ISEL – Instituto Superior de Engenharia de Lisboa

DEETC – Departamento de Engenharia Eletrónica e Telecomunicações e de Computadores

MEET

Mestrado em Engenharia de Eletrónica e Telecomunicações

Dissertação

Manutenção Preventiva na Gestão de Falhas em Redes Móveis usando Machine Learning

MÁRCIO FILIPE GODINHO PEREIRA

Orientadores	
Professor Doutor	Pedro Manuel A. C. Vieira
Mestre	David E. C. Duarte (externo, CELFINET)

Júri	
Presidente:	*Professor Especialista* Nuno Cota
Vogais:	*Professor Especialista* Luís Mata *Professor Doutor* Pedro Vieira

Fevereiro, 2022

Resumo

A gestão de falhas numa rede móvel pode tirar partido de algoritmos de *Machine Learning* (ML), tornando a sua manutenção mais proativa e preventiva. Atualmente, os centros de operações (NOCs) em redes móveis ainda funcionam em modo reativo, onde o diagnóstico e resolução de problemas apenas é feito após a ocorrência dos mesmos. A evolução para uma manutenção preventiva da rede possibilita a prevenção ou a rápida resolução do problema, levando a uma maior disponibilidade da rede e dos serviços, mais eficiência e uma maior satisfação do cliente.

O principal objetivo deste trabalho passa por criar uma solução para a manutenção preventiva de alarmes capaz de: detetar grupos de alarmes e as relações entre eles, formando regras de associação; aprender continuamente a partir de novos dados e torne-se cada vez mais eficaz com a experiência que desenvolve no domínio da manutenção; definir os alarmes antecedente e consequente num padrão sequencial, em que são ordenados cronologicamente; reconhecer os padrões mais frequentes de modo a realçar quais são as falhas mais preocupantes e quais os ganhos que advêm da sua prevenção.

Para isso, são explorados diferentes algoritmos de reconhecimento de padrões de alarmes e definição de regras de associação entre os mesmos, com recurso a dados reais de *Fault Management* (FM) de uma rede *Long Term Evolution* (LTE). Foi elaborada uma análise comparativa do desempenho dos algoritmos, tendo-se verificado uma diminuição de 3.31% no número total de alarmes e 70.45% no que toca ao número de alarmes do mesmo tipo. Observaram-se ainda reduções no número de alarmes por nó da rede, identificando-se 39 nós que deixaram de ter qualquer alarme por resolver.

Estes resultados demonstram que o reconhecimento de padrões sequenciais permite uma manutenção preventiva da rede do operador de redes móveis.

Palavras-chave: gestão de falhas, *machine learning*, manutenção preventiva, reconhecimento de padrões sequenciais.

Abstract

Mobile networks' fault management can take advantage of Machine Learning (ML) algorithms, making its maintenance more proactive and preventive. Currently, Network Operations Centers (NOCs) still operate in reactive mode, where the troubleshoot is only done after the problem identification. The evolution to a preventive maintenance enables the problem prevention or quick resolution, leading to a greater network and services availability, to a better operational efficiency and, above all, ensures customer satisfaction.

The main objective of this work is to create a solution for the preventive maintenance of mobile networks' alarms capable of: detecting groups of alarms and the relationships between them, forming association rules; continuously learn from new data and become increasingly effective with the experience it develops in the maintenance field; define the antecedent and consequent alarms in a sequential pattern, in which they are ordered chronologically; recognize the most frequent patterns in order to highlight which are the most worrying faults and what gains come from their prevention.

In this work, different algorithms for Sequential Pattern Mining (SPM) and Association Rule Learning (ARL) are explored, using real Fault Management (FM) data from a live Long Term Evolution (LTE) network. A comparative performance analysis between all the algorithms was carried out, having observed a decrease of 3.31% in the total number of alarms and 70.45% in the number of alarms corresponding to the same type. There were also considerable reductions in the number of alarms per network node or zone, identifying 39 nodes that no longer had any unresolved alarm.

These results demonstrate that the recognition of sequential alarm patterns allows the preventive maintenance of a mobile communications network.

Keywords: fault management, machine learning, preventive maintenance, sequential pattern mining.

Agradecimentos

Este trabalho foi realizado no âmbito do projeto internacional Artificial Intelligence for Green Networks (AI4GREEN) C2018/1-5, ao abrigo do CELTIC-NEXT Core Group e do programa EUREKA Clusters. Gostaria de deixar um agradecimento ao programa COMPETE/FEDER pelo financiamento da componente nacional do projeto (16/SI/2019), bem como ao Instituto de Telecomunicações (IT) pelo suporte do mesmo.

Quero ainda deixar umas palavras de gratidão aos meus orientadores pela ajuda e suporte que me ofereceram na elaboração da tese.

Dedicado à minha família mais próxima, especialmente aos meus pais, pela paciência que tiveram e o apoio que me deram ao longo destes meses.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos

ALD *Antenna Line Device*

ANR *Automatic Neighbour Relation*

ARL *Association Rule Learning*

BAM *Back Administration Module*

BFS *Breadth-First Search*

C-SON *Centralized SON*

CapEx *Capital Expenditure*

CCO *Coverage and Capacity Optimization*

CEM *Customer Experience Management*

CGI *Cell Global Identifier*

CIO *Cell Individual Offset*

CQI *Channel Quality Indicator*

D-SON *Distributed SON*

DFS *Depth-First Search*

DID *Deployment IDentifier*

DNN *Deep NN*

DT *Decision Tree*

E-UTRAN *Evolved Universal Terrestrial Radio Access Network*

eNB *Evolved Node B*

ES *Energy Saving*

FM *Fault Management*

GLM *Generalized Linear Model*

HII *High Interference Indicator*

ICIC *Inter-Cell Interference Coordination*

ISO *International Organization for Standardization*

k-NN *k-Nearest Neighbors*

LTE *Long Term Evolution*

ML *Machine Learning*

MLB *Mobility Load Balancing*

MO *Management Object*

MRO *Mobility Robustness Optimization*

MTBF *Mean Time Between Failure*

MTTR *Mean Time To Repair*

NB *Naïve Bayes*

NE *Network Element*

NMS *Network Management System*

NN *Neural Network*

NOC *Network Operations Center*

NR *Neighbour Relation*

NRT *NR Table*

OAM *Operations, Administration and Maintenance*

OI *Overload Indicator*

OMC *Operation and Maintenance Center*

OpEx *Operational Expenditure*

PCI *Physical Cell Identifier*

PDCCH *Physical Downlink Control Channel*

PDSCH *Physical Downlink Shared Channel*

PLMN *Public Land Mobile Network*

PM *Performance Management*

PUSCH *Physical Uplink Shared Channel*

QoE *Quality of Experience*

QoS *Quality of Service*

RACH *Random Access Channel*

RAE *Remote Antenna Extension*

RAT *Radio Access Technology*

RET *Remote Electrical Tilt*

RF *Random Forest*

RL *Reinforcement Learning*

RLF *Radio Link Failure*

RNTP *Relative Narrowband Transmit Power*

ROI *Return On Investment*

RSRP *Reference Signal Received Power*

RSRQ *Reference Signal Received Quality*

SASU *Same-band Antenna Sharing Unit*

SCTP *Stream Control Transmission Protocol*

SINR *Signal-to-Interference-plus-Noise Ratio*

SL *Supervised Learning*

SLA *Service Level Agreement*

SOC *Service Operations Center*

SON *Self-Organizing Network*

SPM *Sequential Pattern Mining*

SQM *Service Quality Management*

SV *Single Value*

SVM *Support Vector Machine*

TMA *Tower Mounted Amplifier*

TTM *Time to Market*

TTT *Time to Trigger*

UE *User Equipment*

UL *Unsupervised Learning*

VSWR *Voltage Standing Wave Ratio*

X2AP *X2 Application Protocol*

Capítulo 1

Introdução

Este capítulo apresenta a motivação, os principais objetivos, alguns trabalhos relacionados e a estrutura do trabalho elaborado.

1.1 Motivação

A otimização da gestão de falhas numa rede móvel pode tirar partido de técnicas de *Machine Learning* (ML) para tornar a manutenção operacional da rede mais proativa e preventiva.

Os *Network Operations Centers* (NOCs) dos operadores de redes móveis ainda operam em modo reativo, isto é, o diagnóstico e resolução de problemas começam apenas depois de ocorrer uma falha na rede, um serviço ser afetado ou existir uma insatisfação do cliente. Os engenheiros têm acesso a imensas informações sobre a rede, mas carecem de uma forma rápida e eficaz para diagnosticar e resolver os problemas. Assim, o *Mean Time To Repair* (MTTR) é prejudicado, levando a uma menor disponibilidade da rede e dos serviços, a uma pior eficiência operacional e, por conseguinte, à insatisfação do utilizador final. Num cenário típico de um NOC reativo, o processo que o engenheiro tem que realizar para resolver um problema é ineficiente e longo, pelo que a evolução para um NOC proativo é necessária para acelerar este processo.

Com uma solução para a manutenção preventiva da rede, os operadores podem servir-se de algoritmos de ML para reduzir custos operacionais,

melhorar a disponibilidade da rede e dos serviços, garantir a satisfação do cliente, e reduzir os *Service Level Agreements* (SLAs) perdidos. Progredir em direção a uma gestão da qualidade de serviço (*Service Quality Management* (SQM)) centrada no cliente, ser proativo e prevenir que problemas ocorram em primeiro lugar, é sem dúvida a melhor estratégia. Esta estratégia encontra-se enquadrada no âmbito das *Self-Organizing Networks* (SONs).

Uma SON é uma rede autónoma capaz de se configurar, otimizar e regenerar a ela própria. Com a implementação de uma SON, é possível reduzir os principais custos que advêm das operações na rede e, ao mesmo tempo, melhorar o seu desempenho global. Além disso, como não requer grande envolvimento humano, é possível concentrar esforços de modo a reduzir o *Time to Market* (TTM) e aumentar a diversidade de serviços disponíveis. Por forma a traçar o comportamento e costumes dos clientes para que sejam criados serviços personalizados, feitos à medida de cada utilizador, é possível tirar partido de técnicas de *Machine Learning* (ML).

ML é uma das formas de inteligência artificial, que permite extrair informações relevantes escondidas nos dados históricos, e aprender continuamente com a experiência adquirida. Existem diversos algoritmos de ML mas, regra geral, podem ser divididos em três categorias: *Supervised Learning* (SL), *Unsupervised Learning* (UL) e *Reinforcement Learning* (RL). Dentro destas categorias, existem diferentes tipos de algoritmos que podem ser úteis à gestão de falhas preventiva, nomeadamente, os de *Association Rule Learning* (ARL) e os de *Sequential Pattern Mining* (SPM).

Os algoritmos de ARL conseguem extrair correlações interessantes escondidas nos dados e, com isso, definir regras de associação que podem ser mais ou menos frequentes tendo em conta as métricas que servem para as avaliar. Contudo, este tipo de algoritmos não tem em consideração a ordem temporal dos dados e, por isso, não permite reconhecer a sequência dos eventos que ocorreram. Já os algoritmos de SPM permitem descobrir padrões sequenciais, onde a ordem dos itens já é considerada e é essencial para a prevenção de alarmes sequenciais numa rede de comunicações móveis.

Alguns dos trabalhos relacionados com a gestão de falhas preventiva podem ser encontrados em: [1], [2], [3]. Em [1] é abordada a gestão proativa de alarmes como forma de tirar partido de algoritmos de ML para acompanhar a

evolução das operações em redes e serviços móveis. Em [2] e [3] é realizada a pesquisa sobre a aplicação de técnicas de ML e mineração de dados na gestão de falhas em redes de telecomunicações, de um ponto de vista operacional.

1.2 Objetivos

Esta secção define o principal objetivo do trabalho - criar uma solução que conduz à manutenção preventiva de alarmes de redes móveis que:

1. Deteta grupos de alarmes e as relações entre eles, formando regras de associação;
2. Aprenda continuamente a partir de novos dados e torne-se cada vez mais eficaz com a experiência que desenvolve no domínio da manutenção;
3. Define os alarmes antecedente e consequente num padrão sequencial, em que são ordenados cronologicamente;
4. Reconhece os padrões mais frequentes de modo a realçar quais são as falhas mais preocupantes e quais os ganhos que advêm da sua prevenção.

Neste trabalho foram utilizados dados reais de operadores de redes móveis, tendo sido executado em parceria com a empresa CELFINET[1].

1.3 Estrutura do relatório

Esta secção apresenta sucintamente a forma como a tese está organizada e que tópicos são estudados e discutidos nos próximos capítulos.

No Capítulo 2 é realizado o Estado da Arte, onde são apresentados os conceitos estudados na teoria essenciais ao desenvolvimento prático do trabalho. É feita uma introdução sobre as Operações em Redes Móveis e as SONs e é realizado um estudo sobre os principais tipos de ML e os algoritmos utilizados em cada uma das Áreas de Gestão da Rede.

[1] https://www.celfinet.com/

No Capítulo 3 é feito o estudo e análise de dados reais sobre registos de alarmes numa rede de telecomunicações móvel. Primeiramente é feita a descrição dos alarmes e depois a análise exploratória dos dados utilizados.

No Capítulo 4 são estudados alguns algoritmos de ARL e de SPM e, ainda, formas de os avaliar.

No Capítulo 5 são apresentados os resultados experimentais da implementação de todos os algoritmos abordados, sendo feita a comparação ao nível da eficiência, e no que diz respeito aos algoritmos de cada tipo.

O Capítulo 6 conclui o trabalho e refere alguns aspetos para trabalho futuro.

1.4 Publicações

No contexto desta tese, foi publicado o seguinte artigo científico:

- "Manutenção Preventiva na Gestão de Falhas em Redes Móveis utilizando *Machine Learning*", escrito por Márcio Pereira, David Duarte e Pedro Vieira.

Este artigo foi submetido ao *Best Student Paper Award*, patrocinado pela ANACOM, e apresentado no 15.º Congresso do Comité Português da URSI, com o tema "Sustentabilidade ambiental no uso do espectro radioelétrico". Foi também submetida uma versão alargada deste mesmo artigo para o *Radio Science Bulletin* da URSI.

Capítulo 2

Estado da Arte

Neste capítulo serão apresentados os principais conceitos teóricos necessários ao desenvolvimento da dissertação. Em primeiro lugar, será feita uma introdução sobre as Operações em Redes Móveis e as *Self-Organizing Networks*. De seguida, é feito o estudo sobre os principais tipos e algoritmos de *Machine Learning*, e em que Áreas de Gestão da Rede podem ser utilizados.

2.1 Operações em Redes Móveis

As operações de uma rede móvel, como a monitorização, o controlo e a gestão da rede, são exercidas no NOC. Os operadores de telecomunicações podem ter mais do que um NOC, seja para monitorizarem diferentes elementos da rede, seja para haver redundância geográfica na ocorrência de algum evento que deixe a sede inoperável. Para além de supervisionarem toda a infraestrutura da rede, também ficam a par de outras condições externas para se acautelarem de eventuais disrupções que possam comprometer o estado saudável da rede, como o tráfego excessivo e anormal na rede ou incidentes que possam ter ocorrido nalgum dispositivo do sistema.

Genericamente, os principais objetivos de um NOC são:

1. Assegurar a resposta imediata e eficaz na resolução e seguimento de ocorrências, ao atentar certas mudanças na rede que possam causar degradação de serviço, como faltas de energia, falhas na rede ou outras anormalidades;

2. Garantir o cumprimento dos níveis de serviços prestados, isto é, a conformidade com os *Service Level Agreements* (SLAs) definidos entre o operador e os seus clientes, para que estes tenham total disponibilidade de serviço sempre que precisem.

Os NOCs tradicionais ainda operam em modo reativo e, por isso, focam-se principalmente na supervisão da rede, assumindo que se a rede está operacional, os serviços também estarão sem problemas, o que é verdade até certo ponto. Os operadores precisam de uma rede saudável para poderem oferecer um serviço de qualidade, no entanto, as ferramentas tradicionais para a gestão da mesma não são capazes de identificar os principais indicadores de degradação da rede, que primeiro manifestam-se como degradação de serviços e má experiência de utilização. Os engenheiros do NOC têm acesso a imensos indicadores e dados de *performance* tal como alarmes, medidas de desempenho, topologia da rede, dados transacionados na rede e serviços, entre outros, mas carecem de uma forma rápida e eficaz para diagnosticar e resolver os problemas. Os operadores também tentam tirar partido de dados históricos para ajudar na resolução de incidentes recorrentes, mas muitas vezes sem sucesso.

Num NOC tradicional que funcione em modo reativo, um operador tem de passar por um conjunto extenso e moroso de procedimentos para solucionar um problema, o que torna a resolução do problema ineficiente. Poucos são os eventos que despoletam alarmes que são tratados imediatamente, pelo que a maioria das avarias requerem investigações extensivas, tanto em termos de tempo como de recursos [4]. Devido a esta inaptidão para corrigir as anomalias rapidamente, o MTTR e o *Mean Time Between Failure* (MTBF) são prejudicados, levando a uma menor disponibilidade da rede e dos serviços, a uma pior eficiência operacional e, por conseguinte, à insatisfação do cliente.

Num cenário típico, o operador do NOC deve:

1. Abrir manualmente um *ticket* de problema para registar o evento;
2. Pesquisar em *knowledge databases* por casos similares, podendo ou não encontrar a solução;
3. Se não encontrar uma solução efetiva, deve procurar ajuda noutro tipo de documentos, tais como os manuais de fabricantes dos dispositivos instalados na rede;
4. Além de tudo isso, precisa de verificar métricas de desempenho para avaliar se houve degradação no funcionamento do componente em causa, ou se o problema advém de um componente vizinho, estando informado de atividades de manutenção paralelas que possam estar a afetar o serviço;
5. Por fim, soluciona o problema caso tenha achado a solução, ou remete o mesmo para um superior diagnosticar.

Este processo pode ser longo e tedioso, o que afeta negativamente a disponibilidade do serviço assim como a satisfação do cliente. Além disso, devido aos avanços tecnológicos, a rede de telecomunicações num todo tornou-se cada vez mais complexa, ao ponto do ser humano não ser capaz de lidar com esta complexidade acrescida sem afetar o desempenho e a eficiência do serviço. Não é então surpresa que os operadores estejam a explorar novas tecnologias para transformarem as operações de uma rede móvel, no sentido de se tornarem mais autónomas e inteligentes. Para atingir esse objetivo, é necessário amadurecer os NOCs de modo a torná-los proativos, para que possam evitar a degradação dos serviços ou, pelo menos, prevê-la para que seja corrigida imediatamente, não atingindo a rede e sendo transparente para o utilizador final. A maturação do NOC passa por implementar ferramentas para a gestão da qualidade de serviço, SQM, e gestão da experiência de consumidor, *Customer Experience Management* (CEM), de modo a evoluí-lo e transformá-lo num *Service Operations Center* (SOC) centrado no cliente. Juntamente com a computação cognitiva, é possível criar uma solução que aprenda por si própria e, com isso, se torne melhor ao longo do tempo, e ainda

possa ser treinada para, a qualquer momento, apresentar recomendações e até mesmo soluções prontas a serem postas em prática.

Adotando uma solução para um NOC cognitivo, os operadores podem servir-se de técnicas de Inteligência Artificial e de *Machine Learning*, em conjunto com métodos de análise e estatística avançada, para que as próprias redes se possam otimizar continuamente. Para isso, é necessário haver uma alta correlação entre os incidentes ocorridos e os alarmes, e ser capaz de reconhecer as melhores decisões a tomar com base em previsões ou padrões encontrados. A aprendizagem contínua permitirá a adaptação rápida e eficiente a mudanças na rede, sendo possível alocar dinamicamente recursos a cada cliente dependendo das suas exigências. Com isto é possível reduzir custos operacionais, melhorar a disponibilidade da rede e dos serviços e reduzir os SLAs perdidos, levando a um maior *Return On Investment* (ROI) e a uma melhor experiência do consumidor. Progredir em direção a uma gestão da qualidade de serviço centrada no cliente, ser proativo e prevenir problemas, é a melhor estratégia.

2.2 Redes Auto-Organizáveis

Uma SON é uma rede inteligente e autónoma, capaz de se configurar, otimizar e regenerar a ela própria, sendo um conceito inovador que permite impulsionar as atividades de *Operations, Administration and Maintenance* (OAM).

