Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

BENCHMARK DE SISTEMAS DE DETECÇÃO DE INTRUSÕES BASEADOS EM COMPORTAMENTO COM RECURSO A ALGORITMOS BIOINSPIRADOS

PAULO JORGE GOMES FERREIRA

Leiria, Setembro de 2020

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

BENCHMARK DE SISTEMAS DE DETECÇÃO DE INTRUSÕES BASEADOS EM COMPORTAMENTO COM RECURSO A ALGORITMOS BIOINSPIRADOS

PAULO JORGE GOMES FERREIRA
Número: 2180047

Dissertação realizada sob orientação do Professor Doutor Mário João Gonçalves Antunes.

Leiria, Setembro de 2020

AGRADECIMENTOS

A realização deste trabalho não teria sido possível sem a ajuda e colaboração de algumas pessoas. Não pretendendo ser exaustivo e, sobretudo, não tendo a intenção de omitir alguém, gostaria de destacar:

- Prof. Dr Mário Antunes, o meu orientador
- Prof. Dr. Paulo Angelo Alves Resende, da Universidade de Brasília; o Prof. Paulo Angelo foi o criador do Hogzilla IDS;
- Prof. Dr. Arash Habibi Lashkari, professor na Universidade de New Brunswick, no Canadá; o prof. Arash foi um dos responsáveis pela criação do dataset CSE-CIC-IDS2018, utilizado neste trabalho;
- Os meus familiares, que me deram todo o apoio e força necessários para que este projecto se concretizasse.

Para eles e para todos os outros que, de alguma forma, contribuíram para a realização deste trabalho, o meu muito obrigado.

RESUMO

Com a massificação e diversificação dos ataques por via informática, torna-se imprescindível a implementação de medidas de prevenção e detecção dos mesmos, tendo como finalidade a limitação de potenciais danos que possam causar.

Um sistema de detecção de intrusões é, como o próprio nome indica, um sistema que analisa o tráfego que circula na rede da organização e que emite alertas caso seja detectada uma intrusão.

Quanto ao seu funcionamento, podemos, muito sumariamente, caracterizar os sistemas de detecção de intrusões em dois tipos distintos: baseados em comportamento e baseados em assinaturas. Os sistemas baseados em assinaturas apoiam-se numa "base de dados" de padrões ou assinaturas de ataques, reagindo apenas aos ataques que constam nessa mesma base de dados. Já os sistemas baseados em comportamento adquirem informação sobre o denominado "comportamento normal" da rede e, com base nisso, reportam qualquer desvio a essa normalidade.

Neste trabalho pretende-se fazer uma análise ao desempenho de sistemas de detecção de intrusões baseados em comportamento, recorrendo a metodologias de aprendizagem automática e algoritmos bioinspirados, tais como os baseados no sistema imunológico humano e redes neuronais.

Recorrendo a um *dataset* público desenvolvido especificamente para avaliação de sistemas de detecção de intrusões, serão realizados testes em que os algoritmos serão parametrizados com configurações diferentes, permitindo avaliar qual o algoritmo e respectiva configuração que melhor desempenho apresenta na detecção de possíveis intrusões.

Paralelamente aos dados de cada algoritmo, os resultados individuais serão combinados num processo de votação, com o objectivo de determinar se a conjugação de vários resultados, através de uma política de *majority voting*, contribui ou não para uma melhoria do desempenho do sistema em si.

ABSTRACT

With the widespread and diversification of attacks by computer, it is essential to implement prevention and detection measures, aiming to limit the potential damage that they can cause.

An intrusion detection system is, as its name implies, a system that analyzes the traffic circulating on the organization's network and that issues alerts if an intrusion is detected.

As for its operation, we can, very briefly, characterize the intrusion detection systems in two distinct types: based on behavior and based on signatures. Signature-based systems rely on a "database" of attack patterns or signatures, reacting only to attacks that appear in that database. Behavior-based systems acquire information about the so-called "normal behavior" of the network and, based on this, report any deviation from this normality.

This work intends to analyze the performance of intrusion detection systems based on behavior, using automatic learning methodologies and bioinspired algorithms, such as those based on the human immune system and neural networks.

Using a public *dataset* specifically developed for the evaluation of intrusion detection systems, tests will be carried out in which the algorithms will be parameterized with different configurations, allowing to evaluate which algorithm and respective configuration has the best performance in detecting possible intrusions. .

Parallel to the data of each algorithm, the individual results will be combined in a voting process, with the aim of determining whether the combination of several results, through a *majority voting* policy, contributes or not to an improvement in the performance of the system itself.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

A-IDS	Anomaly-based IDS.
AIS	Artificial Immune Systems.
ANN	Artificial Neural Networks.
API	Application Programming Interface.
CIC	Canadian Institute for Cybersecurity.
CSE	Communications Security Establishment.
CSV	Comma-separated values.
DDoS	Distributed Denial of Service.
DHCP	Dynamic Host Configuration Protocol.
DNS	Domain Name System.
DoS	Denial of Service.
FNR	False Negative Rate.
FPR	False Positive Rate.
FTP	File Transfer Protocol.
IDS	Intrusion Detection System.
IP	Internet Protocol.
LVQ	Learning Vector Quantization.
ML	Machine Learning.
MLP	Multi-Layer Perceptron.
NIST	National Institute of Standards and Technology.
TCP	Transmission Control Protocol.

TNR True Negative Rate.

TPR True Positive Rate.

URL Uniform Resource Locator.

1

INTRODUÇÃO

Vivemos, actualmente, o paradigma do *online*: há uma necessidade cada vez maior de estar ligado, de estar em rede. De facto, quer as empresas quer os indivíduos procuram ter uma ciber-presença tão activa e visível quanto possível, de forma a facilitar quer o contacto quer eventuais transações entre eles.

Essa presença, contudo, não está isenta de riscos. Cada vez mais frequentemente as empresas são vítimas de ciber-ataques visando desde o "simples" roubo ou exfiltração de dados até à completa destruição dos sistemas informáticos, com os inerentes prejuízos daí resultantes.

De facto, o relatório anual da Symantec ISTR 2019 [1], relativo ao ano de 2018, aponta, precisamente, para um crescimento global do número de ciber-ataques, como sejam os ataques a sites web (aumento de 56%), ataques de *ransomware* (com aumento de 12 % no número de ataques a empresas, apesar de, na globalidade, o número total de ataques de *ransomware* ter descido cerca de 20%) ou ataques via Windows Powershell (um aumento de 1000% do número de scripts maliciosos).

Atendendo aos riscos existentes, e tendo como objectivo a defesa e manutenção da sua reputação no mercado, as empresas devem tomar medidas que permitam quer a detecção atempada desses ataques - permitindo uma resposta que se deseja adequada tão cedo quanto possível - quer, na medida do possível, o bloqueio de ataques já conhecidos.

Uma dessas ferramentas, que se destina a fazer, precisamente, a monitorização do tráfego de rede da empresa ou organização, são os Intrusion Detection System (IDS). Devido ao seu modo de funcionamento, é possível, através dessas aplicações, detectar, com algum grau de exactidão e em tempo útil, possíveis intrusões na rede ou sistemas da organização, permitindo a tomada de medidas de contenção ou de limitação da ocorrência.

Consoante os métodos de detecção, podemos considerar dois grandes tipos de IDS: os baseados em assinaturas e os baseados em comportamento. Estes últimos têem a vantagem de poder reagir face a ataques ou intrusões novas, ou seja, para os quais ainda não existe assinatura. Essa possibilidade de reacção está intimamente

relacionada com o facto de estes sistemas realizarem uma "aprendizagem" sobre o tráfego de rede, permitindo, assim, estabelecer um padrão normal de actividade.

A aprendizagem ou treino destes sistemas, tendo em conta o volume de dados (tráfego de rede), é um processo complexo e de grande exigência computacional, recorrendo, frequentemente, a ferramentas de análise de Big Data e a algoritmos de aprendizagem automática (Machine Learning (ML), na designação anglo-saxónica).

De entre a multiplicidade de algoritmos de ML existentes, neste trabalho serão abordados os denominados algoritmos bioinspirados, ou seja, os algoritmos cuja inspiração para o seu funcionamento se encontra em fenómenos ou processos existentes na natureza, nomeadamente nos seres vivos.

Para garantir a eficácia e eficiência dos IDS, é necessário que os algoritmos de detecção que eles implementam sejam os mais adequados e com a melhor resposta, pelo que se torna essencial realizar testes comparativos dos mesmos antes de os colocar em ambiente de produção.

Nesta dissertação pretende-se fazer uma avaliação da aplicação de alguns algoritmos bioinspirados na implementação de sistemas de detecção de intrusões baseados em comportamento.

1.1 OBJETIVOS

Tendo como base o modo de funcionamento dos IDS baseados em comportamento, que recorrem à aprendizagem e definição de um padrão de comportamento normal do tráfego de rede, pretende-se avaliar a adequabilidade e eficácia do uso de algoritmos bioinspirados para esta função. Serão analisados e testados vários algoritmos de ML bio-inspirados, designadamente os baseados no sistema imunitário (Artificial Immune Systems (AIS)) e em algoritmos evolucionários baseados em redes neuronais (Artificial Neural Networks (ANN)).

A motivação para investigação realizada no âmbito desta dissertação consiste em avaliar a comparação dos algoritmos selecionados e, não menos importante, os resultados obtidos com uma estratégia de *ensemble*, onde um comité formado pelos vários algoritmos base, decidirá sobre a classificação final de cada exemplo apresentado. A estratégia de ensemble utilizada baseia-se no conceito de *majority voting*.

Os objectivos deste trabalho são os seguintes:

- Desenvolver uma arquitectura, baseada em aplicações de código aberto (*open-source* na terminologia inglesa), que permita fazer o pré-processamento do dataset utilizado;
- Definir e implementar um fluxo de processamento do dataset no WEKA [2], tendo como base o dataset pré-processado;
- Desenvolver uma ferramenta que permita automatizar os testes realizados com os classificadores CLONALG; [3, 4], LVQ [5] e Back-MLP [6];
- Comparar os resultados obtidos pelos algoritmos bioinspirados isoladamente;
- Comparar os resultados obtidos através da estratégia de *majority voting*, relacionando-os com os resultados isolados.

1.2 CONTRIBUTOS

Este trabalho analisa a possibilidade de aplicação de algoritmos bioinspirados, nomeadamente imunoinspirados e baseados no sistema neurológico, aos sistemas de detecção de intrusões baseados em comportamento.

Um dos contributos desta dissertação é a metodologia que foi definida para realizar e automatizar os testes com três algoritmos bioinspirados e avaliar o impacto do uso de um ensemble.

Foram efectuados testes, utilizando os algoritmos CLONALG, Multi-Layer Perceptron e Learning Vector Quantization, quer individualmente quer combinados num *ensemble* com uma estratégia de *majority voting*.

Para a realização dos testes foi utilizado o *dataset* CSE-CIC-IDS2018, um *dataset* relativamente recente que está a despertar cada vez mais a atenção da comunidade académica. Este *dataset* classificado foi desenvolvido com o propósito de servir de base de testes para sistemas de detecção de intrusões, incluindo tráfego relativo a diferentes e variadas tipologias de ataques.

Em paralelo a este trabalho, e como ferramenta de suporte, foi desenvolvida uma aplicação em Java que, utilizando a API do WEKA, implementa módulos e funções que permitiram a execução dos cenários de testes. O código fonte da aplicação, de código aberto, está alojado na plataforma GitHub (`https://github.com/paulo-ferreira-mcif/benchmarkids`), estando disponível para utilização pela comunidade.

Esta dissertação serviu, também, de base para um artigo intitulado "Benchmarking machine learning bioinspired algorithms for behaviour-based network intrusion detection". O artigo foi submetido e aceite na conferência "Eighth International Symposium on Security in Computing and Communications (SSCC'20)".

1.3 ORGANIZAÇÃO DO DOCUMENTO

No **capítulo 2** serão abordados os conceitos que suportam este trabalho. Serão definidos os conceitos de aprendizagem automática e algoritmos bioinspirados e seus derivados. O conceito de dataset, assim como a motivação para o seu uso, serão, também alvo de análise. De especial relevância para este trabalho, a definição e taxonomia dos sistemas de detecção de intrusão será, também abordada.

O **capítulo 3** incluirá um breve resumo de literatura sobre os diversos temas que compõem este trabalho.

A descrição do ambiente de testes, bem como das aplicações e ferramentas utilizadas será efectuada no **capítulo 4**. A caracterização do dataset CSE-CIC-IDS2018 é, também, feita neste capítulo.

Toda a parte operacional relativa aos testes será descrita no **capítulo 5**. Aqui, encontra-se toda a informação relacionada com a preparação dos dados, o treino dos algoritmos e geração dos modelos, os testes realizados propriamente ditos e os resultados dos mesmos. O capítulo termina com uma análise aos resultados obtidos.

O **capítulo 6** contém as conclusões deste trabalho, incluindo algumas sugestões para trabalho futuro.

2

CONCEITOS FUNDAMENTAIS

Neste capítulo serão abordados alguns conceitos que servem de base a este trabalho, nomeadamente no que diz respeito à aprendizagem automática, aos sistemas de detecção de intrusões,aos datasets e aos sistemas bioinspirados.

2.1 APRENDIZAGEM AUTOMÁTICA

De acordo com [7], a aprendizagem automática é um ramo da inteligência artificial cujo objectivo é o desenvolvimento de métodos e algoritmos que permitam aos sistemas aprender a partir dos dados, em vez de através de programação explícita.

A aprendizagem automática pode ser aplicada a inúmeros domínios, com especial destaque para o processamento da linguagem natural, reconhecimento de imagens e cibersegurança. Neste último caso, a aprendizagem automática pode ser utilizada, por exemplo, para treinar IDS baseados em comportamento e para detectar e/ou classificar potenciais ameaças a partir da análise do tráfego de rede.

O processo de aprendizagem automática passa por várias fases [8]: a preparação dos dados, o treino do algoritmo e geração do modelo, a produção de previsões e o refinamento das mesmas.

2.1.1 *Tipos de aprendizagem automática*

Podemos considerar, essencialmente, dois tipos de aprendizagem automática [8]: supervisionada e não-supervisionada.

Na aprendizagem automática **supervisionada** o treino dos algoritmos é feito com recurso a dados previamente classificados (ver Figura 1). A razão de se designar supervisionada está relacionada com o facto de a previsão efectuada pelo algoritmo ser comparada com o valor original, permitindo avaliar se a previsão está ou não correcta. O algoritmo pode, assim, ajustar o modelo, conduzindo a melhores previsões.

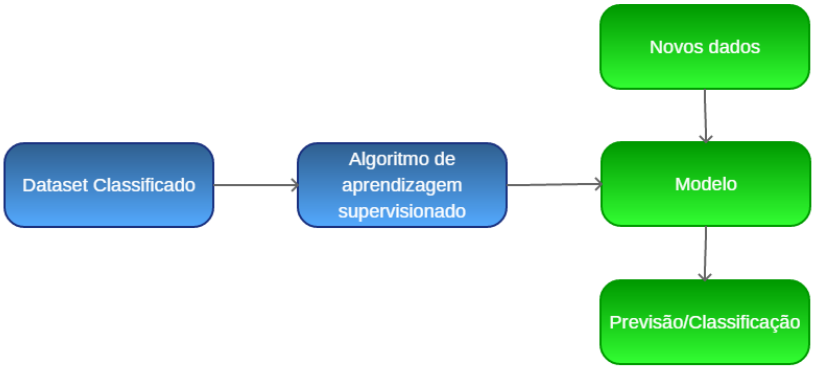

Figura 1: Aprendizagem automática supervisionada (adaptado de [9])

Na aprendizagem supervisionada, é criado um modelo utilizando dados classificados, que consistem em dados e a classificação respectiva, que é o que se pretende obter. A supervisão faz com que o modelo seja ajustado, por comparação da saída do modelo com o valor da classificação original. O modelo assim gerado e treinado pode, *a-posteriori*, ser aplicado a novos dados, que serão classificados tão correctamente quanto possível [9]. Alguns algoritmos de aprendizagem automática supervisionada incluem o Naive-Bayes e Support Vector Machines, por exemplo.

Já a aprendizagem automática **não-supervisionada** recorre a dados não classificados, normalmente em quantidades massivas, tentando identificar possíveis padrões que permitam agrupar os dados e efectuar, assim, uma classificação dos mesmos. Como os dados não estão classificados, não há possibilidade de avaliar se a previsão está ou não correcta, sendo esta a maior e mais importante diferença face à aprendizagem supervisionada [10].

O agrupamento dos dados (*clustering*, na designação inglesa) através de aprendizagem automática não-supervisionada permite que se possa obter características ou propriedades sobre os dados que de outra forma não seria possível.

Figura 2: Aprendizagem automática não-supervisionada (adaptado de [10])

2.1.2 *A metodologia aplicada na aprendizagem automática*

Conforme já foi referido, a aprendizagem automática envolve várias fases: a preparação dos dados, o treino do algoritmo e geração do modelo, a produção de previsões e o refinamento das mesmas.

A **preparação dos dados** tem como objectivo a análise, formatação e condicionamento dos dados de forma a possibilitar a geração de um modelo de aprendizagem.

A preparação dos dados envolve, tipicamente, as seguintes operações [8]:

- **Seleccionar um sub-conjunto de dados** da amostra, fazendo suposições acerca dos dados de forma a seleccionar os atributos pertinentes e adequados ao problema que se pretende resolver; tipicamente, esta operação pode ser utilizada para, por exemplo, eliminar atributos desnecessários ou com pouca relevância para o problema em questão;
- **Seleccionar o formato e a ordenação** dos dados. É importante decidir em que formato serão guardados os dados (em ficheiro de texto ou numa base de dados, por exemplo). Por outro lado, alguns algoritmos podem exigir que os dados sejam ordenados de uma determinada forma;
- **Limpar os dados**, removendo ou substituindo os valores em falta; para isso, há várias ferramentas estatísticas disponíveis que podem ajudar na inspecção dos dados, permitindo detectar erros e desvios. O objectivo é garantir que os dados sejam tão exactos, completos e relevantes quanto possível;
- **Normalizar os dados**, ou seja, ajustar os valores medidos em escalas diferentes para uma escala comum. Alguns algoritmos (por exemplo, o Multi-Layer Perceptron, utilizado nesta dissertação) podem exigir que, para efectuarem comparação de dados, os valores estejam normalizados numa escala comum.

