da aplicação.

Este capítulo é dedicado ao levantamento e descrição dos requisitos do *Findings Engine*. Na primeira secção estão descritos os requisitos funcionais do sistema. Na segunda secção é feita a descrição dos atributos de qualidade. Finalmente, todas as restrições impostas são apresentadas na quarta secção.

4.1 Requisitos funcionais

Nesta secção serão descritos os requisitos funcionais do sistema. Para tal, serão apresentados os casos de uso que descrevem o comportamento da aplicação, bem como as interações entre os utilizadores e a mesma. Uma vez que objetivo do estágio não é desenvolver um *software* como um todo mas sim um pacote que será integrado numa aplicação existente, foi desenhado um diagrama para o sistema completo mas apenas serão desenvolvidas as funcionalidades que integram o *Findings Engine*. Na figura 13 está representado o diagrama de casos de uso.

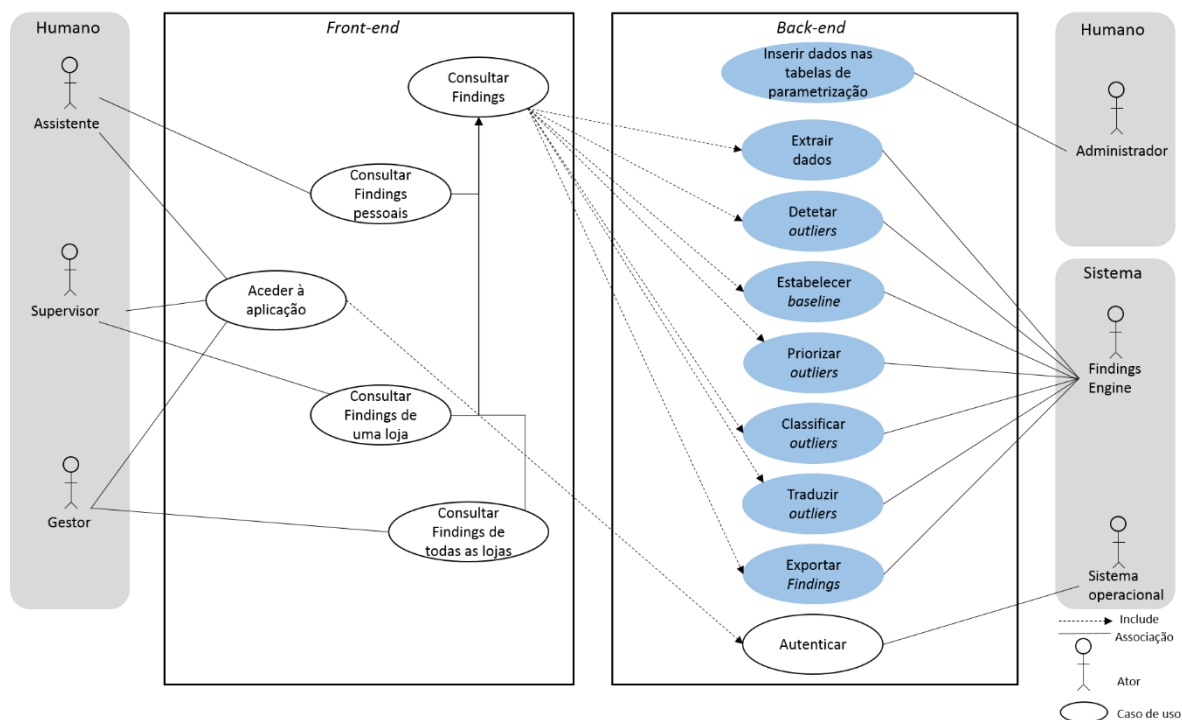


Figura 13. Diagrama de casos de uso identificados para o *Findings Engine*. Apenas serão desenvolvidas as funcionalidades assinaladas a azul.

O sistema é composto pelo *Findings Engine* que faz o processo de produção de *Findings* e pelo próprio sistema operacional nativo que é responsável por todas as funcionalidades inerentes às aplicações de suporte à decisão, nomeadamente a autenticação de utilizadores.

Com base nas necessidades que o negócio das telecomunicações apresenta foi identificado um administrador e três utilizadores com diferentes tipos de permissões a quem os

Findings interessam ser apresentados. As próximas tabelas descrevem todos os atores do sistema.

Tabela 5. Descrição do ator do sistema – Assistente.

ID	A1
Nome	Assistente
Tipo	Humano
Descrição	Colaboradores da empresa de telecomunicações que trabalham no terreno, ou seja, o pessoal operacional incluindo os empregados que estão ao balcão a vender produtos.
Permissões	Os assistentes podem consultar <i>Findings</i> relativamente ao seu desempenho pessoal, comparativamente com os seus colegas de equipa ou com o seu próprio historial.

Tabela 6. Descrição do ator do sistema – Supervisor.

ID	A2
Nome	Supervisor
Tipo	Humano
Descrição	Colaboradores da empresa de telecomunicações responsáveis pela monitorização de uma ou mais lojas.
Permissões	Os supervisores podem consultar <i>Findings</i> relativamente às suas lojas.

Tabela 7. Descrição do ator do sistema – Gestor.

ID	A3
Nome	Gestor
Tipo	Humano
Descrição	Colaborador da empresa de telecomunicações responsável por gerir todas as lojas.
Permissões	Os gestores podem consultar <i>Findings</i> relativamente a todas as lojas.

Tabela 8. Descrição do ator do sistema – Administrador.

ID	A4
Nome	Administrador
Tipo	Humano
Descrição	Utilizador que administra o sistema. É o responsável por inserir os dados nas tabelas parametrizáveis que vão dar origem aos <i>Findings</i> . A qualidade de cada <i>Finding</i> produzido depende diretamente deste ator.
Permissões	Os administradores podem inserir os dados nas tabelas parametrizáveis que vão ser utilizadas para produzir os <i>Findings</i> .
Casos de uso	UC1

Tabela 9. Descrição do ator do sistema - *Findings Engine*.

ID	A5
Nome	<i>Findings Engine</i>
Tipo	Sistema
Descrição	Componente responsável por produzir os <i>Findings</i> e exportá-los para o sistema operacional.
Casos de uso	UC2, UC3, UC4, UC5, UC6, UC7, UC8

Tabela 10. Descrição do ator do sistema - Sistema Operacional.

ID	A6
Nome	Sistema operacional
Tipo	Sistema
Descrição	Componente responsável pela lógica de negócio e pelas funcionalidades que tipicamente compõem um sistema de suporte à decisão.

O ator representado na última tabela – sistema operacional – é o sistema em que o *Findings Engine* será integrado, pelo que já se encontra desenvolvido. Este ator e o caso de uso relativamente à autenticação só estão presentes no diagrama de forma a contextualizar que o sistema operacional continua a ser responsável por toda a lógica de negócio. O próprio trata de mostrar os *Findings* produzidos pelo *Findings Engine*.

Assim que um utilizador deseja consultar um determinado *Finding* são desencadeados um conjunto de processos no *back-end* da aplicação. Esses processos, assinalados a azul no

diagrama da figura 11, serão o foco de desenvolvimento durante o estágio. As próximas tabelas descrevem na íntegra cada caso de uso.

Tabela 11. Cenário do caso de uso - UC1.

Título	<u>Inserir dados nas tabelas de parametrização</u>
ID	UC1
Descrição	<p>Processo de inserção dos dados na tabela de combinações e na tabela de configurações. A tabela de combinações contém, entre outros, a combinação de atributos que dão origem a um <i>Finding</i>.</p> <p>Exemplo de duas combinações diferentes:</p> <ul style="list-style-type: none"> • Número de vendas feitas em cada loja • Número de vendas feitas por cada colaborador <p>A tabela de configurações contém o nome das tabelas que irão ser úteis no processo de criação de um <i>Finding</i>.</p>
Atores envolvidos	A4
Pré-condições	<ul style="list-style-type: none"> • Ligação à base de dados
Fluxo principal	<p>Tabela <i>combination</i>:</p> <ol style="list-style-type: none"> 1. Inserir atributos que dão origem a um <i>Finding</i> 2. Inserir atributos que servem de <i>baseline</i> ao <i>Finding</i> 3. Inserir a métrica que irá ser analisada 4. Inserir a métrica que serve de <i>baseline</i> ao <i>Finding</i> 5. Inserir o tipo de classificação esperada 6. Inserir a frase em que o <i>Finding</i> é traduzido 7. Inserir a importância do <i>Finding</i> 8. Inserir a urgência do <i>Finding</i> <p>Tabela <i>configuration</i>:</p> <ol style="list-style-type: none"> 1. Inserir nome da tabela de dados operacionais 2. Inserir nome da tabela de combinações 3. Inserir nome da tabela onde são inseridos os <i>Findings</i>
Pós-condições	Tabela de combinações populada com dados
Exceções	Nenhuma
Dependências	Nenhuma

Tabela 12. Cenário do caso de uso - UC2.

Título	<u>Extrair dados</u>
ID	UC2
Descrição	Processo de extração dos dados da fonte, baseado nos atributos da tabela de combinações. Para cada entrada na tabela de combinações é gerada uma <i>query</i> que extrai os dados da fonte consoante a combinação de atributos estipulada pelo administrador.
Atores envolvidos	A5
Pré-condições	<ul style="list-style-type: none"> • Ligação à base de dados • Tabela de combinações populada com dados
Fluxo principal	<ol style="list-style-type: none"> 1. Sistema conecta-se à base de dados 2. Lê os dados da tabela de combinações 3. Extrai os dados da fonte com base na combinação de atributos
Pós-condições	O conjunto de dados que irá dar origem a um <i>Finding</i> está agora pronto para ser analisado.
Exceções	<p>Sistema não tem ligação à base de dados:</p> <ol style="list-style-type: none"> 1. Utilizador é reportado que não há conexão à BD <p>Tabela de combinações não está populada com dados:</p> <ol style="list-style-type: none"> 1. Não é produzido qualquer <i>Finding</i>
Dependências	UC1
Prioridade	<i>MUST</i>

Tabela 13. Cenário do caso de uso – UC3.

Título	<u>Detetar outliers</u>
ID	UC3
Descrição	Processo de deteção de <i>outliers</i> baseado num algoritmo estatístico da literatura
Atores envolvidos	A5
Pré-condições	<ul style="list-style-type: none"> • Conjunto de dados já extraído e pronto a ser analisado
Fluxo principal	<ol style="list-style-type: none"> 1. Recebe os dados que foram extraídos da BD

	2. Aplica um algoritmo de detecção de <i>outliers</i> aos dados extraídos
Pós-condições	O resultado deste processo é um ou mais dados que se desviam do comportamento esperado.
Exceções	<p>Não são detetados <i>outliers</i> no conjunto de dados:</p> <ol style="list-style-type: none"> 1. Não é produzido qualquer <i>Finding</i> <p>O conjunto de dados extraído é composto por menos que duas observações:</p> <ol style="list-style-type: none"> 1. Não é produzido qualquer <i>Finding</i>
Dependências	UC1, UC2
Prioridade	<i>MUST</i>

Tabela 14. Cenário do caso de uso - UC4.

Título	<u>Estabelecer <i>baseline</i></u>
ID	UC4
Descrição	<p>Processo que determina a <i>baseline</i> do conjunto de dados. Exemplos de <i>baseline</i>:</p> <ul style="list-style-type: none"> • Média • Mediana • Moda • Limites máximos/mínimos do MAD
Atores envolvidos	A5
Pré-condições	<ul style="list-style-type: none"> • <i>Outliers</i> detetados
Fluxo principal	<ol style="list-style-type: none"> 1. Recebe os dados que foram extraídos da BD 2. Calcula a <i>baseline</i> do conjunto de dados
Pós-condições	O resultado deste processo é o valor da <i>baseline</i>
Exceções	Nenhuma
Dependências	UC1, UC2
Prioridade	<i>MUST</i>

Tabela 15. Cenário do caso de uso - UC5.

Título	<u>Priorizar outliers</u>
ID	UC5
Descrição	Processo de priorização dos <i>outliers</i> do mais prioritário para o menos prioritário.
Atores envolvidos	A5
Pré-condições	<ul style="list-style-type: none"> • <i>Outliers</i> detetados • <i>Baseline</i> calculada
Fluxo principal	<ol style="list-style-type: none"> 1. Recebe conjunto de <i>outliers</i> 2. Prioriza os <i>outliers</i> com base na sua importância e urgência
Pós-condições	O resultado deste processo é um conjunto de <i>outliers</i> com uma percentagem de prioridade. Os <i>Findings</i> mais prioritários tem maior percentagem de prioridade que os restantes.
Exceções	Nenhuma
Dependências	UC1, UC2, UC3, UC4
Prioridade	<i>MUST</i>

Tabela 16. Cenário do caso de uso - UC6.

Título	<u>Classificar outliers</u>
ID	UC6
Descrição	<p>Processo de classificação de um <i>outlier</i> como “positivo” ou “negativo”. Depois de priorizados os <i>outliers</i>, interessa agora perceber o significado do <i>outlier</i> mais relevante. Um <i>outlier</i> que esteja acima da <i>baseline</i> não significa necessariamente que seja positivo ou negativo. Diferentes métricas levam a diferentes classificações. Por exemplo:</p> <ul style="list-style-type: none"> • Colaborador demora <u>mais</u> tempo do que o esperado a fazer uma determinada venda. O <i>outlier</i> está <u>acima</u> da <i>baseline</i> e é classificado <u>negativamente</u>, dado que se demorou mais que o esperado. • Colaborador faz <u>mais</u> vendas do que era esperado. O <i>outlier</i> está <u>acima</u> da <i>baseline</i> e é classificado <u>positivamente</u>, dado que foram vendidos mais produtos.
Atores envolvidos	A5

Pré-condições	<ul style="list-style-type: none"> • Colunas referentes à classificação populadas com dados
Fluxo principal	<ol style="list-style-type: none"> 1. Acede à tabela de combinações para ir buscar o tipo de classificação esperada 2. Verifica se o <i>outlier</i> está acima ou abaixo da <i>baseline</i> 3. Classifica o <i>outlier</i> dependendo da sua posição em relação à <i>baseline</i> e da classificação esperada inserida na tabela de combinações.
Pós-condições	O resultado deste processo é o <i>outlier</i> classificado como “positivo” ou “negativo”
Exceções	Nenhuma
Dependências	UC1, UC2, UC3, UC4, UC5
Prioridade	<i>MUST</i>

Tabela 17. Cenário do caso de uso - UC7.

Título	<u>Traduzir outliers</u>
ID	UC7
Descrição	<p>Processo de tradução dos <i>outliers</i> num <i>Finding</i>, ou seja, em formato textual. Exemplos de <i>Findings</i> possíveis:</p> <ul style="list-style-type: none"> • “Hoje estás a demorar mais tempo a registar um processo de venda, comparativamente com os teus colegas de equipa” • “Na 3ª semana de Janeiro fizeste 140 vendas diárias, +100% que os restantes colegas da tua equipa” • “O agente John é o melhor vendedor da loja” • “A loja <i>Alpha</i> fez menos vendas esta semana, comparativamente com as lojas da mesma região”
Atores envolvidos	A5
Pré-condições	<ul style="list-style-type: none"> • Coluna referente à tradução populada com dados
Fluxo principal	<ol style="list-style-type: none"> 1. Acede à tabela de combinações e guarda a frase em que o <i>Finding</i> será traduzido 2. Preenche os espaços vazios da frase com as variáveis estipuladas pelo administrador

Pós-condições	O resultado deste processo é um <i>Finding</i> , ou seja, uma frase que está pronta a ser mostrada ao utilizador
Exceções	Tabela de traduções não está populada com dados: 1. Não é produzido qualquer <i>Finding</i>
Dependências	UC1, UC2, UC3, UC4, UC5, UC6
Prioridade	<i>MUST</i>

Tabela 18. Cenário do caso de uso - UC8.

Título	<u>Exportar <i>Findings</i></u>
ID	UC8
Descrição	<p>Processo de inserção dos <i>Findings</i> na base de dados e da apresentação dos mesmos através de um <i>web service</i>. A estrutura de um <i>Finding</i> é a seguinte:</p> <ul style="list-style-type: none"> • Data de produção do <i>Finding</i> • <i>Finding</i> - frase textual • Classificação do <i>Finding</i> - positivo vs negativo • Permissões do <i>Finding</i> - que utilizadores têm acesso a ele • Desvio absoluto - desvio entre o <i>outlier</i> que deu origem ao <i>Finding</i> e o valor esperado • Desvio percentual - percentagem do desvio absoluto • <i>Baseline</i> • Prioridade do <i>Finding</i> - há <i>Findings</i> que têm maior prioridade em ser mostrados • Urgência • Importância
Atores envolvidos	A5
Pré-condições	<ul style="list-style-type: none"> • <i>Finding</i> criado • Ligação à BD
Fluxo principal	<ol style="list-style-type: none"> 1. Conectar à base de dados 2. Inserir os <i>Findings</i> na tabela de <i>output</i> 3. Disponibilizar os <i>Findings</i> através de um <i>web service</i>
Pós-condições	O <i>Finding</i> é inserido na BD
Exceções	Sistema não tem ligação à BD:

	1. O <i>Finding</i> é escrito num ficheiro de log, que será carregado na base de dados mais tarde
Dependências	UC1, UC2, UC3, UC4, UC5, UC6, UC7
Prioridade	<i>MUST</i>

É de notar que durante os cenários de caso de uso é feita referência às tabelas de combinações, traduções e *output*, estando estas descritas na íntegra no capítulo 6. Os restantes detalhes técnicos, nomeadamente o que é a *baseline* e a importância que tem para o nosso motor, estão também descritos nos próximos capítulos.

4.2 Atributos de qualidade

O *Findings Engine* será, numa primeira fase, integrado numa aplicação de suporte à decisão direccionada a uma empresa de telecomunicações. Porém, é expectável que o componente possa vir mais tarde a ser integrado noutras aplicações do mesmo âmbito. O *Findings Engine* será desenvolvido para uma empresa de telecomunicações em específico mas de tal forma parametrizável que, possa vir a ser integrado nos SSD de outras empresas.

Uma vez que no mercado das telecomunicações as empresas têm diferentes necessidades de negócio, é necessário desenvolver um componente de tal forma parametrizável que permita ser integrado em qualquer sistema de suporte à decisão. Assim é garantido que cada empresa define as suas necessidades, ou por outras palavras, cada empresa é responsável por definir o tipo de *Findings* que serão produzidos e reportados aos seus colaboradores. Existe portanto um conjunto de requisitos que devem ser garantidos de forma a assegurar a qualidade do sistema.

As próximas tabelas descrevem os atributos de qualidade que o *Findings Engine* deve garantir.

Tabela 19. Cenário do atributo de qualidade - QA1.