O principal objetivo de uma SON é reduzir os custos relacionados com as operações da rede, nomeadamente o *Capital Expenditure* (CapEx) e o *Operational Expenditure* (OpEx), ao minimizar a intervenção humana ao mesmo tempo que melhora o desempenho da rede, isto é, em termos de cobertura e capacidade tal como qualidade do serviço prestado.

A introdução de SONs por parte dos operadores permite dar resposta à crescente complexidade das novas e futuras tecnologias de acesso rádio (*Radio Access Technology* (RAT)) ao integrar o planeamento, a configuração e a otimização da rede móvel num único processo automatizado que, em grande parte, não requer o envolvimento dos técnicos. Existem cada vez mais parâmetros de rede que podem ser ajustados, mas as dependências entre eles começam a ser bastante complexas, ao ponto de deixar de ser viável o ajuste manual. Além disso, devido ao número acrescido de terminais móveis, é necessário reduzir o TTM e assim aumentar a diversidade de serviços oferecidos para responder aos requisitos dos clientes, que estão cada vez mais exigentes.

2.2.1 Tipos de Arquiteturas

Existem dois tipos de arquiteturas principais de SON: *Centralized SON* (C-SON) e *Distributed SON* (D-SON). As suas implementações vão desde as funcionalidades C-SON, onde os algoritmos que permitem a auto-organização da rede se encontram no sistema de gestão da rede, quer seja no *Operation and Maintenance Center* (OMC) ou no *Network Management System* (NMS), até a uma solução D-SON onde as funções SON estão distribuídas, no plano de controlo, tipicamente pelas estações-base. Por um lado, num cenário C-SON, consideram-se os dados vindos de todos os nós da rede para identificar e tratar os problemas. Esta solução pode ser mais robusta face às instabilidades

da rede causadas pelo conflito entre a operação de várias funções SON, uma vez que o controlo é centralizado.
No entanto, os sistemas centralizados tendem a ser lentos na resposta, ainda mais com o crescente número de nós nas redes. Por outro lado, as funcionalidades D-SON estão desenhadas para responderem em quase tempo-real, o que torna as funções SON altamente dinâmicas, permitindo que a rede se adapte rapidamente a mudanças de comportamento locais. Os algoritmos D-SON são executados em nós individuais localizados no *edge* da rede, que trocam informações diretamente entre si sem necessidade de comunicar com um ponto central. Cada nó pode iniciar processos SON e tomar decisões de otimização independentemente ou em coordenação com outros nós. A principal desvantagem quando comparado com uma implementação C-SON é que é bastante vulnerável a instabilidades da rede causadas por operações concorrentes entre funções SON. Ainda existem soluções híbridas que operam algumas funções de forma centralizada, mas as mais críticas, que requerem tempos de resposta mais rápidos são executadas nos nós distribuídos pela rede.

2.2.2 Auto-Configuração

A auto-configuração é a operação de introduzir um novo elemento na rede com o mínimo de intervenção dos técnicos, tipicamente feita nas fases de planeamento e execução da rede móvel [5]. Os algoritmos de auto-configuração gerem toda a parte da configuração de uma estação-base. Aquando da sua ativação, a estação-base deteta a ligação de transporte e estabelece uma conexão com os vários elementos da rede. De seguida, estabelece as ligações OAM, S1 e X2 para finalmente passar para o modo operacional. Depois da sua configuração, a estação-base realiza um auto-teste para elaborar um relatório sobre o seu estado que irá ser entregue ao nó de gestão da rede.
Dentro do conjunto de funções SON dedicadas à auto-configuração tem-se, por exemplo, a *Automatic Neighbour Relation* (ANR) e a atribuição automática do *Physical Cell Identifier* (PCI) [6, 7].

- ANR: Esta função está presente na estação-base e gere a *NR Table* (NRT). É capaz de detetar novas células vizinhas e adicioná-las à NRT

ou identificar as *Neighbour Relations* (NRs) desatualizadas que devem ser removidas. Ao consultar as suas NRs, a estação-base que controla a célula de origem conhece o *Cell Global Identifier* (CGI) e o PCI da célula de destino e, por isso, consegue identificá-la univocamente. O *User Equipment* (UE), a partir das medições que faz, consegue identificar os seguintes tipos de células:

1. A célula de serviço - aquela que o serve;
2. As células listadas - aquelas que são indicadas na lista de células vizinhas do *Evolved Universal Terrestrial Radio Access Network* (E-UTRAN);
3. As células detetadas - aquelas que não são indicadas pelo E-UTRAN mas são detetadas pelo próprio UE. Estas células podem ser células *Long Term Evolution* (LTE) a operar quer na mesma frequência da célula servidora ou em frequências diferentes, ou ainda células pertencentes a outras RATs. Para detetar este tipo de células, é necessário que a estação-base servidora comunique ao UE para medir também noutras frequências [8].

A lista de células vizinhas pode ser gerada com base em condições geográficas, padrões de antena ou potências de transmissão [9]. A função ANR pode também usar o *Signal-to-Interference-plus-Noise Ratio* (SINR) das células adjacentes para construir a lista de células vizinhas [10];

- Atribuição automática do PCI: O PCI permite distinguir os sinais provenientes de diferentes estações-base. Em LTE existem 504 PCIs no total, portanto, em redes mais densas, é inevitável a sua reutilização. A função de atribuição automática do PCI faz a gestão da identificação das células de forma autónoma e sem conflitos, ao garantir que todas as células sejam unicamente identificadas na camada física [11].

2.2.3 Auto-Otimização

A auto-otimização é a designação dada ao conjunto de mecanismos que otimizam os parâmetros da rede durante a sua operação, com base nas

medições feitas pelos UEs como, por exemplo, o *Reference Signal Received Power* (RSRP), o *Reference Signal Received Quality* (RSRQ) e o *Channel Quality Indicator* (CQI). As principais funções de auto-otimização são descritas de seguida:

- *Mobility Load Balancing* (MLB): Função SON responsável por equilibrar a carga das células ao transferi-la entre si. O seu principal objetivo é melhorar a experiência do utilizador e atingir maiores capacidades do sistema ao distribuir o tráfego pelos vários recursos rádio existentes na rede. Esta função é tipicamente implementada de forma distribuída, e as mensagens contendo informações úteis à sua operação são transmitidas na interface X2 [12]. O seu processo de funcionamento passa por ajustar o parâmetro *Cell Individual Offset* (CIO). Este contém os *offsets* das células servidora e vizinhas que todos os UEs (dessa célula) devem usar para satisfazer a condição de *handover* A3 [13];

- *Mobility Robustness Optimization* (MRO): Função SON responsável por garantir uma mobilidade adequada, isto é, em modo *connected*, realizar *handovers* quando necessário e, em modo *idle*, fazer a reseleção da célula. O seu objetivo passa pela minimização de: quedas de chamadas; *Radio Link Failures* (RLFs); *handovers* desnecessários, devido a erros na configuração dos seus parâmetros; problemas que ocorrem em modo *idle*. A sua implementação é normalmente distribuída e foca-se na parametrização dos modos *connected* e *idle*. No modo *connected*, é a função MRO que realiza o ajuste dos parâmetros mais importantes no desencadeamento do processo de *handover*, tais como o *offset* do evento A3 ou o *Time to Trigger* (TTT). Em modo *idle*, ajusta valores de *offset* como, por exemplo, o "Qoffset" num cenário de *handover* entre portadoras intra-RAT;

- *Inter-Cell Interference Coordination* (ICIC): Função SON responsável por minimizar a interferência entre células a operar na mesma frequência, ao garantir a coordenação dos recursos físicos entre células vizinhas. Pode funcionar tanto em *uplink* como em *downlink*, nos canais de dados *Physical Downlink Shared Channel* (PDSCH) e *Physical Uplink Shared Channel* (PUSCH), ou no canal de controlo *Physical Downlink Control Channel* (PDCCH). Esta função pode ser estática, semi-estática

ou dinâmica. A ICIC dinâmica ajusta frequentemente os parâmetros, sendo suportada pela sinalização trocada entre células na interface X2. Para suportar uma coordenação entre células proativa, são utilizados os indicadores *High Interference Indicator* (HII) e *Relative Narrowband Transmit Power* (RNTP), enquanto que numa coordenação reativa é utilizado o *Overload Indicator* (OI) [12];

- Otimização do *Random Access Channel* (RACH): Função SON responsável por otimizar os canais de acesso aleatório, com base no *feedback* proveniente dos UEs, principalmente as configurações do RACH dos seus *Evolved Node Bs* (eNBs) vizinhos. Esta otimização pode ser feita ao ajustar o parâmetro de controlo de potência, ou alterar o formato do preâmbulo de modo a reduzir o atraso no acesso [14];

- *Coverage and Capacity Optimization* (CCO): Função SON que se foca na criação de algoritmos que permitam alcançar o compromisso ideal entre cobertura e capacidade da rede. Para além desses, podem ainda ser otimizados os ritmos de transmissão tanto dentro da célula como na sua fronteira, ou uma combinação ponderada de todos eles;

- *Energy Saving* (ES): Função SON responsável por manter a qualidade de experiência do utilizador final, minimizando o impacto energético e ambiental. O seu objetivo passa por otimizar o consumo energético ao reduzir os gastos de energia dos *Network Elements* (NEs) e suspender os nós que não estão a ser usados, quando não realizam tráfego suficiente [15]. É comum ser usada para ligar/desligar os eNBs ou *small cells* de forma eficiente, de modo a garantir níveis satisfatórios de *Quality of Service* (QoS)/*Quality of Experience* (QoE) e a otimizar a energia consumida.

2.2.4 Auto-Regeneração

Os sistemas de comunicações móveis são propensos a falhas, pelo que devem existir mecanismos de reparação e manutenção da rede. Todos os eNBs são responsáveis por servir uma determinada área, com pouca ou mesmo nenhuma redundância, e, quando não são capazes de atingir os níveis de

qualidade de serviço desejados, ocorre degradação no desempenho do serviço prestado aos utilizadores ali conectados, representando uma potencial perda de receitas avultada para o operador. Os métodos de auto-reparação visam regenerar a rede quando se dá uma perda de cobertura ou capacidade devido a falhas nas células [16]. Existem soluções que passam por ajustar automaticamente os parâmetros de rede das células ao redor da célula onde se deu a falha. Quando a falha estiver resolvida, esses parâmetros são restaurados aos seus valores originais. Alguns dos cenários onde este tipo de algoritmos podem ser usados são apresentados de seguida:

- Auto-recuperação do *software* do NE: Se o *software* instalado num dado NE falhar devido à utilização de uma versão ou configuração defeituosa, deve ser assegurado que o NE voltará a funcionar ao remover o *software* em causa ou restaurar a sua configuração;
- Auto-reparação de falhas no *hardware* [17];
- *Cell Outage Management*: Este caso de utilização divide-se em duas partes:
    1. *Cell Outage Detection*: O principal objetivo é detetar interrupções na célula a partir da monitorização dos indicadores de desempenho. Estes indicadores são comparados com perfis pré-estabelecidos, onde se verifica se estão dentro dos limiares caraterísticos de uma célula em bom funcionamento;
    2. *Cell Outage Compensation*: Serve para aliviar o impacto sentido pela perda do serviço induzida por uma falha na célula servidora [18]. Refere-se à mitigação automática da degradação sentida pelo *outage* sofrido, ao ajustar criteriosamente os parâmetros rádio adequados, tais como a potência da piloto e os parâmetros das antenas das células vizinhas. Para este caso é necessário uma reação eficaz para assegurar a continuidade do serviço, pelo que a deteção do *outage* deve ser o mais eficiente possível.

2.3 *Machine Learning*

ML é um tipo de inteligência artificial que permite que uma "máquina" consiga extrair informações a partir dos dados à sua disposição e melhorar continuamente com a experiência adquirida. É utilizado em tarefas complexas, que possam ser difíceis de realizar manualmente pelo ser humano, com a vantagem de ser mais eficaz ajudar a máquina no desenvolvimento do seu próprio algoritmo do que na especificação de cada passo a executar, à semelhança da programação tradicional.

Os algoritmos de ML constroem um modelo com base nos seus *inputs*, conhecidos como "dados de treino", para descobrirem que decisões tomar acerca dos "dados de teste". O conjunto de todos os dados designa-se por *dataset* e só no processo de aprendizagem é que são divididos em conjuntos de treino e de teste. O rácio da divisão (*split*) pode seguir a regra de 80:20 ou 70:30, ou então usar técnicas mais sofisticadas de validação cruzada (*cross validation*). Os *datasets* devem ser representativos da grandeza em estudo, isto é, garantir a qualidade dos dados usados no treino e na validação dos modelos é fundamental, pois utilizar dados não-representativos do ambiente a avaliar pode ter um impacto bastante negativo na precisão dos modelos produzidos. Ter acesso a este tipo de *datasets* é importante, mas nem sempre é fácil devido à sua natureza sensível e confidencial [19].

Algo que também se deve considerar é que para alcançar elevadas precisões, perto dos 100%, o tempo de computação para treinar o modelo de ML poderá ser elevado, podendo até tornar-se incomportável ao ponto de se optar por algoritmos mais rápidos em detrimento da própria precisão do modelo. É então fundamental efetuar a chamada extração de caraterísticas (*feature extraction*) para determinar as melhores métricas a usar no processo de aprendizagem [20], o que reduz a quantidade de dados que são tidos em conta para reduzir o tempo de processamento, sem sacrificar a precisão do modelo, que tipicamente até é melhorada porque assim são ignorados os dados que iriam prejudicá-la.

Existem diversos algoritmos de ML que, dependendo do *dataset* em estudo, podem ser mais ou menos precisos na aprendizagem do modelo relacional entre as variáveis independentes, também chamadas de caraterísticas, e a variável dependente que queremos prever com o treino apropriado. Um dos principais fatores em ML é a forma como os modelos são avaliados. Os vários tipos de modelos que existem têm diversos parâmetros de entrada que são definidos inicialmente e são fixos durante todo o treino, chamados de hiperparâmetros [21]. Durante a aprendizagem, são inferidos outros parâmetros, chamados de parâmetros de modelo ou parâmetros de aprendizagem, que são distintos entre modelos e, por isso, os caracterizam unicamente. O objetivo é determinar os melhores hiperparâmetros que levam aos melhores parâmetros de aprendizagem, por forma a criar o derradeiro modelo que melhor se ajusta aos dados.

Existem três categorias principais de ML, dependendo de como a aprendizagem é feita [22]: *supervised learning*, *unsupervised learning* e *reinforcement learning* que serão descritas de seguida.

2.3.1 Aprendizagem supervisionada

São fornecidos dados de treino com *inputs* e respetivos *outputs* definidos para que a máquina aprenda um modelo, relacionado-os. Tipicamente, SL é usado na resolução de problemas de regressão e de classificação, onde se pretende prever *outputs* contínuos ou discretos, respetivamente.

Os hiperparâmetros de um modelo de SL definem a flexibilidade do modelo, isto é, por exemplo, como o modelo se deve comportar em situações de *under-fitting* e de *over-fitting*. O *under-fitting* ocorre quando o modelo não consegue, de todo, ajustar-se aos dados de treino, o que resulta num grande erro de treino. Quando ocorre, significa que ainda existe algo a melhorar na aprendizagem do modelo. Já o *over-fitting* ocorre quando o modelo se ajusta demasiadamente bem aos dados de treino, e, por isso, apesar do erro de treino ser mínimo, caso os dados de teste sejam bastante diferentes dos dados de treino, o erro de teste resultante será bastante elevado. Estes dois conceitos são fundamentais para: medir o erro resultante das suposições erradas do algoritmo de aprendizagem (desvio); quantificar a variação na aprendizagem

de uma amostra para outra (variância). Por um lado, se o desvio for elevado, o algoritmo pode perder relações relevantes entre as caraterísticas e a variável dependente dos dados. Por outro lado, se a variância for elevada, o algoritmo pode ajustar-se erradamente aos dados de treino, resultando num modelo pouco preciso. Por isso, deve haver um controlo apertado destes fenómenos para garantir a aprendizagem adequada, algo que é feito a partir do ajuste rigoroso dos hiperparâmetros do modelo.

De seguida, são apresentados os principais algoritmos de SL:

- *Generalized Linear Model* (GLM) [23]: Modelo de regressão linear que descreve a relação entre o(s) *input(s)* e o *output* a partir de uma equação linear. O modelo é treinado para otimizar os coeficientes da função, de modo a minimizar o erro de treino. Tipicamente, são feitas suposições sobre a linearidade do *dataset* que muitas vezes não se verificam, uma vez que a maioria dos dados tem uma relação não-linear.

- *k-Nearest Neighbors* (k-NN): Modelo não-linear em que o *input* consiste nos k dados mais próximos no conjunto de dados de treino. O *output* previsto para um certo ponto é dado pela média dos valores dos seus k vizinhos mais próximos. A distância entre vizinhos é normalmente calculada como sendo a distância euclidiana entre os dois pontos. Este modelo tem as vantagens de ser facilmente interpretável, de ter poucos parâmetros para ajustar e ser bastante rápido na fase de treino, no entanto, devido à sua simplicidade, não consegue atingir precisões muito altas. Pode ser usado tanto para regressão como para classificação.

- *Support Vector Machine* (SVM) [24]: Adaptável a problemas lineares e não-lineares, a SVM constrói um hiperplano apoiado em vetores de suporte que melhor se ajuste aos dados num problema de regressão ou que melhor separe as categorias num problema de classificação. Este modelo tem as vantagens de: ter um elevado desempenho em relações não-lineares; ser robusto a *outliers*; não ser afetado por *over-fitting*. Não é aconselhável ser usado num *dataset* que tenha muitas caraterísticas, pois é necessário aplicar *Feature Scaling* para normalizar os dados, o que o torna num modelo complexo e de dificil interpretação.

- *Naïve Bayes* (NB): Este modelo é usado em problemas de classificação

e baseia-se no teorema de Bayes, isto é, em calcular novas probabilidades tendo em conta probabilidades já conhecidas. Este modelo supõe que as caraterísticas têm todas a mesma relevância estatística e são condicionalmente independentes, mas não deixa de ser eficiente, principalmente quando o conjunto de dados a treinar é de dimensão elevada [25]. Não é influenciável por *outliers* e pode ser utilizado em *datasets* com caraterísticas não-lineares.

- *Decision Tree* (DT): Modelo análogo a uma "árvore" de decisões, em que se efetuam testes nos vários atributos dos dados de treino na tentativa de os distinguir. Quanto mais "folhas" tiver, maior será a distinção feita, mas torna o modelo suscetível a *over-fitting*. Este modelo é facilmente interpretável, não necessita de *Feature Scaling* e funciona tanto em problemas lineares como não-lineares. Pode ser usado para regressão e para classificação. Não é aconselhável se o *dataset* tiver muito poucos dados.

- *Random Forest* (RF): Este modelo tira partido do "trabalho de equipa" de um conjunto de *Decision Trees* na tentativa de melhorar a aprendizagem feita por apenas uma delas. Num problema de regressão, é feita a média das previsões de cada uma das DTs [21], enquanto que em classificação é usada a previsão que tiver mais votos por parte das DTs. Este modelo é bastante eficaz e preciso, e consegue obter um bom desempenho na maioria dos problemas, mesmo naqueles que são caraterizados por serem não-lineares. É necessário otimizar o número de árvores usadas, o *over-fitting* pode facilmente ocorrer e é um modelo de dificil interpretação.