Na fase de **treino do algoritmo e geração do modelo** a aprendizagem automática vai utilizar os dados pré-preparados para treinar um algoritmo de aprendizagem automática, resultando desta operação um modelo de aprendizagem.

O algoritmo a utilizar deve ser seleccionado tendo em conta o problema que se pretende resolver, de forma a que a resposta do mesmo seja tão fiável quanto possível. Há vários tipos de algoritmos disponíveis, sendo que cada um destes será mais adequado a determinadas situações [8]:

- **Algoritmos de regressão** - este tipo de algoritmo é especialmente indicado nas situações em que se pretende prever o valor de uma variável dependente

tendo como base o valor de uma variável independente (regressão linear) ou quando a variável dependente é binária por natureza (regressão logística);

- **Árvores de decisão** - As árvores de decisão usam uma estrutura tipo árvore para ilustrar os resultados de uma decisão. São usadas para mapear todos os possíveis resultados de uma decisão, sendo que cada nó representa, precisamente, um possível resultado [7].
- **Algoritmos baseados em instâncias** - Este tipo de algoritmos é usado quando se pretende classificar novos pontos de dados tendo como base algumas similaridades com os dados de treino. Este conjunto de algoritmos é, por vezes, denominado de *lazy learners* ("aprendiz preguiçoso"), pois não há fase de treino. Em vez disso, os novos dados são comparados com os dados de treino e classificados com base na semelhança com os dados de treino. Os algoritmos basedaos em instâncias não são recomendáveis para conjuntos de dados com variação aleatória, com dados irrelevantes ou com valores em falta [7].
- **Algoritmos de agrupamento (*clustering*)** - Neste tipo de algoritmos os dados não estão classificados, antes são agrupados de acordo com a similaridade entre eles, ou seja, objectos com atributos semelhantes são agrupados. Todos os membros de um determinado grupo são mais semelhantes entre si do que com elementos de outros grupos. Este tipo de algoritmos é utilizado na aprendizagem automática não-supervisionada.

Depois de treinado e criado o modelo, o mesmo pode ser aplicado a novos dados, de forma a poder fazer previsões ou determinações acerca dos mesmos - é a fase habitualmente designada de **fase de teste**.

Há algoritmos em que o modelo vai, também, e à medida que vai sendo sujeito a mais dados, aprendendo dinamicamente, ajustando-se de forma a melhorar e tornar mais eficazes as suas previsões. Esta operação denomina-se **refinamento de previsões**, e tem como base o mecanismo de adaptabilidade subjacente aos sistemas de aprendizagem automática.

2.2 SISTEMAS DE DETECÇÃO DE INTRUSÕES

O surgimento dos sistemas de detecção de intrusões (IDS) está intrinsecamente ligado à necessidade de processamento e tratamento de logs gerados pelos sistemas, tendo em vista a detecção de anomalias ou intrusões. Fuchsberger [11] faz, precisamente, essa súmula histórica, revelando que, inicialmente, o software de detecção de

Tabela 1: Classificação dos IDS - NIST

Método de Detecção	Signature-based
	Anomaly based
	Stateful Protocol Analysis
Tecnologia do IDS	Network-based
	Wireless
	Network Behavior Analysis
	Host-based

intrusões, como era chamado, se resumia a isso mesmo: um programa que processava e analisava os logs gerados pelos diversos sistemas. Com o surgimento e expansão das comunicações em rede, houve necessidade, também e com o mesmo objectivo, de passar a "escutar" o tráfego de rede. Surgiram, então, os Sistemas de Detecção de Intrusões ou IDS.

2.2.1 *Taxonomia dos IDS*

Atendendo às diferentes realidades de análise, em [11] propõe-se, também, uma classificação para os IDS, designadamente:

- De acordo com o objecto de análise, podem ser classificados em *host-based* ou *network-based*;
- De acordo com o método de detecção, classificam-se em baseados em conhecimento ou assinaturas (*knowledge-based*) ou baseados em comportamento (*behaviour-based*).

O National Institute of Standards and Technology (NIST), através da Special Publication 800-94 (Rev. 1 Draft) [12], faz uma classificação muito semelhante, adicionando alguns tipos mais específicos (Tabela 1).

Com base nas classificações descritas acima, e tomando como ponto de partida os **métodos de detecção**, pode dizer-se que um IDS pode ser classificado das seguintes formas:

- ***Signature-based***, quando a detecção é efectuada comparando o tráfego ou os dados em análise com uma base de dados de conhecimento ou assinaturas, um pouco à semelhança do que se passa com os anti-vírus;

- ***Anomaly-based***, quando a detecção se baseia num desvio ao chamado "comportamento normal" do sistema; este tipo de IDS será analisado na secção 2.2.3;
- ***Stateful Protocol Analysis***, quando o tráfego em análise é comparado com perfis pré-definidos de actividade benigna de cada protocolo, com o objectivo de detectar desvios; ao contrário dos sistemas *anomaly-based* - que recorrem a perfis específicos de rede ou do sistema - estes sistemas recorrem a perfis universais desenvolvidos pelos fabricantes, perfis esses que definem a forma como o protocolo em questão pode ou não ser utilizado [12].

Relativamente à **tecnologia dos IDS**, eles podem classificar-se em

- ***Network-based***, quando o objecto de análise é o tráfego de rede;
- ***Host-based***, quando o objecto de análise é um determinado sistema; este processo envolve, geralmente, a análise e processamento de ficheiros de log do sistema em causa;
- **Wireless**, quando se trata de análise de tráfego de redes wireless;
- ***Network Behavior Analysis***, que analisa o tráfego de rede para identificar ameaças que potencialmente possam causar fluxos anómalos de tráfego, tais como sejam ataques Distributed Denial of Service (DDoS) ou certas formas de *malware*.

Independentemente da classificação, o objectivo final é sempre o mesmo: a detecção atempada de possíveis intrusões. Caso seja detectada uma intrusão, o IDS emite um alerta para a administração de sistemas, permitindo, assim, que sejam tomadas as eventuais medidas de contenção.

2.2.2 *IDS baseados em assinaturas*

Na detecção de intrusões baseada em assinaturas parte-se do princípio que os ataques - que visam explorar vulnerabilidades e pontos fracos do sistema operativo e aplicações - seguem padrões bem definidos [13].

Esses padrões ou assinaturas são salvaguardados num repositório ou base de dados acessível ao IDS, sendo por ele utilizados para analisar o tráfego de rede, permitindo, assim, detectar possíveis intrusões.

O modo de funcionamento destes sistemas é bastante simples. Todo o tráfego de rede é captado pelo IDS e comparado com as assinaturas disponíveis, utilizando um

sistema de regras. No caso de alguma regra ser validada de forma positiva, é gerado um alerta.

Ainda de acordo com Kumar e Sangwan [13], podemos verificar que este tipo de sistemas tem as seguintes vantagens:

- Possuem uma baixa taxa de falsos positivos, pois os ataques estão bem definidos à partida;
- São bastante eficientes na detecção de ataques conhecidos, *i. e.*, ataques para os quais já exista uma assinatura;
- A detecção de intrusões baseada em assinaturas é fácil de usar; é, também, um mecanismo de fácil implementação.

Por outro lado, este tipo de sistema tem, algumas desvantagens, sendo que a mais significativa está relacionada com a (não) detecção de ataques não conhecidos [13]:

- É necessário que o IDS tenha, previamente, conhecimento específico sobre o ataque, ou seja, é necessário que o padrão associado ao ataque em questão já conste na base de dados de assinaturas;
- É necessário manter a base de dados de assinaturas devidamente actualizada;
- Os ataques do tipo *zero-day* não são, à partida, detectados, pois não há padrão pré-definido para os mesmos;
- As assinaturas de um determinado ataque são muito dependentes do ambiente em que são executados os ataques, tais como o tipo e versão do sistema operativo ou o tipo e versão da aplicação;
- Este tipo de detecção pode não ser indicado para alguns tipos de ataque, como, por exemplo, ataques a partir do interior da organização que envolvam abuso de privilégios.

Estão disponíveis, no mercado, inúmeras soluções para implementação de IDS baseados em assinaturas, desde soluções comerciais proprietárias a implementações *open-source* gratuitas. Em termos de soluções *open-source* gratuitas, há, claramente, dois sistemas que se destacam: o Snort[1] e o Suricata[2]. O Snort disponibiliza, ainda, bases de dados de assinaturas, sendo que para a mais completa é necessário efectuar registo no *site*; a versão *community* das assinaturas está disponível para descarregar livremente, mas o número de assinaturas disponíveis é bastante inferior.

1 https://snort.org/
2 https://suricata-ids.org/

2.2.3 *IDS baseados em comportamento*

Conforme mencionado acima (secção 2.2.2), a grande vantagem dos IDS baseados em assinaturas prende-se com o facto de serem extremamente eficientes a detectar os ataques conhecidos, ou seja, aqueles para os quais já possuam uma assinatura na sua base de dados. O mesmo não se pode dizer dos ataques para os quais ainda não existam assinaturas, como sejam os *exploits zero-day*. Neste caso, os IDS baseados em assinaturas pura e simplesmente não detectam este tipo de ataques, com as consequências óbvias para os utilizadores e administradores de sistemas.

Numa tentativa de combater esta situação surgem, então, os IDS baseados em comportamento, designados, na língua inglesa, de **Anomaly-based IDS (A-IDS)**. De acordo com [11], estes sistemas analisam o tráfego e tentam estabelecer um comportamento dito normal da rede. Tendo como base esse comportamento normal, os desvios são considerados tráfego anómalo e, portanto, reportados como possíveis positivos.

Este método de funcionamento, embora permita reagir a ataques não catalogados, tem contudo a desvantagem de poder gerar falsos alertas, pelo que se torna absolutamente essencial treinar devidamente o sistema e ajustar os patamares de reacção.

Apesar dos diferentes métodos utilizados na implementação, de um modo geral todos os IDS baseados em comportamento são constituídos pelos seguintes módulos ou fases [14]:

- *Parametrização* - módulo responsável por parametrizar o tráfego observado, convertendo-o para um formato pré-estabelecido.
- *Fase de Treino* - o comportamento normal (ou anormal) do sistema é caracterizado, sendo construído o respectivo modelo. Dependendo do tipo de A-IDS, esta operação pode ser realizada manual ou automaticamente.
- *Fase de Detecção* - O modelo entretanto disponibilizado é comparado com o tráfego previamente parametrizado. Se for detectado um desvio que não cumpra os patamares de tolerância estipulados, é gerado um alerta.

Conforme se pode verificar em [14], os IDS baseados em comportamento, tendo em conta o método-base utilizado, podem classificar-se em *statistical-based*, *knowledge-based* e *machine learning-based*.

Nos IDS **statistical-based** o tráfego de rede capturado é utilizado para criar um perfil representativo do seu comportamento estocástico. As métricas utilizadas

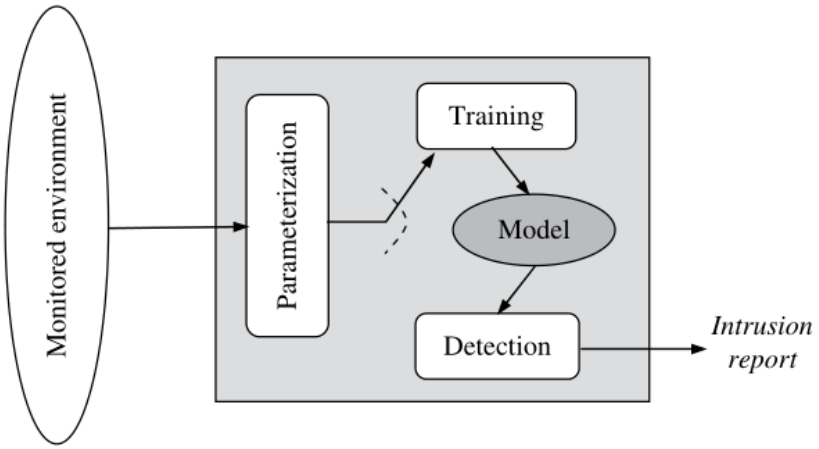

Figura 3: Arquitetura de um A-IDS [14]

podem incluir, por exemplo, o número de pacotes para cada protocolo, a taxa de ligações ou o número de endereços IP diferentes.

Durante o processo de detecção de anomalias, são considerados dois *datasets*, sendo que um deles corresponde ao perfil dos dados que estão correntemente a circular e o segundo corresponde ao perfil estatístico gerado na fase de treino. À medida que os eventos de rede ocorrem, o perfil corrente é determinado e comparado com o perfil de treino, resultando um *score* de anomalia. Este *score* traduz o grau de irregularidade para o evento em questão, sendo que o IDS irá despoletar um alerta se este *score* ultrapassar os patamares pré-determinados.

Este tipo de IDS não exige conhecimento prévio acerca da actividade normal e permite uma notificação muito precisa das actividades maliciosas. As principais desvantagens prendem-se com o facto de que o sistema pode ser treinado pelos atacantes e com a dificuldade na definição dos parâmetros e das métricas.

Os IDS **knowledge-based** caracterizam-se, sobretudo, pelo recurso a *expert systems* ou sistemas periciais, sendo esta a técnica mais frequentemente utilizada neste tipo de IDS.

É suposto que os sistemas periciais classifiquem os dados capturados de acordo com um conjunto de regras, processo esse que envolve três passos. Primeiramente, é feita a identificação das diferentes classes e atributos a partir dos dados de treino; de seguida, o sistema deduz um conjunto de regras, parâmetros e procedimentos de classificação; por fim, os dados capturados são classificados de acordo com as regras deduzidas.

As vantagens deste tipo de abordagem incluem a sua robustez, flexibilidade e escalabilidade; o ponto negativo é que o desenvolvimento e obtenção de conhecimento de alta qualidade é extremamente difícil e exige muito tempo.

Seguindo as mais recentes tendências, surgiram, também, IDS com base em técnicas de aprendizagem automática, normalmente designadas por ***machine learning***. Estes sistemas estabelecem um modelo explícito ou implícito com o objectivo de possibilitar a classificação dos padrões analisados. Um pormenor deste tipo de sistemas é a sua necessidade de um conjunto de dados pré-classificado para treinar o modelo de comportamento. Esta é uma operação que, tipicamente, consome muitos recursos do sistema.

Os sistemas baseados em *machine learning* tentam, utilizando o conhecimento já disponível, melhorar o modelo actual, adaptando-se, assim, às novas circunstâncias e promovendo um aumento de desempenho do mesmo.

De entre os diferentes algoritmos de *machine learning* existentes, os mais representativos no que aos IDS diz respeito são as redes Bayesianas, as redes neuronais, os modelos de Markov, técnicas de *fuzzy-logic* e algoritmos genéticos. Na secção 2.4 serão descritos os principais algoritmos bioinspirados com possibilidade de serem utilizados em A-IDS, nos quais se incluem os algoritmos genéticos e as redes neuronais.

Estes sistemas apresentam uma elevada flexibilidade e adaptabilidade; os pontos negativos deste tipo de sistema são o seu elevado consumo de recursos e a grande dependência relativamente à assumpção sobre o comportamento aceite para o sistema.

2.3 DATASETS PARA TESTAR IDS

O desenvolvimento de um IDS implica, naturalmente, que o mesmo seja testado antes de ser disponibilizado aos futuros utilizadores.

Essa fase de testes permite avaliar se o sistema em teste corresponde, efectivamente, ao que se pretende, permitindo aferir o grau de eficácia e eficiência do mesmo.

Para garantir que os resultados dos testes sejam tão fiáveis quanto possível, é desejável que os dados de teste correspondam, tanto quanto possível, a situações reais.

Por outro lado, é, também, desejável que esses dados estejam disponíveis publicamente, de forma a ter *benchmarks* conclusivos sobre o desempenho da aplicação de várias técnicas de ML sobre os mesmos dados.

Sendo os dados recolhidos em ambientes empresariais, é normal, por motivos de privacidade e confidencialidade, que esses dados não possam ser disponibilizados publicamente.

Surge, assim, a necessidade de gerar dados tão anonimizados e tão aproximados à realidade quanto possível, garantindo a sua disponibilização pública para efeitos de análise ou teste.

De acordo com [15], o uso de *datasets* para efeitos de teste de novos sistemas é recomendável, pois permite:

- *Repetibilidade dos testes*: o uso de datasets permite que investigadores diferentes possam repetir os testes nas mesmas condições e obter os mesmos resultados;
- *Validação de novos métodos*: surgem novos métodos e algoritmos de detecção de anomalias, que necessitam, obviamente, de ser testados e validados;
- *Comparação de diferentes métodos*: O uso das mesmas condições de teste, nomeadamente os mesmos dados, permite estabelecer comparações de performance dos diferentes métodos de detecção utilizados; pode ser o caso de, por exemplo, validar se um novo método tem melhor resposta do que um dos outros existentes;
- *Afinação de parâmetros*: o recurso a testes com datasets permite testar as diversas parametrizações de um determinado algoritmo, possibilitando a optmização de performance e eficácia do mesmo;
- *Dimensionamento do número de features*: os testes permitem definir quais os dados mais relevantes para o desempenho do algoritmo, bem como identificar os menos relevantes e que podem, eventualmente, ser ignorados.