ID	QA1
Atributo	Interoperabilidade
Descrição	O <i>Findings Engine</i> deve ter a capacidade comunicar com qualquer tipo de sistema, através de um formato <i>standard</i> , garantindo assim que possa vir a ser integrado com qualquer SSD
Fonte de estímulo	Administrador
Estímulo	Integrar o <i>Findings Engine</i> com outros SSD
Ambiente	Fase de integração do componente
Artefacto	<i>Findings Engine</i>
Resposta	O <i>Findings Engine</i> é facilmente integrado com qualquer outra aplicação, através da disponibilização dos <i>Findings</i> num formato <i>standard</i> .

Medir a resposta	O componente pode ser integrado em qualquer aplicação de suporte à decisão, na indústria das telecomunicações.
-------------------------	--

Tabela 20. Cenário do atributo de qualidade - QA2.

ID	QA2
Atributo	Parametrização
Descrição	O <i>Findings Engine</i> deve ter a capacidade de ser parametrizável, de forma a responder às diferentes necessidades de negócio que cada empresa de telecomunicações apresenta
Fonte de estímulo	Administrador
Estímulo	Produzir novos <i>Findings</i> com diferentes métricas
Ambiente	Condições normais
Artefacto	<i>Findings Engine</i>
Resposta	O <i>Findings Engine</i> tem tabelas de parametrização que permitem ao administrador definir os atributos/métricas que dão origem a um <i>Finding</i>
Medir a resposta	Para diferentes combinações de atributos, o componente produz diferentes <i>Findings</i> .

Tabela 21. Cenário do atributo de qualidade - QA3.

ID	QA3
Atributo	Escalabilidade
Descrição	O <i>Findings Engine</i> deve ter a capacidade de manter o desempenho com o aumento da quantidade de dados que tem de processar
Fonte de estímulo	Base de dados
Estímulo	Aumento da quantidade de dados a ser analisada
Ambiente	Condições normais
Artefacto	<i>Findings Engine</i>
Resposta	O <i>Findings Engine</i> mantém o desempenho

Medir a resposta	O tempo que o <i>Findings Engine</i> demora a gerar um <i>Finding</i> é inferior a 1 minuto
-------------------------	---

4.2.1 Interoperabilidade

A interoperabilidade do componente revela ser importante, uma vez que no futuro se pretende integrá-lo com outras aplicações. Para tal, o *Findings Engine* será disponibilizado através de um *web service*, permitindo assim que qualquer aplicação invoque os seus métodos, independentemente da linguagem em que foi desenvolvida. O *web service* será baseado na arquitetura REST e para gerar um *Finding* o sistema operacional apenas tem de invocar um determinado método do *web service*. No capítulo 6 é demonstrado um exemplo de como invocar um método do *web service*.

Para além disso os *Findings* são disponibilizados em JSON, um formato de dados *standard* que permite a qualquer SSD trocar informação com o *Findings Engine*. Durante o capítulo 6 é demonstrado o formato de *output* dos *Findings*.

4.2.2 Parametrização

Uma vez que o *Findings Engine* é um componente que poderá ser integrado em qualquer sistema de suporte à decisão das empresas de telecomunicações, é importante garantir que o mesmo seja parametrizável, visto que nesta indústria as empresas não partilham necessariamente as mesmas necessidades de negócio.

Sendo o *Findings Engine* parametrizável, a própria empresa é a responsável por definir a combinação de atributos (e.g. número de vendas feitas num dia por cada assistente) que irá dar origem a um *Finding* (e.g. “Hoje o assistente João foi o que vendeu mais produtos”).

A parametrização pode ser garantida através de tabelas na base de dados, às quais só o administrador do sistema tem acesso, que permitem configurar os atributos que dão origem a um *Finding*, bem como a frase que será mostrada para cada tipo de *Finding*. Desta forma não é restringida qualquer necessidade de negócio que a empresa apresente, permitindo assim que a própria tenha total controlo sobre o *Findings Engine*.

4.2.3 Escalabilidade

Numa indústria como a das telecomunicações são registados milhares de dados diariamente. Uma empresa guarda a informação relativamente aos seus clientes, aos seus colaboradores, aos produtos que são vendidos diariamente nas suas lojas, ao tipo de produtos que são vendidos e até ao tempo que um assistente demora a registar um processo de venda de um determinado produto.

Dado este enorme volume de dados, e uma vez que o *Findings Engine* os tem de analisar, é fundamental assegurar a escalabilidade do componente. Durante o capítulo 7 serão feitos testes de carga com grandes quantidades de dados e será medido o tempo que o *Findings Engine* demora a produzir um *Finding*, de forma a verificar se a arquitetura do componente é ou não escalável.

4.3 Restrições

Esta secção é dedicada às restrições impostas que terão de ser adotadas durante o desenvolvimento do *Findings Engine*. A tabela 22 contém todas as restrições, neste caso impostas pelo cliente, que deverão ser obrigatoriamente garantidas na arquitetura do componente.

Tabela 22. Restrições do *Findings Engine*.

Restrição	Flexibilidade	Alternativa	Fonte
<i>Findings</i> têm de ter formato textual	×	×	Cliente
Um <i>Finding</i> só poderá ser produzido se existirem <i>outliers</i> no conjunto de dados	×	×	Cliente
O <i>Findings Engine</i> tem de poder ser integrado com qualquer sistema de suporte à decisão	×	×	Cliente

5 Arquitetura

Este capítulo contém a definição dos aspetos arquiteturais relativos ao componente a desenvolver durante o estágio. A arquitetura apresentada é resultante da análise feita durante as 3 *drops* e visa garantir todos os atributos enunciados no capítulo anterior.

A próxima lista contém uma descrição sobre as camadas que compõem o *Findings Engine* e que estão descritas na figura 14.

- *Findings Engine*: é a camada responsável pela lógica de negócio do componente que irá ser desenvolvido ao longo do estágio. Esta camada, que comunica diretamente com a base de dados, é a responsável por detetar os comportamentos anormais e produzir os respetivos *Findings* que serão mostrados ao utilizador final.
- Base de dados: é a camada que trata de guardar um histórico dos *Findings* produzidos, bem como armazenar informação útil que é utilizada no processo de criação de um *Finding*.

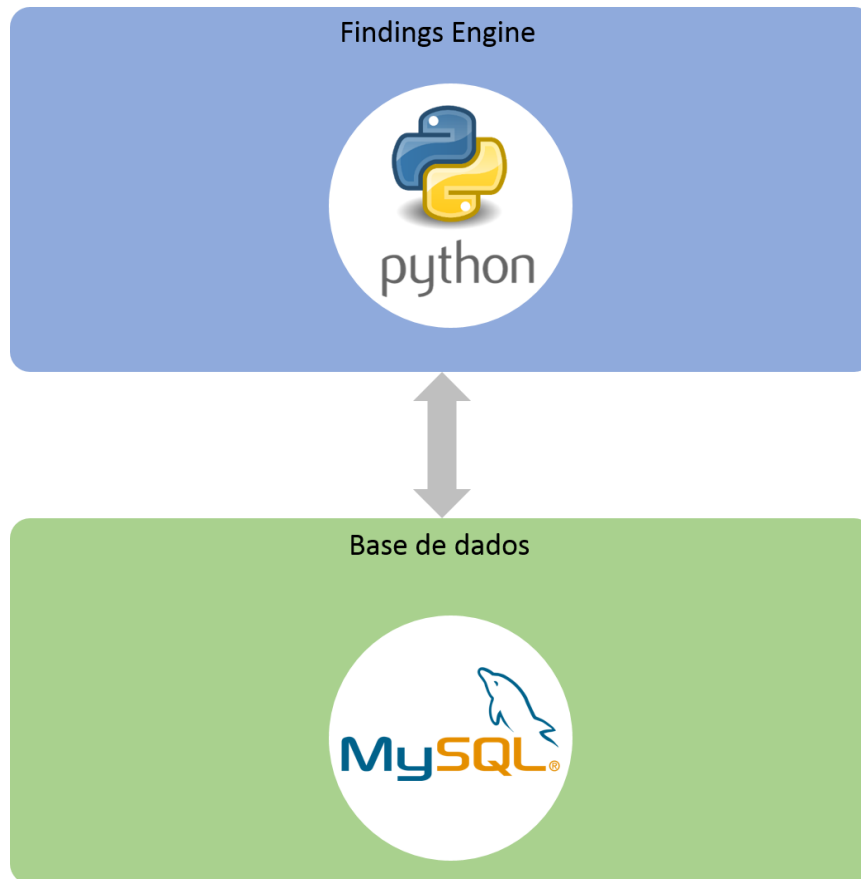


Figura 14. Diagrama de camadas que representa a arquitetura de alto-nível do *Findings Engine*.

Como é possível verificar através da figura 14 o componente desenvolvido ao longo do estágio segue uma arquitetura de 2 camadas. Esta decisão arquitetural prende-se com o facto do *Findings Engine* vir a ser integrado com um produto previamente desenvolvido, pelo que não terá de seguir um modelo rígido de MVC que é composto por três camadas: visualização, lógica de negócio e base de dados. Assim sendo, uma vez que o sistema operacional onde o *Findings Engine* será integrado é o responsável pela visualização, optou-se por um sistema com duas camadas.

5.1 Tecnologias

Numa fase inicial do projeto, a tecnologia utilizada na camada da base de dados foi o *MySQL*. Esta tecnologia tem a vantagem de ser *open-source* garantido assim redução de custos. Porém, procedeu-se a uma fase de análise para comparar a *performance* do *MySQL* com *Oracle* e percebeu-se que esta última tem melhores resultados. Para além disso, dentro da empresa existe maior familiaridade com esta tecnologia o que torna o processo de desenvolvimento mais fácil. A próxima figura ilustra o tempo que a tecnologia *MySQL* e *Oracle* demoram a extrair o mesmo tipo de dados, no mesmo cenário com as mesmas condições. Neste caso pretende-se extrair da base de dados um total de 34168 linhas e 5 colunas.

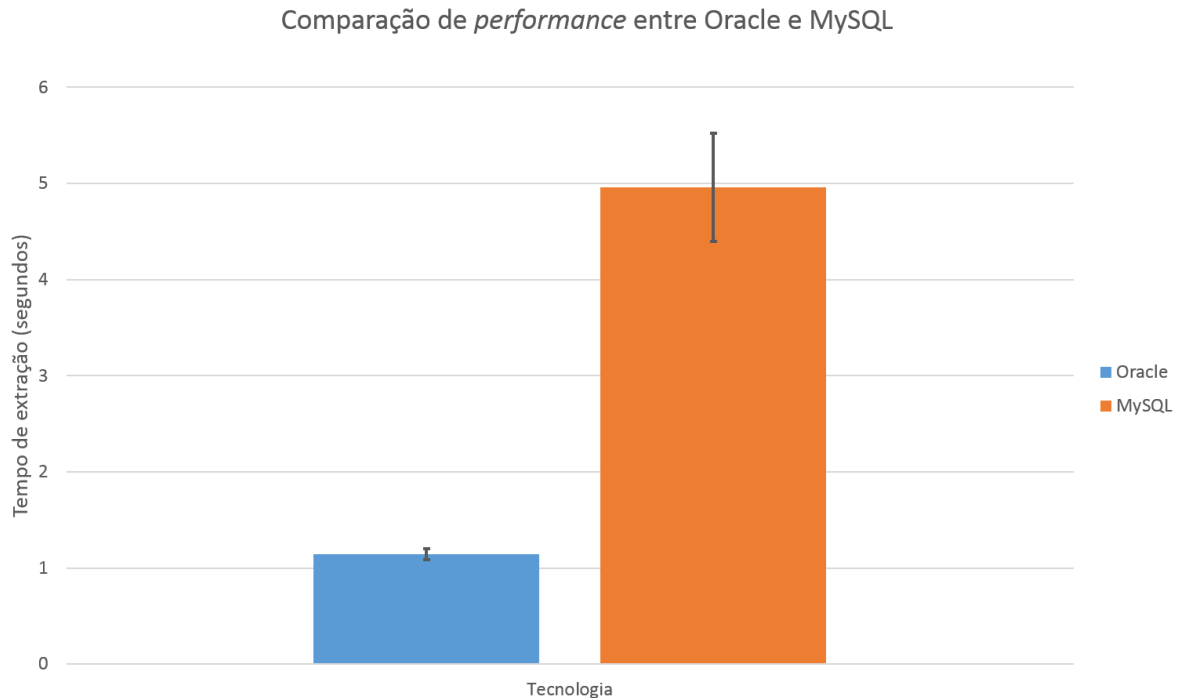


Figura 15. Comparação da *performance* entre *Oracle* e *MySQL*

Através da figura facilmente se conclui que, num cenário com as mesmas condições o *MySQL* tem menor *performance* que o *Oracle* no que toca à extração de dados.

Quanto à camada da lógica de negócio, a opção dividiu-se entre separá-la por completo da camada da base de dados e implementá-la em *Python* ou implementá-la em PL/SQL dentro da camada de base de dados. A vantagem de utilizar o *Python* é o facto de esta tecnologia ser amplamente utilizada na área de análise de dados e fornecer um conjunto de bibliotecas úteis no desenvolvimento do projeto. Porém, a escolha de uma tecnologia como o PL/SQL que permite o desenvolvimento de procedimentos dentro da própria base de dados, pode ser uma vantagem no que toca à *performance* do mecanismo, uma vez que não existe o *overhead* em estabelecer a ligação e extrair os dados da base de dados.

Após uma fase de análise deste *trade-off* concluiu-se que a utilização do *Python* seria o mais adequado ao projeto, uma vez que as bibliotecas fornecidas auxiliam tanto no processo de análise de dados como no processo de tornar o motor interoperável. A *performance* do mecanismo pode ser garantida através de outras técnicas, como será explicado no capítulo 7.

5.2 Base de dados

A camada da base de dados contém quatro tabelas que são geridas pelo administrador do sistema, sendo elas fundamentais para a qualidade do *Findings Engine*. Em seguida é apresentada a lista das tabelas que compõem a camada da base de dados:

- Tabela *combination*: guarda informação sobre as combinações de atributos que dão origem a um *Finding*.

- Tabela *output*: tabela onde são introduzidos todos os *Findings* produzidos pelo *Findings Engine*. No fundo contém um histórico de todos os *Findings* produzidos.
- Tabela *data*: contém os dados produzidos pelo sistema operacional que serão alvo de análise e dos quais irão ser produzidos os *Findings*. Uma vez que não faz parte do âmbito do projeto, é assumido que estes dados já se encontram previamente carregados nesta tabela e é excluída a necessidade de realizar um processo de ETL.
- Tabela *configuration*: guarda o nome das tabelas anteriores, garantindo a modificabilidade do motor.

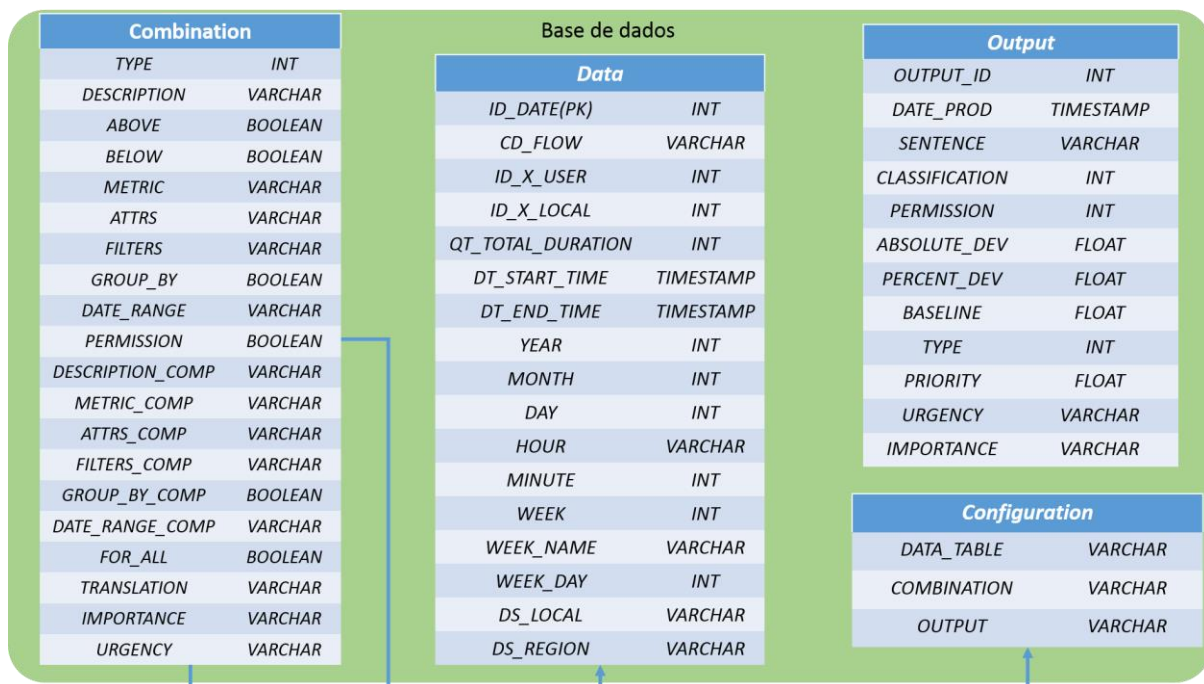


Figura 16. Diagrama físico da base de dados. (PK) – Primary Key; (FK) – Foreign Key;

As tabelas representadas no diagrama anterior são fundamentais para o bom funcionamento do componente. A tabela *data* contém os dados operacionais, ou seja, funciona como uma tabela de factos que contem toda a informação registada pelo sistema operacional que dará origem aos *Findings*.

Como foi referido anteriormente, o processo de carregamento desta tabela é da responsabilidade do cliente e não faz parte do âmbito do estágio, pelo que se assume que o processo de ETL já se encontra feito.

As restantes tabelas foram criadas para garantir que o *Findings Engine* é parametrizável, ou seja, que não exista um conjunto de *Findings* estáticos mas sim que cada empresa possa definir o tipo de *Findings* que deseja ver produzidos.

As próximas secções descrevem cada uma das tabelas na íntegra.

5.2.1 Data

A tabela *data* é uma tabela de factos e contem as vendas registadas pelo sistema operacional. O carregamento dos dados nesta tabela é da total responsabilidade do sistema operacional em que o *Findings Engine* está integrado.

Neste caso foi criada a tabela *data* com as colunas representadas na tabela 23, uma vez que estes foram os dados disponibilizados pelo cliente. Porém, é expectável que estas colunas sejam diferentes quando for necessário integrar o *Findings Engine* noutros SDD. A descrição dos dados fornecidos pelo cliente é feita mais pormenorizadamente na secção 6.1.

Tabela 23. Descrição das colunas da tabela *data*.