2.3.2 Aprendizagem não supervisionada

São fornecidos dados de treino sem distinção entre *inputs* e *outputs* para que a máquina aprenda um modelo que reconheça um padrão dominante escondido na informação que dispõe [26]. Tipicamente, o UL é usado para analisar agrupamentos de dados (*clustering*) onde se procuram semelhanças entre si, detetar valores atípicos (*outliers*) e fazer estimativas de densidade probabilística, que permitem fazer a chamada redução de dimensionalidade

e também a deteção de anomalias. Para além de técnicas que são igualmente usadas em SL como k-NN, DTs e *Neural Networks* (NNs), as principais aplicações de algoritmos de UL são apresentadas de seguida:

- *Clustering*: Esta técnica permite agrupar os dados de *input* segundo as semelhanças entre si, dividindo o *dataset* em vários *clusters*.
  - *K-Means* [27]: Este modelo é simples de entender, facilmente adaptável, rápido, eficiente e eficaz. No entanto, é necessário escolher o número de *clusters* ótimo para um dado *dataset*, o que pode tornar o processo mais moroso pois, para escolher o melhor modelo, é necessário testar vários com diferentes números de *clusters*.
  - *Hierarchical Clustering* [21]: Este modelo, ao contrário do *K-Means*, consegue obter o número ótimo de *clusters* por ele próprio, não sendo necessário treinar vários modelos para ver qual é o melhor. Usam-se dendrogramas para visualizar os resultados práticos. Não é apropriado para *datasets* extensos.
- *Dimensionality Reduction*: Em muitos dos problemas, não é necessário usar todas as caraterísticas presentes no *dataset* na aprendizagem do modelo. Para reduzir a complexidade desses problemas e tornar o algoritmo mais eficiente, podem-se usar métodos de *Feature Extraction* e de *Feature Selection*.
- *Anomaly Detection*: Usada para identificar eventos fora do comum, a aprendizagem passa por inferir a partir do *dataset* o comportamento normal de um determinado sistema e, com isso, detetar os eventos que são anormais [28]. As duas técnicas mais usadas são:
  - *Rule-based Learning*: Permite aprender regras sobre o *dataset* em que é treinado, sejam combinações de caraterísticas frequentes ou relações interessantes entre variáveis que permitam descobrir algo novo sobre o problema em causa. Similar às DTs mas mais flexível, pois novas regras podem ser adicionadas sem afetar as que já existem.

– *Pruning techniques*: Utilizadas para identificar *outliers*, onde existem anomalias em qualquer combinação de caraterísticas possível.

2.3.3 Aprendizagem por reforço

A máquina utiliza informação sobre o ambiente que a rodeia, de modo a aprender quais as ações que deve tomar para atingir um determinado objetivo, recebendo em troca *feedback* análogo a uma recompensa que tenta maximizar continuamente. Ao contrário do SL e UL que não vêem para além dos dados que lhes são facultados, o RL consegue alcançar maiores recompensas futuras em detrimento de ganhos a curto prazo. Por isso, o RL é a abordagem adequada para fazer escolhas cognitivas, tais como tomar decisões ou efetuar planeamentos e agendamentos [29, 30]. A técnica mais proeminente em RL é a *Q-Learning* juntamente com NNs e *Deep NNs* (DNNs), podendo chamar-se de *Deep RL*.

A indústria das telecomunicações está agora preparada para o ML, pois os operadores, mais do que nunca, têm acesso a um imenso conjunto de dados (dados sobre os clientes, desempenho da rede, tráfego na rede, entre outros) prontos para serem aprendidos. Para além disso, a dificuldade na gestão de uma rede móvel já ultrapassa a administração manual, o que leva a que a principal causa de falhas seja por erro humano [22], sendo por isso recomendada a proliferação de ML na supervisão do sistema, na medida de reduzir a carga de trabalho dos engenheiros neste processo e deixar as máquinas fazerem o que melhor sabem.

2.4 Áreas de Gestão da Rede

Uma rede, no geral, pode ser dividida em várias áreas que devem ser geridas o melhor possível para otimizar o funcionamento do sistema. Para entender as principais funções de um sistema de gestão da rede e simplificar o processo de implementação de técnicas de ML na automatização das operações, foram definidas pela *International Organization for Standardization* (ISO) cinco áreas de gestão conhecidas como FCAPS [31] (*Fault, Configuration, Accounting, Performance, Security*) que são exploradas de seguida.

2.4.1 Gestão de falhas

As falhas numa rede móvel, ao contrário do desejado, surgem frequentemente e constituem um dos principais problemas na sua gestão, pois podem tornar-se dispendiosas quando levam à perda de clientes por parte da operadora. O tempo de resposta lento e as pobres soluções que as técnicas tradicionais para a gestão de falhas apresentam, ampliam ainda mais este custo. A procura por melhores soluções tem passado por potencializar o uso de ML na predição proativa de falhas, que permite localizar e mitigar rápida e automaticamente as eventuais falhas que possam surgir no sistema, de modo a minimizar o tempo de inatividade (*downtime*) da rede e a intervenção humana tanto do lado dos colaboradores, que deixariam de ter que resolver o problema manualmente, como do lado dos clientes, que se queixariam menos. Quanto menos falhas, maior a satisfação do cliente.

Como descrito em [19], a aplicação de ML poderá também ser crucial na prevenção de falhas, ao identificar e mitigar a causa de uma falha que tenha sido prevista acontecer. Desta maneira, o problema é resolvido antes mesmo de se tornar um problema. Como a prevenção proativa de falhas requer a execução de um conjunto de medidas, aplicar técnicas de RL seria o mais adequado neste caso. Na localização de falhas são usadas técnicas de ML como NN, k-NN, *k-Means* e DT, enquanto que na mitigação reativa de falhas são normalmente usadas NB e SVM. Na predição de falhas pode ser usada qualquer uma destas técnicas.

2.4.2 Gestão de configuração

Os operadores devem implementar políticas de rede cada vez mais sofisticadas que se ajustem a mudanças nas condições da rede, como quando, por exemplo, há degradação de desempenho ou existem intrusões na rede. Como o estado da rede está em constante mudança, os operadores vêem-se obrigados a ajustar constantemente a configuração da rede para se adaptar a estas alterações, o que pode chegar a ser um processo exaustivo e propício a erros. Desta feita, os algoritmos de ML podem ajudar a automatizar este processo, ao treinarem modelos para identificar as ações ótimas a tomar em cada instante, dependendo do comportamento da rede ao longo do tempo. A aplicação de ML neste âmbito, demonstra benefícios em procedimentos como, por exemplo, a alocação dinâmica de recursos na rede e a configuração de serviços adaptativa, que procuram melhorar as condições que a operadora pretende oferecer aos seus clientes.

O uso de ML pode passar por construir um modelo que relacione comandos de baixo nível, como, por exemplo, os recursos que devem ser alocados, com requisitos de alto nível dos quais fazem parte o desempenho e a segurança da rede ou até mesmo a tolerância a falhas já falada. Neste contexto, pode-se proceder à utilização de técnicas de RL em que a recompensa, que advém da escolha acertada duma configuração de rede, pode ser encarada como a importância desse ajuste no cumprimento dos requisitos definidos, que dependem também da condição da rede. Na configuração adaptativa de serviços pode ser usado *Q-Learning*, sendo ainda possível tirar partido de *Deep RL* para a alocação dinâmica de recursos [19].

2.4.3 Gestão de contabilidade

A área da contabilidade está intimamente ligada aos modelos de negócios, que têm em conta os dados contábeis na tomada de decisões, no planeamento e entrega de serviços, e na designação de tarifas e planos de pagamento. Por isso, é essencial garantir a integridade destes dados ao detetar e impedir ações fraudulentas durante a recolha dos mesmos. A prática de técnicas de ML para a gestão da contabilidade de uma rede móvel ainda não é muito explorada [19], mas os potenciais benefícios na sua utilização seriam a redução de custos

e o aumento da rentabilidade dos serviços prestados pela operadora.

Dada a importância da satisfação do cliente, poderiam ser usadas técnicas de SL e UL para aprender os hábitos de consumo dos utilizadores e tornar a gestão de contabilidade mais inteligente ao personalizar tarifários e serviços conjuntos com vantagens para cada cliente, como a oferta dinâmica de recursos, por forma a melhorar a qualidade de serviço e, consequentemente, proporcionar uma melhor experiência ao consumidor.

2.4.4 Gestão de desempenho

As redes atuais providenciam uma panóplia de serviços, com requisitos de desempenho diferentes, a um número crescente de clientes tendo estes perfis distintos. Para garantir a perfeita funcionalidade do sistema ao longo do tempo, é necessário ter a capacidade de prever corretamente o comportamento da rede. Nesse sentido, têm sido usadas técnicas de ML na predição do tráfego e desempenho do serviço, e na melhoria da correlação entre a QoE e a QoS, resultando numa gestão adaptativa da performance da rede de forma proativa [32].

A obtenção regular de medidas de desempenho é fulcral na monitorização do comportamento da rede. No entanto, a quantidade de dispositivos e a variedade de parâmetros a medir resultam num tráfego excessivo que pode degradar o desempenho da rede. ML pode ser usado para compreender a relevância dos valores medidos e, assim, a rede ter apenas em conta os parâmetros mais importantes, ao utilizar modelos de SL que ajudem na regulação do *overhead* no tráfego. Na predição de medidas de desempenho e tráfego podem ser usadas técnicas como NN, NB e SVM e para melhorar a correlação QoE-QoS são usadas DTs ou até mesmo *Q-Learning* [19].

2.4.5 Gestão de segurança

A medida de segurança mais popular em redes móveis consiste em monitorizar o sistema na tentativa de identificar padrões de ameaças já conhecidas. No entanto, a rede torna-se vulnerável a ataques de dia-zero (*0-day*), porque não está preparada para se defender de algo que desconhece. São necessárias

estratégias de segurança robustas para colmatar esta vulnerabilidade, que consigam automaticamente reconhecer e distinguir as verdadeiras ameaças - que podem comprometer o sistema - dos acessos inocentes por parte do cliente comum, pelo que o papel do ML para esse fim tem sido investigado extensivamente [33].

A utilização de ML passa por usar dados históricos para aprender padrões de ataque complexos e gerar modelos genéricos que permitam identificar variações em ataques já conhecidos e, com isso, detetar o uso indevido dos serviços. A deteção de anomalias usando ML tem sido também explorada para reconhecer ataques de dia-zero, ao aprender como o sistema normalmente se comporta e identificar desvios à norma. Na deteção de uso indevido podem ser usadas NN, DT, NB e SVM, tal como na deteção de anomalias onde são ainda usadas DNN, k-NN e *k-Means*.

Capítulo 3

Análise Exploratória dos Dados

Neste capítulo será feito o estudo e análise sobre os registos de alarmes utilizados na indústria das telecomunicações, nomeadamente em estações-base de um fornecedor específico e disponibilizados pela CELFINET.

Na primeira secção deste capítulo é feita uma pequena síntese de informações referentes aos alarmes que serão úteis no desenvolvimento do trabalho. Posteriormente, é feita uma análise exploratória dos dados, recorrendo às bibliotecas de Python: pandas [34], Matplotlib [35], NumPy [36], seaborn [37], missingno [38] e Dython [39].

3.1 Descrição dos alarmes

Alguns dos principais dados disponíveis de *Fault Management* (FM) usados ao longo do trabalho são:

- Falha na ligação *Stream Control Transmission Protocol* (SCTP);
- Falha na interface X2;
- Limite licenciado excedido na configuração dos dados;
- Erro na topologia da unidade RF durante a sua execução;
- Célula bloqueada;

- Falha na interface S1;
- Capacidade da célula degradada;
- Falha na ligação externa;
- Falha na ligação de manutenção *Antenna Line Device* (ALD);
- Problema na referenciação do *clock* externo;
- Conflito na configuração dos *cabinets* entre sistemas;
- Serviço da *Public Land Mobile Network* (PLMN) degradado;
- Limiar do *Voltage Standing Wave Ratio* (VSWR) da unidade RF ultrapassado.

Os alarmes são classificados quanto à sua severidade, por ordem decrescente, da seguinte forma:

1. ***Critical***: Um alarme crítico afeta os serviços do sistema. Assim que é gerado devem ser tomadas ações imediatas, mesmo quando a falha ocorre fora do horário de trabalho;
2. ***Major***: Um alarme grave afeta a qualidade de serviço e requer ações imediatas, apenas se ocorrer durante o horário de trabalho;
3. ***Minor***: Um alarme ligeiro normalmente não afeta a qualidade do serviço. Deve ser tratado assim que possível ou observado para evitar falhas mais graves;
4. ***Warning***: Um alarme de aviso indica um possível erro que pode afetar a qualidade de serviço. Requer diferentes ações dependendo dos erros.

Os alarmes são ainda categorizados pelos seguintes tipos, dependendo da sua função:

- ***Power***: Alarme gerado quando há uma falha no sistema de energia;

- ***Environment***: Alarme gerado quando são atingidos valores anormais para as variáveis de ambiente da sala onde estão os equipamentos, tais como temperatura, humidade, e controlo do estado da porta;

- ***Signaling***: Alarme gerado quando há um problema na sinalização de canal associado (por exemplo, SS1) e/ou de canal comum (por exemplo, SS7);

- ***Trunk***: Alarme gerado quando ocorrem falhas no sistema de transmissão, incluindo circuitos e placas de transmissão;

- ***Hardware***: Alarme gerado quando são detetadas anomalias nos módulos de *hardware*, tais como a unidade de relógio (*clock*) e CPU (processador);

- ***Software***: Alarme gerado quando existem erros no *software* a correr no sistema;

- ***System***: Alarme gerado quando existe um problema com as operações do sistema;

- ***Communication***: Alarme gerado quando há alguma falha na comunicação. Por exemplo, comunicação entre placas *host* e o *Back Administration Module* (BAM));

- ***Service quality***: Alarme gerado quando se verifica que a qualidade de serviço (QoS) não é suficiente;

- ***Unexpected operation***: Alarme gerado quando ocorre uma exceção inesperada no sistema;

- ***OMC***: Alarme gerado quando o OMC não está a funcionar devidamente;

- ***Integrity***: Alarme gerado quando alguma informação é adicionada, modificada ou removida indevidamente;

- ***Operation***: Alarme gerado quando o serviço está indisponível ou inacessível devido a operações inapropriadas;

- ***Physical resource***: Alarme gerado quando são visados recursos físicos por um ataque suspeito;
- ***Security***: Alarme gerado quando os serviços ou algum mecanismo de segurança detetou que o sistema se encontra a ser atacado;
- ***Time domain***: Alarme gerado quando um evento ocorre inesperadamente;
- ***Running***: Alarme gerado durante a execução de alguma tarefa;
- ***Processing***: Alarme gerado quando ocorre um erro durante o processo de configuração de uma portadora.

As principais causas para o aparecimento destes alarmes enquadram-se nas seguintes áreas:

- ***Data configuration***: Configuração imprópria ou incorreta, por exemplo, um limiar (*threshold*) inadequado;
- ***Software***: Erros de *software*;
- ***Hardware***: Falhas de *hardware*, por exemplo, *hardware* danificado ou simplesmente conectores soltos;
- ***Other***: Qualquer outra causa, exceto as anteriores.

Os registos dos alarmes podem ainda conter algumas informações adicionais:

- **RAT_INFO**: RAT da estação-base que reporta o alarme. Para uma estação-base multimodo, este parâmetro é definido como o RAT multimodo, por exemplo, GL (GSM-LTE);
- **AFFECTED_RAT**: RAT da estação-base que é afetada pelo alarme;
- ***Deployment IDentifier* (DID)**: Identificador da estação-base que reporta o alarme;
- **RF Unit Name**: Nome da unidade RF. Este parâmetro é utilizado apenas quando é uma unidade RF a reportar o alarme.

3.2 Análise estatística dos dados

Para melhor compreender a natureza dos alarmes foi feita uma análise exploratória dos dados em estudo. Como existe um elevado número de registos para ler, seria exaustivo realizar este processo manualmente, pelo que a inferência estatística dos dados é suficientemente capaz de descrever a população de alarmes em causa, identificando os seus aspetos que serão mais interessantes de analisar.

3.2.1 Identificação das variáveis

Numa primeira fase são processadas algumas informações gerais sobre o *dataset* em questão, resumidas na Figura 3.1.

```
RangeIndex: 224885 entries, 0 to 224884
Data columns (total 9 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   ALARM_DATETIME  224885 non-null  datetime64[ns]
 1   SEVERITY        224885 non-null  string
 2   ALARM_NAME      224885 non-null  string
 3   ALARM_TYPE      194002 non-null  string
 4   MO_NAME         224885 non-null  string
 5   INFO            224881 non-null  string
 6   NODE_NAME       224885 non-null  string
 7   TECH            224533 non-null  string
 8   Vendor          221936 non-null  string
dtypes: datetime64[ns](1), string(8)
```

Figura 3.1: Identificação das variáveis.

É possível verificar que o *dataset* é uma matriz que contém 224885 linhas e 9 colunas, ou variáveis, sendo elas: **ALARM_DATETIME; SEVERITY; ALARM_NAME; ALARM_TYPE; MO_NAME; INFO; NODE_NAME; TECH; Vendor**. Observa-se na coluna "*Non-Null Count*" que as variáveis ALARM_TYPE, INFO, TECH e Vendor têm alguns valores em falta. Das 9 variáveis, 8 são categóricas (*strings*) enquanto que o ALARM_DATETIME contém a data e hora (*timestamp*) em que o alarme foi despoletado.

Para cada uma das variáveis presentes no *dataset* foram feitas algumas observações:

1. **ALARM_DATETIME**: Nenhum valor em falta. Foram identificados 192511 *datetimes* distintos. Os dias destes registos estão compreendidos entre 7 de Novembro de 2019 e 15 de Abril de 2021, o que equivale a uma janela temporal de 525 dias. Verifica-se que os dias com maior número de ocorrências de alarmes são, na sua maioria, nos meses de Verão, nomeadamente durante o mês de Julho;

2. **SEVERITY**: Nenhum valor em falta. Foram identificadas 7 severidades únicas, sendo *Major* a mais frequente com 182451 alarmes, o equivalente a 81.13% dos alarmes. Foram encontradas 3 severidades extra (m, M, w) que não pertencem ao grupo das 4 severidades documentadas (Critical, Major, Minor, Warning). Considerando pelos seus nomes que tenham havido erros na escrita destes registos, foi feita uma pesquisa mais específica onde se concluiu o esperado: 'm' é na realidade "Minor", 'M' é "Major" e 'w' é "Warning";

3. **ALARM_NAME**: Nenhum valor em falta. Foram identificados 217 nomes de alarme únicos, sendo o mais frequente "SCTP Link Fault" com 168353 alarmes;

4. **ALARM_TYPE**: 30883 valores em falta. Foram identificados 11 tipos de alarme únicos, sendo "Trunk" o mais frequente com 169018 alarmes. Foram apenas encontrados os seguintes tipos documentados: Trunk, Signaling, Hardware, Running, Communication, Environment, QoS, Power, Security, Software, Processing;

5. **MO_NAME**: Nenhum valor em falta. Foram identificados 1316 *Management Objects* (MOs) únicos, sendo "eNodeB" o mais frequente com 175062 alarmes;

6. **INFO**: 4 valores em falta. Foram identificadas 165967 informações únicas. O número elevado de *info's* distintas deve-se ao facto de apresentarem valores unívocos (como identificadores) e, por isso, serem diferentes de alarme para alarme. Ainda assim, verificou-se que a *info* mais frequente, presente em 5822 alarmes, é a de "Neighboring Base Station Type=eNodeB, Specific Problem=Lower-layer link fault";

7. **NODE_NAME**: Nenhum valor em falta. Foram identificados 2418 nomes de nós distintos, sendo que o mais frequente é o "VA14OL" com 5916 alarmes;

8. **TECH**: 352 valores em falta. Foram identificados 4 tecnologias únicas, sendo elas: 4G, 3G, SRAN e 2G. A mais frequente é a 4G, presente em 191699 alarmes;

9. **Vendor**: 2949 valores em falta. Foram identificados 2 fornecedores únicos, Fornecedor 1 e Fornecedor 2. O mais frequente é o Fornecedor 1 com 191053 alarmes, enquanto que só existem 30883 registos do Fornecedor 2.

3.2.2 Pré-processamento dos dados

Como o *dataset* está ordenado cronologicamente, é possível averiguar qual a origem dos valores em falta. Observando a Figura 3.2, onde o foco se centra sobre a variável ALARM_TYPE, verifica-se que os dados em falta (representados como os espaços em branco) são completamente aleatórios, não tendo qualquer relação temporal entre si, pelo que não é possível ter certezas sobre a sua origem.