Em termos de *datasets* disponíveis, em [16] apresenta-se um estudo bastante recente sobre os *datasets* existentes para efeitos de teste de IDS. Na sequência do levantamento efectuado, os autores elaboraram um resumo das principais características e campos de aplicação de cada *dataset*, destacando, também, se estavam disponíveis publicamente ou não. Os *datasets* foram classificados de acordo com vários parâmetros, tais como o ano da criação, o tipo de dados que o compõem, qual ou quais os tipos de ataque que abrange, se engloba apenas tráfego normal ou se também possui tráfego malicioso e qual a duração do período de recolha.

Um dos datasets mais popular e que foi, até muito recentemente, utilizado para teste e avaliação de IDS, é o KDDCup1999[3]. Este dataset surgiu numa competição anual de Data Mining cujo objectivo, no ano de 1999, era desenvolver um modelo de um sistema de detecção de intrusões de rede baseado em comportamento. Apesar da sua popularidade, o uso deste dataset já não é recomendado [17, 18], pois desde então as tipologias de ataques de rede mudaram, tendo, inclusivamente, surgido novos tipos de ataques não contemplados no dataset.

Em termos de datasets mais recentes, o artigo analisa ainda outros dois *datasets*: o UNSW-NB15[4] e o CICIDS[5]. Moustafa e Slay [19] descrevem a criação do *dataset* UNSW-NB15, um dataset disponível para efeitos de investigação e pesquisa. Este *dataset* foi desenvolvido tendo como base as características de tráfego normal e com anomalias dos dias de hoje, numa aproximação tanto quanto possível ao ambiente real.

De acordo com Sharafaldin. et al. [20], o CICIDS foi desenvolvido pelo Canadian Institute for Cybersecurity (CIC). Utilizando um ambiente simulado, e durante cinco dias, foram realizados vários tipos de ataques, incluindo ataques DoS, de infiltração, ataques a plataformas web e ataques *brute-force*. Durante esse período, todo o tráfego foi capturado, sendo depois analisado e trabalhado para produzir o *dataset*.

Com o objectivo de manter o dataset tão actualizado quanto possível, o CIC disponibilizou novas versões do mesmo, sendo que a última data de 2018, tendo resultado de uma parceria entre o CIC e o Communications Security Establishment (CSE). Esta versão será analisada com mais detalhe na secção 4.4.

2.4 ALGORITMOS BIOINSPIRADOS

Algoritmos bioinspirados são todo um conjunto de algoritmos cuja base de funcionamento assenta sobre sistemas ou mecanismos da natureza ou do organismo humano.

No artigo [21], Mahboubian e Hamid referem como típicas destas analogias as aplicações:

- nas áreas das redes neuronais - com inspiração no funcionamento do cérebro humano;

3 http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
4 https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/
5 https://iscxdownloads.cs.unb.ca/iscxdownloads/

- na computação evolucionária e por DNA - com base nas teorias da evolução e que conduz aos algoritmos genéticos;
- os sistemas imunológicos artificiais, que se baseiam no sistema imunológico dos seres vivos vertebrados, nomeadamente o sistema imunitário do ser humano. De referir que as plantas também possuem um sistema imunitário, mas não é tão interessante do ponto de vista das ciências da computação, já que não possui as componentes adaptativa e de aprendizagem.

2.5 ALGORITMOS BASEADOS NO SISTEMA IMUNOLÓGICO HUMANO

O sistema imunológico humano e o seu funcionamento servem de base aos AIS ou Sistemas Imunológicos Artificiais. Estes sistemas englobam todo um conjunto de algoritmos cuja base de funcionamento assenta em algumas particularidades do sistema imunológico humano, permitindo a sua aplicação nos mais diversos campos.

A principal função do sistema imunológico humano é a protecção do organismo contra o ataque de agentes patogénicos externos. A analogia com os IDS é imediata, pois estes também como missão principal a detecção de potenciais ameaças e protecção da rede informática contra as mesmas.

De seguida, será abordado, de forma tão simples quanto possível, o funcionamento do sistema imunológico humano, a transposição desse funcionamento para os sistemas imunológicos artificiais e o funcionamento do algoritmo utilizado nesta dissertação, o CLONALG.

2.5.1 *O sistema imunológico humano*

O objectivo desta secção é descrever, de forma tão simples e sintética quanto possível, o funcionamento genérico do sistema imunológico humano. A informação que se segue tem como base o artigo [22].

A missão ou principal função do sistema imunológico humano é a protecção e defesa do organismo contra ataques de agentes patogénicos. Essa protecção é garantida através de métodos bastante sofisticados de reconhecimento de padrões e de mecanismos de resposta dependentes do tipo de invasor, do dano que possa causar ao organismo e da forma de entrada no mesmo. Os mecanismos de resposta podem destruir ou neutralizar o efeito do agente invasor.

Analogamente, a função dos IDS é, precisamente, proteger os sistemas e redes informáticos, tentando detectar ameaças ou intrusões e reportando e enviando alertas para os administradores de sistemas.

O sistema imunitário humano é constituído por vários orgãos (ver Figura 4), que asseguram, entre si, a defesa do organismo. Também os IDS são constituídos por vários componentes e sensores, também eles responsáveis por assegurar a detecção e notificação de ciber-ameaças.

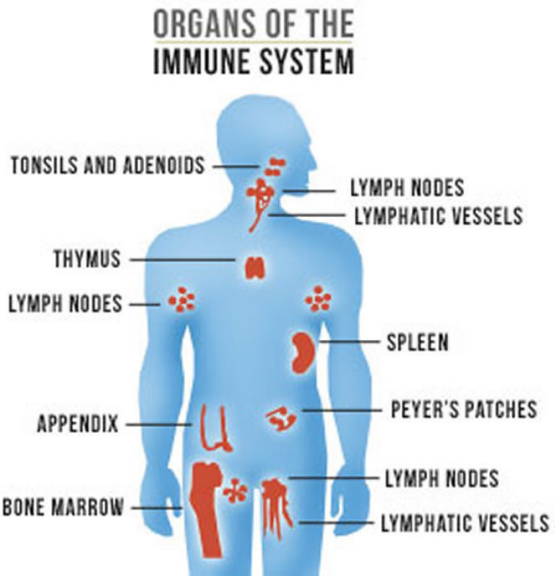

Figura 4: Sistema Imunológico Humano [23]

O funcionamento do sistema imunológico humano baseia-se, sobretudo, na identificação e reconhecimento de agentes externos - os potenciais agentes patogénicos.

Os agentes patogénicos possuem, à superfície, um conjunto de proteínas denominadas antigénios, sendo que essas proteínas são específicas para cada agente patogénico, permitindo, assim, a identificação ou reconhecimento do agente patogénico. Estes antigénios designam-se por antigénios *nonself*. Por outro lado, também o organismo humano, nomeadamente os seus tecidos constituintes, possuem antigénios, conhecidos como antigénios *self*.

O processo pelo qual o organismo efectua a discriminação entre antigénios *self* e antigénios *nonself* denomina-se discriminação *self/nonself*, e é este o processo que permite detectar se há ou não invasão do organismo.

Ao nível dos IDS, também estes sistemas têem que decidir, perante o tráfego de rede, se o tráfego é legítimo/normal (*self*) ou se há alguma intrusão (*nonself*), num processo em tudo semelhante ao que se passa no organismo humano.

Pode dizer-se que o sistema imunológico humano garante dois tipos de imunidade, a imunidade específica ou adaptativa e a imunidade não específica ou inata (ver Figura 5).

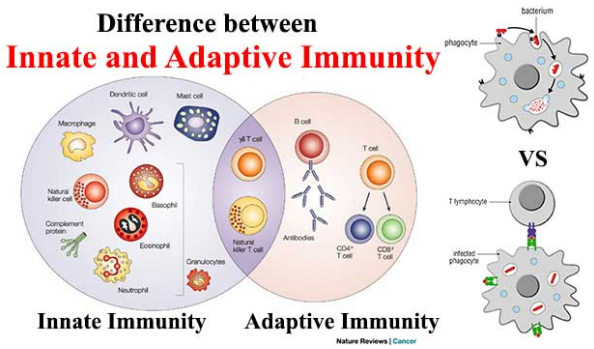

Figura 5: Imunidade Inata vs. Adaptativa [24]

A imunidade não específica é responsável pela defesa genérica, não identificando tipos específicos de agentes patogénicos. É também conhecida por sistema imunitário inato, já que o corpo humano possui, naturalmente, a capacidade de atacar e destruir tipos específicos de antigénios. Este sistema é composto por diversas barreiras defensivas, destinadas a combater partículas estranhas, vírus, parasitas e bactérias, disponibilizando mecanismos de defesa genéricos que permitem anular a influência dos invasores junto das células.

Nos sistemas informáticos, a analogia com a imunidade inata conduz-nos às assinaturas dos anti-vírus ou dos IDS baseados em assinaturas. São bases de dados que permitem reconhecer determinados tipos de ataques tendo em conta determinados padrões/assinaturas dos mesmos.

A imunidade específica, também designada por adaptativa, é um mecanismo de resposta mais específica e direccionada mas, normalmente mais lento que o da imunidade inata. Assegura uma resposta diferenciada para cada tipo de agente patogénico, através de mecanismos de memória e de produção de anti-corpos específicos.

A imunidade específica, também denominada de adaptativa, é, precisamente, o mecanismo que serve de base à aplicação dos AIS aos IDS baseados em comportamento. Os sensores do IDS analisam o tráfego de rede, tentando detectar e identificar desvios ao comportamento normal da rede. Caso seja detectado comportamento anómalo, é gerado um alerta, ao mesmo tempo que o sistema tenta determinar um padrão que permita reconhecer um futuro ataque, guardando essa informação para referência futura.

2.5.2 *Selecção Clonal*

A selecção clonal é uma teoria, desenvolvida por Jerne, Talmage e Burnet com o objectivo de descrever o funcionamento do sistema imunitário humano, nomeadamente da imunidade adaptativa ou adquirida [25, 26, 27].

De acordo com o artigo «Clonal selection theory & clonalg-the clonal selection classification algorithm (csca)» [28], a teoria da selecção clonal especifica que o organismo possui um conjunto de anti-corpos que podem reconhecer, com algum nível de especificidade, todos os antigénios. Quando um antigénio é detectado, o anti-corpo liga-se quimicamente ao antigénio e replica-se, produzindo mais células com o mesmo receptor.

Durante o processo de replicação, podem ocorrer mutações genéticas que irão assegurar uma melhor afinidade ao antigénio. Como resultado, a capacidade de ligação das células melhora ao longo do tempo e da exposição ao antigénio.

Para garantir uma resposta mais eficaz num futuro ataque pelo mesmo antigénio, são, também, geradas "células memória", que mantém as propriedades dos anti-corpos mais adequados ao combate ao antigénio.

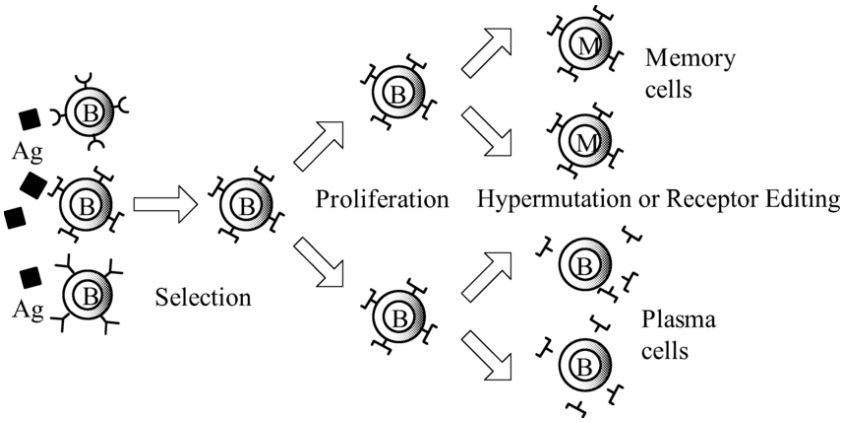

Figura 6: Selecção clonal [29]

Resumindo, a selecção clonal faz uma clonagem dos anti-corpos com maior afinidade com os antigénios, melhorando, assim, a resposta específica do organismo a futuras invasões por parte daquele antigénio específico.

Esta teoria é de fácil adaptação aos sistemas informáticos em geral e aos IDS em particular. Os sensores dos IDS, que irão analisar o tráfego de rede, serão os equivalentes aos anticorpos, e os antigénios correspondem a todo o tráfego malicioso que seja detectado.

2.5.3 *Algoritmos imunoinspirados*

As analogias com o sistema imunológico humano servem de base para o desenvolvimento dos sistemas imunológicos artificiais (AIS). No fundo, trata-se de adaptar e aplicar os métodos utilizados pelo sistema imunitário à resolução de problemas de engenharia e de computação.

Tendo como base esse pressuposto, foram desenvolvidos vários algoritmos, que incorporavam uma determinada característica do modo de funcionamento do sistema imunológico humano.

Foi assim que surgiram algoritmos como o Negative-selection [30], o CLONALG [3, 4]) ou o AIRS (Artificial Immune Recognition System) [31, 32].

Tendo em conta a sua origem, qualquer um destes algoritmos pode ser utilizado como ferramenta de detecção de intrusão de rede, pois a sua principal motivação é, precisamente, a detecção de "corpos estranhos".

A aplicabilidade dos AIS e dos algoritmos imunoinspirados às ciências da computação resulta das analogias com o sistema imunitário dos seres humanos. De facto, a natureza já lida com detecção de anomalias há milhões de anos.

Como já vimos na secção 2.5.1, a função principal do sistema imunitário é, precisamente, a função de detecção e resposta a intrusões. A função dos IDS é, precisamente, a detecção de intrusões, emitindo alertas caso alguma se verifique.

Por outro lado, o sistema imunitário adapta-se a novas situações, aprendendo e memorizando as características do agente invasor de forma a poder combatê-lo eficazmente mais tarde. Os IDS baseados em comportamento também possuem um mecanismo de aprendizagem que lhes permite adaptar-se a novas situações, possibilitando a detecção de ataques ainda não conhecidos.

De entre a variedade de algoritmos disponíveis, o que será utilizado neste trabalho é o CLONALG, pois é um algoritmo de implementação relativamente simples, bastante utilizado pela comunidade científica e cujas características satisfazem os requisitos deste trabalho. O CLONALG tem, ainda, como mais-valia o facto de estar disponível uma implementação do mesmo para a aplicação de Data-Mining utilizada, o WEKA.

2.6 ALGORITMOS BASEADOS NO SISTEMA NEUROLÓGICO

Os algoritmos baseados no sistema neurológico procuram a sua inspiração no funcionamento do cérebro dos vertebrados, nomeadamente nos neurónios e nas redes por eles formadas. Daqui surgem, então, as redes neuronais artificiais (ANN).

Assim, as ANN são uma abstracção computacional que, ao pretender simular o funcionamento do cérebro dos vertebrados, disponibiliza um conjunto de ferramentas, de implementação relativamente simples, que podem ser utilizadas na resolução de problemas com diversos graus de complexidade.

De acordo com o livro *Principles of artificial neural networks* [33], a importância das ANN deve-se ao facto de a sua implementação utilizar apenas operações matemáticas simples para resolver problemas matemáticos complexos, podendo ser utlizadas para resolver qualquer problema. Por oposição, os métodos convencionais utilizam conjuntos de equações bastante complexos e são específicos para um determinado problema.

Resumindo, e ainda de acordo com [33], as ANN são, computacionalmente e algoritmicamente, bastante simples e, para além disso, possuem uma estrutura auto-organizativa que lhes permite serem aplicadas na resolução de diferentes tipos de problemas.

2.6.1 *Redes Neuronais Biológicas*

As redes neuronais biológicas são constituídas por todo um conjunto de neurónios interligados entre si. O neurónio é, portanto, o elemento-base das redes neuronais biológicas, pelo que vamos analisar brevemente a sua constituição e funcionamento.

Na figura 7 podemos ver os elementos que constituem um neurónio.

O neurónio é, então, constituído por [33]:

- dendrites, que são pontos de entrada de impulsos electro-químicos, provenientes de axónios de outros neurónios;
- corpo do neurónio, que inclui o respectivo núcleo; este é, propriamente dito, o centro de processamento do neurónio;
- axónio, que é, essencialmente, um meio de interligação a outros neurónios, através dos terminais do axónio

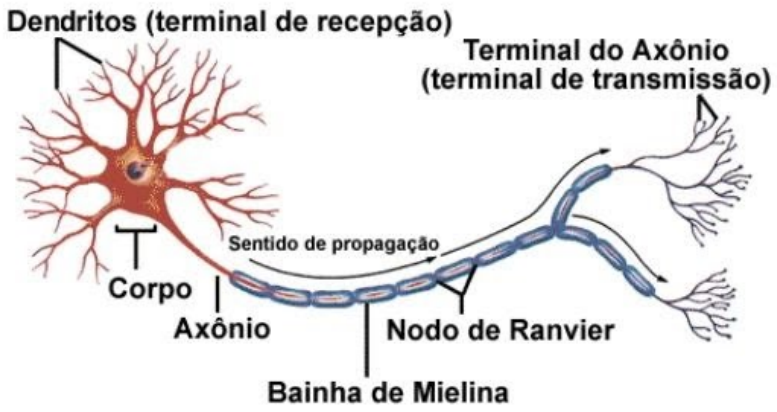

Figura 7: Constituição de um neurónio [34]

As redes neuronais biológicas são, assim, interligações de neurónios. Essas interligações são realizadas entre os terminais do axónio de um neurónio e as dendrites de outro, formando redes extremamente densas de ligações.

Em termos de funcionamento, e ainda de acordo com o livro [33], a actividade neuronal passa de um neurónio para outro através de pulsos electro-químicos, que seguem pelo axónio até à dendrite do(s) neurónio(s) seguinte(s).

É, também, sabido que as ligações entre neurónios não possuem todas a mesma prioridade, sendo algumas delas mais determinantes do que outras na resposta final do neurónio.