<i>Id_date</i>	ID da data em que foi registada a venda
<i>Cd_flow</i>	ID da venda
<i>Id_x_user</i>	ID do assistente que registou a venda
<i>Id_x_local</i>	ID da loja onde foi registada a venda
<i>Qt_total_duration</i>	Tempo total que a venda demorou a ser registada
<i>Dt_start_time</i>	Data e hora em que a venda começou a ser registada
<i>Dt_end_time</i>	Data e hora em que a venda terminou de ser registada
<i>Year</i>	Ano em que a venda começou a ser registada
<i>Month</i>	Mês em que a venda começou a ser registada
<i>Day</i>	Dia em que a venda começou a ser registada
<i>Hour</i>	Hora em que a venda começou a ser registada
<i>Minute</i>	Minuto em que a venda começou a ser registada
<i>Week</i>	Semana em que a venda começou a ser registada
<i>Week_name</i>	Nome da semana em que a venda começou a ser registada
<i>Week_day</i>	Dia da semana em que a venda começou a ser registada
<i>Ds_local</i>	Nome da loja onde a venda foi registada
<i>Ds_region</i>	Região da loja onde a venda foi registada

5.2.2 Combination

A tabela *combination* guarda informação útil para o processo de criação de um *Finding*. Esta tabela contém um conjunto de informações que irão ser fundamentais para a criação de duas *queries* – relativamente ao conjunto de dados de análise e de comparação.

Nesta tabela podem ser parametrizados um ou mais *Findings*, sendo que cada linha da tabela corresponde a um *Finding* diferente. Quando a aplicação invoca o motor todos os *Findings* parametrizados nesta tabela são executados. A tabela 24 descreve cada uma das colunas da tabela *combination*.

Tabela 24. Descrição das colunas da tabela *combination*.

Type	ID do <i>Finding</i>
Description	Descrição sobre os dados que irão servir de análise
Above	Colunas que guardam o tipo de classificação esperada para um <i>Finding</i> . <ul style="list-style-type: none"> • Se <i>Above</i> for 1 e <i>Below</i> 0, então significa que quando o <i>outlier</i> está acima da <i>baseline</i> será classificado como positivo. • Se <i>Above</i> for 0 e <i>Below</i> 1, então significa que quando o <i>outlier</i> está acima da <i>baseline</i> será classificado como negativo
Below	
Metric	Métrica que se pretende analisar. Esta métrica é o nome de uma das colunas da tabela <i>data</i> . Exemplo de métricas: <ul style="list-style-type: none"> • Número de vendas • Tempo demorado a registar uma venda
Attrs	É conjunto de atributos, da tabela <i>data</i> , que irão ser combinadas e dão origem a uma <i>query</i> . O resultado da <i>query</i> é o conjunto de dados que irá ser analisado e posteriormente dará origem a um <i>Finding</i> . Se a coluna estiver preenchida com “ <i>id_utilizador, id_loja</i> ”, a seguinte <i>query</i> é criada: <ul style="list-style-type: none"> • “<i>SELECT id_utilizador, id_loja</i>”
Filters	Conjunto de filtros que iram ser aplicados à <i>query</i> . Se a coluna tiver preenchida com “ <i>id_utilizador = 3, id_loja!= 2</i> ” é criado o seguinte filtro na <i>query</i> : <ul style="list-style-type: none"> • “<i>WHERE id_utilizador = 3 AND id_loja!= 2</i>”
Group_by	Coluna que define se a <i>query</i> criada terá informação agregada: <ul style="list-style-type: none"> • Se <i>Group_by</i> for 1, a <i>query</i> terá informação agregada • Se <i>Group_by</i> for 0, a <i>query</i> não terá informação agregada
Date_range	Coluna que define a <i>range</i> de dados que vão ser selecionados:

	<ul style="list-style-type: none"> • Vendas de um agente relativamente ao mês de Janeiro • Vendas de um agente feitas durante o período de pico (17h-21h) • Vendas de um agente feitas durante os fins de semana
Permission	<p>Coluna que define quem tem permissão de ver o <i>Finding</i>.</p> <ul style="list-style-type: none"> • Se o valor desta coluna for 1, só os assistentes podem ver o <i>Finding</i> • Se o valor desta coluna for 2, só os supervisores podem ver o <i>Finding</i> • Se o valor desta coluna for 3, só os gestores podem ver o <i>Finding</i>
Description_comp	Descrição sobre os dados que irão servir de comparação
Metric_comp	Métrica que serve de comparação. Esta métrica é o nome de uma das colunas da tabela <i>data</i> .
Attrs_comp	É conjunto de atributos, da tabela <i>data</i> , que irão ser combinadas e dão origem a uma <i>query</i> . O resultado da <i>query</i> é o conjunto de dados que irá servir de comparação aos dados de análise.
Filters_comp	Conjunto de filtros que iram ser aplicados à <i>query</i> comparação.
Group_by_comp	Coluna que define se a <i>query</i> de comparação terá informação agregada
Date_range_comp	Coluna que define a <i>range</i> de dados de comparação
For_all	<p>Esta coluna dita se o <i>Finding</i> é calculado para todos os assistentes/lojas ou se para um assistente em específico.</p> <ul style="list-style-type: none"> • Se a coluna for 1, o <i>Finding</i> é calculado para todos os assistentes/lojas • Se a coluna for 0, o <i>Finding</i> é calculado apenas para o assistente/loja indicado/a na coluna <i>Filters</i>
Translation	<p>Coluna que guarda a frase em que o <i>Finding</i> será traduzido. A frase contém variáveis que são substituídas por valores concretos.</p> <ul style="list-style-type: none"> • “Na semana <i>%week</i> o assistente <i>%id_assistente</i> fez <i>%mais/menos</i> vendas que os restantes colegas”
Importance	<p>Coluna que dita qual é a importância de um <i>Finding</i>. Um <i>Finding</i> pode ter importância:</p> <ul style="list-style-type: none"> • <i>High</i> • <i>Medium</i> • <i>Low</i>

<i>Urgency</i>	<p>Coluna que dita a urgência com que cada <i>outlier</i> deve ser tratado. Um <i>outlier</i> pode ter urgência:</p> <ul style="list-style-type: none"> • <i>High</i> • <i>Medium</i> • <i>Low</i>
-----------------------	---

As *drops* II e III trouxeram algumas alterações à tabela *combination*, que inicialmente limitava o tipo de *Findings* que podiam ser construídos. Atualmente é possível definir se um *Finding* é produzido para todos os assistentes/lojas ou se para um assistente/loja em específico. O próprio paradigma da detecção de comportamentos anormais sofreu alterações. Como se pode verificar através da tabela anterior, quando se define um novo *Finding* é necessário estabelecer dois conjuntos de dados diferentes: o de análise e o de comparação. Se o objetivo de um *Finding* for determinar os assistentes que fazem mais vendas comparativamente com os seus colegas da mesma loja, o conjunto de dados de análise são as vendas de cada assistente (individualmente) enquanto o conjunto de dados de comparação são as vendas feitas pelos colegas desse assistente. Este novo paradigma de detecção e todos os aspetos importantes desta tabela são explicados na íntegra no próximo capítulo.

5.2.3 Output

A tabela de *output* contém um histórico de todos os *Findings* produzidos ao longo do tempo.

Tabela 25. Descrição das colunas da tabela *output*.

<i>Output_id</i>	ID do <i>Finding</i>
<i>Date_prod</i>	Data em que o <i>Finding</i> foi produzido
<i>Sentence</i>	Frase textual – <i>Finding</i> .
<i>Classification</i>	Classificação do <i>Finding</i> : positivo ou negativo
<i>Permission</i>	Tipo de utilizador que tem acesso ao <i>Finding</i> .
<i>Absolute_dev</i>	Desvio absoluto entre o <i>outlier</i> e o valor esperado
<i>Percent_dev</i>	Desvio percentual entre o <i>outlier</i> e o valor esperado
<i>Baseline</i>	Valor esperado
<i>Type</i>	Tipo do <i>Finding</i>
<i>Priority</i>	<p>Prioridade de um <i>Finding</i></p> <ul style="list-style-type: none"> • 1 – <i>Finding</i> com prioridade mínima • 2 • 3 • 4 – <i>Finding</i> com prioridade máxima
<i>Urgency</i>	Urgência de cada <i>outlier</i>

	<ul style="list-style-type: none"> • <i>High</i> • <i>Medium</i> • <i>Low</i>
Importance	Importância do <i>Finding</i> <ul style="list-style-type: none"> • <i>High</i> • <i>Medium</i> • <i>Low</i>

5.2.4 Configuration

A tabela *configuration* contém os nomes das tabelas anteriores. A maior vantagem desta tabela é garantir a modificabilidade do motor. Assim sendo, se por ventura o nome das tabelas anteriores for alterado basta proceder também à alteração na tabela *configuration* para que o *Findings Engine* continue a funcionar corretamente.

Tabela 26. Descrição das colunas da tabela *translation*.

Data_table	Nome da tabela de factos que contem o registo das vendas
Combination	Nome da tabela onde o administrador parametriza os atributos que dão origem ao <i>Finding</i>
Output	Nome da tabela que contém o histórico dos <i>Findings</i>

5.3 Findings Engine

O *Findings Engine* é a camada responsável por realizar todos os passos necessários para a produção de um *Finding*. Com base na análise de requisitos funcionais e não-funcionais, foram então identificados quatro componentes fundamentais para o bom funcionamento deste processo:

- Componente de extração: componente é responsável por extrair os dados do sistema operacional de forma a iniciar o processo de produção de um *Finding*. Para cada *Finding* o componente extrai um conjunto de dados que será analisado comparativamente com um conjunto de comparação;
- Componente de deteção: componente que deteta se há ou não *outliers* no conjunto dos dados de análise previamente extraído, comparativamente com o conjunto de dados de comparação. Este componente é também responsável por priorizar os *outliers*;
- Componente de interpretação: trata de classificar um *outlier* como positivo ou negativo e transforma este desvio numa frase *template* previamente definida pelo administrador;
- Componente de *output*: componente que insere os *Findings* na base de dados e os disponibiliza para serem mostrados ao utilizador final;

A figura 14 demonstra os quatro componentes acima descritos, bem como todos os mecanismos que fazem parte de cada componente.

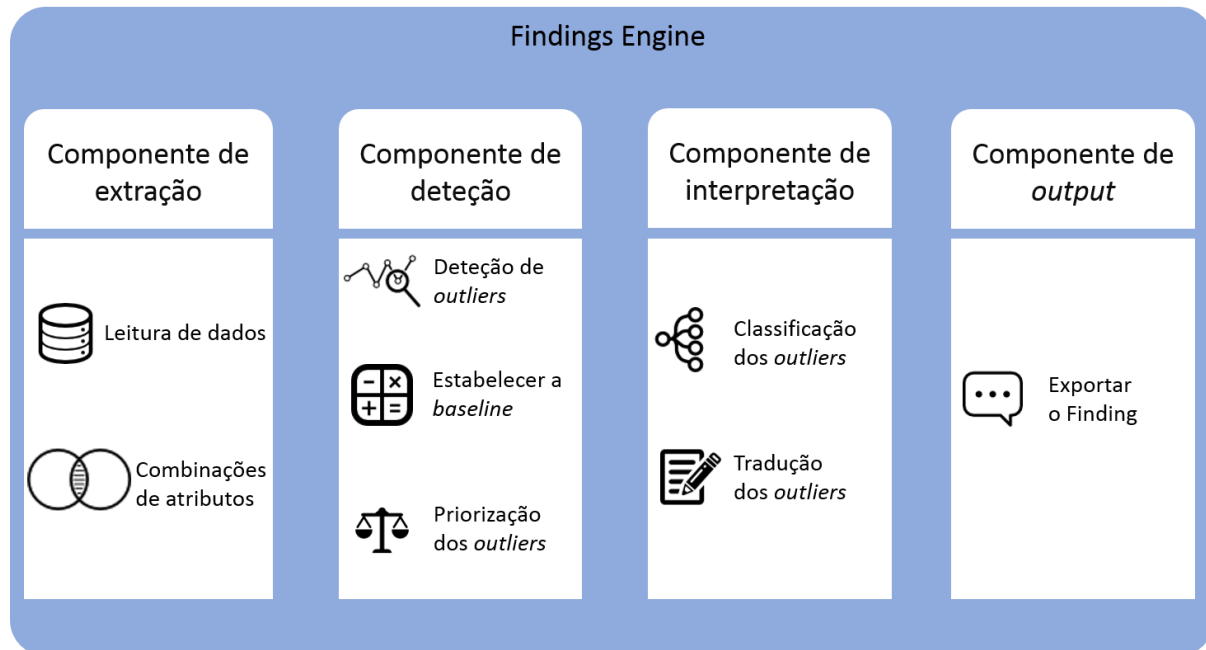


Figura 17. Componentes que integram a camada do *Findings Engine*.

O componente da extração é composto por um mecanismo que faz as combinações dos atributos inseridos na tabela *combination* e a leitura dos dados do sistema operacional com base nos dados inseridos nessa tabela. Este passo serve essencialmente para filtrar os dados que irão dar origem a um *Finding*.

Depois de filtrada, é preciso determinar qual é a informação relevante a ser mostrada ao utilizador final. Um dos aspetos que tem relevância reportar é qualquer comportamento desviante, ou seja, no caso de existir um padrão normal de acontecimentos é necessário reportar ao utilizador os dados que se desviem do esperado. Para tal, o componente da deteção trata de detetar *outliers* (comportamentos desviantes) no conjunto de dados. Por fim este componente prioriza esses comportamentos desviantes com base na urgência e na importância do *Finding*.

Depois de determinado o *outlier* mais prioritário é agora necessário interpretar o significado do mesmo. A classificação de um *outlier* não depende apenas se ele se encontra acima ou abaixo da *baseline* mas também do tipo de métrica que está a ser analisada. Por exemplo, se a métrica que queremos analisar é o número de vendas feitas por um colaborador e se existir um *outlier* acima da *baseline* então esse *outlier* é positivo porque significa que o colaborador está a vender mais do que é esperado, logo a empresa está a faturar mais. Pelo contrário, se a métrica for a quantidade de tempo que um colaborador demora a registar uma venda e se existir um *outlier* acima da *baseline* então esse *outlier* é negativo uma vez que o colaborador está a demorar mais tempo do que é esperado. No final do processo de classificação o componente da interpretação transforma os *outliers* em frases textuais – *Findings*.

Para mostrar os *Findings* ao utilizador, existe o componente de *output* que trata de os inserir numa base de dados e ainda os exportá-los para o *web service*.

A próxima secção descreve o fluxo dos dados na arquitetura do *Findings Engine*.

5.4 Fluxo dos dados

A figura 18 ilustra as interações entre os atores do sistema e o *Findings Engine*, bem como o fluxo de dados.

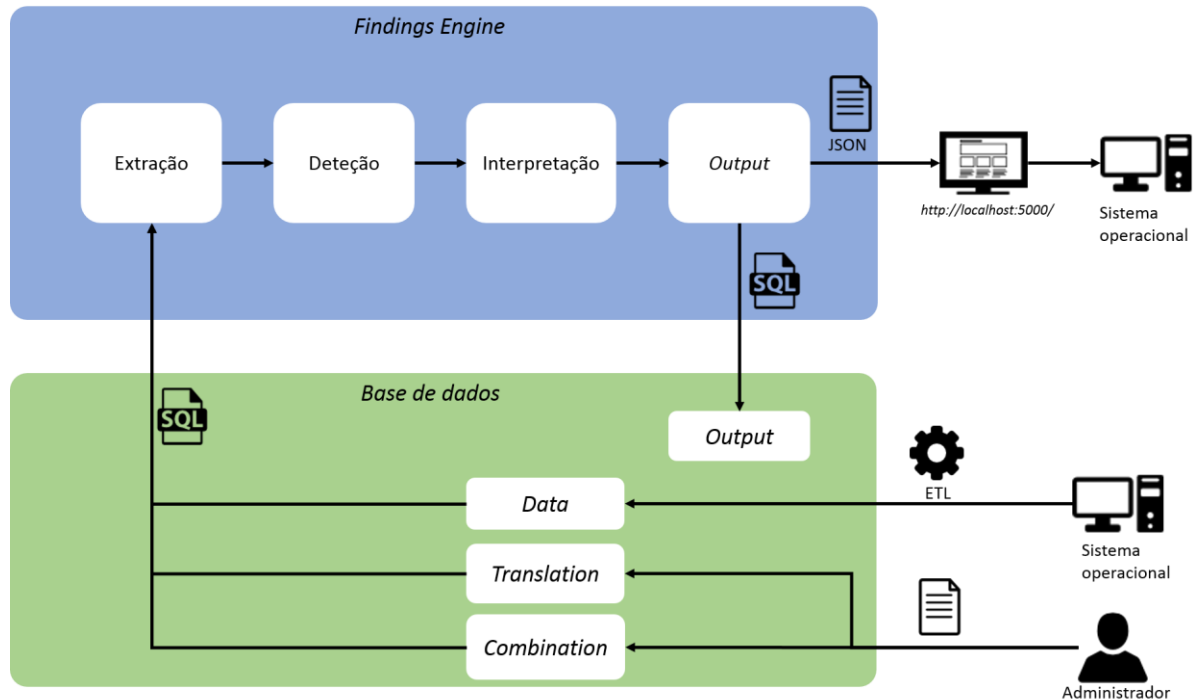


Figura 18. Interação entre os utilizadores/sistema operacional e o *Findings Engine*.

O sistema operacional em que o *Findings Engine* será integrado é o responsável por carregar os dados operacionais na tabela *data* através do processo denominado ETL. O administrador trata de popular as tabelas de parametrização e toda essa informação é lida pelo componente de extração do *Findings Engine*, através de *queries SQL*.

Os dados são então transformados em *Findings* e finalmente, já no componente de *output*, são disponibilizados ao sistema operacional através de um *web service*, neste caso registrado no endereço `http://localhost:5000/`. Os *Findings* são também inseridos na base de dados, na tabela *output*, através de *queries SQL*.

Como o sistema operacional é o próprio responsável por mostrar os *Findings* aos utilizadores a figura 18 não ilustra essa atividade. É fulcral realçar que a frequência de produção de *Findings* depende de cliente para cliente. Neste caso específico o nosso cliente pretende calcular *Findings* diariamente, logo o sistema operacional responsabilizar-se-á por fazer chamadas ao motor diariamente, de forma a executar os *Findings* parametrizados na tabela *combination* e mostrá-los ao utilizador final.

6 Implementação

Este capítulo é dedicado à explicação de todas as funcionalidades desenvolvidas no *Findings Engine*. De forma a facilitar a compreensão de todos os detalhes técnicos, serão explicadas as funcionalidades implementadas com o auxílio de um caso prático.

As próximas seções descrevem os dados com que o *Findings Engine* interage, bem como todas as funcionalidades implementadas.