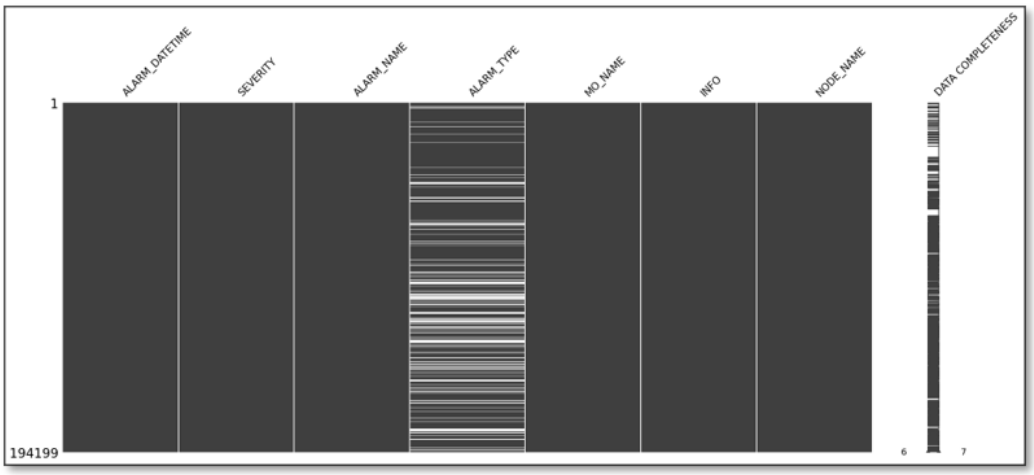


Figura 3.2: Cronologia dos valores em falta.

Num estudo mais aprofundado, verificou-se que estes tipos de alarme em falta ocorriam apenas para os alarmes do Fornecedor 2, algo que se poderia

suspeitar pois são idênticos em número (30883 tipos em falta, 30883 alarmes do Fornecedor 2).

Esta falta de dados pode dever-se ao facto do Fornecedor 2 não categorizar os seus alarmes por tipo de alarme, algo que só se poderia confirmar acedendo à documentação apropriada. Como não foi possível obter essa informação, decidiu-se então utilizar apenas os alarmes do Fornecedor 1 cuja documentação é acessível e foi resumida anteriormente. Assim sendo, a coluna Vendor foi removida, pois ficou apenas com 1 único valor distinto.

Para além disso, observou-se que a remoção dos alarmes do Fornecedor 2 corrigiu não só todos os outros valores em falta, como também eliminou as severidades erradamente registadas (m, M, w). Verificou-se ainda que a variável MO_NAME passou a ter apenas 4 nomes distintos: eNodeB, SRAN, BSC e NodeB. As colunas MO_NAME e TECH passaram a ter uma correlação de 100%, pelo que foram unificadas numa só (MO_NAME), ficando com os valores únicos: eNodeB (4G); SRAN; BSC (2G); NodeB (3G).

Foram ainda identificadas algumas entradas duplicadas no *dataset*, nomeadamente 144 registos de alarmes com mais que uma severidade - em 72 entradas aparecem como sendo alarmes "Minor" e nas outras 72, os mesmos alarmes, são considerados "Major". Acedendo à documentação específica de cada alarme verificou-se que existe realmente a possibilidade de alguns alarmes terem diferentes severidades dependendo da gravidade do problema e, por isso, estas entradas duplicadas não foram removidas.

Todas as inconsistências foram então resolvidas, pelo que a preparação dos dados começou e terminou com a remoção dos registos de alarmes do Fornecedor 2.

3.2.3 Análise univariada

As variáveis podem ser classificadas quanto ao seu tipo e quanto à sua categoria, como descreve a Tabela 3.1.

As variáveis categóricas (*strings*) são qualitativas, enquanto que as variáveis numéricas são quantitativas. Quanto às categorias, as variáveis qualitativas podem ser ordinais, se existir alguma ordem (mesmo que subentendida) entre

Tabela 3.1: Tipos e categorias de variáveis.

Tipo	**Categoria**
Qualitativa	Nominal
	Ordinal
Quantitativa	Contínua
	Discreta

elas, ou nominais, se essa ordem não existir. Já as variáveis quantitativas podem ser discretas, se o seu conjunto de valores suportar apenas números inteiros, ou contínuas, se houver uma continuidade no intervalo de valores dessa variável. As 7 variáveis em análise foram então classificadas como:

1. **Qualitativa Nominal**: SEVERITY, ALARM_NAME, ALARM_TYPE, MO_NAME, INFO, NODE_NAME;

2. **Quantitativa Discreta**: ALARM_DATETIME.

Dependendo do seu tipo e categoria, as variáveis são analisadas individualmente (análise univariada) de diferentes formas. As variáveis qualitativas (sejam elas nominais ou ordinais) e quantitativas discretas podem ser avaliadas construindo uma tabela de frequências, que contenha a contabilização do número de ocorrências de cada valor para uma determinada variável. A tabela de frequências da variável ALARM_TYPE encontra-se em 3.2.

Tabela 3.2: Tabela de frequências (absoluta e relativa) dos tipos de alarme.

ALARM_TYPE	**Freq. Absoluta**	**Freq. Relativa** [%]
Trunk	166528	87.163
Signaling	8981	4.701
Hardware	7969	4.171
Running	2301	1.204
Communication	1860	0.974
Environment	1576	0.825
QoS	1375	0.720
Power	308	0.161
Security	108	0.057
Software	38	0.020
Processing	9	0.005

A partir desta tabela é possível representar graficamente a distribuição de ocorrências dos diversos tipos de alarme, seja por um gráfico de barras (Figura 3.3.a) ou por um gráfico circular (Figura 3.3.b). Optou-se por ilustrar apenas os gráficos referentes à frequência relativa de cada tipo de alarme.

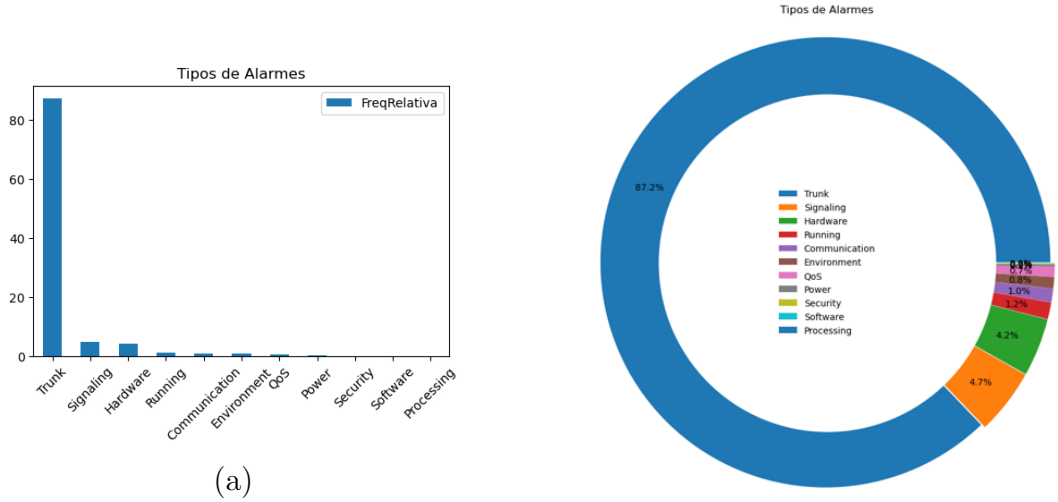


Figura 3.3: Gráfico de barras (a) e Gráfico circular (b) para os tipos de alarme.

Observa-se que a grande maioria dos alarmes são do tipo "Trunk", enquanto que os tipos menos frequentes são tão raros que chegam a ser imperceptíveis nestes gráficos. Por isso, foram também desenhados os gráficos para todos os tipos de alarme exceto o "Trunk", que se encontram na Figura 3.4.

O gráfico (a) desta figura trata-se de uma aproximação aos tipos de alarme menos frequentes da Figura 3.3.a, onde se destacam os alarmes "Signaling" e "Hardware", ambos acima dos 4% de ocorrência.

No gráfico (b) encontra-se a distribuição percentual dentro destes mesmos tipos de alarme, onde se observa que os alarmes "Signaling" e "Hardware" perfazem, em conjunto, 69.1% dos alarmes pertencentes aos 10 tipos mais raros.

Apesar da elevada frequência destes tipos de alarme não implicar que sejam severos e que devam ser tratados de imediato, estes alarmes, principalmente os do tipo "Trunk", devem ser analisados ao detalhe por parte dos

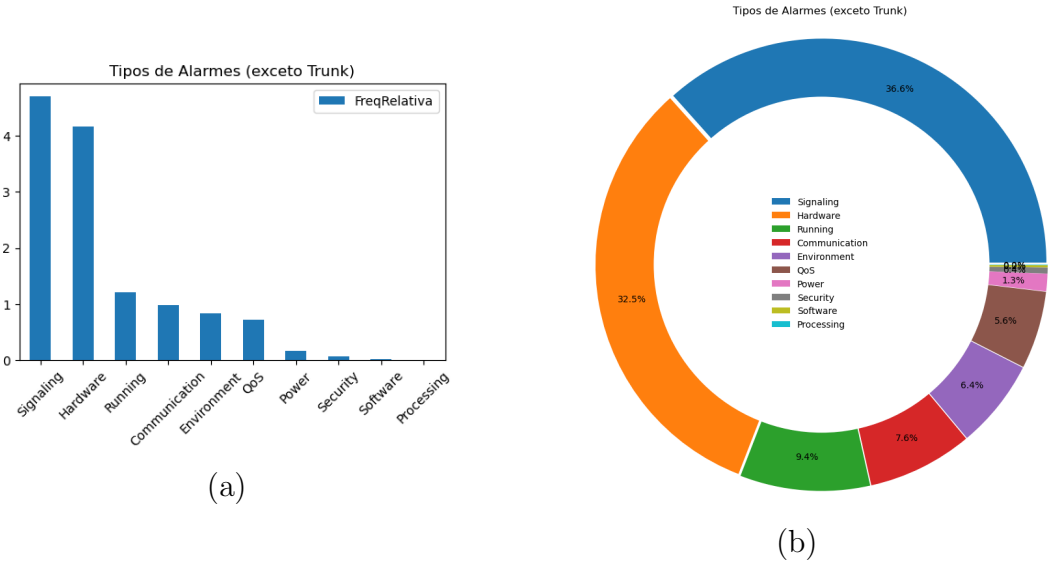


Figura 3.4: Gráfico de barras (a) e Gráfico circular (b) para todos os tipos de alarme à exceção do "Trunk".

engenheiros do NOC, em conjunto com algoritmos de ML, na tentativa de perceber as suas causas e reduzir as suas probabilidades de ocorrência.

3.2.4 Análise bivariada

Por forma a encontrar associações entre 2 variáveis, realiza-se a chamada análise bivariada. A mais importante medida para quantificar a força da dependência entre duas variáveis é a correlação, dada pelo *R de Pearson* e cujos valores estão contidos no intervalo de $[-1, 1]$. No entanto, a correlação só está definida para valores numéricos, quando aqui tem-se como objetivo correlacionar variáveis categóricas.

Para isso é possível usar o *V de Cramer*, uma métrica semelhante à correlação que permite medir a associação entre variáveis categóricas. A única diferença é que não toma valores negativos: é no mínimo 0, se não existir qualquer associação entre as variáveis, e no máximo 1, se houver uma associação total entre elas.

Tal como a correlação, a associação é uma grandeza simétrica, isto é, a associação entre A e B é igual à associação entre B e A. Por isso, podem existir

informações escondidas que são perdidas por se assumir essa simetria. Para resolver este problema foi usada uma variante do *V de Cramer* assimétrica: *U de Theil*. Também conhecido como coeficiente de incerteza, baseia-se na entropia condicional entre as variáveis, que tem em conta a ordem dos termos. Logo, a elevada certeza na associação entre A e B não implica que haja, com certeza, uma associação entre B e A, podendo mesmo ser nula.

A melhor forma de resumir as certezas de associações entre os vários pares de variáveis é representá-las numa matriz de associação (Figura 3.5).

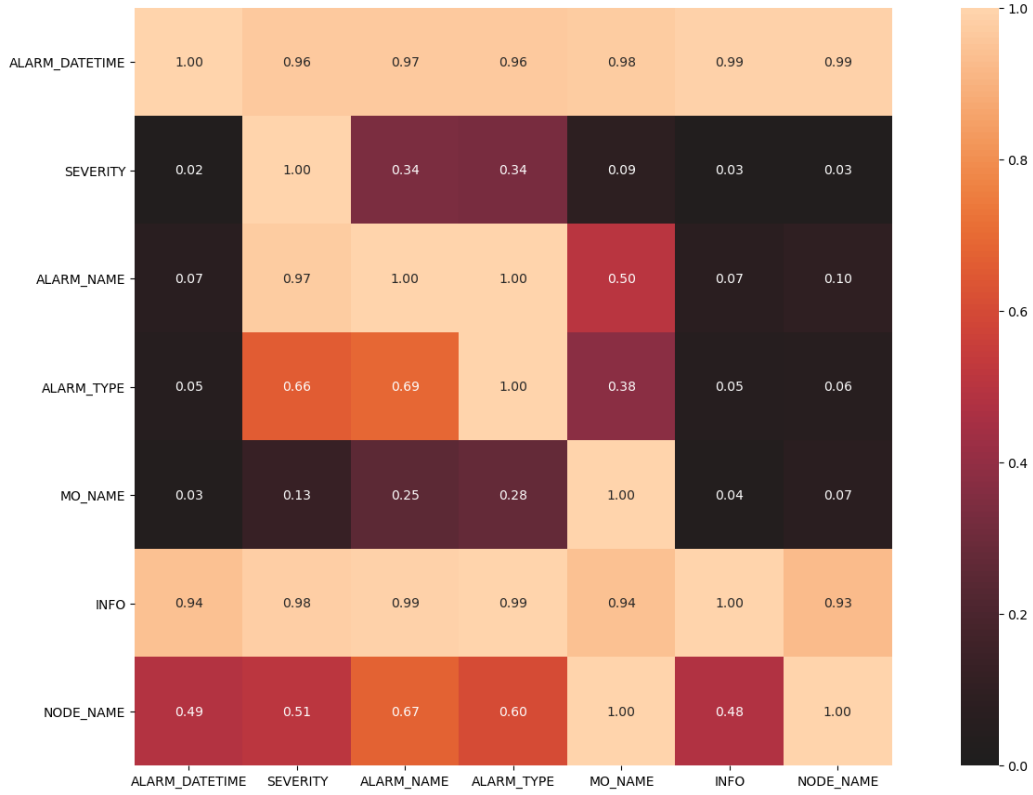


Figura 3.5: Matriz completa de associação entre variáveis.

Para a análise foram excluídas as variáveis com um elevado número de valores distintos (ALARM_DATETIME, INFO e NODE_NAME). Repara-se que a variável ALARM_TYPE é a que tem os coeficientes de incerteza mais elevados, nomeadamente com as variáveis ALARM_NAME (0.69) e SEVERITY (0.66). Optou-se por analisar primeiro a sua relação com a severidade, por ser a que tem menos valores distintos, e só depois, tomando uma das

severidades como exemplo, verificar a sua associação com o nome do alarme.

A distribuição do número de ocorrências dos vários tipos de alarme pelas correspondentes severidades encontra-se representada na Figura 3.6.

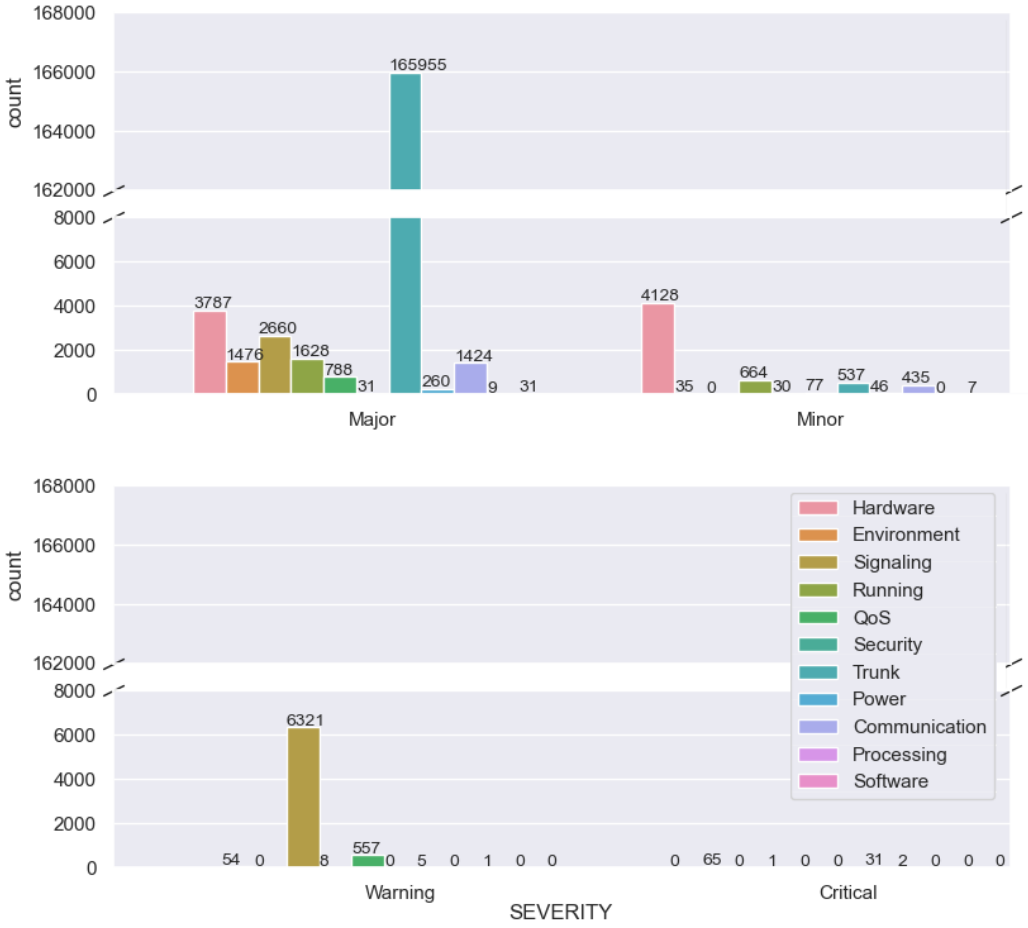


Figura 3.6: Distribuição dos vários tipos de alarme pelas severidades.

Verifica-se que a grande maioria dos alarmes "Minor" são do tipo "Hardware" e os "Warning" são em grande parte do tipo "Signaling". Os alarmes com severidade "Critical" são, quase na totalidade, de apenas 2 tipos: Environment e Trunk. Os alarmes de severidade "Major" são os que estão mais distribuídos por todos os tipos de alarmes e, por isso, serão analisados em maior detalhe. Em primeiro lugar, é calculada novamente a matriz de associação entre todas as variáveis, exceto a SEVERITY que agora só tem um único valor (*Single Value* (SV)), como mostra a Figura 3.7.

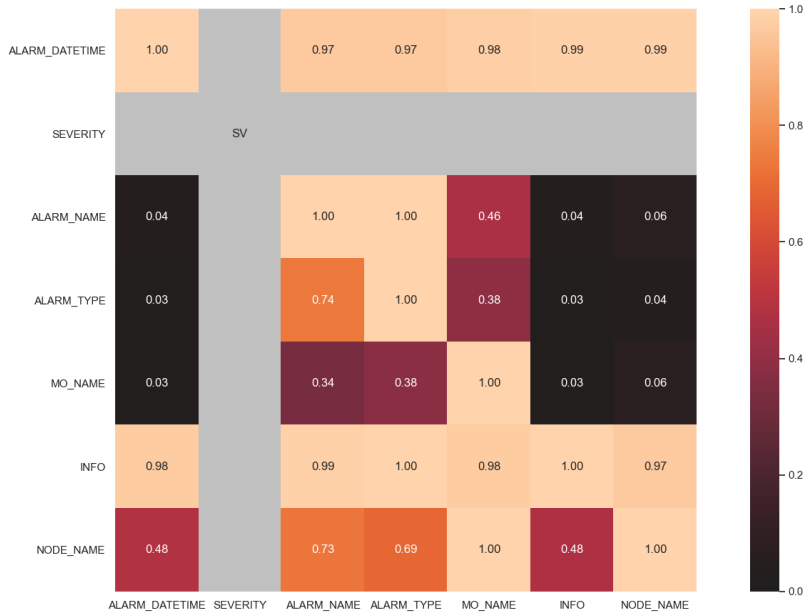


Figura 3.7: Matriz de associação para a severidade "Major".