No que aos IDS diz respeito, podemos considerar uma rede neuronal como sendo um sistema de detecção de intrusões, em que a entrada será o tráfego de rede e a saída será a determinação sobre se esse mesmo tráfego será normal ou malicioso. Este paradigma será analisado na secção seguinte.

2.6.2 *Redes Neuronais Artificiais*

As redes neuronais artificiais são um paradigma computacional que pretende simular o funcionamento das redes neuronais biológicas, permitindo a sua utilização na resolução de vários tipos de problemas.

À semelhança das redes neuronais biológicas, as ANN são constituídas por interligações de elementos-base, denominados perceptrões (da terminologia inglesa *Perceptron*). O perceptrão é, então, o equivalente computacional do neurónio (Figura 8).

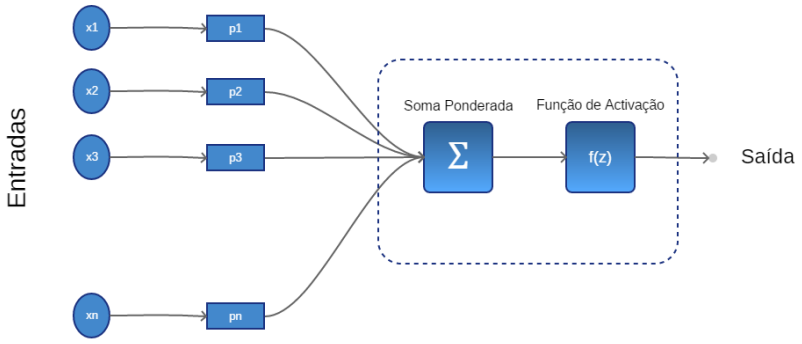

Figura 8: Perceptron - Neurónio artificial (adaptado de [33])

O modo de funcionamento do perceptrão é relativamente simples. A cada entrada x_n está associada uma ponderação p_n, que traduz a importância do respectivo nó para o resultado final. Essa ponderação ou peso é multiplicado pelo respectivo valor de entrada, após o que se procede ao cálculo da soma ponderada. É esta soma que é apresentada à função de activação, que irá, então, determinar o valor de saída. A função de activação é essencial para garantir que os valores da saída se encontrem dentro do intervalo estipulado, tal como [0,1] ou [−1,1].

2.7 OUTROS TIPOS DE ALGORITMOS BIOINSPIRADOS

Para além dos algoritmos imunoinspirados e das redes neuronais artificiais, existem, também os seguintes tipos de algoritmos bioinspirados:

- Algoritmos baseados na genética e evolução
- Sistemas autonómicos
- Swarm Intelligence

Os algoritmos baseados na genética e evolução, também conhecidos por **algoritmos genéticos**, são a representação computacional dos conceitos biológicos da evolução [35, 36]. Estes algoritmos procuram implementar os conceitos que suportam a evolução na natureza, nomeadamente o cruzamento, a mutação, e a selecção dos mais aptos.

De acordo com o livro *Genetic algorithm essentials* [36], da autoria de Kramer, o primeiro passo de qualquer algoritmo genético é gerar uma população inicial - que pode ser, por exemplo, um conjunto possível de soluções para um determinado problema. De seguida, duas ou mais soluções são recombinadas e o resultado desta recombinação é alvo de mutação. As melhores soluções geradas desta forma são, a

posteriori, seleccionadas para a geração seguinte. O processo é repetido durante um número máximo de gerações.

Os **sistemas autonómicos**, como o próprio nome indica, são sistemas que são capazes de funcionar com um mínimo de intervenção humana. De acordo com o artigo «The dawning of the autonomic computing era» [37], da autoria de Ganek e Corbi, os sistemas autonómicos devem possuir as seguintes características:

- Auto-configuração (*self-configuring*) - este tipo de sistemas deve poder adaptar-se automatica e dinamicamente em reacção a mudanças no seu ambiente
- Auto-reparável (*self-healing*) - estes sistemas descobrem, diagnosticam, e reagem a avarias ou interrupções. Para seja auto-reparável, o sistema deve conseguir recuperar à falha de um componente. Para isso, o sistema deve detectar o componente em falha, isolá-lo, desligá-lo, proceder à reparação ou substituição do componente e recolocá-lo em funcionamento. Todos estes procedimentos deverão ocorrer de forma transparente, ou seja, não deve provocar falha ou interrupção visíveis do sistema.
- Auto-optimizável (*self-optimizing*) - os sistemas autonómicos monitorizam e ajustam a utilização de recursos automaticamente. A auto-optimização requer que os sistemas - *hardware* e *software* - sejam capazes de maximizar de forma eficiente e sem intervenção humana, a utilização dos recursos disponíveis de forma a corresponder às necessidades do utilizador final.
- Auto-protecção (*self-protecting*) - os sistemas autonómicos antecipam, detectam, identificam e tomam medidas de forma a se protegerem de todo e qualquer tipo de ataques.

De acordo com o livro *Autonomic computing: principles, design and implementation* [38], os sistemas autonómicos podem ir buscar inspiração nas mais diversas áreas, desde a economia à biologia, passando pelas diversas áreas os sistemas computacionais. No que toca à biologia, a inspiração deve-se ao sistema nervoso autonómico, em particular do sistema nervoso humano.

Ainda de acordo com [38], o sistema nervoso humano é o mais complexo dos que compõem o organismo humano. É, essencialmente, composto por dois sub-sistemas: o sistema nervoso central e o sistema nervoso periférico.

O sistema nervoso central é constituído pelo cérebro e pela espinal medula. Este sistema é responsável pela integração de toda a informação recebida das diferentes partes do corpo e pela transmissão de sinais destinados a controlar as acções do corpo.

Por seu lado, o sistema nervoso periférico é responsável pela interligação entre o sistema nervoso central e os diferentes tecidos do organismo. É formado pelos nervos cranianos e espinais e por diferentes aglomerados de nervos denominados gânglios. Os diferentes componentes do sistema nervoso são interligados por uma complexa e intrincada rede neuronal.

Em termos de analogia com os sistemas informáticos, podemos dizer que o sistema nervoso central pode ser visto como uma unidade de processamento e que o sistema nervoso periférico será a infraestrutura intermediária de ligação entre o sistema nervoso central e os sensores e actuadores.

Um outro tipo de algoritmo bioinspirado , denominado **swarm intelligence**, vai buscar inspiração ao comportamento de enxames (*swarm*, na terminologia inglesa) de insectos ou animais e às suas capacidades para resolver problemas complexos enquanto enxame, embora essas tarefas possam parecer de difícil resolução ao nível individual.

De facto, conforme mencionado no artigo «Swarm intelligence in intrusion detection: A survey» [39], as formigas, abelhas ou até algumas aves ou peixes aparentam possuir uma inteligência limitada enquanto indivíduos, mas quando interagem socialmente entre eles e com o seu ambiente conseguem realizar tarefas bastante complexas, tais como identificar os caminhos mais curtos para uma fonte de alimento, organizar o ninho, sincronizar os seus movimentos de forma a movimentarem-se rapidamente como uma entidade única e coerente.

Os sistemas de detecção de intrusões utilizam, depois, os modelos resultantes para a execução de tarefas como a distinção entre comportamento normal ou anormal, a determinação da fonte do ataque ou a optimização de performance.

Ainda de acordo com o artigo de Kolias et al. [39], a motivação para o uso deste tipo de tecnologia nos IDS deve-se, essencialmente, ao facto de estes sistemas naturais possuirem um conjunto bastante atraente de características que podem, de imediato, ser herdadas pelos sistemas.

Mais especificamente, as características do *swarm intelligence* têm como objectivo a resolução de problemas complexos utilizando vários mas simples agentes, sem necessidade de supervisão. Cada agente colabora com os restantes no sentido da obtenção da solução óptima. Na situação dos IDS, os agentes podem ser utilizados para várias tarefas, como sejam a descoberta de regras de classificação para detecção de uso indevido, a descoberta de clusters para detecção de anomalias ou a manutenção de um registo de rotas de intrusão. Estes atributos, auto-organizáveis e distribuídos, são bastante apreciados, pois permitem a decomposição de um IDS

em tarefas mais simples, divididas pelos diversos agentes. Conforme mencionado no artigo [39], este facto permite que os IDS possam potencialmente ser autónomos, altamente adaptativos, auto-organizados e economicamente eficientes.

2.8 ALGORITMOS UTILIZADOS

Nas secções seguintes serão descritos e analisados os 3 algoritmos seleccionados para esta dissertação, um baseado nos sistemas imunológicos artificiais - e os outros dois baseados em redes neuronais artificiais.

2.8.1 *Algoritmo CLONALG*

O CLONALG (CLONal selection ALGorithm) é um algoritmo bioinspirado que se baseia na teoria da selecção clonal (*clonal selection*, na terminologia inglesa). A implementação deste algoritmo no WEKA teve como referência os trabalhos de de Castro e Von Zuben [3, 4].

2.8.1.1 *O algoritmo*

De acordo com o relatório técnico elaborado por Brownlee [28], o CLONALG assenta nos seguintes princípios da teoria da selecção clonal:

- Criação e manutenção de um conjunto de memória
- Selecção e clonagem dos anti-corpos mais estimulados
- Destruição dos anticorpos não estimulados
- Maturação por afinidade (Mutação)
- Re-selecção dos clones de forma proporcional à afinidade com o antigénio
- Geração e manutenção da diversidade

Ainda de acordo com o artigo [28], o objectivo deste algoritmo é gerar um conjunto (*pool*, na designação inglesa) de anticorpos que seja uma solução para um determinado problema. Neste caso, um anticorpo seria uma solução ou parte da solução, ao passo que o antigénio representa o problema ou parte dele.

O algoritmo e, consequentemente, o conjunto de anticorpos final, resultam da aplicação da teoria da selecção clonal e do processo de maturação por afinidade, ambos já descritos anteriormente (cf. Secção 2.5.2).

O algoritmo 1 contém o pseudo-código do algoritmo CLONALG.

Dados: $G = NumerodeGerações$
Resultado: Conjunto de anticorpos de memória
Preparar conjunto de N anticorpos
para $i \leftarrow 1$ **até** G **faça**
  para cada $ag \in conjunto\_de\_antigenios$ **faça**
    Seleccionar aleatoriamente um antigénio
    Calcular a afinidade do antigénio
    Seleccionar os n anticorpos com melhor afinidade
    Clonar os anticorpos seleccionados
    Maturação de afinidade dos clones
    Calcular afinidade dos clones
    Colocar os clones com maior afinidade no conjunto de anticorpos de memória
    Substituir os d anticorpos com menor afinidade
  fim
fim

Algoritmo 1: Pseudo-código do algoritmo CLONALG (adaptado de [28])

Conforme podemos verificar, o primeiro passo do algoritmo é a **Inicialização**. Nesta fase, é criado um conjunto de anticorpos de tamanho pré-definido e fixo **N**. Este conjunto é, posteriormente, dividido em duas partes, uma secção correspondente à solução do algoritmo **m** e outra, a remanescente ou **r**, cuja função é introduzir diversidade no sistema.

Na fase seguinte, e durante **G** iterações, o algoritmo irá expor o sistema a todos os antigénios. Cada iteração é denominada **Geração**, sendo que o número de gerações é configurada pelo utilizador.

No ciclo principal e para cada antigénio, as operações que ocorrem são as seguintes [28]:

1. Selecciona, aleatoriamente e sem reposição, um antigénio;
2. O sistema é exposto ao antigénio, sendo calculados os valores de afinidade do antigénio face aos anti-corpos existentes. A afinidade é uma medida de semelhança e varia consoante o problema em causa. A afinidade é, usualmente, medida através do cálculo da distância de Hamming;
3. Os **n** anti-corpos com maiores valores de afinidade são seleccionados;

4. Os n anti-corpos seleccionados são objecto de clonagem, proporcionalmente ao respectivo valor da afinidade (maior afinidade produzirá maior número de clones);
5. O conjunto de clones é, então, sujeito a um processo de maturação da afinidade por mutação. Esta operação visa aumentar, ainda mais, a afinidade do anti-corpo ao antigénio, sendo que a mutação será tanto maior quanto menor for a afinidade;
6. Os clones maturados são expostos aos antigénios, com o consequente cálculo das afinidades;
7. O(s) anti-corpo(s) com maior afinidade são, então, promovidos a anti-corpos memória, sendo colocados no pool de memória. Se a afinidade deste anti-corpo é maior do que a do anti-corpo com maior afinidade existente na pool de memória, então este último é substituído pelo novo;
8. No último passo deste ciclo, os **d** anti-corpos com menor afinidade existentes no conjunto remanescente (r) são substituídos por novos anti-corpos

Após as G gerações, o conjunto de anti-corpos que constituem a componente de memória são devolvidos como sendo a solução para o problema.

Como se pode verificar, este algoritmo incorpora noções de selecção e de diversidade. De facto, ao clonar os anti-corpos com maior afinidade está a assegurar a "sobrevivência dos mais adaptados", o princípio básico da selecção natural. Quando substitui os anti-corpos com menor afinidade por outros novos, está a promover a diversidade da solução.

2.8.1.2 *Parâmetros*

De acordo com artigo [28], o CLONALG pode ser configurado através dos seguintes parâmetros:

1. **População de anti-corpos (N)** - Denominada *antibody pool size* na terminologia inglesa, representa o número total de anti-corpos do sistema. Este número inclui o número de anti-corpos de memória (*memory antibody pool size*) e o número de anti-corpos remanescentes (*remainder antibody pool size*). A forma como é realizada a divisão não está explicitada, sendo que a forma mais simples é especificar um número m correspondente ao número de anti-corpos memória. Como é óbvio, $m \leq N$. Por consequência, o número de anti-corpos remanescentes r pode ser obtido através de $r = N - m$.

2. **Conjunto de selecção (n)** - denominado *selection pool size*, representa o número de anti-corpos que será seleccionado para clonagem. O número de anti-corpos a seleccionar n deverá estar compreendido no intervalo $n \in [1,N]$. O valor de n define a pressão selectiva colocada sobre os anti-corpos para atingir altos valores de afinidade e, por conseguinte, proliferar. Valores mais pequenos de n implicam maior pressão, tendo como eventual consequência uma diminuição da diversidade da população, garantindo que apenas os anti-corpos com as melhores (maiores) afinidades são clonados e substituem os elementos existentes no conjunto de memória.

3. **Conjunto remanescente (d)** - no original inglês *remainder pool size*, especifica o número de anti-corpos com as afinidades mais baixas que serão substituídos por anti-corpos aleatórios a cada exposição aos antigénios. Sendo r o número de anti-corpos remanescentes, temos que $d \in [0,r]$. Este parâmetro é responsável por introduzir um mecanismo adicional de diversidade à população de anti-corpos. O mecanismo pode ser desactivado, bastando, para tal, que o valor de d seja 0 (zero).

4. **Factor clonal (β)** - este parâmetro serve de factor de escala para o número de clones criado para cada anti-corpo seleccionado. De acordo com [28], o número de clones criado para cada anti-corpo pode ser obtido pela expressão

$$numClones = \left\lfloor \frac{\beta \times N}{i} + 0.5 \right\rfloor$$

onde β é o factor clonal, N é o número total de anti-corpos e i é a posição actual (*ranking*) do anti-corpo em causa, $i \in [1,n]$.

5. **Número de gerações (G)** - Especifica o número total de iterações que o algoritmo vai executar. Em cada iteração, o sistema será exposto a todos os antigénios. Este parâmetro traduz uma medida da "quantidade de aprendizagem"que o algoritmo irá realizar sobre o problema em questão.

2.8.2 *Algoritmo Multi-Layer Perceptron*

De acordo com [6], Multi-Layer Perceptron é uma classe de redes neuronais constituída por, pelo menos, três nós. Cada um destes nós, exceptuando o nó de entrada (*input*), é um neurónio que utiliza uma função de activação não-linear.

Os nós estão distribuídos por camadas, sendo elas a de entrada, a de saída e as escondidas (*hidden layers*). As camadas escondidas são todas as camadas que se encontram entre a camada de entrada e a de saída (ver Figura 9).

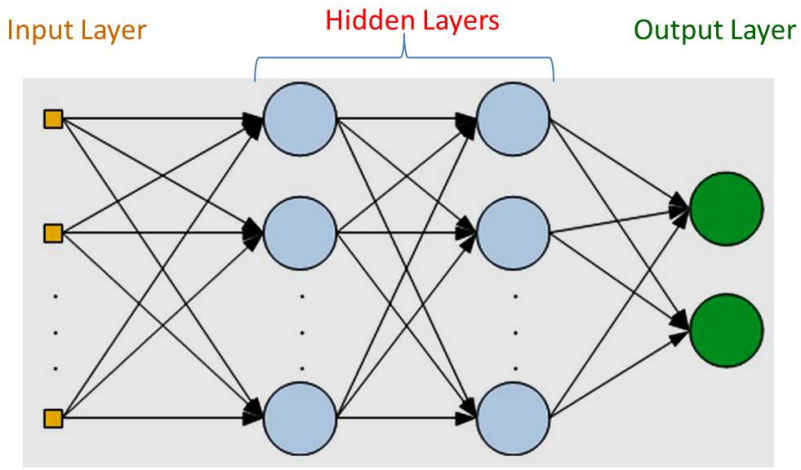

Figura 9: Esquema simplificado do Multi-Layer Perceptron [6]

Como todos os algoritmos de ML, a aplicação deste algoritmo implica uma fase aprendizagem e uma fase de teste. Na secção seguinte, será abordado o mecanismo de aprendizagem.

2.8.2.1 *Mecanismo de Aprendizagem*

A aprendizagem deste algoritmo é realizada em duas fases principais, denominadas *feedforward* e *back propagation* (Figura 10).

Na fase de *feedforward*, cada vector de entrada é propagado pela rede, provocando reacções dos vários perceptrões e produzindo um valor de saída. Esse valor de saída é, então, comparado com o valor esperado, sendo gerado um valor de erro para cada nó da camada de saída.