6.1 Dados operacionais

Os dados operacionais que são carregados periodicamente na tabela de factos *data*, dizem respeito às vendas feitas por uma empresa da indústria das telecomunicações. Estes dados, com os quais o *Findings Engine* interagiu durante o estágio, são referentes a um período de 3 Meses – desde Dezembro de 2015 a Fevereiro de 2016. A granularidade dos dados é ao nível do segundo, uma vez que dizem respeito a todas as vendas feitas pelas lojas da operadora de telecomunicações.

Esta operadora é representada por um total de 2230 lojas, espalhadas por várias as regiões do país, e por 3496 assistentes. Num universo como este, a tabela de factos *data* é consequentemente composta por imensos registos, mais concretamente 1 095 696, o que perfaz um total de 223MB. A próxima figura ilustra uma pequena parte desses dados.

<i>Id_date</i>	<i>Cd_flow</i>	<i>Id_x_user</i>	<i>Id_x_local</i>	<i>Qt_total_duration</i>	<i>Dt_start_time</i>	<i>Dt_end_time</i>	<i>Year</i>	<i>Month</i>	<i>Day</i>	<i>Hour</i>	<i>Minute</i>	<i>Week</i>	<i>Week_name</i>	<i>Week_day</i>	<i>Ds_local</i>	<i>Ds_region</i>
20151201	53b1f115	1712	1621	153054	15.12.01 18:09:00	15.12.01 18:11:33	2015	12	1	18	9	49	1ª semana de Dezembro	3		Abu Dhabi
20151201	921d8881	872	925	106660	15.12.01 14:13:47	15.12.01 14:15:33	2015	12	1	14	13	49	1ª semana de Dezembro	3		Alain
20151201	01f37350	2770	891	162909	15.12.01 17:50:10	15.12.01 17:52:53	2015	12	1	17	50	49	1ª semana de Dezembro	3		Dubai

Figura 19. Amostra dos dados operacionais de uma empresa de telecomunicações.

As duas próximas seções descrevem as métricas e atributos dos dados.

6.1.1 Métricas

Uma métrica é uma medida quantitativa que quando é analisada minuciosamente pode revelar informações úteis com valor para a empresa.

Nos dados fornecidos pelo cliente existem duas métricas: o número de vendas e o tempo que cada venda demora a ser feita. O número de vendas não está explícito em nenhuma coluna, mas como cada registo da tabela *data* diz respeito a uma venda essa métrica pode ser facilmente medida através da contagem de registos. Por sua vez, o tempo que cada venda demora a ser feita é guardado na coluna *Qt_total_duration* e está representado em milissegundos.

As métricas descritas anteriormente podem ser analisadas isoladamente ou pode ser feita uma análise mais complexa, como por exemplo a combinação de métricas. A próxima figura (figura 20) ilustra um exemplo prático da diferença entre a análise de uma métrica (análise unidimensional) e a análise de duas métricas combinadas (análise bidimensional).

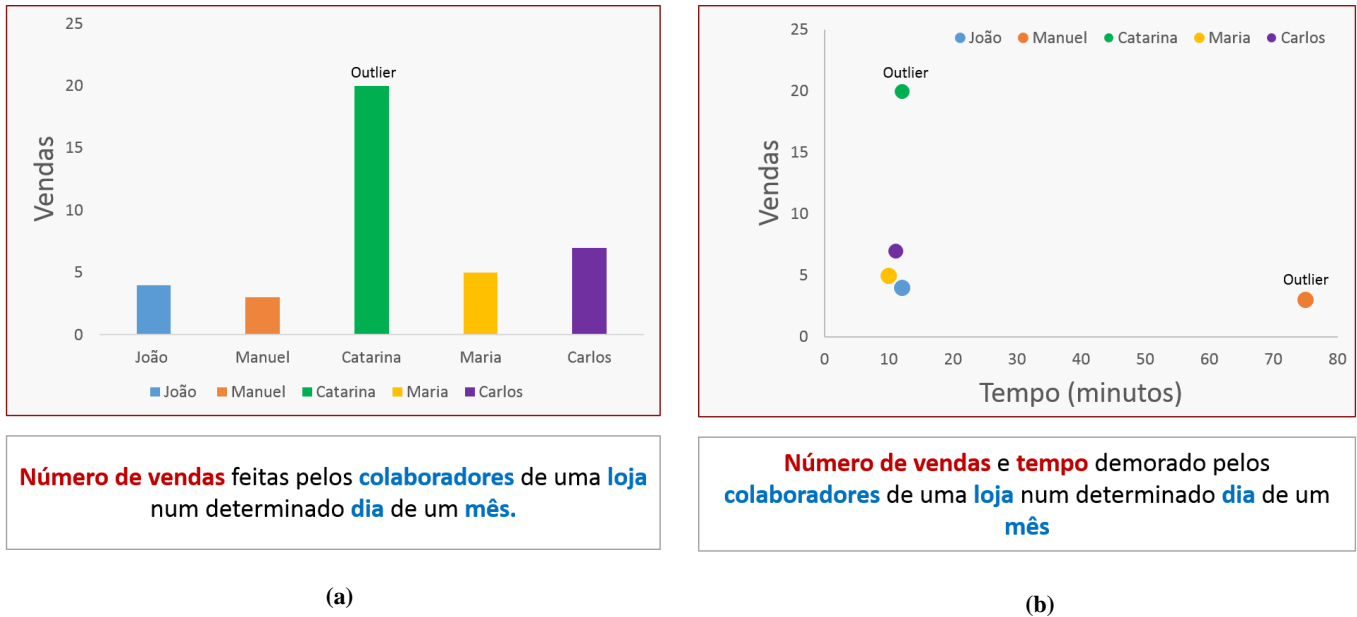


Figura 20. Diferença entre a análise de uma métrica e duas métricas, com vários atributos combinados. **(a)** Análise de uma métrica (vermelho) com quatro atributos (azul) combinados. **(b)** Análise de duas métricas (vermelho) com quatro atributos (azul) combinados.

Através da figura 20 (a) e (b) é possível verificar a diferença entre a análise de uma métrica e duas métricas. Tanto num caso como no outro existe a combinação entre quatro atributos diferentes.

Na figura 20 (a), quando é analisada apenas uma métrica isoladamente, existe apenas um *outlier* uma vez que a Catarina fez mais vendas do que era esperado, comparativamente com os seus colegas. Contudo, quando são analisadas duas métricas combinadas (número de vendas e tempo demorado) para o mesmo conjunto de dados, passa a existir mais um comportamento fora do esperado. Como se pode verificar pela figura 20 (b), apesar do Manuel fazer o mesmo número de vendas que a restante maioria dos colegas, demora demasiado tempo a completá-las.

Apesar de numa fase inicial do projeto se ter ponderado fazer análises bidimensionais, a prioridade do cliente foi o foco na análise unidimensional. Desta forma o *Findings Engine* apenas está preparado para detetar *outliers* em conjuntos de dados com apenas uma métrica.

6.1.2 Atributos

Os atributos são todas as colunas da tabela que não representam uma medida quantitativa. Nos dados fornecidos pelo cliente, existem diversos atributos como a loja em que foi registada uma venda, o assistente que a registou, o dia da semana, o mês, entre outros. Alguns atributos, apesar de não estarem representados na tabela de factos, podem ser facilmente adicionados como é o caso de um atributo que indique se a venda foi feita durante um fim de semana ou se durante a semana, ou até mesmo se a venda foi feita durante um período de pico ou durante um período regular.

O facto de existirem diversos atributos disponíveis nos dados fornecidos pelo cliente e de ser fácil adicionar ainda mais, vários *Findings* diferentes podem ser calculados apenas

através da combinação de diferentes atributos. Eis três exemplos de *Findings* totalmente diferentes em que há apenas a alteração de um atributo:

- “O João faz mais vendas que os seus colegas, durante o período de pico (das 17h às 21h)”
- “O João fez mais 40 vendas (+100%) que os seus colegas, durante a 3ª semana de Dezembro”
- “O João faz em média menos 50 vendas que os restantes colegas, durante o período da noite”

Como se pode verificar pelos exemplos anteriores, apenas com o auxílio de uma métrica (número de vendas) e a combinação entre vários atributos diferentes, podem ser criados inúmeros *Findings* com valor para o negócio do cliente.

Durante as próximas seções será mostrado um exemplo prático do processo de produção de um *Finding*, explicando em detalhe cada componente que faz parte da arquitetura do *Findings Engine*. De forma a facilitar a compreensão de cada componente, irá ser produzido o seguinte *Finding* para a seguinte loja e assistente:

- **Loja:** *Indirect Sales*
 - Loja com 14 assistentes
 - Faz cerca de 1500 vendas por semana
 - É uma loja com grande impacto, pois faz cerca de 2% do total de vendas da operadora
- **Assistente:** John
 - Faz cerca de 200 vendas por semana
 - É um assistente que faz 12% do total de vendas da *Indirect Sales*
- **Período de análise:** Da primeira semana de Janeiro à quarta semana de Fevereiro (8 semanas)
- **Objetivo do *Finding*:** Determinar se existe alguma semana em que o número médio de vendas diárias do John é irregular, comparativamente com o número médio de vendas diárias dos seus colegas de loja

6.2 Extração

O componente da extração é o responsável por extrair os dados operacionais que irão dar origem ao *Finding*. Este componente começa por construir as *queries*, relativamente aos dados inseridos na tabela *combination* e subsequentemente extrai os dados operacionais da tabela de factos. Tal como foi referido anteriormente, para produzir um *Finding* são sempre extraídos dois conjuntos de dados: conjunto de análise e conjunto de comparação.

Neste caso, em que se pretende determinar se existe alguma semana em que o número médio de vendas diárias do John é irregular, comparativamente com o número médio de vendas diárias dos seus colegas de loja, o conjunto de dados de análise contém o número médio de

vendas diárias feitas pelo John e o conjunto de dados de comparação contém o número médio de vendas diárias feitas pelos seus colegas.

A figura 21 ilustra o conjunto de dados de análise e o conjunto de dados de comparação.

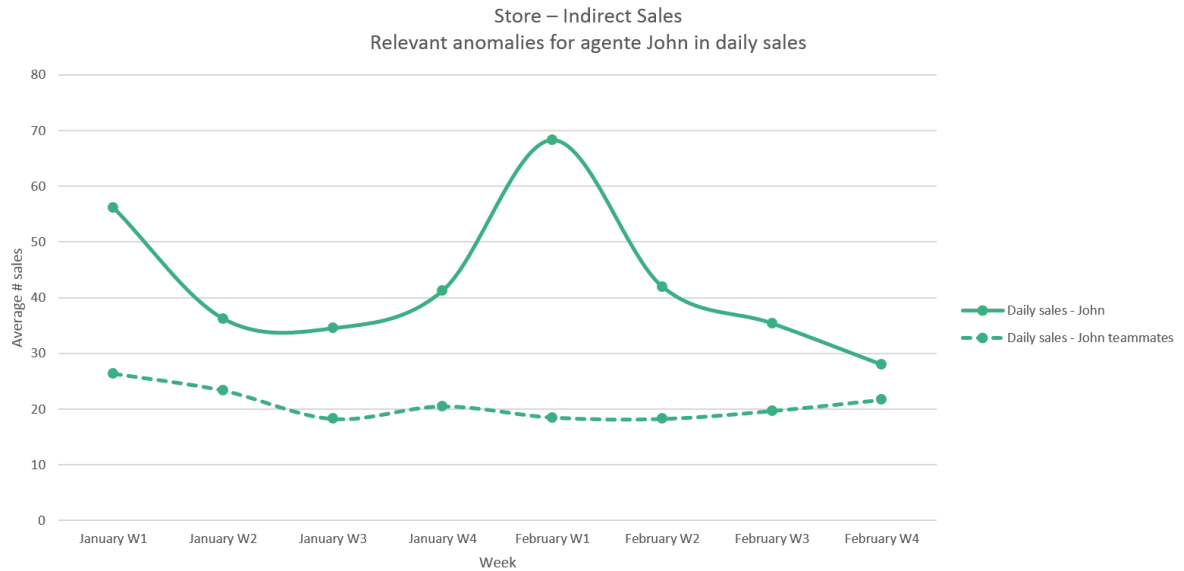


Figura 21. Conjunto de dados de análise – vendas feitas pelo John – e de comparação – vendas feitas pelos colegas; Linha a tracejado: conjunto de comparação; Linha contínua: conjunto de análise.

A linha contínua é o conjunto de dados de análise, ou seja, aquele em que pretende identificar se existem *outliers*. Esses *outliers* são determinados com base no conjunto de dados de comparação – representado através da linha a tracejado.

É importante referir que a linha a tracejado é uma média das vendas diárias feitas pelos colegas do John, de forma a facilitar a visualização do gráfico. Na realidade, o conjunto de dados de comparação é o número médio de vendas diárias feitas por cada colega do John, como se pode verificar através da próxima figura.

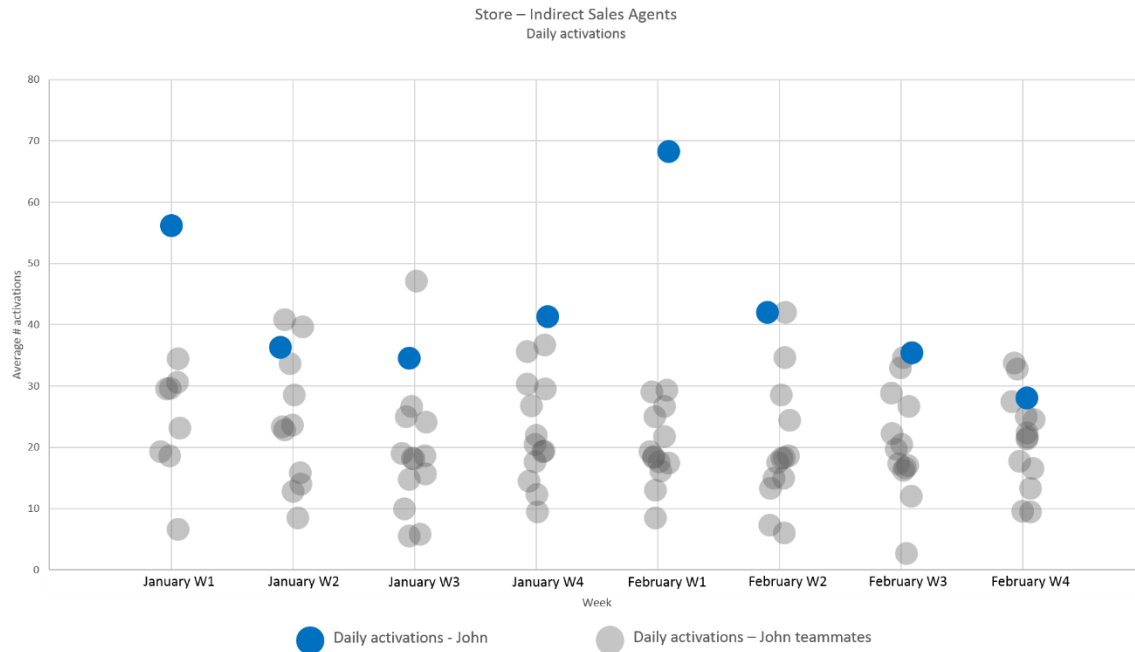


Figura 22. Número médio de vendas diárias feitas pelo John e pelos seus colegas; Azul - vendas do John; Cinzento - vendas dos colegas

Os pontos a azul indicam o número médio de vendas diárias feitas pelo John em cada semana – conjunto de dados de análise. Os pontos a cinzento são o número médio de vendas diárias feitas por cada colega do John em cada semana – conjunto de dados de comparação.

O objetivo do *Finding* é determinar se existe alguma semana em que o John fez um número irregular de vendas, comparativamente com os seus colegas. Por outras palavras pretende-se determinar se para cada semana algum dos pontos a azul é *outlier*, com base nos pontos cinzentos.

Estes dados são extraídos através da criação de duas *queries* com base nos dados parametrizados na tabela *combination*. A próxima seção descreve como é que o componente da extração cria as *queries* para extrair estes dados.

6.2.1 Combinações

O componente das combinações cria duas *queries* que irão selecionar os dados operacionais. Este componente utiliza a tabela *combination* para criar as *queries* e a tabela *configuration* para saber o nome da tabela onde se encontram os dados operacionais.

Para este *Finding* a tabela *combination* encontra-se parametrizada da seguinte forma:

Tabela 27. Tabela *combination* populada com dados que irão dar origem ao *Finding*.

Type	1
Description	Número médio de ativações feitas pelo John
Above	1
Below	-1
Metric	sales
Attrs	id_x_local, week_name, id_x_user
Filters	id_x_user = 3526 and id_x_local = 3114
Group_by	0
Date_range	week = thisWeek()
Permission	3526
Description_comp	Número médio de ativações feitas pelos colegas de loja do John
Metric_comp	sales
Attrs_comp	id_x_local, week_name, id_x_user
Filters_comp	id_x_user != 3526 and id_x_local = 3114
Group_by_comp	0
Date_range_comp	week = thisWeek()
For_all	0
Translation	Na %week_name o assistente %id_x_user teve uma %subida/descida de %deviation vendas (%+/- %percentage%), comparativamente com os restantes agentes da loja %id_x_local
Importance	High
Urgency	0 : 100; : 0; 100 :

Para a criação das *queries* são utilizadas todas as colunas exceto a *above*, *below*, *translation*, *importance* e *urgency*, que são utilizadas por outros componentes e as quais serão explicadas nas próximas secções.

A *query* de análise é criada da seguinte forma:

- `SELECT metric, attrs FROM data_table WHERE filters AND date_range GROUP BY attrs`

A *query* de comparação é criada da mesma forma que a *query* de análise, mas através das colunas que tem a terminação *comp*. Dado a forma a tabela *combination* se encontra parametrizada, eis as *queries* que são criadas para extrair o conjunto de dados de análise e de comparação, respetivamente:

Tabela 28. *Queries* criadas pelo mecanismo da extração.

QUERY (Análise)	<pre>SELECT sales, id_x_local, week_name, id_x_user FROM SALES_PER_WEEK WHERE id_x_user = 3526 AND id_x_local = 3114 AND week = thisWeek();</pre>
QUERY (Comparação)	<pre>SELECT sales, id_x_local, week_name, id_x_user FROM SALES_PER_WEEK WHERE id_x_user != 3526 AND id_x_local = 3114 AND week = thisWeek();</pre>

A *query* de análise extrai o número médio diário de vendas que o John (ID 3526) faz durante a semana atual, enquanto a *query* de comparação extrai os mesmos dados relativamente a todos os colegas que trabalham na mesma loja (ID 3114) que o John.

Tipicamente, os dados são extraídos diretamente da tabela de factos *data*. Porém, neste caso específico são extraídos da tabela *SALES_PER_WEEK*, uma vez que a tabela *configuration* se encontra parametrizada da seguinte forma (tabela 29).

Tabela 29. Tabela *configuration* populada com o nome das tabelas com que o *Findings Engine* irá interagir.