Observa-se que a variável com maior certeza na associação com a coluna ALARM_TYPE continua a ser o ALARM_NAME e, por isso, foi analisada a sua distribuição de valores em relação às duas variáveis. Como existem vários nomes de alarme, a Figura 3.8 ilustra apenas os 10 mais frequentes.

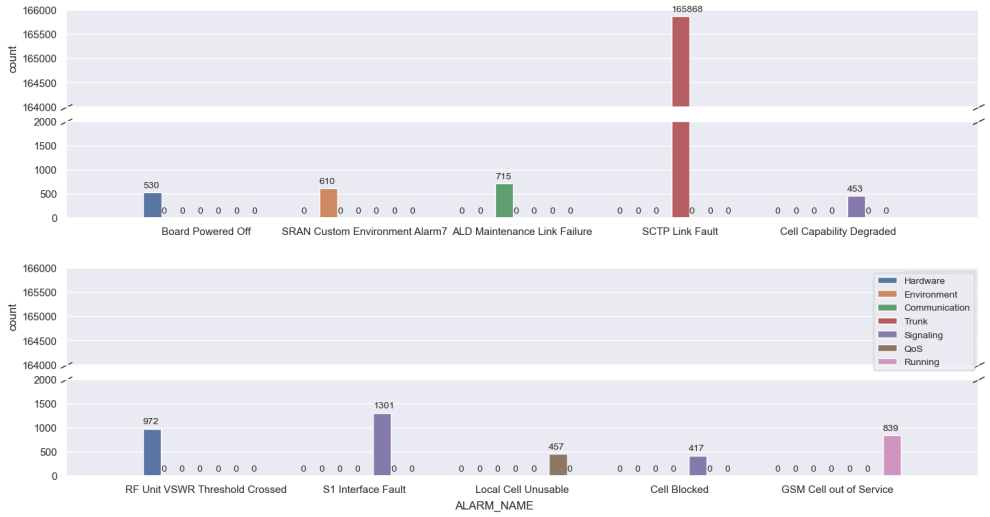


Figura 3.8: Distribuição dos vários tipos de alarme de severidade "Major" pelos 10 nomes de alarme mais frequentes.

Verifica-se que, para os 10 alarmes "Major" mais frequentes, existe uma dependência única com o tipo de alarme, ou seja, a certeza de associação entre as duas variáveis seria igual a 1. As barras representadas graficamente estão associadas aos tipos de alarme a que estes 10 alarmes pertencem e, por isso, só são ilustrados 7 tipos de alarme (num total de 11), distribuídos da seguinte forma:

- **Hardware**: Board Powered Off (530), RF Unit VSWR Threshold Crossed (972);
- **Environment**: SRAN Custom Environment Alarm7 (610);
- **Communication**: ALD Maintenance Link Failure (715);
- **Trunk**: SCTP Link Fault (165868);
- **Signaling**: Cell Capability Degraded (453), S1 Interface Fault (1301), Cell Blocked (417);
- **QoS**: Local Cell Unusable (457);
- **Running**: GSM Cell out of Service (839).

Para além do cálculo das associações entre variáveis, pode-se também construir uma tabela de dupla entrada (*two-way table*). Continuando com o foco na variável ALARM_TYPE, a tabela diária associada ao mês de Julho de 2020 é apresentada na Tabela 3.3.

É possível representar esta tabela por um gráfico de colunas empilhadas (*stacked column chart*), apresentado na Figura 3.9, onde se observa a distribuição diária dos vários tipos de alarme para o mês de Julho de 2020.

Para além disso, pode-se focar num dos tipos de alarmes como, por exemplo, o mais frequente "Trunk" e construir a sua tabela de dupla entrada desta vez horária e representá-la graficamente por um *heatmap* calendarizado, como apresentado na Figura 3.10.

Tabela 3.3: Tabela de dupla entrada da variável ALARM_TYPE, para Julho 2020.

DIA	Communication	Environment	Hardware	Power	Processing	QoS	Running	Security	Signaling	Software	Trunk
2020-07-01	1	16	26	1	0	3	5	6	28	0	1129
2020-07-02	10	4	61	0	0	20	24	0	55	0	1325
2020-07-03	0	2	18	1	0	1	3	3	21	0	946
2020-07-04	3	1	6	0	0	0	2	3	3	0	3
2020-07-05	2	2	0	0	0	2	6	0	10	0	108
2020-07-06	7	5	18	1	0	0	6	0	18	0	1043
2020-07-07	6	13	11	1	0	2	8	0	30	0	1157
2020-07-08	10	5	26	4	0	4	20	0	28	0	1238
2020-07-09	13	8	45	3	0	6	24	0	48	0	1424
2020-07-10	6	1	5	0	0	4	8	0	21	0	1268
2020-07-11	0	1	4	0	0	0	4	0	2	0	3
2020-07-12	2	0	4	1	0	0	8	0	6	0	1
2020-07-13	8	2	20	2	0	0	10	0	21	0	221
2020-07-14	24	14	43	1	0	8	13	0	42	0	1253
2020-07-15	22	4	39	2	0	8	30	2	52	0	1415
2020-07-16	17	17	107	2	0	64	18	0	63	0	1664
2020-07-17	11	1	12	0	0	5	12	0	24	0	1175
2020-07-18	0	1	1	0	0	2	0	0	5	0	1
2020-07-19	21	2	25	0	0	9	4	1	22	0	46
2020-07-20	8	6	45	1	0	5	5	0	36	0	1276
2020-07-21	4	18	27	1	0	5	10	0	47	0	1363
2020-07-22	13	9	37	2	0	13	22	0	49	0	1361
2020-07-23	12	9	25	1	0	10	17	1	224	0	1505
2020-07-24	2	2	28	0	0	4	1	0	21	0	1253
2020-07-25	2	0	9	0	0	3	4	0	4	0	3
2020-07-26	1	1	8	0	0	0	0	1	7	0	38
2020-07-27	11	8	30	2	0	3	12	0	27	0	1472
2020-07-28	8	11	37	2	0	2	12	2	35	0	1420
2020-07-29	4	8	46	0	0	4	3	0	35	0	1386
2020-07-30	13	8	21	0	0	15	16	0	42	0	890
2020-07-31	9	6	47	1	0	35	12	0	29	0	1249

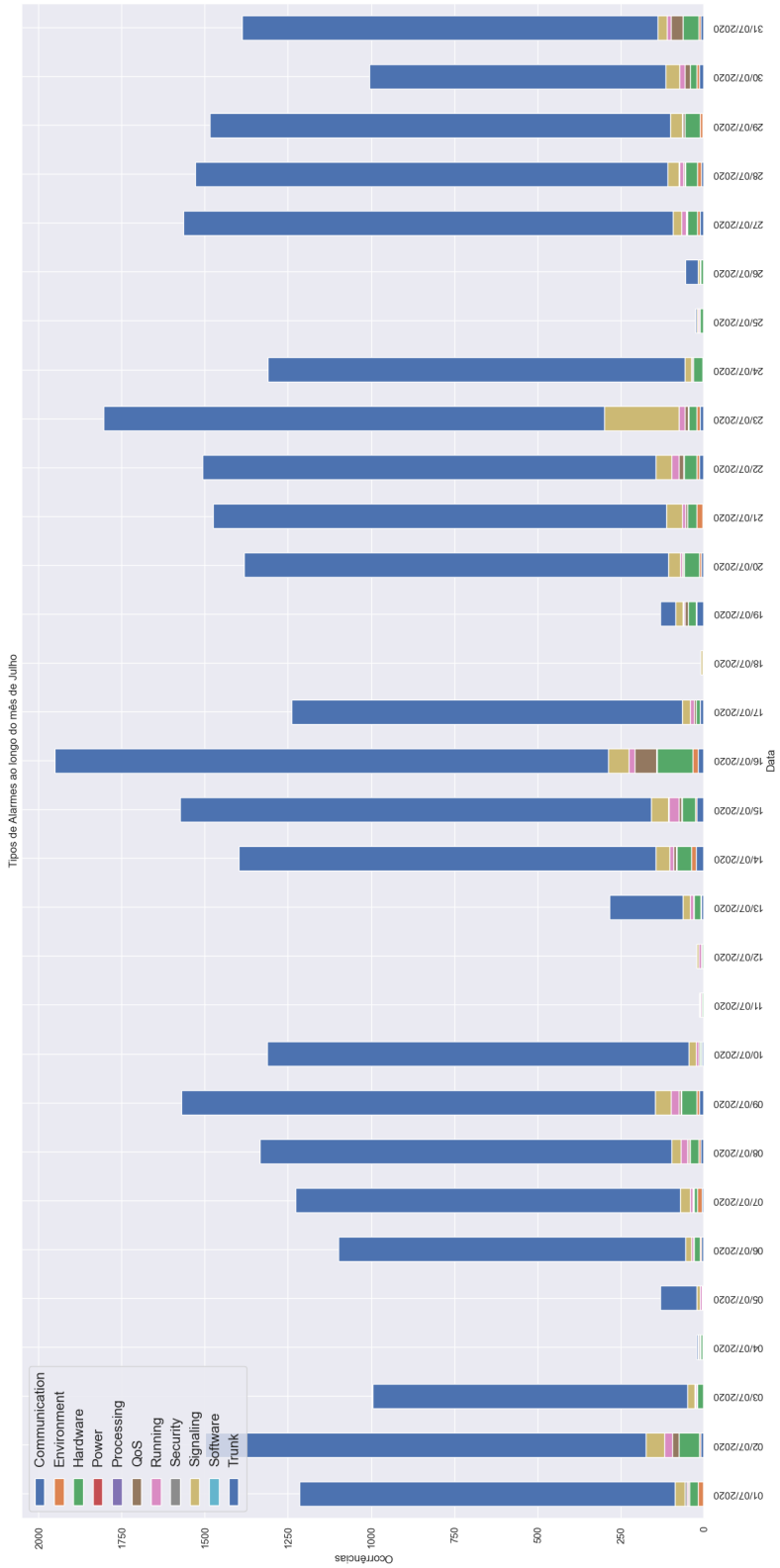


Figura 3.9: Gráfico de colunas empilhadas da variável ALARM_TYPE, para Julho 2020.

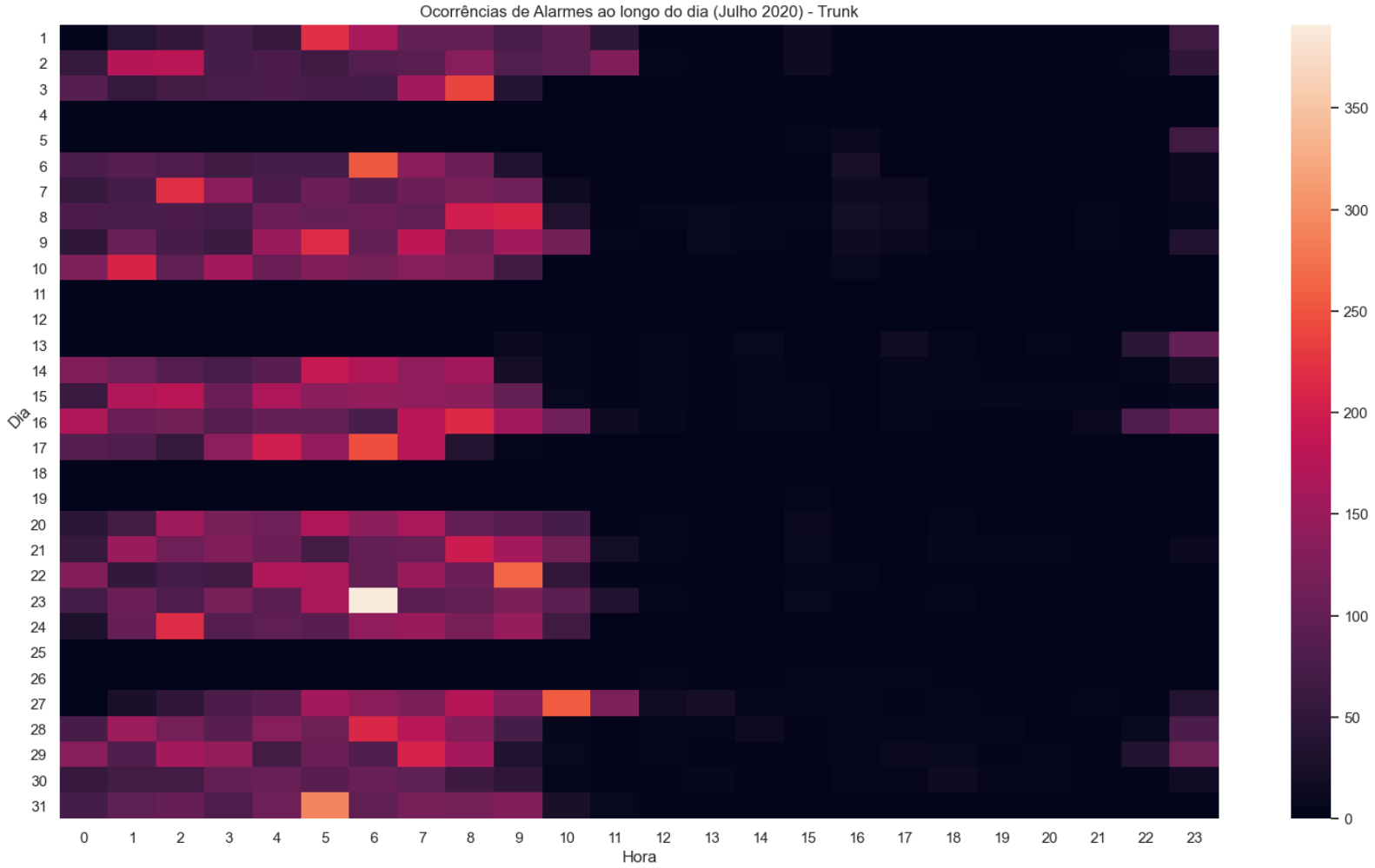


Figura 3.10: *Heatmap* calendarizado dos alarmes "Trunk", para Julho 2020.

Em qualquer uma das figuras, é possível retirar a conclusão que os alarmes ocorridos no mês de Julho de 2020 são muito mais frequentes durante os dias da semana (de segunda a sexta) e quase inexistentes durante o fim-de-semana.

Pela Figura 3.10, conclui-se que os alarmes do tipo "Trunk" que ocorreram durante o mês de Julho de 2020, são na sua maioria registados fora do horário de trabalho, aproximadamente entre as 22h e as 10h. Isto leva a crer que os problemas deste tipo são imediatamente tratados quando a rede e os serviços estão sob vigia.

A baixa densidade de alarmes ao fim-de-semana pode ser justificada por haver uma vigilância mais apertada durante todo o dia de Sábado e Domingo ou, pelo contrário, não haver a devida monitorização dos dispositivos da rede, em que poucos são os alarmes que acabam por ser registados.

3.2.5 Engenharia de dados

A engenharia de dados é o processo de criação de novas variáveis, baseadas nas variáveis já existentes. Existem dois tipos de variáveis que podem ser criadas: derivadas e *dummy*.

As variáveis derivadas são, como o próprio nome indica, variáveis que derivam de outras já existentes nos dados. No *dataset* em estudo, existe uma variável que pode dar origem a outras mais: ALARM_DATETIME. Esta variável, que contém o *timestamp* completo do registo de cada alarme, pode ser dividida em várias novas colunas como, por exemplo: ano, mês, dia, hora, minuto e segundo. A partir do ano, mês e dia é possível saber qual o dia da semana em que o registo foi efetuado. Esta variável poderá revelar-se útil pois, como analisado anteriormente, existe uma grande diferença entre os volumes diários de alarmes durante os dias úteis e o fim-de-semana.

As variáveis *dummy* são sempre necessárias quando existem colunas categóricas no *dataset* em estudo. Como os algoritmos de ML, que irão utilizar estes dados como *input*, só conseguem processar valores numéricos, é necessário transformar as variáveis qualitativas em variáveis quantitativas. Este processo é exemplificado na Figura 3.11.

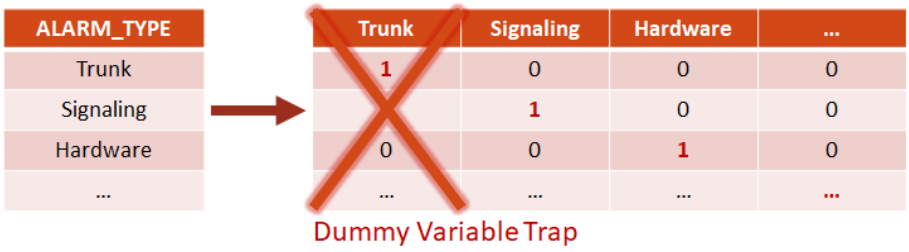


Figura 3.11: Exemplo do processo de criação de variáveis *dummy*.

No exemplo ilustrado foi realizada a transformação da coluna categórica ALARM_TYPE em várias colunas *dummy*, tantas quanto o número de tipos de alarme distintos existentes. Estas colunas tomarão, em cada linha do *dataset*, os valores 1 ou 0, conforme o alarme pertença ou não ao tipo associado a uma dada coluna *dummy*.

Por causa desta dependência intrínseca, que pode facilmente ser capturada e indevidamente modelada pelos algoritmos de ML, é necessário resolver o problema de *dummy variable trap*. Para isso, basta remover completamente uma das colunas *dummy* para eliminar quaisquer dependências inerentes a estas variáveis que foram criadas.

Nenhuma informação é perdida ao remover a coluna "Trunk", pois, para um dado alarme pertencente a esse tipo, bastaria verificar que todas as outras variáveis *dummy* se encontram a 0 para saber que existiria um 1 nessa coluna e inferir qual o seu tipo de alarme.

Capítulo 4

Metodologia

Neste capítulo serão apresentados os tópicos-chave que foram o foco do trabalho desenvolvido na prática. É feito o estudo sobre os principais algoritmos de *Association Rule Learning* e de *Sequential Pattern Mining* e, ainda, formas de os avaliar.

4.1 Aprendizagem de Regras de Associação

A aprendizagem de regras de associação permite extrair correlações ou associações interessantes, escondidas entre conjuntos de itens presentes em transações que se encontrem num *dataset* [40].

Este problema das regras de associação foi introduzido pela primeira vez por Agrawal [41], que define uma transação como sendo um conjunto de itens onde cada item pode ter diferentes atributos. O *dataset* em causa contém várias transações. Uma regra de associação é uma implicação, escrita como $A \rightarrow C$, onde A e C são conjuntos de itens, denominados antecedente e consequente, respetivamente.

Geralmente, existem duas medidas para avaliar uma regra de associação: suporte e confiança [41]. Como os *datasets* costumam armazenar grandes volumes de informação mas só as transações mais frequentes são interessantes, são definidos valores mínimos para estas métricas de avaliação, que ajudam a filtrar as regras menos frequentes.

- **Suporte**: Frequência relativa ou probabilidade de ocorrência de uma transação. Pode tomar valores entre 0 e 1.

$$Suporte(A \rightarrow C) = \frac{\text{Número de ocorrências de } A \cup C}{\text{Número total de transações}}$$

  O suporte de um item mede a sua importância estatística. Se o seu suporte for baixo (por exemplo, abaixo de 1%) então o item é raro e pode não ser interessante considerá-lo. Mas, dependendo do caso em estudo, pode ser importante não descartar os itens menos frequentes, quer seja para revelar essa situação e tentar torná-lo mais frequente ou porque todos os itens, sem exceção, devam ser considerados.

- **Confiança**: Probabilidade de uma transação conter o consequente sabendo que também contém o antecedente. A confiança é máxima (igual a 1) se o consequente e o antecedente ocorrerem sempre em conjunto. Não é simétrica, isto é, a confiança de $A \rightarrow C$ é diferente da confiança de $C \rightarrow A$.

$$\text{Confiança}(A \rightarrow C) = \frac{Suporte(A \rightarrow C)}{Suporte(A)}$$

  A confiança de uma regra mede a força da sua associação. Se a sua confiança for elevada (por exemplo, acima de 50%) significa que a maioria das transações que contêm A, também contêm C, o que demonstra a probabilidade da co-ocorrência desses dois itens.