Como cada nó contribuiu, em certo grau, para o erro presente na camada de saída, os valores dos erros são transmitidos para cada nó da camada escondida imediatamente anterior à camada de saída, e assim sucessivamente, até que cada nó na rede tenha recebido um sinal de erro que atesta a sua contribuição para o erro na camada de saída. Esta fase denomina-se ***back propagation***.

Assim que esteja determinado o valor do erro para cada nó, esse valor é, depois, utilizado por cada nó para ajustar as ponderações de cada entrada, até que a rede convirja para um estado que permita que todos os vectores de entrada possam ser representados.

O objectivo da fase de *back propagation* é minimizar o valor da função de erro. O conjunto das ponderações que minimizam a função de erro são considerados a solução do problema.

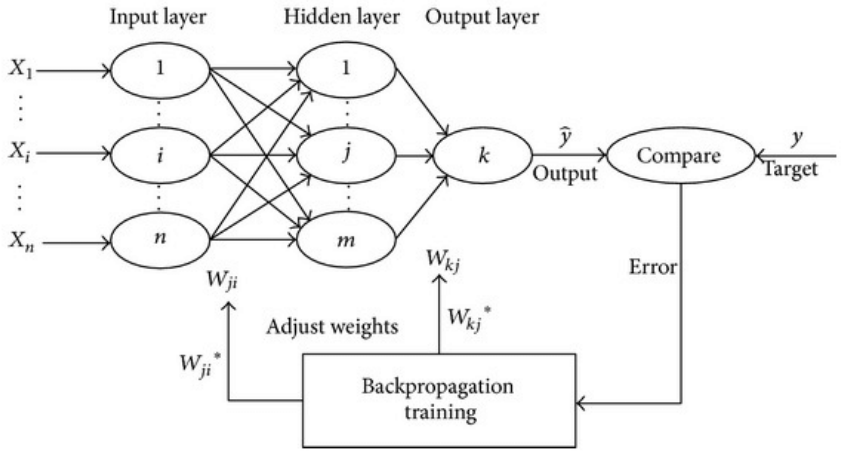

Figura 10: MLP e back propagation [40]

2.8.2.2 *Parâmetros*

De acordo com a implementação do Multi-Layer Perceptron (MLP) no WEKA, os parâmetros utilizados pelo algoritmo são os seguintes:

1. **Hidden Layers** - especifica a constituição das camadas escondidas (*hidden layers*, na terminologia inglesa). É uma lista de números inteiros positivos separados por vírgulas em que cada número representa o número de perceptrões na respectiva camada. Este parâmetro admite três valores especiais "a", "t","o"e "i", sendo que o número de perceptrões correspondente encontra-se na tabela 2.
2. **Learning Rate** - é a taxa de aprendizagem, utilizada na actualização dos pesos de cada entrada; tipicamente é um valor decimal entre 0 e 1, sendo 0.3 o valor mais comum;
3. **Momentum** - valor utilizado na actualização das ponderações de cada entrada, em conjunto com a taxa de aprendizagem; é, também, um valor decimal entre 0 e 1, sendo 0.2 o mais utilizado;
4. **Iterations (epochs)** - número de iterações que o algoritmo irá executar.

Tabela 2: Hidden Layers - Valores especiais

Valor	Nº Perceptrões
a	(Nº atributos + Nº classes)/2
t	Nº atributos + Nº classes
o	Nº classes
i	Nº atributos

2.8.3 *Algoritmo Learning Vector Quantization*

O Learning Vector Quantization (LVQ) é um algoritmo da classe dos algoritmos neurológicos, descrito por Kohonen no livro *Self-Organizing Maps* [5]. Este algoritmo é considerado uma versão supervisionada do algoritmo *Self-Organizing Map*, do mesmo autor.

O funcionamento do algoritmo baseia-se numa colecção de elementos denominados *codebook vectors*, que formam uma rede neuronal.

De acordo com o site *Machine Learning Mastery* [41], da autoria de Brownlee, um *codebook vector* é uma lista de valores que têem os mesmos atributos de entrada e de saída dos dados de treino. A representação do modelo é um conjunto, de dimensão fixa, de *codebook vectors*, que resultam do processo de treino.

Em termos de redes neuronais, um *codebook vector* será o equivalente ao neurónio, cada atributo do *codebook vector* corresponde a uma ponderação e o conjunto de *codebook vectors* será a rede neuronal.

De acordo com Brownlee [42], a fase de treino deste algoritmo é significativamente mais rápida do que outros métodos neuronais, como sejam os mecanismos com *back-propagation*. Outras vantagens incluem o facto de não necessitar que os dados de entrada estejam normalizados (embora a normalização possa produzir um incremento de performance), de conseguir lidar com valores em falta nos dados e de poder ser uma solução para qualquer problema de classificação desde que os atributos possam ser comparados através de uma função de cálculo de distância.

Por outro lado, e ainda de acordo com o site *WEKA Classification Algorithms* [42], o modelo apresenta algumas desvantagens, das quais se destacam:

- O modelo, para produzir resultados fiáveis, necessita de ser capaz de calcular distâncias entre todos os atributos (para os atributos numéricos é comum utilizar distâncias Euclidianas);

- a precisão e fiabilidade do modelo tanto dependem em grande escala da sua inicialização como dos valores utilizados para os parâmetros de aprendizagem (ver secção 2.8.3.2);
- a precisão também depende da distribuição das classes no *dataset* de treino; uma boa distribuição das amostras produz melhores modelos
- é difícil prever, para um determinado problema, o número de *codebook vectors*.

2.8.3.1 *Funcionamento do algoritmo*

O algoritmo constrói a colecção de *codebook vectors* a partir dos dados de treino. A explicação que se segue tem como base o artigo "Learning Vector Quantization for Machine Learning"[6], da autoria de Jason Brownlee, disponível no site *Machine Learning Mastery* [41].

O primeiro passo é definir o número de *codebook vectors*. Este número, normalmente, é obtido empiricamente, através da realização de várias configurações de teste.

Definido o número de *codebook vectors*, é criada a colecção de *codebook vectors* inicial. Os valores dos atributos de cada um destes *codebook vectors* inicial podem ser gerados aleatoriamente (garantindo que os valores de cada atributo se encontram dentro da gama de valores de cada atributo no *dataset*) ou corresponderem a instâncias seleccionadas aleatoriamente a partir do *dataset* de treino. Conforme mencionado acima, cada *codebook vector* possui o mesmo número de atributos dos dados de entrada. Possui, também, uma variável correspondente à classe de saída.

De seguida, o algoritmo processa cada instância do *dataset* de treino. Para cada instância, é seleccionando, de entre os *codebook vectors* da colecção, o que mais se assemelha a essa instância. Se o valor da variável de saída do codebook vector for igual à da instância de treino, o *codebook vector* é movido para mais perto da instância de treino; se não fôr, o *codebook* é afastado. A distância de aproximação ou afastamento é controlada por um parâmetro denominado **taxa de aprendizagem** (*learning rate*, na terminologia inglesa).

Para melhor ilustrar o funcionamento, vamos considerar um exemplo. Suponhamos que temos um codebook vector que possui entre as suas variáveis a variável x. Se o valor da classe da instância em análise for igual ao valor da classe do *codebook*

6 https://machinelearningmastery.com/learning-vector-quantization-for-machine-learning/

vector, então a variável x do *codebook vector* pode ser movida para mais perto da instância de treino correspondente t através da fórmula seguinte:

$$x = x + learning\_rate \times (t - x)$$

De igual forma, o afastamento da variável x pode ser efectuado através da fórmula

$$x = x - learning\_rate \times (t - x)$$

Este cálculo é repetido para cada atributo do codebook vector, relativamente ao atributo correspondente na instância de treino.

Esta operação é repetida, depois, para cada instância do *dataset* de treino. A iteração por todo o *dataset* de treino é denominada de época (*epoch*, na designação inglesa). O número de épocas - e, portanto, de iterações de todo o *dataset* - é parametrizado pelo utilizador no início do algoritmo. Todo o processo de aprendizagem termina quando fôr atingido o número total de épocas.

De referir, ainda, que o valor da taxa de aprendizagem não se mantém constante ao longo das iterações. De facto, o valor da taxa de aprendizagem sofre um decaimento, sendo que o valor para cada iteração é obtido pela fórmula:

$$learning\_rate = \alpha \times (1 - (epoch / max\_epoch))$$

em que *learning_rate* é a taxa de aprendizagem para a iteração *epoch*, α é a taxa de aprendizagem inicial e *max_epoch* é o número total de iterações.

2.8.3.2 *Parâmetros*

A implementação deste algoritmo no WEKA admite os seguintes parâmetros:

1. **Epsilon** - modificador da ponderação de aprendizagem, utilizado quando ambas as unidades que mais se assemelham são da classe da instância em análise; os valores recomendados são 0.1 ou 0.5;
2. **Modo de Inicialização** - Initialization Mode, na literatura inglesa; é o modo de inicialização dos codebook vectors; define como são criados os *codebook vectors* iniciais. Há seis opções possíveis, que englobam a geração aleatória ou a selecção aleatória de instâncias dos dados de treino;

3. **Função de aprendizagem** - Função a utilizar para a taxa de aprendizagem durante a fase de treino. Há três opções disponíveis, sendo que, tipicamente, a opção que possibilita melhores resultados é a função de decaimento linear;
4. **Taxa de aprendizagem** - *Learning Rate*, na terminologia inglesa. É o valor inicial da taxa de aprendizagem, tipicamente 0.3 ou 0.5;
5. **Número total de *codebook vectors*** - corresponde ao número total de *codebook vectors*, ou seja, a dimensão do conjunto de *codebook vectors*;
6. **Número de iterações** - É o número de épocas, ou seja, é o número de vezes que o *dataset* de treino é processado;
7. **Com Voto** - *Use Voting*, na designação inglesa, é a opção que permite o recurso a uma votação dinâmica para determinar a classe de cada codebook vector; permite o tratamento automático de instâncias classificadas erradamente;
8. ***Window Size*** - intervalo de valores em que os codebook vectors que mais se assemelham à instância têem que pertencer; os valores típicos são 0.2 ou 0.3.

3

REVISÃO DE LITERATURA

Neste capítulo será efectuada uma revisão de literatura sobre os diferentes assuntos abordados. A pesquisa será efectuada com recurso a ferramentas disponíveis da Internet, como, por exemplo, o Google Scholar[1].

3.1 APRENDIZAGEM AUTOMÁTICA

No que à aprendizagem automática diz respeito, a pesquisa nos motores de busca devolve inúmeros resultados, desde vídeos online até livros e artigos.

Os livros, para além da explicitação dos conceitos associados à aprendizagem automática, focam-se na aplicação da mesma a várias linguagens de programação - Python é uma das mais populares - ou a uma determinada *framework*, como sejam a sci-kit learn [2], a Tensor Flow [3] ou até o Apache Spark [4].

Pela sua generalidade e neutralidade no que a linguagens de programação ou *frameworks* diz respeito, gostaria de destacar o livro de Domingos [43].

Neste livro, o autor descreve as várias escolas de aprendizagem automática, quais as suas bases de pensamento e analisa os pontos fortes e fracos de cada uma delas.

O objectivo final é a definição de um método que, combinando os pontos fortes de cada escola, permita chegar ao que Domingos denomina de Algoritmo Mestre, um algoritmo que, partindo da análise dos dados, seja capaz de criar algoritmos que permitam extrair conhecimento a partir desses mesmos dados.

3.2 DATASETS

No que diz respeito aos *datasets* existentes para teste de sistemas de detecção de intrusões, Creech e Hu [44] fazem uma análise ao popular *dataset* KDD Cup 99,

1 https://scholar.google.pt/
2 https://scikit-learn.org/stable/
3 https://www.tensorflow.org/
4 https://spark.apache.org/

apresentando algumas razões pelas quais o referido *dataset* deve ser preterido em favor de outros mais recentes. Aliás, esta perspectiva já tinha sido abordada por McHugh [18], um artigo apresentado no ano 2000 (apenas um ano depois de surgir o *dataset*).

Num artigo mais recente, Al-Tobi e Duncan [17] abordam, precisamente esta problemática, analisando, uma vez mais, os problemas resultantes dos métodos utilizados na geração do *dataset* KDD Cup 99.

Relativamente ao *dataset* utilizado neste trabalho, o CSE-CIC-IDS2018, há, também, vários artigos reportando a utilização deste *dataset* para teste de modelos de IDS. Os modelos baseiam-se, por exemplo, em redes neuronais (Kim et al. [45]) ou em técnicas de aprendizagem profunda (*deep learning*, na designação inglesa), como se pode verificar no trabalho de Ferrag et al. [46].

No artigo [47], os autores fazem uma avaliação de alguns classificadores de aprendizagem automática, tendo como base a utilização do dataset CIC-AWS-2018, muito semelhante ao utilizado neste trabalho.

3.3 CLONALG

O CLONALG, apesar de ser um algoritmo com alguns anos, ainda é bastante utilizado pela comunidade científica.

De facto, desde actualizações ao algoritmo em si até novas aplicações, a pesquisa na Internet devolveu bastantes artigos, alguns deles bastante recentes.

No que toca a actualizações ou melhoramentos ao algoritmo podem referir-se dois trabalhos: o relatório técnico de Brownlee [28] e o artigo de Das et al. [48].

Ambos os artigos propõem alterações ao algoritmo base, tendo em vista um melhor desempenho do mesmo. Brownlee propõe o algoritmo CSCA (Clonal Selection Classification Algorithm), ao passo que Das et al. propõe o algoritmo E-CLONALG (Enhanced CLONALG).

Quanto a novas aplicações do algoritmo, de entre os vários artigos pesquisados pode destacar-se, pela sua aplicação à área da cibersegurança, o artigo de Elshafie et al. [49]. Este artigo aborda a forma de melhorar o desempenho de um IDS baseado em assinaturas - no caso, o Snort - utilizando a teoria da selecção clonal.

3.4 MULTI-LAYER PERCEPTRON

Os algoritmos baseados no sistema neurológico em geral e o algoritmo Multi-Layer Perceptron em particular são bastante utilizados na comunidade académica. De facto, uma pesquisa efectuada no Google Scholar [5] devolveu bastantes artigos produzidos em 2019 e 2020. Os artigos descrevem não só a aplicação deste algoritmo para resolução de problemas nas mais diversas áreas mas também algumas optimizações e melhoramentos realizados.

No artigo «An efficient hybrid multilayer perceptron neural network with grasshopper optimization» [50], os autores propõem um novo algoritmo de treino estocástico híbrido usando o recentemente proposto algoritmo Grasshopper Optimization Algorithm (GOA) para redes neuronais baseadas em MLP. O algoritmo GOA é uma técnica emergente com alto potencial para resolver problemas de optimização com base nos seus mecanismos de busca flexíveis e adaptáveis. O resultado é um novo modelo denominado GOAMLP.

Também o artigo «Forecasting gold price fluctuations using improved multilayer perceptron neural network and whale optimization algorithm» [51], da autoria de Alameer et al., refere a aplicação do algoritmo Whale Optimization Algorithm a redes neuronais baseadas em MLP para criar um modelo que permita prever as variações do preço do ouro. Os resultados desta combinação são, *a posteriori*, comparados com os resultados obtidos através de outros algoritmos aplicados a redes neuronais, como sejam as redes neuronais clássicas, algoritmos genéticos para redes neuronais ou *grey wolf optimization* para redes neuronais.

3.5 LEARNING VECTOR QUANTIZATION

Também este algoritmo, desenvolvido na década de 1990 por Kohonen, continua a ter bastante aceitação junto da comunidade científica.

Da pesquisa efectuada através do Google Scholar [5] podemos verificar a aplicação deste algoritmo à resolução de múltiplos tipos de problemas. São, também, apresentados artigos com propostas de optimização do algoritmo ou de combinação deste com métodos de optimização.

Um exemplo é o artigo intitulado «Comparative Analysis of Backpropagation With Learning Vector Quantization (LVQ) to Predict Rainfall in Medan City» [52],

5 https://scholar.google.pt/

da autoria de Mahrina et al. Este artigo faz um estudo comparativo da aplicação de dois dos algoritmos utilizados nesta dissertação - LVQ e MLP - à geração de modelos destinados a prever a queda de chuva numa determinada cidade.

O LVQ também pode ser aplicado ao reconhecimento facial. O artigo «The Student Attendance Controlling Based on Face Recognition by using Learning Vectorization Quantization (LVQ) Algorithm» [53], descreve a aplicação do LVQ ao controlo de absentismo de estudantes, recorrendo, para tal, à técnologia de reconhecimento facial. O objectivo do sistema descrito é permitir o registo, de forma automática, da presença dos alunos na sala de aula.

No campo da optimização, Arifando et al., no artigo «Hybrid Genetic Algorithm Learning Vector Quantization for Classification of Social Assistance Recipients» [54], descreve a aplicação de algoritmos genéticos para optimizar o vector de ponderações do LVQ, de forma a melhorar o resultado final do algoritmo.

Ainda relativamente à optimização do algoritmo, Semadi e Pulungan [55] descreve a aplicação do método da redução de dados utilizando proximidade geométrica de dados como forma de optimizar o LVQ. O objectivo é eliminar conjuntos de dados que apresentam similaridades, mantendo apenas uam representação para cada conjunto. Este método, com a parametrização correcta, permite reduzir a quantidade de dados envolvida no processo de aprendizagem mantendo a precisão existente.

4

ARQUITETURA

O objectivo deste capítulo é descrever a arquitectura, quer ao nível do *hardware* quer ao nível do *software*, utilizada e implementada para realizar os testes. Será exposto o ambiente de testes, com especial enfoque nas características da máquina utilizada e, de seguida, serão descritas as aplicações utilizadas.

Por último, será feita uma caracterização do dataset utilizado.

A figura 11 ilustra, de forma esquemática, a arquitectura implementada.