<i>Data_table</i>	SALES_PER_WEEK
<i>Combination</i>	Combination
<i>Output</i>	Output

A tabela *SALES_PER_WEEK* é uma vista materializada, criada especificamente para responder ao *Finding* enunciado, que contém o número médio de vendas diárias que cada assistente faz, estando essa informação agregada por semana. Uma vez que a informação já se encontra agregada por semana, as *queries* que selecionam os dados não necessitam da cláusula “*GROUP BY*”, justificando o facto da colunas *group_by* e *group_by_comp* estarem com o valor 0. A tabela 30 ilustra uma parte dos dados guardados na *SALES_PER_WEEK*.

Tabela 30. Tabela de factos *SALES_PER_WEEK* criada para responder ao *Finding*.

<i>week_name</i>	<i>sales</i>	<i>week</i>	<i>id_x_user</i>	<i>id_x_local</i>
1ª semana de Janeiro	18,6	1	4590	3114
1ª semana de Janeiro	56,1	1	3526	3114
1ª semana de Janeiro	29,5	1	2507	3114
1ª semana de Janeiro	6,6	1	4247	3114
1ª semana de Janeiro	34,4	1	2899	3114
1ª semana de Janeiro	29,6	1	3528	3114
1ª semana de Janeiro	30,5	1	4249	3114
1ª semana de Janeiro	23,1	1	4248	3114
1ª semana de Janeiro	19,3	1	4591	3114
2ª semana de Janeiro	28,5	2	2899	3114
2ª semana de Janeiro	33,6	2	2507	3114
2ª semana de Janeiro	8,5	2	4245	3114
2ª semana de Janeiro	39,6	2	3192	3114
2ª semana de Janeiro	22,8	2	3528	3114
2ª semana de Janeiro	40,8	2	4248	3114
2ª semana de Janeiro	23,6	2	4249	3114
2ª semana de Janeiro	23,3	2	4246	3114
2ª semana de Janeiro	12,8	2	4590	3114
...

Como se pretende produzir um *Finding* para um período de 8 semanas, a tabela *SALES_PER_WEEK* tem de ser atualizada ao final de cada semana com as vendas de cada assistente. Por sua vez, o *Findings Engine* tem de ser chamado pelo sistema operacional no final de cada semana. A cláusula “*WHERE week = thisWeek()*” garante que os dados extraídos dizem respeito à semana atual.

É importante realçar a importância que a coluna *for_all* tem para o motor. Quando o valor desta coluna é 1, significa que o *Finding* é produzido não só para um assistente em específico mas sim para todos os assistentes/lojas. Neste caso o *Finding* apenas é produzido para o John, uma vez que a coluna tem o valor 0. Porém se a coluna tivesse o valor 1, o *Finding* seria produzido para cada assistente da loja *Indirect Sales* e ainda para os restantes assistentes de todas as lojas da operadora de telecomunicações.

Um aspeto importante é também a coluna *filters_comp* da tabela *combination*. Como se pode verificar a coluna está preenchida com “*id_x_user != 2730, id_x_local = 3359*” uma vez que é necessário extrair as vendas feitas pelos colegas do John. Porém, se a coluna *for_all* estivesse com o valor 1, ou seja, se o *Finding* fosse produzido para todos os assistentes e todas as lojas, a coluna *filters_comp* poderia ficar preenchida com “*id_x_user != self, id_x_local = self*”.

A keyword “*self*” extrai automaticamente o ID do assistente/loja que está a ser analisado/a, permitindo assim que o *Findings Engine* produza um *Finding* para todos os assistentes tendo apenas de ser parametrizada uma linha na tabela *combination*. Esta decisão vem assim automatizar a extração dos dados, facilitar a parametrização dos *Findings* e garante que cada linha na tabela de combinações corresponde a um *Finding* diferente.

6.2.2 Leitura dos dados

Depois das *queries* serem criadas os dados são extraídos da base de dados. Para tal, o *Findings Engine* estabelece uma conexão com a base de dados *Oracle* através de um conector disponibilizado por uma biblioteca do *Python*.

Depois de as *queries* serem executadas, os dados extraídos da tabela *SALES_PER_WEEK* são guardados e estão prontos para serem analisados pelo mecanismo da deteção. O componente da leitura de dados é apenas responsável por executar as *queries* criadas pelo componente anterior e executá-las na camada da base de dados.

6.3 Deteção

O componente da deteção é o responsável por detetar se o conjunto de dados contém amostras irregulares. Como se tem vindo a referir ao longo do relatório, para detetar amostras irregulares no conjunto de dados é preciso ter uma base de comparação. Num cenário real em que por exemplo se quer detetar se um determinado assistente fez muitas vendas no mês de Julho, é necessário ter uma base de comparação, como por exemplo as vendas feitas pelo mesmo assistente no mesmo mês do ano anterior, ou as vendas feitas pelos seus colegas no mesmo mês e no mesmo ano.

Numa operadora de telemóveis como o nosso cliente, o volume de vendas varia de dia para dia e essa sazonalidade implica que os dados não sejam igualmente comparáveis. Tipicamente não se deve comparar um mês de Fevereiro, em que há pouco volume de vendas, com um mês de Dezembro, onde o volume de vendas é bastante acrescido. Desta forma, o facto do *Findings Engine* permitir parametrizar os dados em que se quer detetar comportamentos anormais e os dados que servem de termo de comparação, torna o sistema totalmente configurável permitindo ao administrador, que tem um profundo conhecimento do seu negócio, estabelecer o que é ou não comparável.

6.3.1 Detecção de *outliers*

O componente da deteção de *outliers* utiliza um algoritmo estatístico – MAD – para detetar se o conjunto de análise tem amostras irregulares. O MAD é um algoritmo que quando é aplicado a um conjunto de dados, retorna um limite superior e inferior. Todas as amostras que estejam acima/abaixo do limite superior/inferior são consideradas *outliers*.

De forma a perceber se existem irregularidades no conjunto de dados de análise com base nos dados de comparação, o algoritmo MAD é aplicado aos dados de comparação e é retornado um limite superior e inferior. Finalmente, todos os pontos do conjunto de dados de análise que estejam acima/abaixo desses limites são considerados amostras anormais - *outliers*. A próxima figura (figura 23) ilustra na íntegra o algoritmo de deteção de *outliers*.

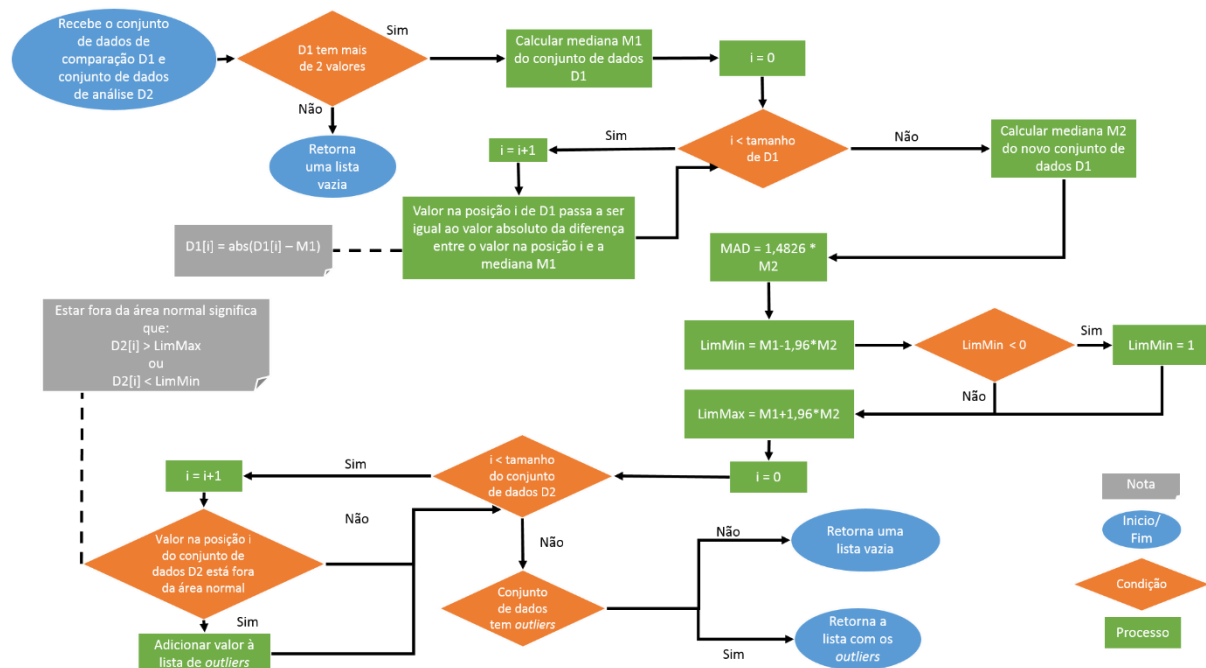


Figura 23. Diagrama de fluxo do algoritmo de deteção de *outliers* - MAD.

Neste caso prático, em que se quer perceber se o John fez um número irregular de vendas, aplica-se o algoritmo MAD às vendas feitas pelos seus colegas. O resultado desse algoritmo é um limite superior – número máximo de vendas que o John pode fazer até não ser considerado um *outlier* – e um limite inferior – número mínimo de vendas que o John pode fazer até não ser considerado um *outlier*. A próxima figura ilustra o resultado da aplicação do MAD neste exemplo em concreto.

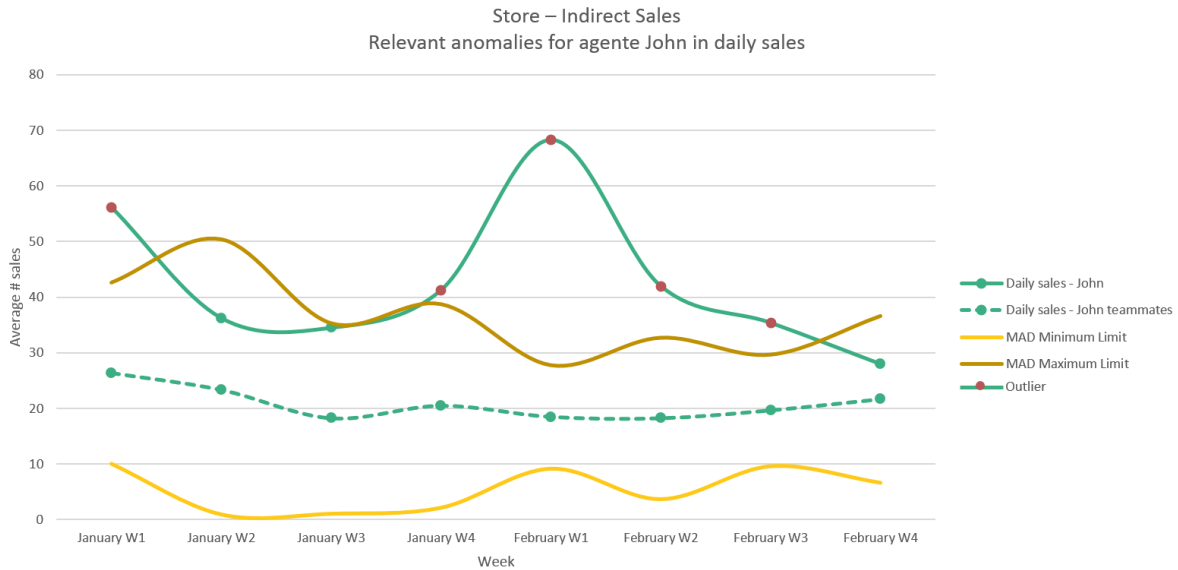


Figura 24. Resultado da aplicação do MAD ao conjunto de dados de comparação – vendas feitas pelos colegas do John; Pontos vermelhos: *outliers*.

Como se pode verificar através da figura 24, depois de o MAD ser aplicado ao conjunto de dados de comparação, que se encontra representado na figura 22 através dos pontos cinzentos, foram retornados 2 limites. Por exemplo, como na primeira semana de Janeiro o John fez cerca de 56 vendas diárias, ultrapassando o limite máximo esperado que é 42, o *Findings Engine* considera que nessa semana o John fez um número anómalo de vendas. O mesmo aconteceria se o John fizesse menos de 10 vendas.

O facto de os limites serem calculados através do conjunto de dados de comparação, significa que as vendas feitas pelos colegas do John determinam qual é o número de vendas que se espera que o John faça.

Através da análise da 24 facilmente se conclui que na 1ª e 4ª semana de Janeiro, bem como na 1ª, 2ª e 3ª semana de Fevereiro o John fez um número irregular de vendas diárias.

6.3.1.1 Trade-off

Numa fase inicial do projeto foi utilizado o teste de quartis, um algoritmo estatístico de deteção de *outliers*. Este algoritmo, à semelhança do MAD, é um algoritmo que retorna um limite superior e inferior e os pontos que ultrapassarem esses limites são considerados *outliers*. No entanto o teste de quartis tem a desvantagem de não obter bons resultados quando é aplicado a conjuntos de dados com poucas amostras. A figura seguinte ilustra os resultados do MAD e do teste de quartis para o *Finding* que está a ser apresentado.

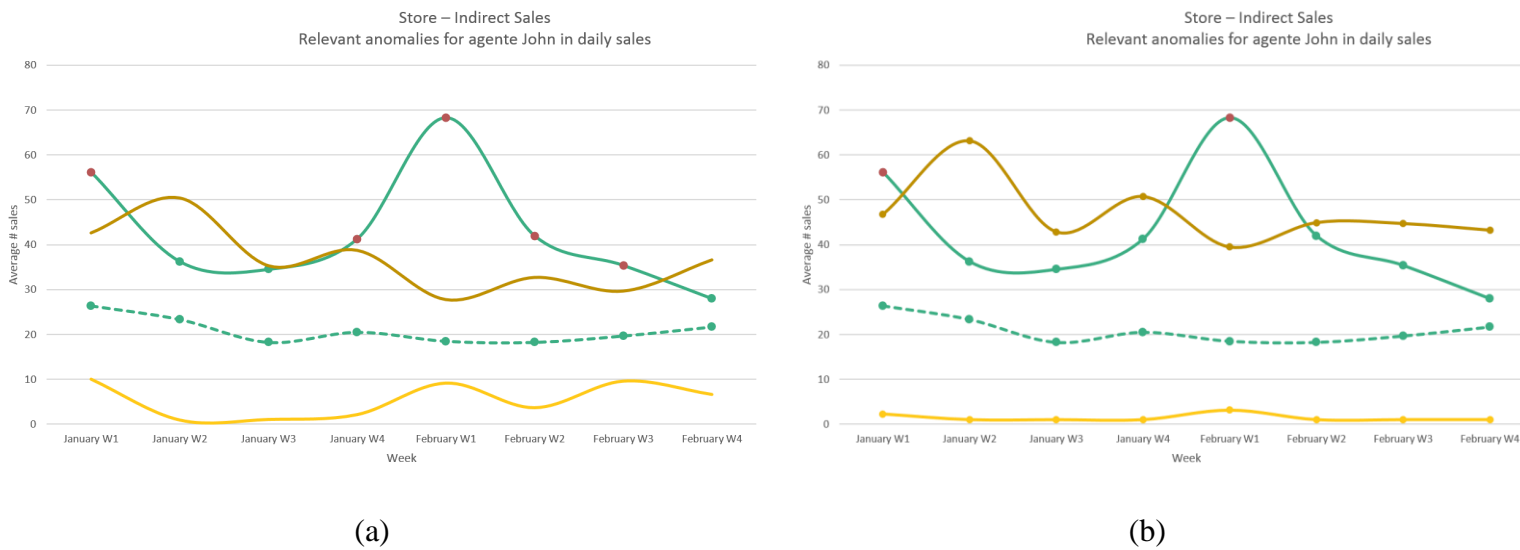


Figura 25. Comparação entre a aplicação do MAD e do teste de quartis ao número de vendas feitas pelos colegas do John. (a) Limites resultantes do MAD; (b) Limites resultantes do teste de quartis.

Neste caso prático o teste de quartis deteta que na 1^a semana de Janeiro e Fevereiro, o John fez um número irregular de vendas. Claramente o teste de quartis deteta menos *outliers* que o MAD e isso deve-se principalmente ao facto do conjunto de dados de comparação, ao qual o algoritmo é aplicado, conter poucas amostras.

Apesar de parecer que o teste de quartis obtém bons resultados para este caso prático, se for feita uma análise mais cuidada dos dados facilmente se percebe que este algoritmo é menos eficaz que o MAD. Através da figura 22, que representa as vendas feitas pelo John e pelos seus colegas, pode-se verificar que na 2^a semana de Fevereiro a grande maioria dos colegas do John fez menos de 20 vendas, tendo este feito um número muito superior à dos colegas. Porém, o teste de quartis não deteta esta semana como um *outlier*. O facto do teste de quartis ser menos eficaz para conjuntos com poucos dados ditou a utilização do algoritmo MAD.

6.3.2 Estabelecer a *baseline*

Depois de terem sido detetados os *outliers* é necessário agora estabelecer a *baseline*. A *baseline* tal como o nome indica é o valor esperado. Neste caso prático, a *baseline* é o número de vendas que se espera que o John faça, consoante as vendas feitas pelos seus colegas. Esta variável é essencial para o processo de priorização de *outliers*.

Se o *outlier* estiver acima do limite superior retornado pelo algoritmo MAD, a *baseline* desse *outlier* é o limite superior. Caso o *outlier* esteja abaixo do limite inferior, a *baseline* desse *outlier* é o limite inferior. A próxima imagem ilustra os *outliers* detetados pelo MAD, bem como os desvios dessas amostras em relação à *baseline*.

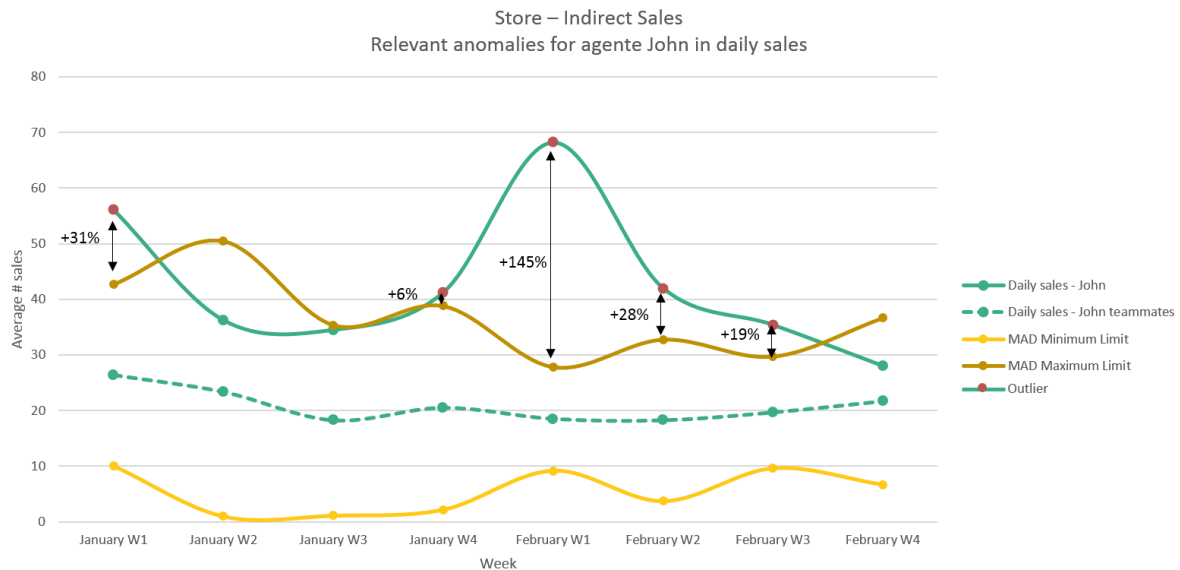


Figura 26. Desvio percentual entre as vendas irregulares feitas pelo John – *outliers* – e a *baseline*.