Para além destas duas medidas, podem ser usadas outras para melhor classificar as regras associativas, ao considerar outras propriedades que tanto o suporte como a confiança não consigam quantificar [42]. No âmbito deste trabalho foram usadas mais três métricas de avaliação: *lift* [43], *leverage* [44] e convicção [43]. Estas métricas não servem para filtrar regras mas sim para melhor avaliá-las.

- ***Lift***: Quantifica o quão mais frequente é a ocorrência simultânea de A e C comparativamente ao que seria de esperar se fossem estatisticamente independentes. Se A e C forem independentes, o *lift* será exatamente igual a 1.

$$Lift(A \rightarrow C) = \frac{\text{Confiança}(A \rightarrow C)}{Suporte(C)}$$

- ***Leverage***: Diferença entre a frequência de A e C ocorrerem em conjunto e a frequência que seria de esperar se A e C fossem independentes. Um valor de 0 indica independência entre os dois conjuntos de itens.

$$Leverage(A \rightarrow C) = Suporte(A \rightarrow C) - Suporte(A) \times Suporte(C)$$

- **Convicção**: Um valor elevado significa que C é fortemente dependente de A. Por exemplo, no caso da confiança ser 1, o denominador será 0, pelo que a convicção será ∞. Tal como o *lift*, se os itens forem independentes, a convicção será igual a 1.

$$\text{Convicção}(A \rightarrow C) = \frac{1 - Suporte(C)}{1 - \text{Confiança}(A \rightarrow C)}$$

Existem diversos algoritmos que dão resposta a este problema de mineração de regras de associação. Neste trabalho o foco estará direcionado para os três mais populares: *Apriori* [45], *Eclat* [46] e *FPGrowth* [47].

4.1.1 *Apriori*

Este algoritmo acede várias vezes à base de dados para obter os conjuntos de itens frequentes. Como exemplo utilizar-se-á o *dataset* representado na Tabela 4.1.

Na primeira leitura, o suporte é contabilizado para cada item individualmente, construindo-se a Tabela 4.2. Com o valor de suporte mínimo definido como 50%, isto é, 2 ocorrências num total de 4 transações, o item I_2 é excluído da Tabela 4.3.

Para a construção da lista de candidatos com 2 itens (candidatos de nível 2), já não são tidas em conta os conjuntos que contêm o item I_2, porque se I_2 não faz parte da lista de itens frequentes de nível 1, então todos os sub-conjuntos que o contenham também não serão frequentes (propriedade *Apriori*).

Observando a Tabela 4.4, verifica-se que apenas dois dos candidatos de 2º nível ocorrem pelo menos 2 vezes e portanto, o conjunto de itens $\{I_1, I_3\}$ é removido e constrói-se a lista de itens frequentes de nível 2 da forma apresentada na Tabela 4.5.

Tabela 4.1: Exemplo de base de dados (representação horizontal).

TID	**Conjunto de itens**
1	$\{I_1, I_2, I_4\}$
2	$\{I_3, I_4\}$
3	$\{I_1, I_4\}$
4	$\{I_1, I_3, I_4\}$

Tabela 4.2: Número de ocorrências de candidatos de nível 1.

Item	**Nº ocorrências**
$\{I_1\}$	3
$\{I_2\}$	1
$\{I_3\}$	2
$\{I_4\}$	4

Tabela 4.3: Itens frequentes de nível 1.

Itens frequentes
$\{I_1\}$
$\{I_3\}$
$\{I_4\}$

Tabela 4.4: Número de ocorrências de candidatos de nível 2.

Item	**Nº ocorrências**
$\{I_1, I_4\}$	3
$\{I_3, I_4\}$	2
$\{I_1, I_3\}$	1

Tabela 4.5: Itens frequentes de nível 2.

Itens frequentes
$\{I_1, I_4\}$
$\{I_3, I_4\}$

Este processo é novamente repetido para níveis superiores até deixar de existir conjuntos de itens frequentes. Neste exemplo não existem candidatos de nível 3 e, por isso, os últimos conjuntos de itens frequentes são os de nível 2.

A lista final de itens frequentes é a junção de todas as listas criadas para cada um dos níveis, onde constam os valores de suporte calculados associados a cada conjunto de itens frequente.

4.1.2 *Eclat*

O *Eclat* é uma versão melhorada do algoritmo *Apriori* [46]. Enquanto que o *Apriori* utiliza uma representação horizontal da base de dados, exemplificada na Tabela 4.1, o *Eclat* transforma-a na sua representação vertical (Tabela 4.6) onde, em vez de estarem indicados os conjuntos de itens que pertencem a cada transação, são listadas as transações em que cada item ocorre.

Tabela 4.6: Exemplo de base de dados (representação vertical).

Item	**Lista de transações**
I_1	$\{1, 3, 4\}$
I_2	$\{1\}$
I_3	$\{2, 4\}$
I_4	$\{1, 2, 3, 4\}$

As listas de transações para conjuntos de itens de nível superior são criadas recursivamente, calculando a interseção das listas de transações (do nível anterior) de cada um dos itens. Se a interseção for nula, o conjunto de itens é removido da lista. Ora, por exemplo, a lista de transações para o conjunto de itens $\{I_1, I_2\}$ será dada pela interseção entre $\{1, 3, 4\}$ e $\{1\}$, resultando na lista $\{1\}$. A lista completa de nível 2 encontra-se representada na Tabela 4.7. Para o nível 3 foi construída a Tabela 4.8. Não existe nenhum conjunto de itens de nível 4.

Tabela 4.7: Lista de transações para conjuntos de itens de nível 2.

Conjunto de itens	**Lista de transações**
$\{I_1, I_2\}$	$\{1\}$
$\{I_1, I_3\}$	$\{4\}$
$\{I_1, I_4\}$	$\{1, 3, 4\}$
$\{I_2, I_4\}$	$\{1\}$
$\{I_3, I_4\}$	$\{2, 4\}$

Assumindo um valor de suporte mínimo de 50%, ou seja, 2 ocorrências num total de 4 transações, a lista final com todos os conjuntos de itens frequentes pode ser verificada na Tabela 4.9.

Tabela 4.8: Lista de transações para conjuntos de itens de nível 3.

Conjunto de itens	**Lista de transações**
$\{I_1, I_2, I_4\}$	$\{1\}$
$\{I_1, I_3, I_4\}$	$\{4\}$

Tabela 4.9: Lista final de conjuntos de itens frequentes.

Conjunto de itens frequentes
$\{I_1\}$
$\{I_3\}$
$\{I_4\}$
$\{I_1, I_4\}$
$\{I_3, I_4\}$

O resultado final é idêntico ao do algoritmo *Apriori*, mas o *Eclat* ocupa menos memória durante todo o seu processo, consegue ser mais rápido por utilizar uma abordagem vertical - em que os seus cálculos são feitos de forma paralela - e acaba por executar menos leituras à base de dados, porque é possível calcular os valores de suporte em qualquer uma das listas criadas para cada um dos níveis dos conjuntos de itens.

4.1.3 *FPGrowth*

O algoritmo *FPGrowth* [47] começa por ler a base de dados das transações para calcular o suporte de todos os conjuntos de itens individuais (conjuntos de itens de nível 1). Todos os itens raros, isto é, aqueles cujo suporte seja menor que o valor mínimo de suporte especificado pelo utilizador, são descartados, pois nunca farão parte de um conjunto de itens frequente.

Para além disso, em cada transação, os itens são ordenados pelo seu suporte por ordem decrescente. Embora o algoritmo não imponha esta ordenação, testes experimentais demonstraram que assim são obtidos tempos de execução menores comparativamente com aqueles que se registam quando os itens são dispostos aleatoriamente [48].

Esta fase do processo é demonstrada na Tabela 4.10, recuperando o exemplo da Tabela 4.1. À esquerda encontra-se a lista de transações inicial, ao centro está a contabilização do número de ocorrências de cada item (ordenadas de forma decrescente) e à direita encontram-se as mesmas transações mas com os itens ordenados do mais frequente para o menos frequente.

Assumindo um valor mínimo de suporte de 50%, isto é, 2 ocorrências num total de 4 transações, o item I_2 é descartado e por isso removido de todas as transações onde se encontrava inicialmente presente.

Tabela 4.10: Ordenação dos itens de cada transação.

Transação
$\{I_1, I_2, I_4\}$
$\{I_3, I_4\}$
$\{I_1, I_4\}$
$\{I_1, I_3, I_4\}$

Item	**Ocorrências**
I_4	4
I_1	3
I_3	2
I_2	1

Transação ordenada
$\{I_4, I_1\}$
$\{I_4, I_3\}$
$\{I_4, I_1\}$
$\{I_4, I_1, I_3\}$

Na próxima fase, é construída a chamada *FP-tree*, uma árvore que contém os prefixos das transações. Cada caminho da árvore representa um conjunto de transações que partilham o mesmo prefixo, onde cada nó corresponde a um único item. Além disso, todos os nós referentes a um mesmo item são ligados entre si, para que todas as transações que contenham um determinado item possam ser facilmente encontradas e contabilizadas ao percorrer esta árvore.

A *FP-tree* correspondente à lista de transações ordenadas da tabela anterior é apresentada na Figura 4.1.

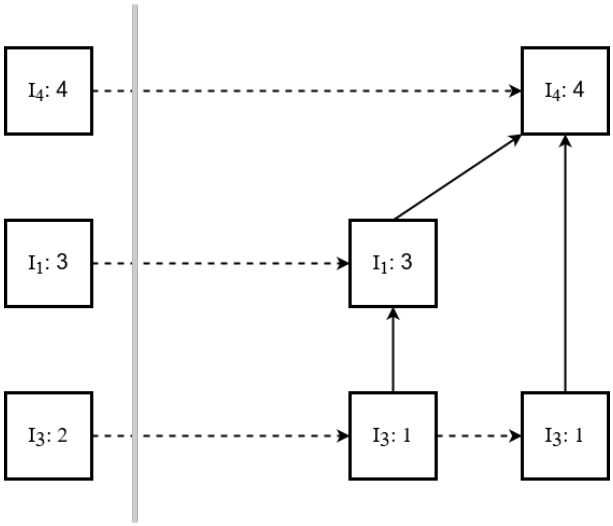


Figura 4.1: *FP-tree* da lista de transações ordenadas da Tabela 4.10.

A principal operação que o algoritmo *FPGrowth* tem de realizar é a de construir a *FP-tree* de uma base de dados projetada, isto é, uma base de dados com as transações que contêm um certo item, com esse item removido. Esta base de dados projetada é processada recursivamente, não esquecendo que os conjuntos de itens frequentes encontrados partilham do mesmo prefixo - o item que foi removido. A *FP-tree* para a base de dados projetada do item I_3 é ilustrada na Figura 4.2.

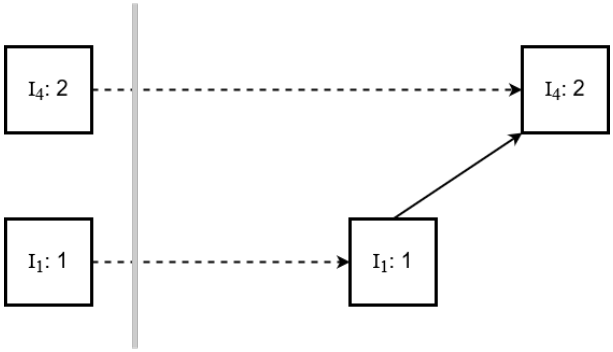


Figura 4.2: *FP-tree* projetada do prefixo I_3.

Note-se que nesta árvore o item I_1 é raro, pois só ocorre 1 única vez quando o suporte mínimo é de 2 ocorrências. Logo, não seria tido em conta pelo algoritmo, mas foi incluído na figura para melhor ilustrar a sua relação com a Figura 4.1.

Depois de construir as *FP-trees* para todas as projeções da base de dados necessárias, é realizado o processo de eliminação de alguns dos nós associados a itens raros de modo a simplificar a árvore e acelerar o processo. Para exemplificar esta situação foi construída uma *FP-tree* que nada tem a ver com os exemplos anteriores mas que ajuda a entender melhor esta fase da execução do algoritmo. A árvore inicial encontra-se à esquerda da Figura 4.3, enquanto que a final, resultante da remoção do item raro I_2, situa-se à direita. Veja-se que, como o item que foi removido dava origem a uma ramificação, a árvore acabou por ficar bastante mais simples, tornando-se numa árvore com um ramo único.

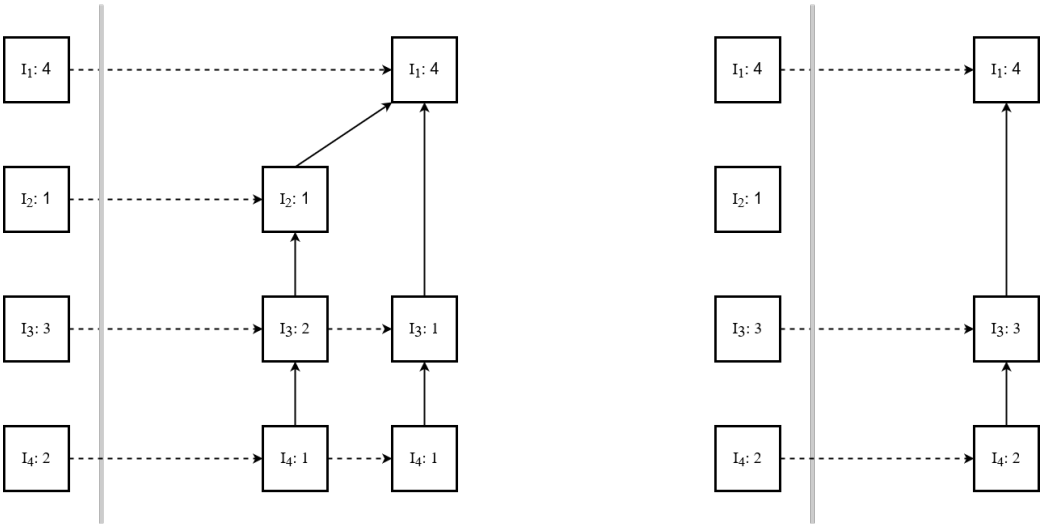


Figura 4.3: Remoção dos nós raros de uma *FP-tree*.

Graças a uma implementação eficiente das *FP-trees*, o algoritmo *FP-Growth* supera em larga escala os algoritmos apresentados anteriormente (*Apriori* e *Eclat*), tanto em termos de tempo de execução como da utilização de memória necessária, já que o armazenamento da base de dados utilizando uma representação em árvore é mais compacta do que a lista completa de transações.

4.2 Reconhecimento de Padrões Sequenciais

A mineração de dados permite extrair informações relevantes armazenadas em *datasets* por forma a compreendê-los melhor e tomar decisões com base nessa recolha. O reconhecimento de padrões consiste em descobrir padrões escondidos nos dados que possam revelar-se úteis de serem explorados.

Esta área de investigação iniciou-se com o trabalho de Agrawal e Srikant nos anos 90 [45]. Nesse artigo é introduzido o algoritmo *Apriori* que permite encontrar, por exemplo, nas transações de clientes de supermercado, conjuntos de itens frequentes, isto é, produtos que costumem comprar em conjunto. Com isso, é possível estudar o comportamento dos consumidores e tomar decisões estratégicas para aumentar o volume de vendas, como, por exemplo, fazer promoções em vários produtos de forma conjunta.

Existem diversas técnicas de reconhecimento de padrões, mas as mais populares, como as de aprendizagem de regras de associação, não têm em conta a ordem dos itens numa determinada transação. Assim, estas técnicas não podem ser usadas para resolver problemas com séries temporais, em que é vital conhecer a sequência dos eventos.

Para abordar esta questão, foi proposto o reconhecimento de padrões sequenciais. Tal como o reconhecimento de padrões, permite descobrir conjuntos de itens frequentes numa determinada transação, com a diferença de que agora é tida em conta a ordem dos itens, passando a designar-se por "sequência" em vez de "transação".

O objetivo do reconhecimento de padrões sequenciais é o de encontrar todos os padrões (sub-sequências) que têm um suporte maior ou igual ao valor de suporte mínimo definido pelo utilizador. Logo, existe uma única resposta a um problema deste tipo. O que diferencia os vários algoritmos existentes é apenas a sua eficiência para encontrar essa mesma resposta [49].

Para resolver esta questão, poderia-se pensar (ingenuamente) que bastaria calcular o suporte para todos os padrões possíveis e remover aqueles que não respeitassem o limite mínimo imposto. No entanto, tal abordagem é bastante ineficiente porque o número de padrões encontrados é, tipicamente, bastante

elevado. Uma sequência que tenha n itens pode chegar a $2^n - 1$ padrões. Por isso, é necessário implementar algoritmos que não precisem de explorar todo o leque de possíveis padrões, ao utilizarem diferentes estratégias que os permitam ser eficientes.

A principal estratégia para evitar explorar todos os padrões possíveis é ter em conta a propriedade *Apriori*, que diz que qualquer sub-sequência tem, de certeza, um suporte menor ou igual ao suporte da sequência que a origina. Isto significa que se uma sequência não cumpre o requisito mínimo para o valor de suporte, para além dela própria, todas as suas sub-sequências podem ser descartadas.

Em geral, os algoritmos de reconhecimento de padrões sequenciais podem ser divididos em dois tipos: *Breadth-First Search* (BFS) e *Depth-First Search* (DFS).

Os algoritmos BFS, como o *GSP* [50], exploram primeiro todo o *dataset* à procura dos padrões frequentes de nível 1 (que contêm apenas um item) e utilizam-nos para gerar os padrões de nível 2. Estes são usados para gerar os de nível 3 e assim sucessivamente até não existirem padrões de nível superior. O suporte é calculado em cada uma dessas fases e é tirado proveito da propriedade *Apriori* para acelerar o processo.
Algumas das desvantagens deste tipo de algoritmos são os múltiplos acessos ao *dataset* e o armazenamento em memória da lista de candidatos (que pode levar ao seu uso excessivo).

Os algoritmos DFS começam também pelas sequências de primeiro nível, mas depois utilizam apenas uma dessas sequências para gerarem, recursivamente, sequências de nível superior. Assim que não existam padrões maiores, o algoritmo volta ao primeiro passo para utilizar outra das sequências de nível 1 e repete o mesmo ciclo para todas essas sequências base.

Nesta secção é feito o estudo, aplicação e análise comparativa de três dos mais populares algoritmos deste tipo, nomeadamente o *PrefixSpan* [51], o *SPADE* [52] e o *SPAM* [53].

4.2.1 *PrefixSpan*

Para além de ser DFS, o *PrefixSpan* é também um algoritmo *pattern-growth*, tendo sido baseado no algoritmo *FPGrowth* [47]. É o único dos três algoritmos estudados que não considera todas as combinações possíveis para os padrões que podem ser encontrados. Acede recursivamente ao *dataset* para ir concatenando novos itens até formar o padrão completo, por isso, só considera os padrões que existem mesmo no *dataset*. Estes sucessivos acessos podem, no entanto, ser demorados, pelo que foi introduzido o conceito de "*dataset* projetado", de modo a diminuir a sua dimensão, otimizando o acesso. Em termos de memória, criar várias projeções do *dataset* pode ocupar bastante espaço no armazenamento dos dados.

4.2.2 *SPADE*

Inspirado no *Eclat* [46], este algoritmo utiliza uma representação vertical do *dataset*, criada durante a primeira leitura, que indica em que conjuntos de itens e em que sequência cada um dos itens se encontra.

A título de exemplo será usado o *dataset* representado nas Tabelas 4.11 e 4.12, que conta com 2 sequências, contendo cada uma 2 conjuntos de 2 itens.

Tabela 4.11: Representação horizontal do *dataset*.

SID	**Sequência**
1	[{a,b}, {a,c}]
2	[{b,c}, {a,b}]

Analisando a primeira tabela, é possível verificar que a primeira sequência contém os conjuntos {a,b} e {a,c} e a segunda os conjuntos {b,c} e {a,b}. Da mesma forma, podemos ver pela segunda tabela que, por exemplo, o item 'a' encontra-se nos 2 conjuntos da primeira sequência e apenas no segundo conjunto da segunda sequência.