Figura 11: Ambiente de testes

4.1 HARDWARE

Ao nível do hardware, foi utilizado um computador portátil de marca ASUS, com processador Intel Core i7 de sétima geração. A memória RAM instalada é de 16GB.

Em termos de placa gráfica, o computador possui instalada uma placa NVidia GeForce GTX, que permite a instalação das bibliotecas de processamento gráfico CUDA. Estas bibliotecas permitem transferir algumas tarefas de processamento do CPU para a placa gráfica, permitindo, assim, um melhor desempenho global graças à memória dedicada e capacidades do processador gráfico.

4.2 APLICAÇÕES UTILIZADAS

Na realização deste trabalho foram utilizadas várias aplicações, quer para realizar, em ambiente gráfico, a definição dos fluxos de tratamento do *dataset*, quer para a operacionalização de alguns passos no tratamento prévio dos dados. Para além disso, foi implementada, com recurso à API Java do WEKA, uma ferramenta de automatização dos testes. Essas aplicações são descritas nas secções que se seguem.

4.2.1 *WEKA*

A aplicação WEKA[1] - de Waikato Environment for Knowledge Analysis - é uma aplicação desenvolvida em Java na Universidade de Waikato, na Nova Zelândia. É uma aplicação em ambiente gráfico (ver Figura 12), que disponibliza um conjunto de ferramentas para tratamento massivo de dados (*data mining*, na terminologia inglesa) e aprendizagem automática. A aplicação encontra-se descrita no artigo de Frank et al. [2], que é um apêndice, disponível *online*, do livro "Data Mining: Practical Machine Learning Tools and Techniques", dos mesmos autores.

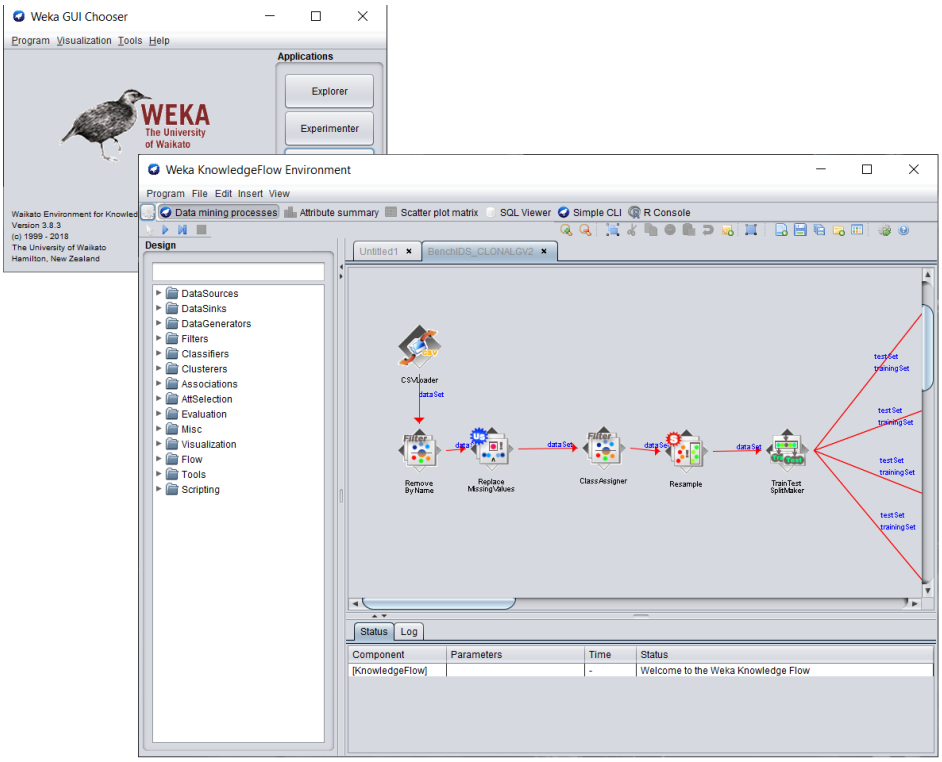

Figura 12: Ambiente gráfico do WEKA

1 https://www.cs.waikato.ac.nz/ml/weka/

A aplicação incorpora vários módulos, que permitem, por exemplo, descrever um fluxo de conhecimento em que vários componentes são aplicados aos dados consecutivamente, até à produção do resultado final.

Os dados de entrada e saída podem assumir diversos formatos, desde o típico ficheiro Comma-separated values (CSV) até formatos mais elaborados como ficheiros de Matlab ou escrita directa em base de dados.

4.2.2 *Orange*

A aplicação Orange é uma aplicação de *data-mining* desenvolvida pelo laboratório de bio-informática da Universidade de Ljubljana, na Eslovénia, em colaboração com a comunidade *open-source*. Foi apresentada no artigo de Demšar et al. intitulado «Orange: Data Mining Toolbox in Python» [56].

A aplicação, desenvolvida em Python, possui um ambiente gráfico bastante intuitivo, sendo possível adicionar mais funcionalidades ou algoritmos através de um sistema de *plugins/add-ons*.

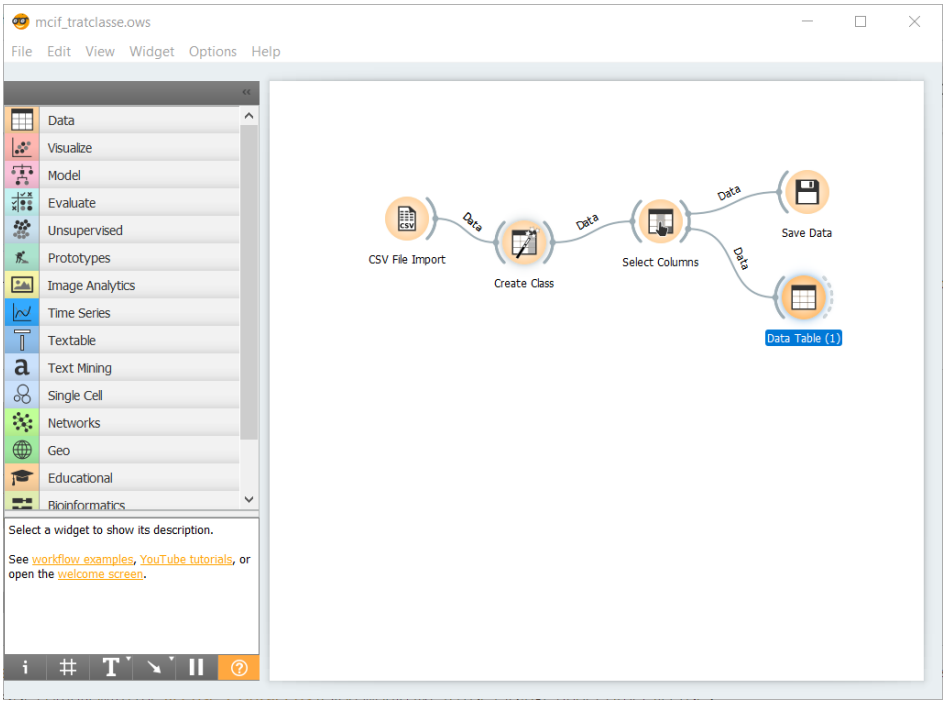

Figura 13: Ambiente gráfico do Orange

A aplicação pode ser descarregada a partir do Uniform Resource Locator (URL) `https://orange.biolab.si/`.

4.3 A APLICAÇÃO BENCHMARKIDS

A aplicação BenchmarkIDS surgiu na sequência de suprir algumas necessidades relacionadas com este trabalho, nomeadamente a flexibilidade proporcionada pela Application Programming Interface (API) e tentar ultrapassar algumas limitações do ambiente gráfico. A aplicação, desenvolvida em Java, não possui ambiente gráfico e os resultados são apenas mostrados no écran (não são guardados em ficheiro).

4.3.1 *Estrutura*

A aplicação é constituída por 4 módulos: o módulo de inicialização (*setup*), o módulo de treino, o módulo de testes e o módulo de cenários.

O módulo de inicialização é responsável pela criação dos ficheiros de dados que irão ser utilizados pelos módulos de treino e testes.

Partindo dos parâmetros passados na linha de comando, abre os ficheiros correspondentes do dataset, faz o pré-processamento (ver Secção 5.2) e gera os ficheiros para treino do modelo (retira, aleatoriamente, 200000 linhas do ficheiro e divide-as em dois ficheiros, um com 140000 linhas (70%) para treino e outro com as restantes 60000 (30%), utilizado para testar o modelo) e para teste de simulação de ataque zero-day (ver Figura 14)

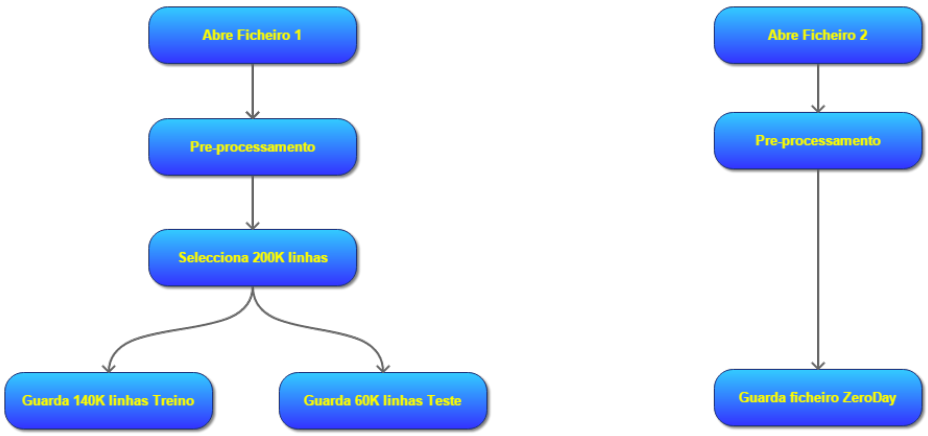

Figura 14: BenchmarkIDS - Setup

O módulo de treino lê o ficheiro de dados previamente gerado e treina o modelo, utilizando os algoritmos pré-definidos. A parametrização dos algoritmos está, também, pré-definida (ver Secção 5.3), sendo gerados 3 modelos de cada algoritmo.

O módulo de teste carrega os modelos previamente gerados, lê o ficheiro de testes e o ficheiro de simulação de ataque Zero-Day e testa e avalia os modelos, gerando um conjunto de métricas indicativas da performance de cada modelo.

O módulo de cenários, que implementa os 4 cenários de teste deste trabalho (ver Secção 5.4), carrega os ficheiros previamente gerados a partir do dataset CSE-CIC-IDS2018 pela aplicação Orange, selecciona de um deles 140000 linhas para a aprendizagem e 60000 linhas do outro que se destina à fase de teste e aplica esses dados aos 3 algoritmos utilizados. Por motivos de significância estatística, este procedimento é repetido dez vezes, sendo que os dados seleccionados em cada uma das vezes são completamente independentes dos seleccionados nas restantes.

4.3.2 *Utilização*

Para utilizar esta aplicação é necessário abrir a linha de comandos do Windows (ou uma *shell* em Linux) e mudar para a pasta (directório) onde está localizada a distribuição da aplicação.

Para executar o módulo de inicialização e gerar, assim, os ficheiros de dados necessários, tem que se executar um comando com a seguinte estrutura:

```
java␣-jar␣BenchmarkIDS.jar␣setup␣<fich1>␣<fich2>
```

Os parâmetros significam o seguinte:

- <fich1> - número do ficheiro que irá servir de base ao treino do modelo
- <fich2> - número do ficheiro que irá servir de base à simulação de ataque zero-day

Assim, por exemplo, o comando

```
java -jar BenchmarkIDS.jar setup 02 03
```

irá executar o módulo de inicialização, acedendo ao ficheiro Dados_02.csv para gerar os dados treino do modelo e ao ficheiro Dados_03.csv para criar o ficheiro que irá simular o ataque zero-day.

No módulo de treino, a estrutura dos comandos é a seguinte:

```
java␣-jar␣BenchmarkIDS.jar␣training␣<algoritmo>
```

O parâmetro `algoritmo` apenas pode assumir 3 valores distintos:

- clonalg - para seleccionar o algoritmo CLONALG

- mlp - para seleccionar o algoritmo Multi-Layer Perceptron
- lvq - para seleccionar o algoritmo Learning Vector Quantization

Se, por exemplo, pretendêssemos treinar os modelos CLONALG, o comando a executar seria, então:

```
java -jar BenchmarkIDS.jar training clonalg
```

O módulo de testes engloba duas opções distintas, uma para testar um determinado algoritmo e outra para gerar e testar os *ensembles* dos 3 algoritmos, agrupados 2 a 2.

A estrutura do comando para testar um determinado algoritmo, incluindo o teste de simulação de ataque zero-day, é a seguinte:

```
java␣-jar␣BenchmarkIDS.jar␣test␣<algoritmo>
```

Os valores admissíveis para o parâmetro `algoritmo` são iguais aos do parâmetro com o mesmo nome no módulo de treino.

Para gerar os *ensembles* e realizar os respectivos testes, basta executar o comando:

```
java␣-jar␣BenchmarkIDS.jar␣ensemble
```

Para utilizar o módulo de cenários, basta executar o comando:

```
java␣-jar␣BenchmarkIDS.jar␣training␣<cenario>
```

O parâmetro `cenario` pode assumir os valores `cenario1`, `cenario2`, `cenario3` e `cenario4`.

4.4 DATASET CSE-CIC-IDS2018

Para a realização dos testes, e de entre os datasets disponíveis, foi seleccionado o CSE-CIC-IDS2018[2]. As razões desta escolha prendem-se, sobretudo com:

- É um dataset bastante recente e que está agora a ser utilizado pela comunidade científica para fazer benchmark de IDS;
- Inclui um conjunto bastante alargado de ataques e de características;
- Tráfego gerado dinamicamente, simulando uma rede empresarial.

2 https://www.unb.ca/cic/datasets/ids-2018.html

De acordo com [57], o dataset inclui sete cenários diferentes de ataques, sendo que os ataques são executados por um conjunto de 50 máquinas que têem como alvo uma organização dividida em 5 departamentos, com um total de 420 máquinas de utilizadores e 30 servidores.

Os cenários de ataques contemplam ataques de *brute-force*, Heartbleed, Botnet, Denial of Service (DoS), DDoS, ataques web e infiltração a partir do interior da organização.

4.4.1 *Tipos e duração dos ataques*

Na **infiltração a partir do interior**, foi enviado um email com conteúdo malicioso, destinado a explorar uma vulnerabilidade de uma aplicação, por exemplo, o Adobe Reader. O *software* malicioso, depois de instalado, abre uma *backdoor* no computador do utilizador, permitindo aos atacantes pesquisar, detectar e explorar vulnerabilidades noutras máquinas da organização.

Os **ataques de DoS e DDoS** visaram, essencialmente, o serviço web, sendo que o objectivo a atingir seria a total inacessibilidade do serviço web.

Para realizar estes ataques os atacantes recorreram a diversas ferramentas, das quais se destacam o Slowloris [58], o LOIC (Low Orbit Ion Cannon) [59] e o HOIC[3] (High Orbit Ion Cannon).

Os **ataques web** tiveram como alvo a aplicação Damn Vulnerable Web App[4] (DVWA). Esta aplicação apresenta várias vulnerabilidades e é desenvolvida especificamente para "ser atacada", tendo como objectivo servir de ajuda e alerta para os profissionais que, assim, podem testar as suas capacidades neste domínio.

Em termos de metodologia, o ataque foi iniciado por um *scanning* de vulnerabilidades utilizando uma ferramenta de pesquisa de vulnerabilidades em aplicações web, sendo, *a posteriori* e tendo como base as vulnerabilidades detectadas, efectuados vários tipos de ataques, tais como SQL Injection, injecção de comandos e *upload* de ficheiros.

Foram, também, realizados **ataques de força bruta**, com o objectivo de obter credenciais de acesso quer ao MySQL quer ao serviço SSH. A ferramenta utilizada foi o Patator, ferramenta essa que, para a realização das tentativas de acesso,

3 https://en.wikipedia.org/wiki/High_Orbit_Ion_Cannon
4 http://www.dvwa.co.uk/

recorre a um dicionário ou lista de palavras que são utilizadas como tentativa de palavra-passe.

No que toca a tipos de ataques mais recentes, foi utilizada a ferramenta Heartleech para realizar **ataques heartbleed**. A ferramenta pesquisa os sistemas vulneráveis à falha heartbleed [60] e, nesses sistemas, pode ser utilizada para fazer exfiltração de dados.

O *dataset* incorpora, também, dados resultantes de um **ataque de Botnet**. Para tal, e de acordo com [57], foi utilizada a ferramenta Zeus, que é um *malware trojan* para sistemas Windows. Esta ferramenta pode ser utilizada para várias actividades maliciosas, tais como o roubo de informação bancária via *key-logging* no browser ou a instalação do *ransomware* Crypto-Locker. A disseminação deste *malware* é realizada principalmente com recurso a *phishing*.

Em complemento a este ataque foi, também, utilizada a botnet open-source Ares, que permite, de entre outras capacidades, a execução remota de comandos, *download* e *upload* de ficheiros e funcionalidades de *key-logging*.

4.4.2 *Calendarização dos ataques*

De acordo com [57], a calendarização dos ataques e as ferramentas utilizadas encontram-se explicitados na tabela 3.

4.4.3 *Os dados*

O *dataset* está disponível para download a partir do AWS, bastando, para tal e de acordo com [57], instalar a ferramenta AWS CLI.

O *dataset* disponibiliza quer os ficheiros .pcap - que incluem todo o tráfego capturado - quer ficheiros pré-processados, em formato CSV, para análise e tratamento em software de ML.

Neste último caso, o pré-processamento foi levado a cabo através da ferramenta CICFlowMeter-V3[5].