Neste caso prático, como só existem *outliers* acima do limite superior do MAD, o limite inferior do MAD não é *baseline* para nenhum ponto. Como se pode verificar pela figura 26, na 1ª semana de Fevereiro o John fez cerca de 68 vendas diárias e era esperado que fizesse 27, logo o John fez mais 145% do que era esperado comparativamente com os seus colegas. Facilmente pode-se concluir que este é o *outlier* mais discrepante, uma vez que o seu desvio em relação à *baseline* é superior ao dos restantes.

6.3.3 Priorização

Após terem sido calculados os desvios entre cada *outlier* e a *baseline* falta agora priorizar os *outliers*. Apesar de na 1ª semana o John ter feito muito mais vendas que os colegas não implica que o *Finding* relativamente a essa semana seja mais prioritário que os restantes. Isto significa que existe uma forte componente de negócio que dita o que é mais/menos prioritário para cada cliente.

O componente de priorização baseia-se no princípio de Eisenhower [25], proposto por Dwight David Eisenhower. Este princípio tem por base a importância e a urgência de uma tarefa. A importância de uma tarefa está relacionada com o facto de esta ter ou não valor e a sua urgência depende se esta tem de ser executada imediatamente. Quanto mais urgente e importante for uma tarefa, mais prioritária ela é. A figura 27 ilustra a matriz urgência/importância definida por Eisenhower.

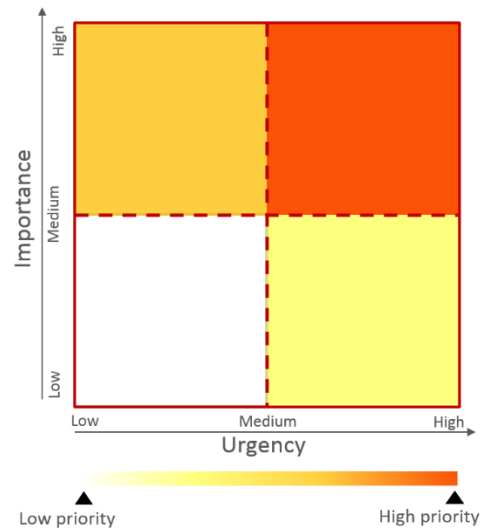


Figura 27. Matriz urgência/importância.

As colunas *urgency* e *importance* da tabela *combination* têm um papel fulcral no mecanismo de priorização. A coluna *importance* dita a importância que um *Finding* tem. A coluna *urgency* dita com que urgência é que cada *outlier* deve ser tratado. A figura 31 ilustra novamente a informação parametrizada nas colunas *urgency* e *importance*.

Tabela 31. Dados parametrizados nas colunas *urgency* e *importance* da tabela *combination*.

<i>Importance</i>	High
<i>Urgency</i>	0 : 100; : 0; 100 :

É fácil de perceber que o *Finding* que está a ser apresentado tem uma prioridade alta. A coluna urgência está dividida em pequena, média e alta urgência. A primeira *range* de valores, de 0 a 100, indica que os *outliers* que se desviam entre 0% e 100% da *baseline* têm uma urgência pequena. A segunda *range*, valores inferior a 0%, indica que os *outliers* que tenham um desvio negativo têm uma urgência média. Finalmente os *outliers* que se desviam mais de 100% da *baseline* têm urgência alta. A figura seguinte ilustra as vendas anómalas feitas pelo John e a respetiva prioridade que cada *Finding* terá.

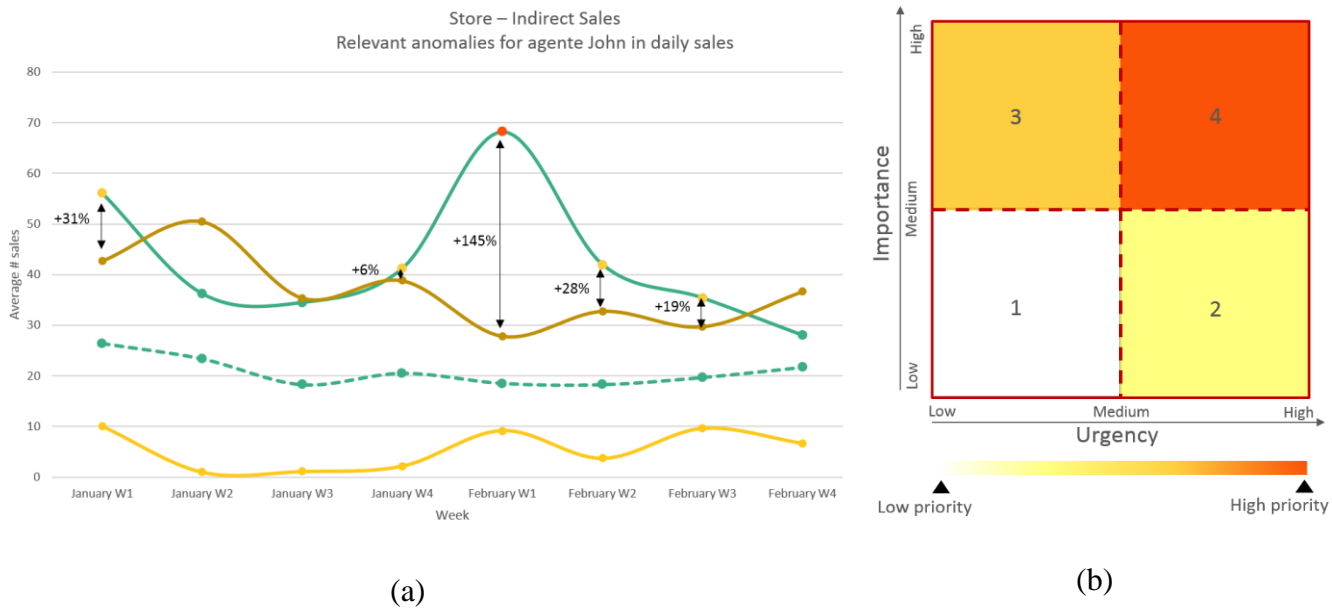


Figura 28. (a) Prioridade com que cada venda feita pelo John deve ser tratada. Ponto vermelho: prioridade 4. Pontos laranja: prioridade 3; (b) Matriz urgência/importância com os respetivos níveis de prioridade. Nível 4: maior prioridade. Nível 1: menor prioridade.

Através da figura 28 pode-se concluir que o *Finding* que irá ser produzido relativamente à 1ª semana de Fevereiro tem uma prioridade 4, uma vez que tem urgência alta e importância alta. Por sua vez os *Findings* relativamente à 1ª e 4ª semana de Janeiro, bem como para a 2ª e 3ª semana de Fevereiro tem uma prioridade 3, uma vez que a sua importância é alta mas a sua urgência é baixa.

O facto das colunas *urgency* e *importance* serem parametrizáveis permite ao administrador, que tem um total conhecimento do negócio, estabelecer a importância entre cada tipo de *Finding* que será produzido e, para cada um deles, dizer ainda a urgência com que cada *outlier* deve ser tratado. É importante referir que os *Findings* com maior prioridade são os primeiros a serem reportados ao utilizador através do componente de *output*.

6.4 Interpretação

O mecanismo da interpretação é o responsável por interpretar o significado do *outlier* e transformá-lo num *Finding*, ou seja, numa frase textual. Este mecanismo é composto pelo componente da classificação e da tradução.

Uma vez que já está estabelecida a *baseline* e já foi calculado o desvio entre o *outlier* e essa *baseline*, é necessário agora classificar o *outlier* – tom positivo ou negativo. O facto de um *outlier* estar acima da *baseline* não implica obrigatoriamente que o *Finding* tenha um tom positivo. Por exemplo no caso prático que está a ser apresentado, uma vez que a métrica que está a ser analisada é o número de vendas e os *outliers* estarem todos acima da *baseline* implica que todos os *Findings* criados terão um tom positivo, porque nessas semanas o John fez mais vendas que o esperado. Porém, se a métrica que estivesse a ser analisada fosse o tempo que as vendas demoram a ser feitas, o tom dos *Findings* seria negativo pois implicaria que o John teria demorado mais tempo que o esperado a fazer as suas vendas.

Finalmente é preciso converter o *outlier* numa frase textual que faça sentido para ser mostrada ao utilizador final. Este processo de conversão do *outlier* no *Finding* propriamente dito é feito pelo componente da tradução.

6.4.1 Classificação

Como foi referido anteriormente, o componente da classificação é uma parte importante do motor uma vez que permite classificar um *outlier*, ou seja, dizer se o *outlier* tem um tom positivo – com valor para o negócio – ou um tom negativo – um aspeto que deve ser rapidamente melhorado. Uma vez que o tom do *outlier* depende da métrica que está a ser analisada, é importante permitir ao administrador que este parametrize qual é o tom esperado para cada métrica, uma vez que este processo envolve uma componente de negócio.

O sistema de classificação é baseado nas colunas *above* e *below* da tabela *combination*. Através destas o administrador pode definir qual a classificação esperada para cada *outlier* esteja ele acima ou abaixo da *baseline*. No caso prático do John essas colunas encontram-se parametrizadas com a seguinte informação:

Tabela 32. Dados parametrizados nas colunas *above* e *below* da tabela *combination*.

<i>Above</i>	1
<i>Below</i>	-1

Através da figura 32 conclui-se que a classificação esperada para um *outlier* que esteja acima da *baseline* é positiva e para um *outlier* que esteja abaixo é negativa. Apesar de ser um mecanismo de classificação simples sem qualquer tipo de inteligência, o negócio exige que assim o seja uma vez que parte do administrador saber qual o impacto que cada métrica tem no seu negócio e subsequentemente a classificação esperada. A figura seguinte (figura 29) ilustra as vendas irregulares feitas pelo John e a classificação de cada uma.

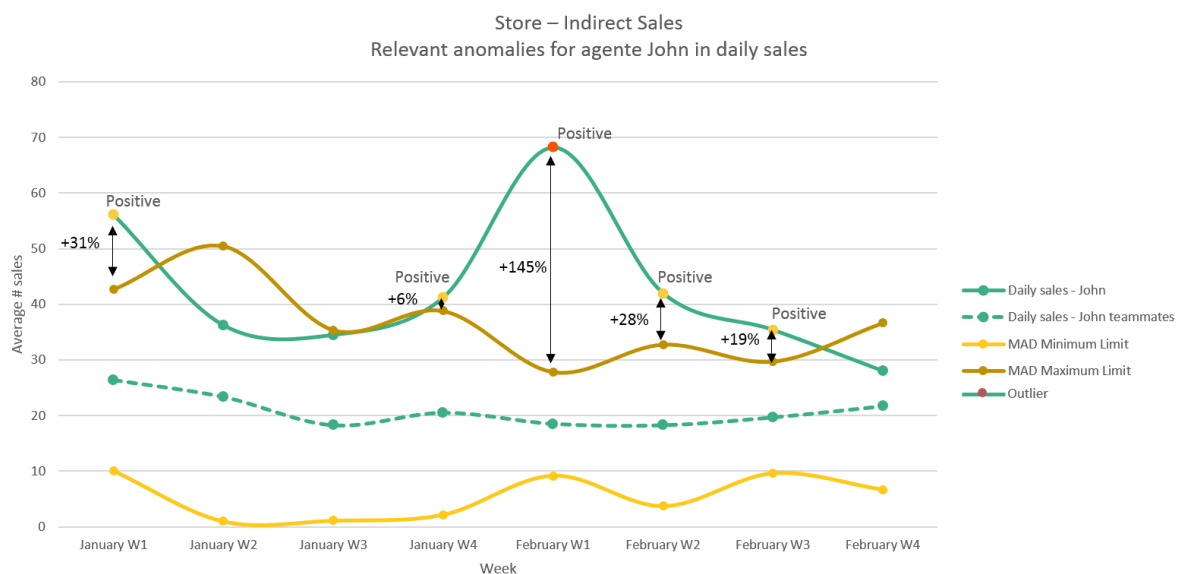


Figura 29. Classificação de cada venda feita pelo John.

6.4.2 Tradução

O componente da tradução é responsável por transformar cada *outlier* num *Finding*, ou seja, numa frase textual com informação relevante para o negócio. O componente da tradução foi feito para que o administrador possa parametrizar a frase textual em que o *outlier* será traduzido, uma vez que essa frase depende do contexto do negócio.

O componente da tradução utiliza a frase que está na coluna *translation* para o processo de tradução de um *outlier* num *Finding*. Para o caso prático dado durante o relatório a coluna *translation* contém os seguintes dados.

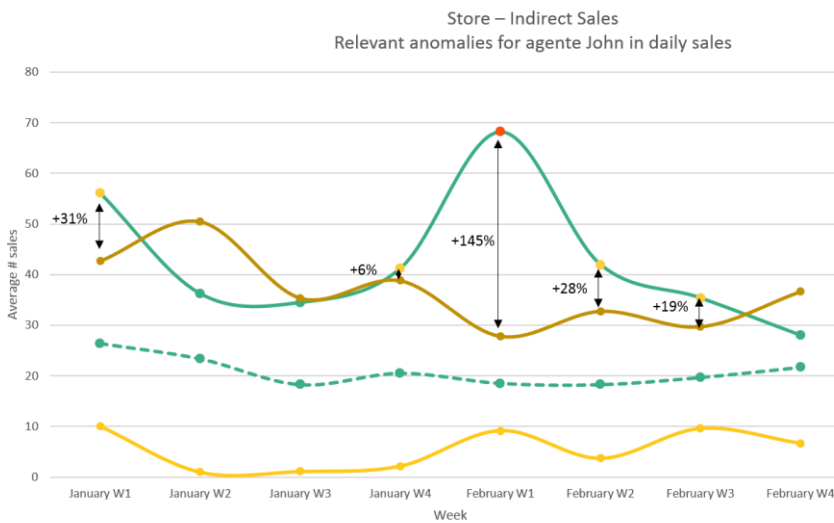
Tabela 33. Frase textual, parametrizada na coluna *translation*, em que o *outlier* será transformado.

<i>Translation</i>	Na %week_name o assistente %id_x_user teve uma %subida/descida de %deviation vendas (%+/- %percentage%), comparativamente com os restantes agentes da loja %id_x_local
--------------------	--

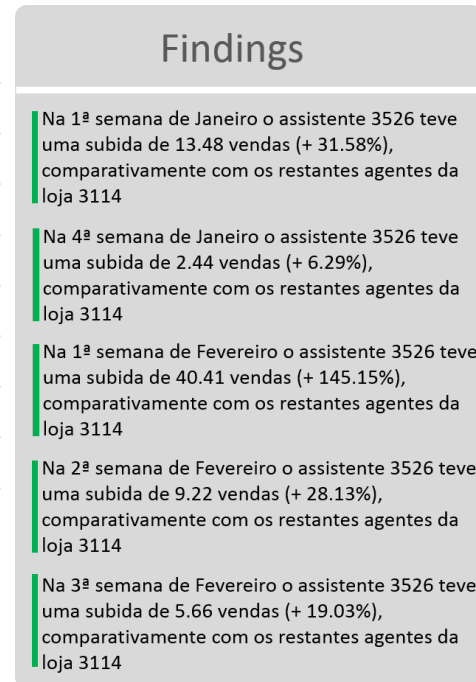
Como se pode verificar pela tabela 33, a frase inserida na coluna *translation* contém texto normal e ainda as palavras como %week_name ou %id_x_user que são preenchidas com as variáveis da tabela de factos, neste caso com o nome da semana e o ID do utilizador, respetivamente. A frase textual contém ainda outro tipo de palavras como %subida/descida ou %+/- que são apresentadas consoante a posição do *outlier* em relação à *baseline*. É importante referir que a palavra à esquerda tem um tom positivo e a palavra à direita um tom negativo, ou seja, %tom positivo/tom negativo.

Por exemplo no caso prático que está a ser apresentado, o facto de o John fazer mais vendas, tem um tom positivo. Isso explica que as palavras “subida” e “+” estejam à esquerda do símbolo “/”. Por sua vez se a métrica analisada fosse o tempo que as vendas demoram a ser feitas, as palavras “subida” e “+” teriam um tom negativo, logo teriam de estar à direita do símbolo “/”.

A próxima figura ilustra as vendas irregulares feitas pelo John com os respetivos *Findings* produzidos pelo motor.



(a)



(b)

Figura 30. (a) Vendas irregulares feitas pelo John com o respetivo desvio percentual face à *baseline*; (b) *Findings* produzidos relativamente às vendas irregulares feitas pelo John

6.5 Output

O mecanismo de *output* garante que os *Findings*, bem como toda a sua informação relevante, são guardados numa base de dados e ainda são disponibilizados ao utilizador através de um *web service*.

Quando o *Findings Engine* é invocado pelo sistema operacional, através de um *web service*, os *Findings* que são produzidos são guardados numa base de dados e são retornados ao sistema operacional através do próprio *web service*.

6.5.1 Base de dados

A base de dados guarda um histórico com todos os *Findings* produzidos pelo motor. Esta base de dados é composta pela tabela *output*, previamente descrita na secção 5.2.3. A próxima figura ilustra os *Findings* guardados na base de dados para o caso prático que está a ser apresentado.

Tabela 34. Tabela de *output* com os *Findings* produzidos pelo sistema.