A representação vertical conta com duas propriedades interessantes para o reconhecimento de padrões sequenciais. A primeira propriedade é que a lista criada para qualquer sequência permite calcular diretamente o seu suporte. Por exemplo, a lista do padrão {a,c} é {(s_1,1), (s_1,2)}, porque este padrão

Tabela 4.12: Representação vertical do *dataset*.

a	
SID	**Conjuntos de itens**
1	1, 2
2	2

b	
SID	**Conjuntos de itens**
1	1
2	1, 2

c	
SID	**Conjuntos de itens**
1	2
2	1

só pode ser encontrado na primeira sequência (s_1). Por existir apenas uma sequência distinta onde esse padrão é encontrado, o seu suporte é igual a 1, ou, em termos relativos, 1/2, por existirem no total 2 sequências.

A segunda propriedade é que a lista de qualquer sequência pode ser obtida, sem aceder diretamente ao *dataset* original, ao fazer *join* das várias listas das sub-sequências que a compõem [52].

Ao tirar partido destas propriedades, algoritmos como o *SPADE* e até mesmo o *SPAM* realizam a sua descoberta por padrões sequenciais sem aceder repetidamente ao *dataset* e, por isso, sem manter em memória um elevado número de padrões (ao contrário dos algoritmos BFS).

4.2.3 *SPAM*

Similar ao algoritmo apresentado anteriormente, o *SPAM* consegue ser ainda mais eficiente ao otimizar a estrutura da lista de padrões. Este algoritmo codifica estas listas como vetores binários, o que permite poupar a memória utilizada para armazenar a mesma informação. Para além disso, acelera as operações matemáticas que necessitam de ser realizadas. Pode ainda ser melhorado com o uso de técnicas de compressão que reduzam o número de bits utilizados.

Foi demonstrado que este algoritmo consegue ser mais rápido que o *SPADE* e o *PrefixSpan*, principalmente para *datasets* relativamente grandes. Em termos de espaço em memória utilizado, o *SPADE* ainda consegue ser o mais eficiente entre os dois.

Capítulo 5

Resultados Experimentais

De modo a descobrir as regras de associação e padrões que surgem na rede, e para obter algumas conclusões práticas que verifiquem os resultados esperados na teoria, foram usados os dados reais de FM do *dataset* descrito e explorado no Capítulo 3.

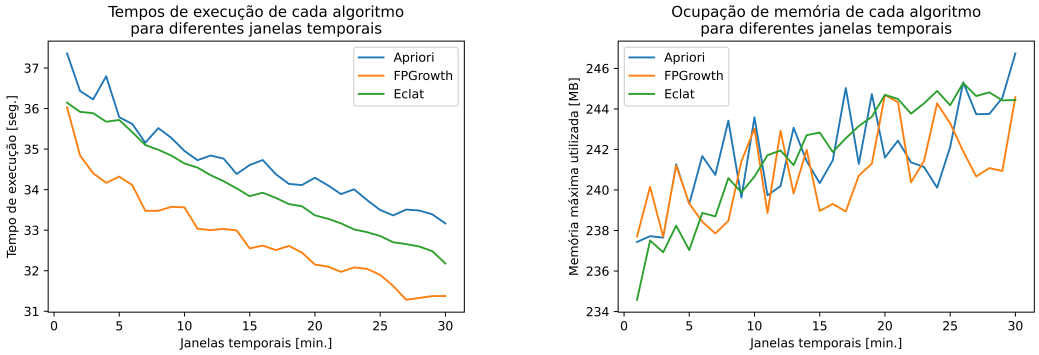
Na execução dos algoritmos implementados, foram apenas usados os nomes dos alarmes presentes na coluna ALARM_NAME, sem distinção do seu tipo ou severidade, dados pelas colunas ALARM_TYPE e SEVERITY, respetivamente. Esta análise foi feita por nó e para todos os nós da rede presentes no *dataset*, extraídos da coluna NODE_NAME. As transações foram definidas como sendo o agrupamento periódico de todos os alarmes que ocorreram durante um intervalo de tempo, utilizando a coluna ALARM_DATETIME.

Por forma a avaliar toda e qualquer regra de associação, o valor de suporte mínimo foi fixado em "2 ocorrências / Número total de transações", ou seja, basta haver uma repetição de um certo conjunto de itens para que seja considerado frequente. Foi ainda imposta uma confiança mínima de 50%, o que indica que, no mínimo, para uma determinada regra, metade das transações que contêm o antecedente contêm também o consequente.

Antes de analisar os resultados, e para avaliar a eficiência de cada algoritmo, foram medidos os tempos de execução e memória utilizada. A obtenção dos valores de memória utilizada foi feita através da biblioteca Memory Profiler [54]. Estes resultados foram obtidos em testes realizados numa máquina com as seguintes especificações: Intel® Core™ i5-7300HQ CPU @ 2.50GHz (4 CPUs), 8GB RAM, Windows 10 Education 64bits.

5.1 Aprendizagem de Regras de Associação

Na implementação dos algoritmos *Apriori* e *FPGrowth* foi usada a biblioteca MLxtend [55], enquanto que para o *Eclat* foi utilizada a PyFIM [56]. Os testes foram realizados para janelas temporais de 1 a 30 minutos, de minuto a minuto. Os gráficos comparativos dos algoritmos, para as diferentes janelas temporais, são apresentados na Figura 5.1.



(a) (b)

Figura 5.1: Tempos de execução (a) e memória utilizada (b) dos diferentes algoritmos, para diferentes janelas temporais.

Como se pode observar no gráfico (a) da Fig. 5.1, os tempos de execução de todos os algoritmos vão diminuindo à medida que se aumenta a janela temporal. Apesar de, com esse aumento, existirem transações com mais conjuntos de itens, o número de transações vai diminuindo, existindo menos métricas de avaliação para calcular, o que justifica a redução do tempo. Comparando os tempos de execução de cada algoritmo, é possível evidenciar que o mais rápido é o *FPGrowth*, seguido do *Eclat* e, por fim, o *Apriori* com o pior desempenho.

Quanto ao gráfico (b) da mesma figura, apesar das constantes oscilações, é possível verificar que a utilização de memória aumenta à medida que a janela temporal aumenta, o que se justifica pelo aumento do número de itens em cada transação. No geral, é possível concluir que o mais eficiente em termos de ocupação de memória é também o *FPGrowth*. O *Eclat* consegue ser bastante otimizado para pequenas transações, mas a partir de certo ponto passa a utilizar mais memória que o *Apriori*.

A lista de regras, idêntica a todos os algoritmos, foi ordenada pela métrica *Lift* de forma decrescente, pois aquelas que possuem o valor mais elevado são as que contam com uma maior dependência entre os alarmes antecedente e consequente da regra. As 10 primeiras regras são ilustradas na Tabela 5.1.

Tabela 5.1: As 10 regras de associação de alarmes, e respetivas métricas de avaliação calculadas pelos três algoritmos, com maior *Lift*.

Minuto(s)	**Nó da rede**	**Antecedente(s)**	**Consequente(s)**	**Suporte**	**Confiança**	**Lift**	**Leverage**	**Convicção**
15 - 19	BX94BL	RF Unit ALD Current Out of Range	ALD Maintenance Link Failure	0.00134 - 0.00150	1.0	741.5 - 665.0	0.00134 - 0.00150	∞
15 - 19	BX94BL	ALD Maintenance Link Failure	RF Unit ALD Current Out of Range	0.00134 - 0.00150	1.0	741.5 - 665.0	0.00134 - 0.00150	∞
1	VA08OL	X2 Interface Fault	Inter-System Cabinet Configuration Conflict	0.00151	1.0	662.0	0.00150	∞
1	VA08OL	Inter-System Cabinet Configuration Conflict	X2 Interface Fault	0.00151	1.0	662.0	0.00150	∞
20 - 21	BX94BL	RF Unit ALD Current Out of Range	ALD Maintenance Link Failure	0.00153 - 0.00157	1.0	650.0 - 634.0	0.00153 - 0.00157	∞
20 - 21	BX94BL	ALD Maintenance Link Failure	RF Unit ALD Current Out of Range	0.00153 - 0.00157	1.0	650.0 - 634.0	0.00153 - 0.00157	∞
2	VA08OL	X2 Interface Fault	Inter-System Cabinet Configuration Conflict	0.00158	1.0	632.0	0.00157	∞
2	VA08OL	Inter-System Cabinet Configuration Conflict	X2 Interface Fault	0.00158	1.0	632.0	0.00157	∞
3	VA83WL	ALD Maintenance Link Failure	RF Unit ALD Current Out of Range	0.00160	1.0	623.0	0.00160	∞
3	VA83WL	RF Unit ALD Current Out of Range	ALD Maintenance Link Failure	0.00160	1.0	623.0	0.00160	∞

Para além destas regras, foram também consideradas as regras com *Lifts* elevados da Tabela 5.2. As restantes regras com *Lift* elevado que não foram incluídas nestas tabelas são apenas variantes daquelas que aqui são apresentadas.

Essas regras foram definidas com base noutras janelas temporais e/ou nós da rede, mas como os conjuntos de itens que as compõem (antecedente/consequente) são idênticos, as métricas de avaliação não diferem muito daquelas que são apresentadas e, por isso, foram omitidas.

Tabela 5.2: Regras de associação adicionais, e respetivas métricas de avaliação calculadas pelos três algoritmos, com valores de *Lift* elevados.

Minuto(s)	Nó da rede	Antecedente(s)	Consequente(s)	Suporte	Confiança	Lift	Leverage	Convicção
2	BX16QL	Certificate Invalid	External Clock Reference Problem	0.00202	1.0	494.0	0.00202	∞
2	BX16QL	{Certificate Invalid, SCTP Link Fault}	External Clock Reference Problem	0.00134	1.0	494.0	0.00134	∞
2	BX16QL	External Clock Reference Problem	{Certificate Invalid, SCTP Link Fault}	0.00134	0.66	494.0	0.00134	2.9959
2	BX16QL	Certificate Invalid	{External Clock Reference Problem, SCTP Link Fault}	0.00134	0.66	494.0	0.00134	2.9959
2	BX16QL	{External Clock Reference Problem, SCTP Link Fault}	Certificate Invalid	0.00134	1.0	494.0	0.00134	∞
2	BX16QL	External Clock Reference Problem	Certificate Invalid	0.00202	1.0	494.0	0.00202	∞

É possível verificar, em qualquer uma das tabelas, que para uma determinada janela temporal e nó da rede, existe sempre um par de regras simétricas, isto é, existe sempre uma segunda regra em que o antecedente é o consequente e o consequente é o antecedente da primeira regra.

Na primeira tabela, nem sequer existem diferenças nas métricas de avaliação de cada um desses pares de regras. Na segunda tabela, já se identificam alterações nos valores de confiança e, por conseguinte, de convicção, nomeadamente nas regras em que o alarme "SCTP Link Fault" faz parte do seu consequente. Isto deve-se ao facto de, como se verificou no Capítulo 3, este alarme ser o mais frequente de todos e, por isso, haver uma elevada probabilidade de o encontrar em muitas das transações definidas.

Apesar dos valores de suporte serem baixos, os algoritmos demonstram convicção de que estas associações são bastante fortes, isto é, quando ocorre um determinado alarme antecedente, irá ocorrer de seguida o consequente.

Mas, se existem sempre pares de regras simétricos, como é possível ter a certeza de qual é mesmo o antecedente/consequente? Será que podem acontecer as 2 situações ou só uma destas regras é plausível? Para responder a estas questões é necessário definir uma ordem dentro de cada conjunto de alarmes e, como as regras de associação são probabilísticas, não têm uma ordem definida, tendo que ser usadas sequências/padrões de alarmes.

5.2 Reconhecimento de Padrões Sequenciais

Na implementação dos algoritmos foi utilizado um *wrapper* [57] para Python da extensiva *framework* de mineração de dados SPMF [58], originalmente desenvolvida em Java.

No desenvolvimento do trabalho relatado na secção anterior, percebeu-se que não havia necessidade de testar tantas janelas temporais (desde 1 até 30 minutos, de minuto a minuto) na definição das transações, pois os resultados são idênticos para todas elas. Assim, para estes algoritmos, as transações foram definidas com base numa única janela dinâmica, cujo tamanho depende do intervalo de tempo entre cada dois alarmes. Existe um intervalo de tempo limite, designado de *timeout*, em que se nenhum alarme acontecer dentro dessa janela, fecha-se a transação atual e o próximo alarme que ocorra já pertencerá à próxima transação. Isto significa que poderão haver, no mínimo, alarmes isolados (transações com 1 único alarme) ou, no máximo, transações que podem demorar horas (provavelmente se existir uma longa rajada de alarmes do alarme mais frequente "SCTP Link Fault"). No entanto, o algoritmo só irá avaliar uma única lista de transações, em vez de 30 listas diferentes como fazia anteriormente.

O tempo de *timeout* foi definido tendo em conta as regras de associação predominantes no capítulo anterior, nomeadamente aquelas que continham alarmes associados ao ALD: RF Unit ALD Current Out of Range e ALD Maintenance Link Failure. Como o objetivo é que esta sequência de alarmes esteja sempre na mesma transação, foi feita uma pesquisa profunda ao *dataset* e verificou-se que este par de alarmes ocorre, no máximo, com um intervalo de tempo de 10 minutos e 36 segundos. Para simplificar, e porque pode dar-se o caso deste intervalo ser ainda maior, este valor foi arredondado por excesso e o tempo de *timeout* foi então definido como 11 minutos.

As medidas de desempenho dos algoritmos, para a janela dinâmica com *timeout* de 11 minutos, são apresentadas na Tabela 5.3.

Tabela 5.3: Tempo de execução e memória ocupada pelos diferentes algoritmos, para a janela dinâmica.

Algoritmo	**Tempo de execução**	**Memória máxima utilizada**
PrefixSpan	14 minutos 43 segundos	179.07 MB
SPADE	15 minutos 9 segundos	178.06 MB
SPAM	13 minutos 50 segundos	180.18 MB

Observando o valor médio de 3 testes executados de seguida, verifica-se que o algoritmo mais rápido é o *SPAM*, seguido do *PrefixSpan*, e finalmente o *SPADE*. Em termos de memória ocupada, a ordem inverte-se, sendo o *SPADE* o mais eficiente, seguido do *PrefixSpan* e, por fim, o menos eficiente é o *SPAM*. Estes resultados validam o que foi estudado sobre cada algoritmo.

Dependendo do *dataset* em estudo e dos recursos disponíveis, qualquer um dos algoritmos é viável e tem os seus prós e contras. Para uma versão final do *software* realizado no âmbito deste trabalho, a escolha recai sobre o *SPAM* pois a diferença na memória utilizada por todos é quase nula, enquanto que em termos de tempo de execução leva uma vantagem de quase 1 minuto para o segundo algoritmo mais rápido.

A lista de padrões, idêntica em todos os algoritmos, foi ordenada pela métrica *Lift* de forma decrescente, pois aquelas que têm este valor mais elevado são as que contam com uma maior dependência entre todos os alarmes que compõem o padrão. Os 5 primeiros padrões são ilustrados na Tabela 5.4.

Como se pode verificar, continuam a surgir os mesmos alarmes já identificados nas regras de associação do capítulo anterior, nomeadamente: *RF Unit ALD Current Out of Range, ALD Maintenance Link Failure, Inter-System Cabinet Configuration Conflict, X2 Interface Fault, Certificate Invalid, External Clock Reference Problem.* No entanto, desta vez já não existem regras simétricas, apenas os padrões propriamente ditos que, por si só, já indicam qual a ordem de ocorrência dos alarmes.

Tabela 5.4: Os 5 padrões sequenciais, e respetivas métricas de avaliação calculadas pelos três algoritmos, com maior *Lift*.

Nó da rede	Padrão	Suporte	Confiança	Lift	Leverage	Convicção
VA08OL	{Inter-System Cabinet Configuration Conflict, X2 Interface Fault}	0.00259	1.0	386.5	0.00258	∞
VA83WL	{RF Unit ALD Current Out of Range, ALD Maintenance Link Failure}	0.00274	1.0	364.33	0.00274	∞
BX16QL	{Certificate Invalid, External Clock Reference Problem}	0.00197	0.66	226.0	0.00196	2.9912
AB42AL	{Inter-System Cabinet Configuration Conflict, X2 Interface Fault}	0.00476	1.0	210.0	0.00474	∞
TX49BL	{ALD Maintenance Link Failure, External Clock Reference Problem}	0.00477	1.0	209.66	0.00475	∞

Da Tabela 5.4 pode-se concluir que:

- A repetição do padrão {Inter-System Cabinet Configuration Conflict, X2 Interface Fault} para 2 nós diferentes da rede (VA08OL e AB42AL) pode significar que este padrão não é específico de um só nó, mas que pode acontecer em qualquer zona da rede;
- A presença do padrão {RF Unit ALD Current Out of Range, ALD Maintenance Link Failure} reforça ainda mais a sua importância e justifica o porquê de ter sido usado para a definição do tempo de *timeout*;
- O padrão {ALD Maintenance Link Failure, External Clock Reference Problem} cujo antecedente é o consequente do padrão anterior, pode indiciar que o alarme "ALD Maintenance Link Failure" que foi despoletado por causa do "RF Unit ALD Current Out of Range", pode ele próprio dar origem ao alarme "External Clock Reference Problem", caso esta sequência não se verifique apenas localmente;
- O alarme "External Clock Reference Problem" é consequente em 2 padrões diferentes, o que pode significar que tem diferentes causas e reflete a importância da ordem dentro de uma sequência: apesar de se conhecer o consequente, não implica que tenha acontecido um dado antecedente.

Para validar os resultados obtidos e entender se estes padrões podem mesmo surgir na realidade, foram usadas as documentações de cada alarme, obtidas através da plataforma de documentação associada ao Fornecedor 1, de modo a tentar relacionar os pares de alarmes de cada padrão da tabela anterior. Além disso, assumindo que quando se resolve o antecedente, o consequente já não ocorre e que o intervalo de tempo entre si é suficiente para relatar a falha, despoletar uma *work order* fictícia e resolver o problema, o consequente será removido do *dataset* original, sendo calculada qual a redução máxima do número de alarmes verificada.

RF Unit ALD Current Out of Range → ALD Maintenance Link Failure

Começando pelos alarmes associados ao ALD, criou-se o quadro comparativo presente na Tabela 5.5.

Tabela 5.5: Quadro comparativo entre os alarmes RF Unit ALD Current Out of Range e ALD Maintenance Link Failure.

	RF Unit ALD Current Out of Range	**ALD Maintenance Link Failure**
Descrição	Reportado quando a corrente fornecida pela unidade RF ao ALD através do *feeder* ou de cabos multi-fios está para lá dos valores normais de operação.	Reportado quando a ligação de manutenção entre a unidade RF e o ALD (antena RET, TMA, RAE ou SASU) não funciona.
Severidade	**Minor**/Warning	**Major**
Possíveis causas	O *feeder* está mal conectado ou encharcado; O *feeder* está deformado; O *jumper* entre o *feeder* e a antena está solto; O circuito de alimentação do ALD presente na unidade RF está com problemas; O dispositivo de antena está com problemas.	A corrente na antena vinda da unidade RF está fora do intervalo de valores normais.

Para além de fazer sentido que um alarme *Minor*, ou *Warning*, dê origem a um alarme *Major*, a possível causa do "ALD Maintenance Link Failure" é semelhante à descrição do "RF Unit ALD Current Out of Range", o que valida a sequência encontrada pelos algoritmos.

No âmbito da manutenção preventiva da rede, há que perceber qual o possível ganho ao conseguir reconhecer os padrões que aconteceram no passado de forma a tentar prever e prevenir o que aconteceria no futuro. Neste caso, ao retornar o valor da corrente, que chega à antena vinda da unidade RF, ao intervalo de valores normais, antes que se dê a falha na ligação de manutenção do ALD, 715 alarmes "ALD Maintenance Link Failure" são removidos. Isto traduz-se, principalmente, na redução de:

- Alarmes do tipo "Communication": 1860 → 1145 (-38.44%);
- Alarmes do nó "VX73KL": 40 → 22 (-45%);
- Alarmes do nó "CS12KL": 87 → 51 (-41.38%);
- Alarmes do nó "VX37KL": 179 → 107 (-40.22%);
- Alarmes do nó "CR29VL": 64 → 44 (-31.25%);
- Alarmes do nó "VX58KL": 164 → 122 (-25.61%);
- Alarmes do nó "VX20KL": 373 → 318 (-14.75%).