A aplicação, que foi desenvolvida pelo CIC, analisa os ficheiros .pcap e agrupa os pacotes em fluxos de dados, sumarizando, para cada fluxo, um conjunto de oitenta

5 https://www.unb.ca/cic/research/applications.html#CICFlowMeter

Tabela 3: Calendarização dos ataques

Dia	Hora		Tipo de Ataque	Ferramenta(s)	Ficheiro
	Inicio	Fim			
14/02/2018	10:32	12:09	Brute Force - FTP	Patator	Dados_01
	14:01	15:31	Brute Force - SSH	Patator	
15/02/2018	09:26	10:09	DoS	GoldenEye	Dados_02
	10:59	11:40	DoS	Slowloris	
16/02/2018	10:12	11:08	DoS	SlowHTTPTest	Dados_03
	13:45	14:19	DoS	Hulk	
20/02/2018	10:12	11:17	DDoS	LOIC-HTTP	Dados_04
	13:13	13:32	DDoS	LOIC-UDP	
21/02/2018	10:09	10:43	DDoS	LOIC-UDP	Dados_05
	14:05	15:05	DDoS	HOIC	
22/02/2018	10:17	11:24	Brute Force - Web	DVWA Código Próprio	Dados_06
	13:50	14:29	Brrute Force - XSS		
	16:15	16:29	SQL Injection		
23/02/2018	10:03	11:03	Brute Force - Web	DVWA Código Próprio	Dados_07
	13:00	14:10	Brute Force - XSS		
	15:05	15:18	SQL Injection		
28/02/2018	10:50	12:05	Infiltration	Email phishing Metasploit Nmap	Dados_08
	13:42	14:40	Infiltration		
01/03/2018	09:57	10:55	Infiltration		Dados_09
	14:00	15:37	Infiltration		
02/03/2018	10:11	11:34	BotNet	Zeus Ares Botnet	Dados_10
	14:24	15:55	BotNet		

atributos. A definição de cada um desses atributos pode ser consultada no Apêndice A.

Cada ficheiro CSV gerado corresponde a um dia de tráfego. Dado que o nome original de cada ficheiro era extenso e incluía espaços, e tendo em vista facilitar o tratamento dos mesmos, procedeu-se a uma alteração dos respectivos nomes, sendo que os nomes definitivos são os que constam na tabela 3.

Cada fluxo é, ainda, classificado, permitindo identificar se corresponde a tráfego normal ou não, caso em que o fluxo é classificado de acordo com o tipo de ataque em questão.

Por exemplo, o *dataset* correspondente ao dia 14/02/2018 tem as classes que constam na Tabela 4. Os dados para os restantes dias podem ser consultados na secção A.2 do Apêndice A.

Tabela 4: Classificação dos fluxos do dia 14/02/2018

Classe	**Nº Instâncias**	**%**
Benign	667626	63,67%
FTP-BruteForce	193360	18,44%
SSH-BruteForce	187589	17,89%
Total de Instâncias	1048575	100,00%

5

TESTES

Neste capítulo serão descritos os testes realizados, bem como a metodologia adoptada para a realização dos mesmos.

Primeiramente, e dado que são comuns a todo o processo, serão especificadas as métricas utilizadas para aferir a performance de cada modelo e descrita a fase pré-processamento.

De seguida, será descrito o processo de selecção da parametrização de cada algoritmo utilizado, com recurso à aplicação BenchmarkIDS.

Com base na configuração determinada no passo anterior, será efectuada, com recurso ao ambiente gráfico do WEKA, uma bateria de testes, tentando determinar qual o algoritmo com melhor performance face ao tráfego de rede apresentado.

A figura 15 ilustra, de forma simplificada, o processo de testes.

Figura 15: Processo de testes

5.1 MÉTRICAS

Para aferir o desempenho de cada um dos vários algoritmos, bem como efectuar comparações entre os vários algoritmos, iremos recorrer ao cálculo de algumas métricas, tendo como objectivo a elaboração de uma tabela final comparativa.

Como referido acima, um dos principais problemas dos IDS baseados em comportamento prende-se com a identificação de falsos positivos e/ou falsos negativos. Tendo isso em conta, é lógico que os primeiros indicadores a considerar sejam as taxas de falsos positivos - False Positive Rate (FPR), na língua inglesa) e falsos negativos - False Negative Rate (FNR). Também serão de considerar as taxas de verdadeiros positivos - True Positive Rate (TPR) - e de verdadeiros negativos - True Negative Rate (TNR).

Nas fórmulas que se seguem, FN designa o número de falsos negativos (*false negatives*, na terminologia inglesa), TN o número de verdadeiros negativos (*true negatives*), FP o número de falsos positivos (*false positives*) e TP o número de verdadeiros positivos (*true positives*)

De acordo com o livro *Encyclopedia of Machine Learning and Data Mining*, as diferentes taxas podem calcular-se do seguinte modo:

$$FPR = \frac{FP}{FP + TN} \qquad TPR = \frac{TP}{TP + FN}$$

$$TNR = \frac{TN}{TN + FP} \qquad FNR = \frac{FN}{FN + TP}$$

De acordo com o artigo [20], podem ainda ser consideradas as seguintes métricas:

- Precisão (Pr) - *Precision*, na literatura inglesa, é definida como sendo a razão entre o número de verdadeiros positivos e o total de positivos classificados (soma do número de verdadeiros positivos com o número de falsos positivos);
- Sensibilidade (Rc) - designada como *Sensitivity* ou *Recall* na literatura inglesa, é definida como sendo a razão entre o número de verdadeiros positivos e o total de positivos gerados (soma do número de verdadeiros positivos com o número de falsos negativos);
- Exactidão (Acc) - designada como *Accuracy* na terminologia anglo-saxónica, de acordo com o livro *Encyclopedia of Machine Learning and Data Mining*, define-se como sendo a razão entre o número de objectos correctamente classificados e o número total de objectos;
- Medida-F (F1) - *F-Measure* na terminologia inglesa, é uma combinação harmónica da Precisão e da Sensibilidade numa só unidade.

Matematicamente, temos:

$$Pr = \frac{TP}{TP + FP} \qquad Rc = \frac{TP}{TP + FN} \qquad Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F1 = \frac{2}{\frac{1}{Pr} + \frac{1}{Rc}} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

5.2 PRÉ-PROCESSAMENTO

O objectivo desta fase é preparar os dados do *dataset* para serem consumidos pelos classificadores. Essas operações incluem as seguintes tarefas:

- Redução do número de classes
- Eliminação de atributos desnecessários
- Tratamento de valores em falta (*missing values*)
- Normalização dos dados

5.2.1 *Redução do número de classes*

Como já foi mencionado, cada dia do *dataset* compreende várias classes, permitindo identificar o tipo de ataque em questão. Como o que se pretende, neste trabalho, é avaliar a capacidade de identificar tráfego anómalo independentemente do tipo de ataque, procedeu-se a uma redução do número de classes do *dataset*. Desta forma, todos os fluxos do *dataset* classificados como Benign são considerados tráfego normal. Todos os restantes são considerados como tráfego Malicioso.

Assim, por exemplo, e considerando como exemplo o dia 14/02/2018 (ver Tabela 4), o dataset final terá as seguintes classes:

Tabela 5: Redução de classes - dia 14/02/2018

Classe	**Nº Instâncias**	**%**
Normal	667626	63,67%
Malicioso	380949	36,33%
Total de Instâncias	1048575	100,00%

Esta operação foi realizada através do recurso à aplicação Orange[1], pois o WEKA não dispõe de ferramentas adequadas e de fácil uso para realização da tarefa em questão - nomeadamente de um componente para gerar um atributo com base no

1 https://orange.biolab.si/

valor de outro atributo. O diagrama do fluxo correspondente na aplicação Orange bem como a parametrização do componente que vai criar o novo atributo constam na Figura 16

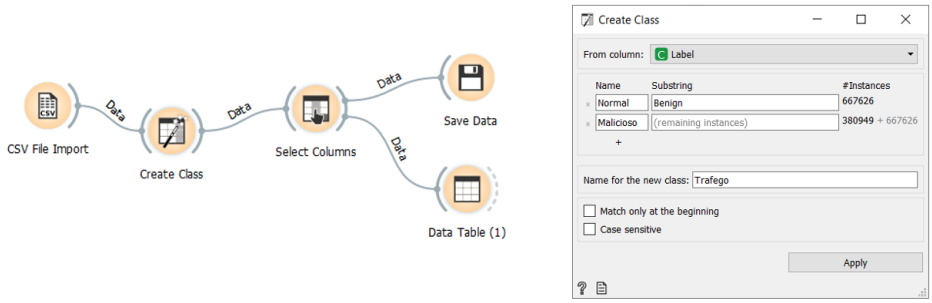

Figura 16: Redução de classes

Na ferramenta de automatização que está a ser desenvolvida em Java, este tratamento é efectuado com recurso à API do WEKA, conforme se pode verificar na Listagem 1.

Listagem 1: Redução das classes em Java/WEKA

```
// Cria o atributo
dados.insertAttributeAt(new Attribute("Trafego",valores), dados.numAttributes());

// Indice do atributo Trafego
indiceTrafego=dados.attribute("Trafego").index();

// Preenche a coluna do novo atributo com base no valor de Label
for (int i=0; i<numLinhas;i++){
        // System.out.print("NumLinha: " + i + " => ");
        // System.out.print(dados.instance(i).toString(indiceClasse));
        if (dados.instance(i).toString(indiceClasse).equalsIgnoreCase("Benign")){
                dados.instance(i).setValue(indiceTrafego, "Normal");
                // System.out.println(" => Normal");
        } else {
                dados.instance(i).setValue(indiceTrafego, "Malicioso");
                // System.out.println(" => Malicioso");
        }
}

// Altera o atributo de classificação
dados.setClassIndex(indiceTrafego);

// Remove atributo Label
dados.deleteAttributeAt(indiceClasse);
```

5.2.2 *Eliminação de atributos desnecessários*

Como já foi mencionado, o *dataset* tem oitenta atributos (cf. Secção A.1), sendo que, à partida, nem todos seriam necessários para que a classificação de cada fluxo

fosse efectuada de modo correcto pelo classificador. Poder-se-ia recorrer a vários métodos de redução de atributos ou de selecção dos atributos mais relevantes, tais como efectuar uma análise de *Principal Components Analysis* [62, 63] ou utilizar algoritmos genéticos para seleccionar os atributos mais relevantes, mas tal não é o objectivo deste trabalho e, sobretudo, desta sub-fase.

Um dos atributos de cada fluxo é o **Timestamp**, que regista a data e hora em que o fluxo ocorreu. Como o que se pretende é que o classificador classifique o fluxo em Normal ou Malicioso independentemente da data em que este ocorreu, o atributo Timestamp pode ser eliminado do dataset.

Outro motivo para remover o atributo Timestamp prende-se com o algoritmo Back-Propagation Multi-Layer Perceptron. De facto, atendendo a que o atributo está identificado como Nominal e dado que a implementação, no WEKA, do algoritmo Back-Propagation Multi-Layer Perceptron não consegue lidar com dois atributos do tipo Nominal (o outro é a classe, Tráfego), para que este algoritmo possa ser aplicado este atributo tem que ser eliminado.

No caso do diagrama de fluxo do WEKA, a eliminação foi efectuada recorrendo ao filtro **RemoveByName**. Na ferramenta de automatização, recorreu-se ao método **deleteAttributeAt** da API do WEKA.

5.2.3 *Tratamento de valores em falta*

Uma das operações essenciais quando se analisam dados é determinar se os dados estão completos e correctos, ou seja, se não há valores em falta (*missing values*, na terminologia inglesa) ou valores errados. A correcção destas anomalias torna-se importante na medida em que alguns algoritmos podem não estar preparados para as tratar. O WEKA, através do seu ambiente gráfico, permite ver se, num determinado atributo do dataset, há ou não valores em falta (Figura 17).

O WEKA dispõe de dois filtros destinados ao tratamento dos valores em falta. Um deles, o `ReplaceMissingWithUserConstant`, substitui os valores em falta por um valor constante fornecido pelo utilizador. O outro, `ReplaceMissingValues`, substitui os valores em falta pelos valores médios e modas dos atributos em questão. Estes filtros estão, também, disponíveis na API do WEKA. Para este trabalho recorreu-se ao filtro `ReplaceMissingValues`.

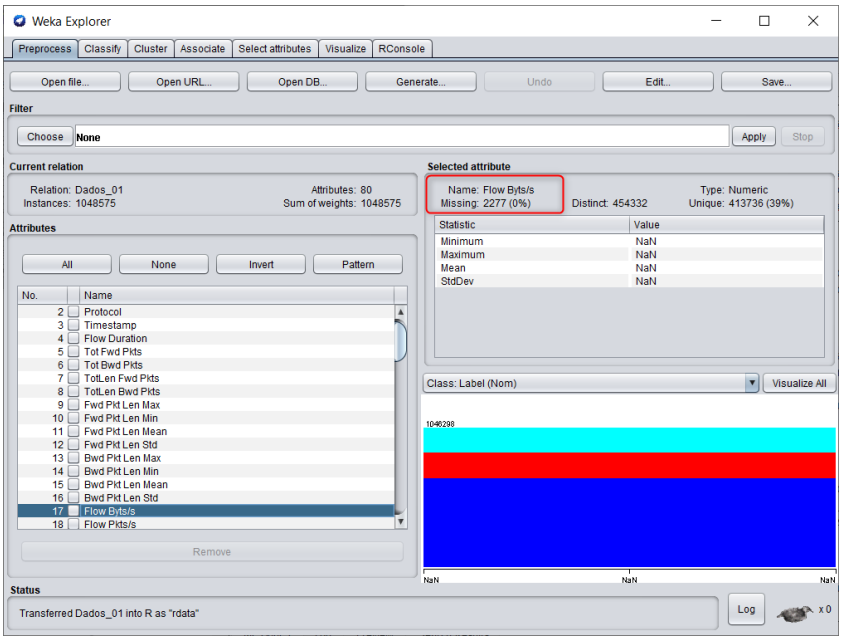

Figura 17: WEKA - valores em falta

5.2.4 *Normalização dos dados*

Algumas implementações de algoritmos necessitam, para o seu correcto funcionamento, que os valores dos atributos estejam compreendidos entre valores limite. É o caso, por exemplo, do algoritmo Back-Propagation Multi-Layer Perceptron, que exige que os valores dos atributos numéricos estejam compreendidos entre 0 e 1.

Para que isso aconteça torna-se necessário, antes de executar o algoritmo, efectuar uma normalização dos dados, operação essa que, de forma estatística, irá assegurar que os valores dos atributos estejam compreendidos entre os valores 0 e 1.

Tanto no ambiente gráfico do WEKA, no Knowledge Flow, como na API, a normalização dos atributos pode ser conseguida através da utilização do filtro **Normalize**.

5.3 PARAMETRIZAÇÃO INICIAL DOS ALGORITMOS

Para esta fase, recorreu-se à aplicação BenchmarkIDS, tendo como objectivo testar, para cada um dos algoritmos seleccionados, um conjunto de 3 configurações, aferindo, depois, a performance de cada uma delas. A configuração com melhor performance será, depois, utilizada na bateria de testes.

De referir que não faz parte do âmbito desta tese a definição e aplicação de metodologias específicas para a obtenção da configuração óptima de cada algoritmo. As configurações apresentadas resultam, apenas, da análise dos parâmetros requeridos pela implementação específica de cada algoritmo, tentando adaptá-los ao conjunto de atributos do *dataset* utilizado.

Também não faz parte do âmbito desta dissertação a definição e aplicação de métodos de selecção dos atributos mais relevantes, pelo que se optou por utilizar todos os atributos do *dataset* (exceptuando o atributo Timestamp - cf. secção 5.2.2).

5.3.1 *CLONALG*

Tendo em conta os parâmetros do algoritmo e a especificidade do *dataset* em análise, foram estabelecidas as três parametrizações seguintes (Tabela 6):

Tabela 6: Parâmetros dos modelos CLONALG

Parâmetro	**CLONALG1**	**CLONALG2**	**CLONALG3**
Antibody Pool Size	120	50	40
Clonal Factor	0.3	0.5	0.3
Number of Generations	40	40	20
Remainder Pool Ratio	0.2	0.2	0.4
Selection Pool Size	80	20	20
Total Replacement	10	10	15
Seed	1	1	1

Para o caso de simulação de ataque zero-day, os resultados obtidos para cada configuração são os que constam na tabela 7. Da análise da tabela, pode verificar-se que a configuração com melhor resultado é a CLONALG1

Tabela 7: Configurações CLONALG - Resultados

Configuração	**TPR**	**FPR**	**TNR**	**FNR**	**Precision**	**Accuracy**	**Recall**	**F1**
CLONALG1	0,0240	0,0001	0,9999	0,9760	0,9957	0,4398	0,0240	0,0468
CLONALG2	0,0235	0,0002	0,9998	0,9765	0,9946	0,4395	0,0235	0,0458
CLONALG3	0,0235	0,0001	0,9999	0,9765	0,9959	0,4395	0,0235	0,0458

5.3.2 *Multi-Layer Perceptron*

As configurações testadas para o algoritmo Multi-Layer Perceptron são as que constam na tabela 8:

Tabela 8: Parâmetros dos modelos Multi-Layer Perceptron

Parâmetro	**BackMLP1**	**BackMLP2**	**BackMLP3**
Hidden Layers	80,40,10	80,40,10	a
Learning Rate	0.1	0.1	0.1
Momentum	0.0	0.3	0.3
Iterations	500	500	500
Seed	1	1	1

Os resultados obtidos constam da Tabela 9. Da análise da tabela, podemos concluir que a configuração com melhor desempenho é a BackMLP3.