<i>Output_id</i>	<i>Date_prod</i>	<i>Sentence</i>	<i>Classification</i>	<i>Permission</i>	<i>Absolute_dev</i>	<i>Percent_dev</i>	<i>Baseline</i>	<i>Type</i>	<i>Priority</i>	<i>Urgency</i>	<i>Importance</i>
1	20-06-2016	Na semana 1ª semana de Janeiro o agente 3526 teve uma subida de 13.48 vendas (+ 31.58%), comparativamente com os restantes agentes da loja 3114	1	3526	13,48	31,58	42,68	1	3	Low	High
2	20-06-2016	Na semana 2ª semana de Fevereiro o agente 3526 teve uma subida de 9.22 vendas (+ 28.13%), comparativamente com os restantes agentes da loja 3114	1	3526	9,22	28,13	32,77	1	3	Low	High
3	20-06-2016	Na semana 3ª semana de Fevereiro o agente 3526 teve uma subida de 5.66 vendas (+ 19.03%), comparativamente com os restantes agentes da loja 3114	1	3526	5,66	19,03	29,74	1	3	Low	High
4	20-06-2016	Na semana 1ª semana de Fevereiro o agente 3526 teve uma subida de 40.41 vendas (+ 145.15%), comparativamente com os restantes agentes da loja 3114	1	3526	40,41	145,15	27,84	1	4	High	High
5	20-06-2016	Na semana 4ª semana de Janeiro o agente 3526 teve uma subida de 2.44 vendas (+ 6.29%), comparativamente com os restantes agentes da loja 3114	1	3526	2,44	6,29	38,80	1	3	Low	High

Toda esta informação apresentada na tabela 34 também é retornada ao sistema operacional através de um *web service*. A próxima secção descreve esse componente.

6.5.2 *Web service*

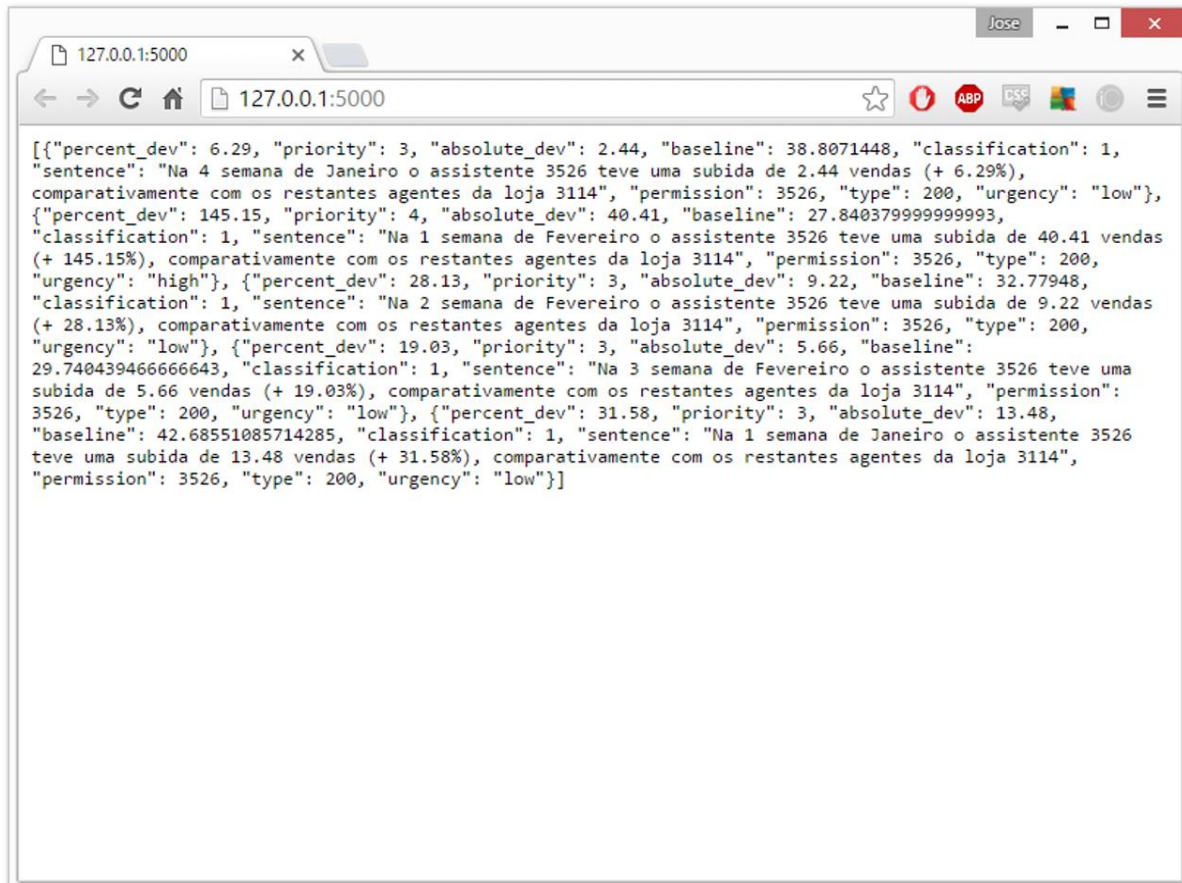
Para invocar o *Findings Engine* o sistema operacional faz uma chamada a um *web service* implementado em REST e disponibilizado através de um *web service*. É importante referir que o *web service* pode facilmente ser disponibilizado através do endereço IP, tornando-o disponível a todos os dispositivos que estejam na mesma rede. O *web service* aceita apenas a chamada do seguinte método:

- `http://{endereço}:{porto}/`

Para efeitos de teste foi utilizada a máquina local e o porto 5000. O *web service* encontra-se portanto à escuta no seguinte endereço:

- `http://localhost:5000/`

Quando o sistema operacional faz uma chamada a este endereço os *Findings* parametrizados na tabela *combination* são produzidos e, depois de serem inseridos na base de dados, são apresentados ao sistema operacional através do próprio *web service*.



```
[{"percent_dev": 6.29, "priority": 3, "absolute_dev": 2.44, "baseline": 38.8071448, "classification": 1,
"sentence": "Na 4 semana de Janeiro o assistente 3526 teve uma subida de 2.44 vendas (+ 6.29%),
comparativamente com os restantes agentes da loja 3114", "permission": 3526, "type": 200, "urgency": "low"},
{"percent_dev": 145.15, "priority": 4, "absolute_dev": 40.41, "baseline": 27.840379999999993,
"classification": 1, "sentence": "Na 1 semana de Fevereiro o assistente 3526 teve uma subida de 40.41 vendas
(+ 145.15%), comparativamente com os restantes agentes da loja 3114", "permission": 3526, "type": 200,
"urgency": "high"}, {"percent_dev": 28.13, "priority": 3, "absolute_dev": 9.22, "baseline": 32.77948,
"classification": 1, "sentence": "Na 2 semana de Fevereiro o assistente 3526 teve uma subida de 9.22 vendas
(+ 28.13%), comparativamente com os restantes agentes da loja 3114", "permission": 3526, "type": 200,
"urgency": "low"}, {"percent_dev": 19.03, "priority": 3, "absolute_dev": 5.66, "baseline":
29.740439466666643, "classification": 1, "sentence": "Na 3 semana de Fevereiro o assistente 3526 teve uma
subida de 5.66 vendas (+ 19.03%), comparativamente com os restantes agentes da loja 3114", "permission":
3526, "type": 200, "urgency": "low"}, {"percent_dev": 31.58, "priority": 3, "absolute_dev": 13.48,
"baseline": 42.68551085714285, "classification": 1, "sentence": "Na 1 semana de Janeiro o assistente 3526
teve uma subida de 13.48 vendas (+ 31.58%), comparativamente com os restantes agentes da loja 3114",
"permission": 3526, "type": 200, "urgency": "low"}]
```

Figura 31. Findings devolvidos em formato JSON através do web service.

Como se pode verificar pela figura 31, os *Findings* são disponibilizados através de um formato *standard* – JSON. Este formato de dados, que foi criado como alternativa ao XML, tem a vantagem de ser mais leve e ser mais fácil de escrever. O JSON permite assim que dois sistemas distintos, neste caso o sistema operacional e o *Findings Engine*, comuniquem entre si. A utilização do *web service* conjuntamente com o formato de dados JSON garante a interoperabilidade do motor, uma vez que este consegue comunicar com qualquer sistema.

7 Testes

O processo de validação do produto desenvolvido é um passo fundamental em qualquer projeto de desenvolvimento de *software*. Este capítulo é dedicado aos testes de *performance* e aos testes de caixa preta – testes unitários e testes de aceitação.

Os testes de performance têm um papel fundamental no processo de desenvolvimento do *Findings Engine*, pois espera-se perceber qual o tempo de resposta do motor e que tipo de soluções podem ser implementadas para garantir a escalabilidade do sistema.

Os testes unitários permitem testar o algoritmo de deteção de *outliers* enquanto os testes de aceitação permitem testar os componentes desenvolvidos.

Todos os testes foram realizados com a mesma máquina:

SQ: Windows 8.1 Enterprise – 64 Bits

CPU: Intel® Core™ i7-4610M @ 3.00 GHz

RAM: 8GB

7.1 Performance

A performance é um aspeto crítico do *Findings Engine*. Tal como foi referido anteriormente, o motor irá lidar com um grande volume de dados pelo que é fulcral garantir a escalabilidade do sistema. O objetivo desta secção é fazer testes de *performance* a fim de perceber qual o tempo de resposta do *Findings Engine*.

Para medir o tempo de resposta do *Findings Engine* foram escolhidas 6 lojas – 1 loja que faz poucas vendas, 1 loja que faz muitas vendas e 4 lojas que fazem um número normal de vendas – e foi executado o mesmo *Finding* para cada loja.

As lojas escolhidas são as seguintes:

Tabela 35. Informação sobre lojas/assistentes utilizados para efeitos de teste.

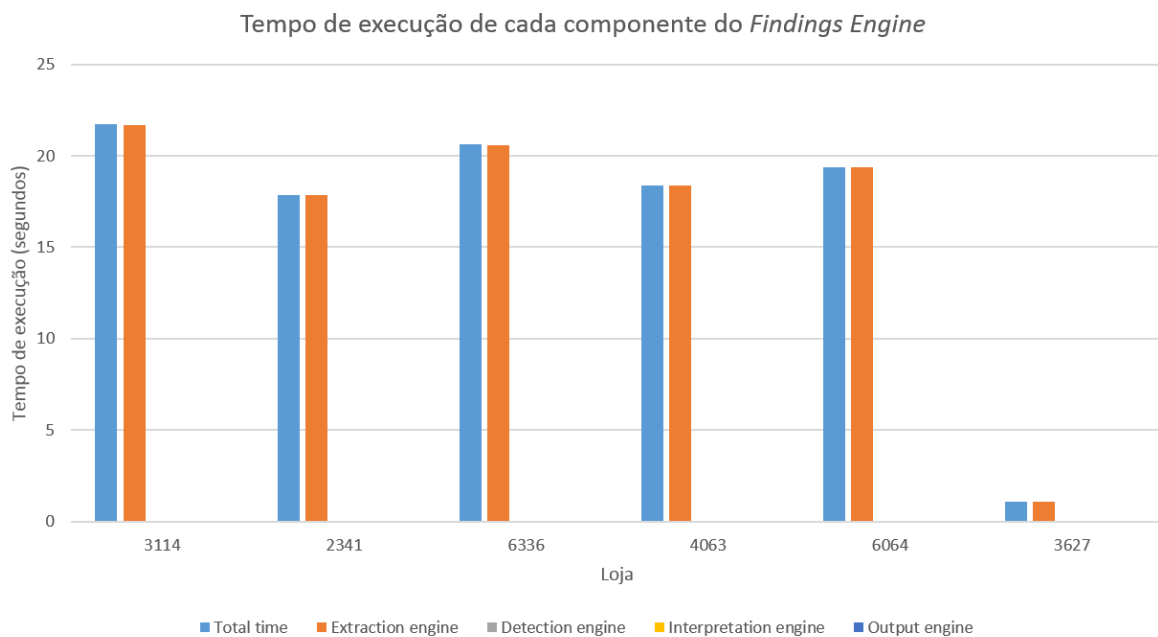
Loja (ID)	Assistente (ID)	Número de vendas (% do total)	Período de análise	Objetivo do <i>Finding</i>
<i>Indirect Sales</i> (3114)	John (3526)	Muitas (2%)	8 Semanas	Determinar se existe alguma semana em que o número médio de vendas diárias do assistente é irregular, comparativamente com o número médio de vendas diárias dos seus colegas de loja
<i>Alaam Khuyool</i> (2341)	Adam (3264)	Normal (0,66%)		
<i>Rukin Suraa</i> (6336)	Calvin (2249)	Normal (0,65%)		
<i>Al Khaleej</i> (4063)	Sophia (1617030)	Normal (0,68%)		
<i>Hethro Devices</i> (6064)	Aria (2390)	Normal (0,58%)		
<i>Palace Mobile</i> (3627)	Isabella (581)	Poucas (0,14%)		

Será então produzido o *Finding* para cada uma das 6 lojas, individualmente, e será medido o tempo que o *Findings Engine* demora a executar, bem como cada um dos componentes. Com este teste pretende-se determinar o tempo de execução do motor como um todo mas também de cada componente, a fim de determinar qual o componente mais crítico – que demora mais a executar. A próxima tabela ilustra as *queries* criadas para a produção do *Finding* descrito.

Tabela 36. *Queries* criadas para responder ao *Finding*. Dados hospedados numa tabela de factos indexada.

<i>Query</i> de comparação	<i>Query</i> de análise
<pre> SELECT avg(sales), week, id_x_user, id_x_local FROM (SELECT count(*) sales, day, week, id_x_local, id_x_user FROM du GROUP BY week,day, id_x_local,id_x_user) WHERE id_x_local = {id_x_local} and id_x_user != {id_x_user} and week = {week} GROUP BY week, id_x_user, id_x_local </pre>	<pre> SELECT avg(sales), week, id_x_user, id_x_local FROM (SELECT count(*) sales, day, week, id_x_local, id_x_user FROM du GROUP BY week,day, id_x_local,id_x_user) WHERE id_x_local = {id_x_local} and id_x_user = {id_x_user} and week = {week} GROUP BY week, id_x_user, id_x_local </pre>

É importante referir que a tabela de factos “du” contém índices nas seguintes colunas: *id_x_user* e *id_x_local*. A próxima figura ilustra o tempo que o *Findings Engine* demorou a executar o *Finding* para cada uma das 6 lojas.

**Figura 32.** Tempo de execução de cada componente do sistema.

Como se pode verificar pela figura anterior, o componente que demora mais tempo a executar é o da extração, que demora uma percentagem muito significativa (quase 100%) do tempo total de execução. Por sua vez, os restantes componentes demoram uma percentagem pouco significativa do tempo total de execução do *Findings Engine*.

Esta percentagem tão grande que o componente de extração demora a executar deve-se ao facto de ter de seleccionar e extrair os dados da camada da base de dados para a camada do *Findings Engine*. Este passo merece uma atenção especial e é necessário otimizá-lo a fim de garantir a escalabilidade de todo o sistema. Os restantes componentes não necessitam de qualquer optimização dado que o seu tempo de execução é bastante satisfatório.

Existem diversos tipos de otimizações que podem ser feitas ao nível do componente da extração tais como:

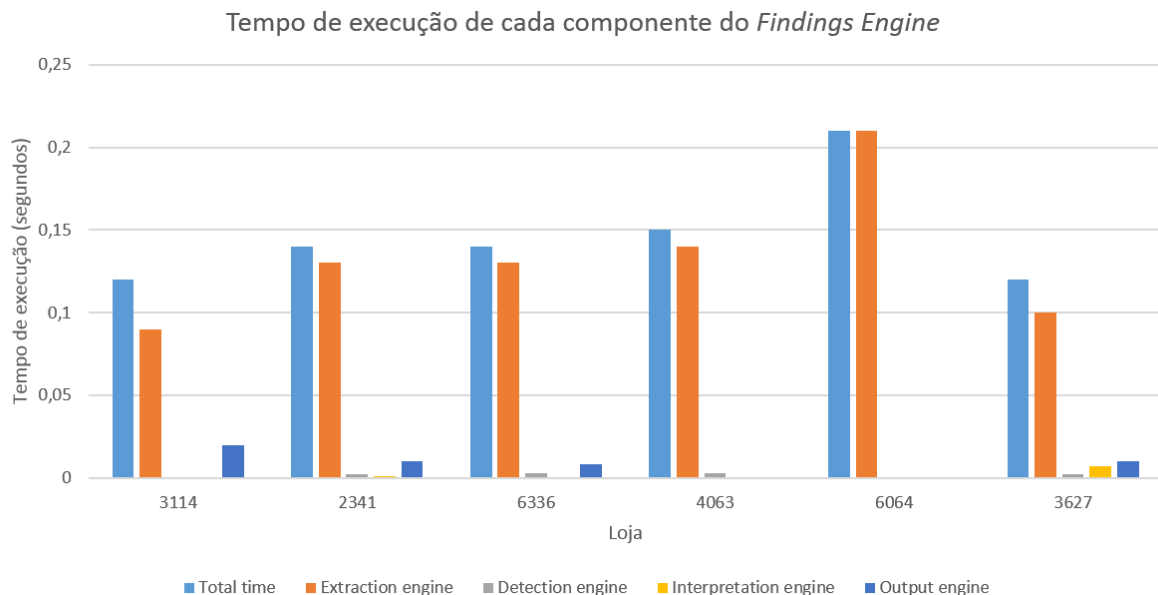
- Índices: é uma estrutura de dados que aumenta a rapidez de resposta da base de dados. Os índices são utilizados para procurar informação numa tabela mais rapidamente, sem ter de percorrer cada linha dessa tabela. Existem diversos tipos de índices e normalmente são aplicados em colunas que são utilizadas com filtro de uma *query*, como é o caso das colunas que aparecem na cláusula *WHERE*.
- Caching: é um *software/hardware* que guarda informação que pode ser reutilizada. O *caching* de informação na base de dados pode implicar guardar resultados de uma *query* para serem usados no futuro. Uma solução que pode ser implementada no *Findings Engine* é adicionar uma nova camada, entre a base de dados e o *Findings Engine*, que tenha como objetivo fazer o *caching* de informação que possa ser útil a diversos *Findings*.
- Purging: é o processo de “limpar” ou agregar dados obsoletos. O processo de remover ou agregar informação que já não é essencial permite diminuir o espaço ocupado nas tabelas, garantindo assim melhor tempo de resposta das mesmas. Uma solução que pode ser implementada no *Findings Engine* é agregar semanalmente os dados com mais de um ano.
- Sharding: é o processo de particionamento horizontal de uma tabela. Existem dois tipos de particionamento – vertical e horizontal. O *sharding* é um processo que permite particionar os dados entre várias tabelas de forma a distribuir a carga de trabalho. No *Findings Engine* pode ser implementado um particionamento pela data de registo. Num cenário como este os dados operacionais ficarão distribuídos por várias tabelas, consoante a data em que tenham sido registados.
- Vistas materializadas: são tabelas que contém dados resultantes de uma *query*. As vistas materializadas são bastante uteis para guardar informação agregada ou para pré-calcular métricas e guardá-las. Um exemplo de aplicação no *Findings Engine* é, por exemplo, criar uma vista materializada com o número médio de vendas diárias feitas por cada assistente, estando essa informação agregada semanalmente. Desta forma é possível responder ao *Finding* que tem vindo a ser descrito durante o relatório.