Inter-System Cabinet Configuration Conflict → X2 Interface Fault

Para a sequência seguinte, a partir dos alarmes "Inter-System Cabinet Configuration Conflict" e "X2 Interface Fault", foi construída a Tabela 5.6.

Mais uma vez, um alarme "Minor" dá origem a um alarme de severidade "Major". Apesar das possíveis causas não oferecerem nenhuma informação relevante quanto à relação entre estes dois alarmes, ambas as descrições falam de configurações erradas ou inconsistentes, o que leva a crer que basta as configurações do *cabinet* não serem consistentes entre todos os modos, numa estação-base multimodo, para que a ligação X2 não possa ser efetuada.

Resolvendo estes erros e conflitos nas configurações que podem levar à falha na interface X2, conseguem-se prevenir 6327 alarmes "X2 Interface Fault" de acontecerem.

Tabela 5.6: Quadro comparativo entre os alarmes Inter-System Cabinet Configuration Conflict e X2 Interface Fault.

	Inter-System Cabinet Configuration Conflict	**X2 Interface Fault**
Descrição	Numa estação-base multimodo, este alarme é reportado quando as configurações do *cabinet* são inconsistentes entre diferentes modos, como, por exemplo, o tipo de *cabinet*.	Reportado quando a ligação X2 não pode ser efetuada porque o X2AP foi mal configurado, o eNB emparelhado está num estado anormal, ou a ligação X2 que suporta o SCTP ficou indisponível durante 18 segundos seguidos.
Severidade	**Minor**	**Major**
Possíveis causas	O tipo de *cabinet* está configurado incorretamente para um modo.	A interface X2 está mal configurada; A estação-base emparelhada não está configurada com uma interface X2; A estação-base local está na lista negra da estação-base à qual se emparelhou; A interface S1 do eNB local/emparelhado está bloqueada.

Isto traduz-se, principalmente, na redução de:

- Total de alarmes: 191053 → 184726 (-3.31%);
- Alarmes do tipo "Signaling": 8981 → 2654 (-70.45%);
- Alarmes do nó "LO01AL": 37 → 18 (-51.35%);
- Alarmes do nó "VX83EL": 74 → 37 (-50%);
- Alarmes do nó "VX49KL": 53 → 27 (-49.06%);
- Alarmes do nó "VX41KL": 195 → 107 (-45.13%);
- Alarmes do nó "VX31DL": 189 → 107 (-43.39%);
- Alarmes do nó "VX78EL": 200 → 120 (-40%);
- Alarmes do nó "VX47EL": 203 → 129 (-36.45%).

Certificate Invalid → External Clock Reference Problem

Para a sequência dos alarmes "Certificate Invalid" e "External Clock Reference Problem", foi obtida a Tabela 5.7.

Tabela 5.7: Quadro comparativo entre os alarmes Certificate Invalid e External Clock Reference Problem.

	Certificate Invalid	**External Clock Reference Problem**
Descrição	Reportado quando o NE deteta que um certificado não está ativo, expirou ou houve uma falha no seu processamento. Este alarme é limpo quando o NE deteta que o certificado foi processado com sucesso e está ativo.	Reportado no caso de existir uma falha na sincronização do *clock* que perdura por um período de tempo (normalmente de 200 segundos). Este alarme é limpo quando a sincronização do *clock* é bem sucedida e mantém-se dessa forma durante um período de tempo (normalmente de 900 segundos).
Severidade	**Major**	**Major**/Minor
Possíveis causas	O certificado expirou ou foi revogado; O ficheiro do certificado foi perdido; O ficheiro do certificado foi danificado; O formato do ficheiro do certificado está incorreto; O certificado não está ativo.	O modo de funcionamento do *clock* ou o *clock* de referência do NE está mal configurado; O modo de sincronização do *clock* do NE está mal configurado; O *hardware* da placa onde o *clock* é originado está avariado; O *hardware* da placa de controlo principal está avariado; O *hardware* de um nó de interconexão está avariado.

Desta vez, é um alarme "Major" que dá origem a outro alarme "Major". Apesar da sua relação não ser óbvia, tanto a descrição do primeiro alarme como as possíveis causas do segundo referem o NE como a entidade que controla a regulação dos certificados e a configuração dos *clocks*. Leva a crer que quando o NE deteta a invalidade do certificado, as suas configurações relacionadas com o *clock* não são devidamente carregadas e originam o problema relatado pelo segundo alarme. No caso do certificado ter expirado, bastará renová-lo para reativar as corretas configurações do *clock*.

Ao validar corretamente o certificado, que pode resolver a falha na sincronização do *clock* de referência externo, podem-se prevenir bastantes alarmes "External Clock Reference Problem" de acontecerem. Como este padrão partilha do mesmo consequente do próximo, o cálculo e análise deste impacto são feitos para o padrão seguinte.

ALD Maintenance Link Failure → External Clock Reference Problem

Para a última sequência, dos alarmes "ALD Maintenance Link Failure" e "External Clock Reference Problem", foi construída a Tabela 5.8.

Tabela 5.8: Quadro comparativo entre os alarmes ALD Maintenance Link Failure e External Clock Reference Problem.

	ALD Maintenance Link Failure	**External Clock Reference Problem**
Descrição	Reportado quando a ligação de manutenção entre a unidade RF e o ALD (antena RET, TMA, RAE ou SASU) não funciona.	Reportado no caso de existir uma falha na sincronização do *clock* que perdura por um período de tempo (normalmente 200 segundos). Este alarme é limpo quando a sincronização do *clock* é bem sucedida e mantém-se dessa forma durante um período de tempo (normalmente 900 segundos).
Severidade	**Major**	**Major**/Minor
Possíveis causas	A corrente na antena vinda da unidade RF está fora do intervalo de valores normais.	O modo de funcionamento do *clock* ou o *clock* de referência do NE está mal configurado; O modo de sincronização do *clock* do NE está mal configurado; O *hardware* da placa onde o *clock* é originado está avariado; O *hardware* da placa de controlo principal está avariado; O *hardware* de um nó de interconexão está avariado.

Outro padrão entre alarmes "Major", que permite deduzir, depois de analisados os 4 casos, que a sequência de alarmes segue sempre uma ordem crescente em termos de severidade. Logicamente, poderão haver alarmes causados por alarmes com a mesma severidade (tal como acontece neste caso), mas será improvável dar-se um alarme menos severo do que o anterior (por exemplo, $Major \rightarrow Minor$).

Neste caso, a relação entre alarmes não é direta, mas pode dar-se o caso de chegar uma corrente fora do intervalo de valores normais ao ALD que, para além de causar o mau funcionamento da sua ligação de manutenção à unidade RF, provoque a falha na sincronização do *clock* de referência externo utilizado pela antena.

Se juntarmos este padrão àquele que foi descrito na Tabela 5.5, e assumindo que se trata de uma sequência que pode acontecer em qualquer zona da rede, obtém-se a seguinte sequência de 3 alarmes:

RF Unit ALD Current Out of Range(1) $\rightarrow$

$\rightarrow$ ALD Maintenance Link Failure(2) $\rightarrow$

$\rightarrow$ External Clock Reference Problem(3)

Desta forma justifica-se o facto de que o padrão de dois alarmes que o algoritmo calculou foi entre os alarmes 2 e 3, e não diretamente entre o 1 e o 3. O alarme "ALD Maintenance Link Failure" só acontece *a posteriori* e devido ao "RF Unit ALD Current Out of Range". O alarme "External Clock Reference Problem" - apesar de ser causado pelos valores inaceitáveis de corrente (relatados pelo 1º alarme) - ocorre só depois do 2º alarme, porque a falha na sincronização deve perdurar durante um certo período de tempo (normalmente 200 segundos) até o alarme ser despoletado.

Este padrão também é o segundo (dos apresentados) em que se encontra o alarme "External Clock Reference Problem" como consequente. Isto pode indicar que os problemas no *clock* de referência externo podem ser causados, indiretamente, por várias fontes da rede, que levem à má configuração do *clock* ou à avaria de algum elemento de *hardware* essencial à devida sincronização do *clock*.

Retomando a correta sincronização do *clock* de referência externo, previne-se a ocorrência de 1414 alarmes "External Clock Reference Problem". Isto traduz-se, principalmente, na redução de:

- Alarmes do tipo "Hardware": 7969 → 6555 (-17.74%);
- Existem 39 nós que deixam de ter qualquer registo de alarme (-100%);
- Alarmes do nó "VX30EG": 18 → 1 (-94.44%);
- Alarmes do nó "CR06VU": 22 → 4 (-81.82%);
- Alarmes do nó "VX88EU": 50 → 12 (-76%);
- Alarmes do nó "AB42AU": 77 → 27 (-64.94%);
- Alarmes do nó "CS55CU": 27 → 11 (-59.26%);
- Alarmes do nó "VX88EG": 61 → 31 (-49.18%);
- Alarmes do nó "VX88EL": 90 → 52 (-42.22%);
- Alarmes do nó "CR13AL": 215 → 181 (-15.81%);
- Alarmes do nó "SO33WL": 379 → 344 (-9.23%);
- Alarmes do nó "AB42AL": 910 → 860 (-5.49%).

Conclui-se que, com o reconhecimento de padrões sequenciais, é possível prever qual será o alarme mais provável de ser o consequente de um dado alarme antecedente. No âmbito do *preventive maintenance*, se a resolução do problema for rápida e eficaz, é possível prevenir por completo que o alarme consequente ocorra, resultando numa diminuição do número de alarmes despoletados e nas falhas causadas.

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Conclusões

Este trabalho teve como objetivo o desenvolvimento de uma solução para a manutenção preventiva e proativa na gestão de falhas em redes móveis LTE, onde se utilizaram dois tipos de técnicas de aprendizagem automática para a manipulação de dados FM: ARL e SPM.

No caso dos algoritmos de ARL, o *FPGrowth* apresentou o melhor desempenho em termos de tempo de execução e de recursos utilizados. O *Eclat* foi o mais eficiente para transações curtas, sendo superado pelo *Apriori* para transações maiores. Em todos eles verificou-se uma simetria nas regras de associação, tendo-se por isso evoluído para os algoritmos SPM, onde a ordem de ocorrência dos alarmes é importante.

Neste caso, o *SPAM* revelou ser o mais eficiente no tempo de execução mas o pior em termos de utilização de recursos. O *SPADE* foi o mais eficiente em termos de utilização de recursos mas o mais lento de todos. Por outro lado, o *PrefixSpan* ofereceu um bom compromisso de desempenho entre os três. Em todos os testes realizados foi possível concluir que a prevenção dos alarmes consequentes resulta numa diminuição no seu número absoluto nos nós da rede onde ocorreram.

No melhor dos casos verificou-se uma diminuição de 3.31% em todos os nós analisados, e 70.45% no que toca a alarmes do mesmo tipo. Constatou-se ainda que 39 nós da rede deixaram de ter qualquer alarme por resolver, ao

detetar atempadamente os alarmes antecedentes. Estes resultados demonstram que o reconhecimento de padrões sequenciais impulsiona a manutenção preventiva de alarmes numa rede móvel LTE, reforçando a importância do conceito de *preventive maintenance* para os operadores.

6.2 Trabalho Futuro

Para trabalho futuro será desenvolvido um *software* que, dado o alarme antecedente, apresenta todos os possíveis alarmes consequentes e qual a probabilidade de cada um ocorrer. Também encontra-se planeado conhecer o intervalo de tempo médio entre os antecedentes e consequentes dos padrões, para avaliar se é possível resolver o problema em tempo útil. Para além dos dados de FM, poderiam também ser processados dados de *Performance Management* (PM) para enriquecer a análise dos problemas na rede. Tudo isto será potencializado pela integração com dados históricos atualizados em tempo-real e num sistema de *tickets* e resoluções de problemas que possa ser acedido pelos engenheiros que trabalham no NOC.

Referências

[1] C. G. Araújo, G. Cardoso, J. Sousa, M. Coutinho, and P. Neves, "Predictive fault management," in *InnovAction #4*, Altice Labs, 2019.

[2] D. Mulvey, C. H. Foh, M. A. Imran, and R. Tafazolli, "Cell fault management using machine learning techniques," *IEEE Access*, vol. 7, pp. 124514–124539, 2019.

[3] M. Nouioua, P. Fournier-Viger, G. He, F. Nouioua, and M. Zhou, "A survey of machine learning for network fault management," in *Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics*, 2021.

[4] Radhakant Das, "Cognitive Operations: The Future of Telecom Networks is Here," White Paper, Tata Consultancy Services, 2019.

[5] 3GPP, "Technical Specification Group Services and System Aspects; Telecommunication management; Self Configuration of Network Elements; Concepts and Requirements (Release 9)," TS 32.501, 3rd Generation Partnership Project (3GPP), 2010.

[6] 3GPP, "Technical Specification Group Services and System Aspects; Telecommunication management; Project scheduling and open issues for SA5 (Release 8)," TR 30.818, 3rd Generation Partnership Project (3GPP), 2010.

[7] M. Amirijoo, P. Frenger, F. Gunnarsson, H. Kallin, J. Moe, and K. Zetterberg, "Neighbor Cell Relation List and Physical Cell Identity Self-Organization in LTE," in *Neighbor Cell Relation List and Physical Cell Identity Self-Organization in LTE*, pp. 37 – 41, 06 2008.

[8] 3GPP, "Telecommunication management; Automatic Neighbor Relation (ANR) management, Concepts and Requirements, (Release 11)," TR 32.511, 3rd Generation Partnership Project (3GPP), 2011.

[9] 3GPP, "From large lists of potential neighbour cells to self-optimised neighbour cell lists," Tech. Rep. R3-071239, Mitsubishi Electric, 2010.

[10] D. Kim, B. Shin, D. Hong, and J. Lim, "Self-configuration of neighbor cell list utilizing e-utran nodeb scanning in lte systems," in *2010 7th IEEE Consumer Communications and Networking Conference*, pp. 1–5, 2010.

[11] 3GPP, "Technical Specification Group Radio Access Network, Evolved Universal Terrestrial Radio Access Network (E-UTRAN), Self configuring and Self optimizing Network (SON) Uses Case and Solutions (Release 9)," TR 36.902, 3rd Generation Partnership Project (3GPP), 2011.

[12] 3GPP, "Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access Network (EUTRAN); X2 Application Protocol (X2AP) (Release 11)," TS 36.423, 3rd Generation Partnership Project (3GPP), 2013.

[13] 3GPP, "Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures (Release 10)," TS 36.213, 3rd Generation Partnership Project (3GPP), 2011.

[14] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2," TS 36.300, 3rd Generation Partnership Project (3GPP), 2016.

[15] 3GPP, "Study on Energy Savings Management (ESM) (Release 10)," TR 32.826, 3rd Generation Partnership Project (3GPP), 2010.

[16] C. Mueller, M. Kaschub, C. Blankenhorn, and S. Wanke, "A cell outage detection algorithm using neighbor cell list reports," in *A Cell Outage Detection Algorithm Using Neighbor Cell List Reports*, pp. 218–229, 12 2008.

[17] S. Hamalainen, H. Sanneck, and C. Sartori, *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*. Wiley, 01 2012.

[18] 3GPP, "Digital cellular telecommunications system; Universal Mobile Telecommunications System (UMTS);LTE; Telecommunication management; Self-Organizing Networks (SON); Self-healing concepts and requirements," TS 32.541, 3rd Generation Partnership Project (3GPP), 2014.

[19] S. Ayoubi, N. Limam, M. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada-Solano, and O. Caicedo Rendon, "Machine learning for cognitive network management," *IEEE Communications Magazine*, vol. 56, 01 2018.

[20] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng, "Data-intensive question answering," *TREC*, 12 2001.

[21] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2nd ed., 2009.

[22] Q. Mahmoud, *Cognitive Networks: Towards Self-Aware Networks*. Wiley-Interscience, 07 2007.

[23] P. McCullagh and J. Nelder, *Generalized Linear Models, Second Edition*. Chapman & Hall, 1989.

[24] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, p. 199–222, Aug. 2004.

[25] H. Zhang, "The optimality of naive bayes," in *The Optimality of Naive Bayes*, vol. 2, 01 2004.

[26] Z. Ghahramani, "Unsupervised learning," in *Advanced Lectures on Machine Learning*, pp. 72–112, Springer-Verlag, 2004.

[27] J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *JSTOR: Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979.

[28] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artif. Intell. Rev.*, vol. 22, p. 85–126, Oct. 2004.

[29] S. Whiteson and P. Stone, "Adaptive job routing and scheduling," *Engineering Applications of Artificial Intelligence*, vol. 17, pp. 855–869, 08 2004.

[30] R. Boutaba, M. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. Caicedo Rendon, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, 05 2018.

[31] A. Clemm, *Network Management Fundamentals*. Cisco Press, 2006.

[32] N. Bui, M. Cesana, S. Hosseini, Q. Liao, I. Malanchini, and J. Widmer, "A survey of anticipatory mobile networking: Context-based classification, prediction methodologies, and optimization techniques," *IEEE Communications Surveys & Tutorials*, vol. PP, pp. 1–1, 04 2017.

[33] A. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, pp. 1–1, 01 2015.

[34] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), pp. 56 – 61, 2010.

[35] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[36] The NumPy development team, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[37] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.

[38] "missingno: Missing data visualization module for Python." `https://github.com/ResidentMario/missingno`. Accessed: 2021-09-21.

[39] "Dython: Data analysis tools in pYTHON." `http://shakedzy.xyz/dython/`. Accessed: 2021-09-21.

[40] M.-S. Chen, J. Han, and P. Yu, "Data mining: an overview from a database perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 866–883, 1996.

[41] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *SIGMOD Rec.*, vol. 22, p. 207–216, June 1993.

[42] "Métricas de avaliação suportadas pela biblioteca MLxtend." `http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/#metrics`. Accessed: 2021-09-21.

[43] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," *SIGMOD Rec.*, vol. 26, p. 255–264, June 1997.

[44] G. Piatetsky-Shapiro, "Discovery, analysis, and presentation of strong rules," in *Knowledge Discovery in Databases* (G. Piatetsky-Shapiro and W. J. Frawley, eds.), pp. 229–248, AAAI/MIT Press, 1991.

[45] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, (San Francisco, CA, USA), p. 487–499, Morgan Kaufmann Publishers Inc., 1994.

[46] M. Zaki, "Scalable algorithms for association mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 3, pp. 372–390, 2000.

[47] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data Min. Knowl. Discov.*, vol. 8, pp. 53–87, 01 2004.

[48] C. Borgelt, "An implementation of the fp-growth algorithm," in *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, OSDM '05, (New York, NY, USA), p. 1–5, Association for Computing Machinery, 2005.

[49] P. Fournier Viger, C.-W. Lin, U. Rage, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Science and Pattern Recognition*, vol. 1, pp. 54–77, 02 2017.

[50] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '96, (Berlin, Heidelberg), p. 3–17, Springer-Verlag, 1996.

[51] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "Mining sequential patterns by pattern-growth: The prefixspan approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp. 1424–1440, Nov. 2004.

[52] M. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," *Machine Learning*, vol. 42, pp. 31–60, 01 2001.

[53] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, (New York, NY, USA), p. 429–435, Association for Computing Machinery, 2002.

[54] "Memory Profiler: A module for monitoring memory usage of a Python program." `https://pypi.org/project/memory-profiler/`. Accessed: 2021-09-21.

[55] S. Raschka, "Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack," *The Journal of Open Source Software*, vol. 3, Apr. 2018.

[56] C. Borgelt, "Frequent item set mining," *WIREs Data Mining and Knowledge Discovery*, vol. 2, no. 6, pp. 437–456, 2012.

[57] "spmf-py: Python Wrapper for SPMF." `https://github.com/LoLei/spmf-py`. Accessed: 2021-09-21.

[58] P. Fournier-Viger, J. C.-W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, and H. T. Lam, "The SPMF Open-Source Data Mining Library Version 2," in *Machine Learning and Knowledge Discovery in Databases* (B. Berendt, B. Bringmann, É. Fromont, G. Garriga, P. Miettinen, N. Tatti, and V. Tresp, eds.), (Cham), pp. 36–40, Springer International Publishing, 2016.