Para perceber o impacto dos mecanismos de otimização, foi criada uma vista materializada para responder ao *Finding* produzido para as 6 lojas. Essa vista materializada, denominada *AGENT_SALES_PER_WEEK*, contém o número médio de vendas diárias feitas por cada assistente, estando essa informação agregada semanalmente. Assim sendo, as *queries* criadas são as seguintes:

Tabela 37. *Queries* criadas para responder ao *Finding*. Dados hospedados numa vista materializada.

<i>Query</i> de comparação	<i>Query</i> de análise
<pre>SELECT sales, week, id_x_user, id_x_local FROM agent_sales_per_week WHERE id_x_local = {id_x_local} and id_x_user != {id_x_user} and week = {week}</pre>	<pre>SELECT sales, week, id_x_user, id_x_local FROM agent_sales_per_week WHERE id_x_local = {id_x_local} and id_x_user = {id_x_user} and week = {week}</pre>

É importante referir que uma vez que a vista materializada já contém o número médio de vendas pré-calculado e que a informação já está agregada, não é necessário adicionar a cláusula *GROUP BY* nas *queries*. A próxima figura ilustra o tempo que o *Findings Engine* demorou a produzir o *Finding* para cada uma das 6 lojas, depois de adicionada esta otimização no componente da extração.

**Figura 33.** Tempo de execução de cada componente do sistema.

Como se pode verificar pela imagem 33, a construção de uma vista materializada para responder ao *Finding* teve um grande impacto na performance do nosso motor. Apesar do componente da extração continuar ainda a demorar uma grande percentagem do tempo total de execução, o tempo de resposta melhorou significativamente.

Como se pode constatar, criar o *Finding* para cada loja numa base de dados com índices, demorou em média 16,5 segundos. Depois de ter sido feita a otimização do componente da extração, o motor passou a demorar em média 0,14 segundos.

É então importante garantir que estas vistas sejam criadas quando se pretende responder a um *Finding* que lide com um grande volume de dados e cujos esses dados tenham de ser agregados. A criação das vistas materializadas, bem como a implementação de um mecanismo de *caching* ou até de um mecanismo de particionamento, depende de negócio para negócio e

do tipo de *Findings* que se querem produzir. Posto isto, as soluções apresentadas devem ser analisadas antes de serem implementadas no cliente.

7.2 Unitários

Os testes unitários são testes que validam partes individuais do código para garantir que cada módulo funciona corretamente. Para as várias unidades individuais (método ou parte de um método) que existam no programa, criam-se vários testes com o resultado que se espera obter para determinados *inputs*. Se o resultado obtido for igual ao esperado então o teste passa.

De forma a validar o algoritmo MAD, implementado para a deteção de *outliers*, foram realizados testes unitários. Para tal foram criados 5 casos de teste com os respetivos valores de *input* e qual o *output* esperado. Através da biblioteca *unittest* fornecida pelo *Python* foram validados os casos de teste. A tabela seguinte ilustra os casos de teste que foram criados e testados.

Tabela 38. Resultados dos testes unitários.

<i>Input</i>	<i>Output</i>	Resultado
12, 12, 12, 2, 22, 13	10.547052, 12.0, 13.452948	OK
1, 1, 1, 2, 2, 2	0.047052, 1.5, 2.952948	OK
4, 3, 9, 12	1, 6.5, 15.2176879	OK
60, 12, 200, 100, 25	1, 60.0, 176.23584	OK
100, 120, 50, 0	1, 75.0, 176.70636	OK

A coluna *input* contém o conjunto de dados de comparação sobre o qual o MAD é aplicado. A coluna de *output* é composta pelo limite mínimo calculado pelo MAD, a mediana do conjunto de dados e o limite máximo, respetivamente. A última coluna contém o resultado obtido do teste unitário.

7.3 Aceitação

Os testes de aceitação permitem testar todos os componentes do *Findings Engine*. Estes testes, à semelhança dos anteriores, não são muito detalhados uma vez que devem ser realizados por uma equipa de qualidade de *software*. Desta forma o objetivo dos testes realizados é validar, de uma forma simples, o funcionamento dos componentes desenvolvidos e garantir que os requisitos são cumpridos.

As próximas tabelas descrevem os testes realizados para cada um dos requisitos funcionais apresentados na secção 4.1.

Tabela 39. Teste de aceitação - T_01.

T_01 Inserir dados nas tabelas de parametrizações	
Sumário	O administrador acede à base de dados e pode parametrizar as tabelas – <i>combination</i> e <i>configuration</i> .
Pré-condições	<ul style="list-style-type: none"> • Acesso à base de dados
Descrição	<p>Tabela <i>combination</i>:</p> <ol style="list-style-type: none"> 9. Inserir atributos que dão origem a um <i>Finding</i> 10. Inserir atributos que servem de <i>baseline</i> ao <i>Finding</i> 11. Inserir a métrica que irá ser analisada 12. Inserir a métrica que serve de <i>baseline</i> ao <i>Finding</i> 13. Inserir o tipo de classificação esperada 14. Inserir a frase em que o <i>Finding</i> é traduzido 15. Inserir a importância do <i>Finding</i> 16. Inserir a urgência do <i>Finding</i> <p>Tabela <i>configuration</i>:</p> <ol style="list-style-type: none"> 4. Inserir nome da tabela de dados operacionais 5. Inserir nome da tabela de combinações 6. Inserir nome da tabela onde são inseridos os <i>Findings</i>
Resultado	OK

Tabela 40. Teste de aceitação - T_02.

T_02 Extrair dados	
Sumário	Processo em que o sistema extrai os dados da base de dados
Pré-condições	<ul style="list-style-type: none"> • Ligação à base de dados • Tabela de combinações populada com dados
Descrição	<ol style="list-style-type: none"> 1. Sistema faz combinações dos dados parametrizados nas tabelas 2. Sistema cria as <i>queries</i> 3. Sistema executa as <i>queries</i> 4. Sistema guarda os dados retornados pela base de dados
Resultado	OK

Tabela 41. Teste de aceitação - T_03.

T_03 Detetar <i>outliers</i>	
Sumário	Processo em que o sistema deteta <i>outliers</i> no conjunto de dados
Pré-condições	<ul style="list-style-type: none"> • Conjunto de dados já extraído e pronto a ser analisado
Descrição	<ol style="list-style-type: none"> 1. Aplicação do algoritmo MAD ao conjunto de dados de comparação 2. Algoritmo MAD retorna 3 valores – limite mínimo, mediana e limite máximo 3. Os dados do conjunto de análise são comparados com o limite mínimo e máximo, a fim de verificar se são <i>outliers</i>.
Resultado	OK

Tabela 42. Teste de aceitação - T_04.

T_04 Estabelecer <i>baseline</i>	
Sumário	Processo em que o sistema determina a <i>baseline</i> do conjunto de dados.
Pré-condições	<ul style="list-style-type: none"> • <i>Outliers</i> detetados
Descrição	<ol style="list-style-type: none"> 1. Recebe os dados que foram extraídos da BD 2. Determina a <i>baseline</i> do conjunto de dados
Resultado	OK

Tabela 43. Teste de aceitação - T_05.

T_05 Priorizar <i>outliers</i>	
Sumário	Processo de priorização dos <i>outliers</i> , do mais prioritário para o menos prioritário.
Pré-condições	<ul style="list-style-type: none"> • <i>Outliers</i> detetados • <i>Baseline</i> calculada
Descrição	<ol style="list-style-type: none"> 1. Recebe o conjunto de <i>outliers</i> 2. Prioriza os <i>outliers</i> conforme a sua urgência e importância
Resultado	OK

Tabela 44. Teste de aceitação - T_06.

T_06 Classificar <i>outliers</i>	
Sumário	Processo de classificação de um <i>outlier</i> como “positivo” ou “negativo”.
Pré-condições	<ul style="list-style-type: none"> • Colunas referentes à classificação populadas com dados
Descrição	<ol style="list-style-type: none"> 1. Acede à tabela de combinações para ir buscar o tipo de classificação esperada 2. Verifica se o <i>outlier</i> está acima ou abaixo da <i>baseline</i> 3. Classifica o <i>outlier</i> dependendo da sua posição em relação à <i>baseline</i> e da classificação esperada inserida na tabela de combinações.
Resultado	OK

Tabela 45. Teste de aceitação - T_07.

T_07 Traduzir <i>outliers</i>	
Sumário	Processo de tradução dos <i>outliers</i> num <i>Finding</i> – em formato textual.
Pré-condições	<ul style="list-style-type: none"> • Coluna referente à tradução populada com dados
Descrição	<ol style="list-style-type: none"> 1. Acede à tabela de combinações e guarda a frase em que o <i>Finding</i> será traduzido 2. Preenche os espaços vazios da frase com as variáveis estipuladas pelo administrador
Resultado	OK

Tabela 46. Teste de aceitação - T_08.

T_08 Exportar <i>Findings</i>	
Sumário	Processo de inserção dos <i>Findings</i> na base de dados e da apresentação dos mesmos através de um <i>web service</i> .
Pré-condições	<ul style="list-style-type: none"> • <i>Finding</i> criado • Ligação à BD
Descrição	<ol style="list-style-type: none"> 1. Conectar à base de dados 2. Inserir os <i>Findings</i> na tabela de <i>output</i> 3. Disponibilizar os <i>Findings</i> através de um <i>web service</i>

Resultado	OK
------------------	----

8 Conclusões e Trabalho futuro

Este capítulo marca o final do estágio na Novabase e inicia-se com uma análise do trabalho realizado durante 9 meses. Depois serão apresentadas algumas sugestões sobre o trabalho que ainda pode ser feito, tais como novas funcionalidades, a fim de melhorar o produto desenvolvido.

8.1 Análise do trabalho realizado

O objetivo principal deste estágio era desenvolver um produto que agilizasse o processo de análise de informação de uma empresa de telecomunicações. O facto de este produto poder vir a ser integrado noutras empresas para além do nosso cliente, fez com que este fosse desenhado para ser o mais parametrizável possível, garantindo que se adapte às necessidades de cada cliente/negócio.

Um fator fulcral que contribuiu para o sucesso do produto foi a metodologia de desenvolvimento utilizada. A abordagem por *drops* permitiu uma maior margem de erro, uma vez que o produto que foi entregue no final da primeira *drop* foi alvo de análise por uma equipa com vários elementos da Novabase e foram sugeridas algumas melhorias. Subsequentemente essas sugestões foram implementadas no produto, garantindo que este ia de encontro com os requisitos do cliente e com os objetivos propostos. Esta metodologia iterativa permitiu, portanto, identificar antecipadamente os problemas que surgiram e corrigi-los nas *drops* seguintes.

Finalmente é importante realçar que os objetivos propostos foram cumpridos e que o produto desenvolvido tem um grande contributo para a indústria das telecomunicações, mais concretamente para o nosso cliente. Esta aplicação, que pode ser embebida em qualquer sistema de suporte à decisão, permite rapidamente tirar ilações sobre o negócio bem como sobre a prestação dos colaboradores de uma empresa. A maior vantagem deste sistema é que garante uma poupança de tempo significativa no que toca à análise do negócio, uma vez que a informação é digerida pelo próprio sistema e as frases textuais apresentadas são mais facilmente interpretadas que os tradicionais gráficos.

8.2 Trabalho futuro

Apesar do produto desenvolvido cumprir todos os objetivos propostos, existem algumas funcionalidades que o podem melhorar.

A primeira funcionalidade que pode vir a ser implementada é um mecanismo que permita que o *Fidings Engine* faça análises de dados bidimensionais. Apesar de o foco do cliente serem análises unidimensionais, a combinação entre duas ou até mais métricas pode ser uma mais-valia para o negócio. Como foi referido na secção 6.1.1, a análise num conjunto de dados com métricas combinadas pode dar a conhecer ao cliente novos *Findings* que tenham um grande impacto na atividade da empresa.

Outro aspeto que pode ser melhorado no futuro é o *web service*. Atualmente o *web service* permite apenas a chamada a um método. Quando esse método é chamado através do *browser*,

todos os *Findings* parametrizados na tabela de combinações são executados. Desta forma, se assumirmos que os *Findings* parametrizados devem ser executados diariamente, todos os dias tem de ser feita uma chamada ao método do *web service*. No futuro podem ser adicionado um método que permita definir com que frequência os *Findings* devem ser produzidos, ou até um métodos que permita executar um *Finding* específico que esteja parametrizado na base de dados. Apesar do *web service* garantir a interoperabilidade do sistema, existe uma grande margem de progressão no que toca às funcionalidades do mesmo.

9 Referências

1. Peter G. W. Keen, Michael S. Scott Morton, *Decision Support Systems: An Organizational Perspective*; Disponível em: <https://books.google.pt>
2. Udo Richard Averweg, *Historical Overview of Decision Support Systems (DSS)*; Disponível em: <http://www.irma-international.org/viewtitle/13813/>; Acesso em: 11/12/2015
3. Daniel J. Power, *Examples of Decision Support Systems (DSS) Aiding Business Decision-Making*; Disponível em: <http://searchbusinessanalytics.techtarget.com/tutorial/How-decision-support-systems-DSS-can-help-business-decision-making>; Acesso em: 11/12/2015
4. NIST/SEMATECH e-Handbook of Statistical Methods, *What are Outliers in the Data?*; Disponível em: <http://www.itl.nist.gov/div898/handbook/>; Acesso em: 11/12/2015
5. Maimon O. e Rockach L. (Eds.), *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*; Disponível em: <http://www.eng.tau.ac.il/~bengal/outlier.pdf>; Acesso em: 11/12/2015
6. Charu C. Aggarwal, *Outlier Analysis, 2013*; Disponível em: <http://charuaggarwal.net/outlierbook.pdf>; Acesso em: 11/12/2015
7. Victoria J. Hodge e Jim Austin, *A Survey of Outlier Detection Methodologies*, 2004; Disponível em: <http://eprints.whiterose.ac.uk/767/1/hodgevj4.pdf>; Acesso em: 11/12/2015
8. *Statistical Rejection of “Bad” Data – Chauvenet’s Criterion*; Disponível em: <http://www.ohio.edu/people/bayless/seniorlab/chauvenet.pdf>; Acesso em: 11/12/2015
9. Stephen M. Ross, *Peirce's Criterion for the Elimination of Suspect Experimental Data*, 2003; Disponível em: <https://www.eol.ucar.edu/system/files/piercescriterion.pdf>; Acesso em: 11/12/2015
10. S. L. R. Ellison, Trevor J. Farrant e Vicki Barwick, *Practical Statistics for the Analytical Scientist: A Branch Guide*; Disponível em: <https://books.google.pt/books?id=buXUWfo7gngC&printsec=frontcover#v=onepage&q&f=false>; Acesso em: 11/12/2015
11. *Dixon's Q-test: Detection of a Single Outlier*; Disponível em: http://www.chem.uoa.gr/applets/AppletQtest/Applet_Qtest2.html; Acesso em: 14/12/2015
12. Cláudia Pascoal, M. Rosário de Oliveira, Rui Valadas, Peter Filzmoser, Paulo Salvador e António Pacheco, *Robust Feature Selection and Robust PCA for Internet Traffic Anomaly Detection*;
13. Subhajit Dutta, Anil K. Ghosh e Probal Chaudhuri, *Some Intriguing Properties Of Tukey's Half-Space Depth*, 2011; Disponível em: <http://arxiv.org/pdf/1201.1171.pdf>; Acesso em: 14/12/2015

14. Ivy Olga Kwok, *An Efficient and Effective Computation of 2-dimensional Depth Contours*, 1999; Disponível em: <https://open.library.ubc.ca/cIRcle/collections/ubctheses/831/items/1.0051489>; Acesso em: 14/12/2015
15. Jiawei Han, Micheline Kamber e Jian Pei, *Data Mining: Concepts and Techniques Third Edition*; Disponível em: [http://www.cse.hcmut.edu.vn/~chauvtm/data_mining/Texts/\[1\]%20Data%20Mining%20-%20Concepts%20and%20Techniques%20\(3rd%20Ed\).pdf](http://www.cse.hcmut.edu.vn/~chauvtm/data_mining/Texts/[1]%20Data%20Mining%20-%20Concepts%20and%20Techniques%20(3rd%20Ed).pdf); Acesso em: 14/12/2015
16. Gustavo H. Orair, Carlos H. C. Teixeira, Wagner Meira Jr., Ye Wang e Srinivasan Parthasarathy, *Distance Based Outlier Detection: Consolidation and Renewed Bearing*; Disponível em: <http://www.vldb.org/pvldb/vldb2010/papers/I09.pdf>; Acesso em: 14/12/2015
17. Yihua Liao e V. Rao Vemuri, *Use of K-Nearest Neighbor Classifier for Intrusion Detection*; Disponível em: <http://web.cs.ucdavis.edu/~vemuri/papers/knn-ss02.pdf>; Acesso em: 14/12/2015
18. Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng e Jörg Sander, *LOF: Identifying Density-Based Local Outliers*; Disponível em: <http://people.cs.vt.edu/badityap/classes/cs6604-Fall13/readings/breunig-2000.pdf>; Acesso em: 14/12/2015
19. Hans-Peter Kriegel, Matthias Schubert e Arthur Zimek, *Angle-Based Outlier Detection in High-dimensional Data*; Disponível em: <http://www.dbs.ifi.lmu.de/~zimek/publications/KDD2008/KDD08-ABOD.pdf>; Acesso em: 14/12/2015
20. Brian S. Everitt, Sabine Landau e Morven Leese, *Cluster Analysis*; Disponível em: <http://serverlib.moe.gov.ir/documents/10157/42675/Cluster+Analysis.pdf>; Acesso em: 14/12/2015
21. Sanjay Chawla e Aristides Gionis, *K-means: A Unified Approach to Clustering and Outlier Detection*; Disponível em: <http://pmg.it.usyd.edu.au/outliers.pdf>; Acesso em: 14/12/2015
22. Hans-Peter Kriegel, Peer Kröger e Arthur Zimek, *Outlier Detection Techniques*, 2010; Disponível em: <https://www.siam.org/meetings/sdm10/tutorial3.pdf>; Acesso em: 14/12/2015
23. Ching-an Hsiao, *On Classification from Outlier View*, 2010; Disponível em: <https://arxiv.org/ftp/arxiv/papers/0907/0907.5155.pdf>; Acesso a: 08/04/16;
24. Edgar Acuña e Caroline Rodriguez, *On detection of outliers and their effect in supervised classification*; Disponível em: <http://academic.uprm.edu/~eacuna/vene31.pdf>. Acesso a: 08/04/16

25. Dwight D. Eisenhower, [Address at the Second Assembly of the World Council of Churches](#); Disponível em: <http://www.presidency.ucsb.edu/ws/?pid=9991>. Acesso a: 14/06/16