Tabela 9: Configurações MLP - Resultados

Configuração	**TPR**	**FPR**	**TNR**	**FNR**	**Precision**	**Accuracy**	**Recall**	**F1**
BackMLP1	0,9979	0,0002	0,9998	0,0021	0,9998	0,9987	0,9979	0,9988
BackMLP2	0,9974	0,0007	0,9993	0,0026	0,9994	0,9982	0,9974	0,9984
BackMLP3	1,0000	0,0002	0,9998	0,0000	0,9998	0,9999	1,0000	0,9999

5.3.3 *Learning Vector Quantization*

As três configurações testadas para o algoritmo Learning Vector Quantization encontram-se na Tabela 10.

Tabela 10: Parâmetros dos modelos Learning Vector Quantization

Parâmetro	**LVQ1**	**LVQ2**	**LVQ3**
Learning Weight Modifier (ε)	0.1	0.1	0.1
Initialization Mode	1	6	1
Learning Function	1	1	1
Learning Rate	0.3	0.5	0.3
Total Codebook Vectors	90	90	50
Iterations	4500	4500	2500
Use Voting	False	True	True
Window Size	0.2	0.2	0.2
Seed	1	1	1

Podemos verificar, analisando a tabela 11, que a configuração que assegura os melhores resultados é a LVQ1.

Tabela 11: Configurações LVQ - Resultados

Configuração	**TPR**	**FPR**	**TNR**	**FNR**	**Precision**	**Accuracy**	**Recall**	**F1**
LVQ1	0,7485	0,0002	0,9998	0,2515	0,9998	0,8556	0,7485	0,8561
LVQ2	0,7251	0,0000	1,0000	0,2749	1,0000	0,8422	0,7251	0,8406
LVQ3	0,0240	0,0002	0,9998	0,9760	0,9953	0,4398	0,0240	0,0468

5.3.4 *Geração dos ficheiros de dados*

Os ficheiros de dados correspondentes a cada cenário foram gerados com a ajuda da aplicação Orange. Tendo como base o ficheiro do dataset CSE-CIC-IDS2018 correspondente, definiu-se um fluxo de forma a gerar os 3 ficheiros distintos para cada dia. O fluxo é muito semelhante, pelo que vamos descrever apenas um deles. Os fluxos podem ser consultados no Apêndice B.

Na figura 18 podemos observar o diagrama de fluxo para o dia 21 de Março.

Após o carregamento do ficheiro de entrada, são efectuadas as operações de pré-processamento atrás identificadas - eliminação do campo Timestamp, normalização dos dados e tratamento dos valores em falta (*missing values*).

De seguida, e dependendo dos requisitos do ficheiro de saída, serão executadas as operações correspondentes aos componentes:

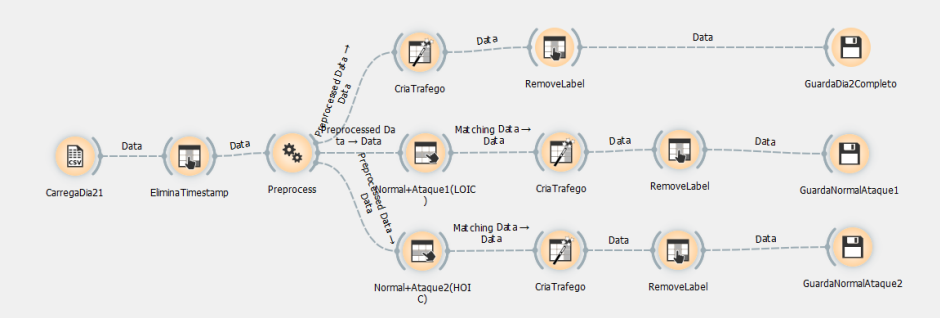

Figura 18: Orange - Diagrama de fluxo

- **CriaTráfego** - este componente é responsável pela geração de um novo campo denominado Tráfego; este campo, resultante da operação de redução de classes (ver secção 5.2.1), poderá assumir apenas dois valores, **Normal** ou **Malicioso**, tendo como base o valor do campo **Label**;
- **RemoveLabel** - este componente elimina o campo Label;
- **Normal+Ataque** - Estes componentes vão filtrar o ficheiro original, tendo como resultado todos os registos cujo campo **Label** tenha o valor **Benign** e um dos ataques existentes; no caso da figura 18, podemos ver que temos um componente que selecciona os registos em que o valor do campo Label seja Benign ou LOIC - Normal+Ataque1(LOIC) - e o outro selecciona os registos em que o campo Label seja Benign ou HOIC - Normal+Ataque2(HOIC);
- **Guarda<Ficheiro>** - componente que guarda em ficheiro os dados seleccionados.

5.4 TESTES REALIZADOS

Para os testes a realizar foram definidos 2 módulos de quatro cenários cada que constam na tabela 12.

Por motivos de limitações computacionais, os testes foram efectuados sobre um conjunto de 200000 registos, sendo que 70% (140000 registos) dos mesmos constituem os dados de treino e os restantes 30% (60000 registos) os dados de teste.

Para além disso, por motivos de significância estatística, cada cenário de testes foi executado dez vezes, sendo que os dados respeitantes a cada iteração são independentes das restantes, mas iguais para os três algoritmos em cada iteração. Os valores finais das métricas corresponderão, para cada conjunto de parâmetros, à media aritmética dos valores obtidos em cada uma das dez iterações.

Tabela 12: Módulos e Cenários de Teste

Módulo	Cenário	Treino		Teste	
		Ficheiro	**Tráfego**	**Ficheiro**	**Tráfego**
A	1	Dados_02	Normal+Ataque1	Dados_02	Normal+Ataque2
	2	Dados_02	Normal+Ataque1	Dados_05	Normal+Ataque1
	3	Dados_02	Normal+Ataque2	Dados_05	Normal+Ataque2
	4	Dados_02	Normal+Ataques	Dados_05	Normal+Ataques
B	1	Dados_03	Normal+Ataque1	Dados_05	Normal+Ataque2
	2	Dados_03	Normal+Ataque1	Dados_05	Normal+Ataque1
	3	Dados_03	Normal+Ataque2	Dados_05	Normal+Ataque2
	4	Dados_03	Normal+Ataques	Dados_05	Normal+Ataques

Assim, para cada iteração são seleccionados 140000 registos do ficheiro com os dados de treino - que irão constituir os dados de treino dos três algoritmos - e 60000 provenientes do ficheiro com os dados de teste, que irão constituir os dados de teste.

A selecção é efectuada com recurso a uma funcionalidade da API do WEKA. De facto, e tendo como objectivo garantir a repetitividade e reprodutibilidade dos testes em igualdade de circunstâncias, o WEKA - tanto na versão gráfica como na API, permite a definição de um valor **Seed**, que serve para inicializar o gerador de números aleatórios. Diferentes valores de Seed implicam valores aleatórios diferentes e, consequentemente, no caso em referência, diferentes registos seleccionados. O mesmo valor de Seed assegurará que as condições da experiência sejam idênticas e, portanto, que os resultados obtidos sejam os mesmos.

Como já foi referido, foram utilizados os algoritmos CLONALG, Backpropagation Multi-Layer Perceptron e Learning Vector Quantization. Para além disso, foi, também, considerado um *ensemble* dos três algoritmos, em que a estratégia de decisão se baseia no critério de decisão por maioria (*Majority Voting* na terminologia anglo-saxónica).

5.5 RESULTADOS

Esta secção tem como objectivo apresentar os resultados obtidos para cada um dos módulos de testes. Os resultados para o módulo A estão na secção 5.5.1 e os do módulo B na secção 5.5.2.

5.5.1 *Módulo A*

As tabelas 13, 14, 15 e 16 apresentam os resultados obtidos para cada um dos cenários do módulo A. Para um dado algoritmo, os valores de cada métrica correspondem à média aritmética dos valores obtidos em cada uma das dez iterações do cenário.

Tabela 13: Resultados - Cenário 1

Algoritmo	**TPR**	**TNR**	**FPR**	**FNR**	**Precision**	**Recall**	**Accuracy**	**F1**
MLP	0,652446	0,999966	0,000034	0,347554	0,995378	0,652446	0,996178	0,788013
Ensemble	0,045719	0,999897	0,000103	0,954281	0,195966	0,045719	0,989496	0,072391
LVQ	0,045719	0,999680	0,000320	0,954281	0,169517	0,045719	0,989281	0,070213
CLONALG	0,019419	0,983010	0,016990	0,980581	0,003042	0,019419	0,972506	0,002821

Tabela 14: Resultados - Cenário 2

Algoritmo	**TPR**	**TNR**	**FPR**	**FNR**	**Precision**	**Recall**	**Accuracy**	**F1**
CLONALG	0,000000	0,999920	0,000080	1,000000	0,000000	0,000000	0,995153	0,000000
Ensemble	0,000000	0,999948	0,000052	1,000000	0,000000	0,000000	0,995182	0,000000
LVQ	0,000000	0,999948	0,000052	1,000000	0,000000	0,000000	0,995182	0,000000
MLP	0,000000	0,999757	0,000243	1,000000	0,000000	0,000000	0,994992	0,000000

Tabela 15: Resultados - Cenário 3

Algoritmo	**TPR**	**TNR**	**FPR**	**FNR**	**Precision**	**Recall**	**Accuracy**	**F1**
CLONALG	0,000000	1,000000	0,00000	1,000000		0,000000	0,344689078	0,000000
Ensemble	0,000000	1,000000	0,00000	1,000000		0,000000	0,344689078	0,000000
LVQ	0,000000	0,999990	9,67071E-06	1,000000	0,000000	0,000000	0,344685745	0,000000
MLP	0,000000	0,999961	3,86828E-05	1,000000	0,000000	0,000000	0,344675745	0,000000

Tabela 16: Resultados - Cenário 4

Algoritmo	**TPR**	**TNR**	**FPR**	**FNR**	**Precision**	**Recall**	**Accuracy**	**F1**
CLONALG	0,000000	0,999961	0,000039	1,000000	0,000000	0,000000	0,344109	0,000000
Ensemble	0,000000	0,999889	0,000111	1,000000	0,000000	0,000000	0,344084	0,000000
LVQ	0,000000	0,999898	0,000102	1,000000	0,000000	0,000000	0,344087	0,000000
MLP	0,000000	0,999792	0,000208	1,000000	0,000000	0,000000	0,344051	0,000000

5.5.2 *Módulo B*

Os resultados obtidos para o módulo B constam das tabelas 17, 18, 19 e 20.

À semelhança do que se verificou para os resultados do módulo A, para cada algoritmo, o valor apresentado de cada métrica corresponde à média aritmética dos valores obtidos em cada iteração do cenário.

Tabela 17: Resultados - Cenário 1

Algoritmo	**TPR**	**TNR**	**FPR**	**FNR**	**Precision**	**Recall**	**Accuracy**	**F1**
CLONALG	0,030591	0,999664	0,000336	0,969409	0,989487	0,030591	0,507060	0,059345
Ensemble	0,030591	0,999664	0,000336	0,969409	0,989487	0,030591	0,507060	0,059345
LVQ	0,030591	0,999644	0,000356	0,969409	0,988870	0,030591	0,507050	0,059344
MLP	0,000056	1,000000	0,000000	0,999944	1,000000	0,000056	0,491703	0,000111

Tabela 18: Resultados - Cenário 2

Algoritmo	**TPR**	**TNR**	**FPR**	**FNR**	**Precision**	**Recall**	**Accuracy**	**F1**
CLONALG	0,008042	0,653670	0,346330	0,991958	0,033723	0,008042	0,650593	0,010255
Ensemble	0,008042	0,653670	0,346330	0,991958	0,033723	0,008042	0,650593	0,010255
LVQ	0,702448	0,003132	0,996868	0,297552	0,003356	0,702448	0,006465	0,006679
MLP	0,000000	0,999745	0,000255	1,000000	0,000000	0,000000	0,994980	0,000000

Tabela 19: Resultados - Cenário 3

Algoritmo	**TPR**	**TNR**	**FPR**	**FNR**	**Precision**	**Recall**	**Accuracy**	**F1**
MLP	1,000000	0,999768	0,000232	0,000000	0,999878	1,000000	0,999920	0,999939
Ensemble	1,000000	0,002597	0,997403	0,000000	0,655898	1,000000	0,656206	0,792196
CLONALG	1,000000	0,002577	0,997423	0,000000	0,655894	1,000000	0,656199	0,792193
LVQ	1,000000	0,000363	0,999637	0,000000	0,655393	1,000000	0,655436	0,791828

Tabela 20: Resultados - Cenário 4

Algoritmo	**TPR**	**TNR**	**FPR**	**FNR**	**Precision**	**Recall**	**Accuracy**	**F1**
MLP	0,897660	0,999598	0,000402	0,102340	0,928373	0,897660	0,932739	0,898737
Ensemble	0,999205	0,003250	0,996750	0,000795	0,656433	0,999205	0,656474	0,792336
LVQ	1,000000	0,000756	0,999244	0,000000	0,656048	1,000000	0,656138	0,792306
CLONALG	0,999205	0,002988	0,997012	0,000795	0,656374	0,999205	0,656384	0,792293

5.6 ANÁLISE DE RESULTADOS

Nesta secção serão analisados e interpretados os resultados obtidos nos testes. Primeiramente, serão feitas algumas ilações de âmbito geral e que, portanto, são aplicáveis a qualquer um dos módulos. Seguidamente, serão analisados em detalhe os resultados obtidos em cada módulo.

5.6.1 *Considerações gerais*

Em primeiro lugar, gostaria de mencionar que o objectivo dos testes é simular, tanto quanto possível, a detecção de um ataque do tipo *zero-day*. Para esse efeito, as estratégias adoptadas foram as seguintes:

- Utilizando o mesmo tipo de ataque, treinar os algoritmos com um ataque gerado por uma ferramenta e tentar detectar o ataque gerado por outra ferramenta (cenário 1). Por exemplo, no caso do módulo A, a fase de treino é efectuada com dados correspondentes ao ataque realizado com GoldenEye e a fase de testes é realizada com dados correspondentes ao ataque realizado com Slowloris, simulando-se, assim, um ataque não conhecido, apesar de ser do mesmo tipo;
- Treino com dados de um tipo de ataque e tentar detectar ataque de outro tipo (cenários 2, 3 e 4). Mais concretamente, o objectivo é treinar o modelo com dados correspondentes a um ataque DoS e tentar detetar a ocorrência um ataque DDoS.

Depois, gostaria de referir que os testes não foram efectuados sobre o tráfego de rede em si, mas sim sobre fluxos de dados gerados a partir do tráfego de rede.

Como é sabido, um ciber-ataque é, essencialmente, uma anomalia ao tráfego normal de rede. Essa anomalia pode traduzir-se, por exemplo, num elevado volume de tráfego num curto intervalo de tempo, sendo importante averiguar, para a sua detecção, qual a cadência - número de pacotes por intervalo de tempo - de pacotes e intervalo de tempo entre cada pacote.

O uso de fluxos nestes testes inviabiliza, precisamente, a determinação do intervalo entre pacotes ou a cadência dos mesmos, não permitindo, assim, caracterizar e detectar o ataque.

Outra questão relevante prende-se com a avaliação sobre a necessidade de se aplicar uma estratégia de *ensemble*. Um dos objectivos deste trabalho era averiguar se o uso de um classificador deste género, com uma estratégia de *majority voting*, contribuia ou não para uma melhoria dos resultados apresentados por cada classificador individualmente.

Como se pode observar nas tabelas acima (ver secções 5.5.1 e 5.5.2), atendendo ao facto de dois dos três classificadores apresentarem sempre resultados muito desfavoráveis, a contribuição do *ensemble* não é significativa.

5.6.2 *Resultados do Módulo A*

De um modo geral, os resultados do módulo A são maus, o que pode significar o seguinte:

- Os algoritmos não são adequados à detecção deste tipo de ataques;
- Os ataques não geram padrões suficientes que permitam que os algoritmos aprendam as suas características e que, por conseguinte, consigam detectar o ataque;
- As parametrizações dos algoritmos poderão não ser as mais adequadas para o *dataset* utilizado.

Comecemos por identificar os tipos de ataque envolvidos. Assim, para treino dos algoritmos, temos ataques do tipo DoS, gerados por duas ferramentas distintas que, consequentemente, geram tráfego e padrões de tráfego distintos: GoldenEye e Slowloris (ver tabela 3).

Os ataques contemplados na fase de teste são do tipo DDoS, também gerados por ferramentas distintas: LOIC e HOIC (ver tabela 3).

No caso do cenário 1, o treino e teste são realizados com ataques do mesmo tipo - DoS - pelo que, apesar de serem gerados por ferramentas diferentes, é possível, para todos os algoritmos, identificar padrões e reconhecer a existência de ataque.

Pela tabela 13, podemos verificar que o algoritmo com melhor resultado - e que, portanto, seria o mais adequado para detectar este tipo de ataques - é o MLP, com um valor de F1 de cerca de 78%.

Para os restantes cenários, como se pode verificar pelos valores da métrica F1 nas tabelas 14, 15 e 16, os algoritmos não conseguem identificar a ocorrência de um ataque DDoS a partir do treino com ataques DoS. Este situação pode dever-se ao facto de que, durante a fase de treino, os algoritmos não conseguem identificar no ataque DDoS, padrões que possam ser identificados com os gerados na fase de treino, e isso tanto se verifica com o um ataque isolado como com os dois ataques.

De referir, pela sua importância, o caso do algoritmo CLONALG no cenário 3. Nesta situação, como se pode verificar na tabela 15, não foi possível calcular o valor da métrica *Precision*, pois tanto a taxa de verdadeiros positivos (TPR) como a taxa de falsos positivos têem valor zero. Isto significa que o CLONALG considerou todo o tráfego do ataque DDoS como sendo tráfego normal, o que, a verificar-se numa rede real, poderia ter consequências graves.

Uma característica interessante e que, apesar de tudo, seria de esperar: o tráfego normal é, na sua esmagadora maioria, correctamente identificado, como se pode confirmar pela elevada taxa de verdadeiros negativos (TNR) presente nas tabelas.

Como se mencionou acima, um dos factores que poderão ter influenciado os resultados nos cenários deste módulo poderá ser a parametrização dos algoritmos. Nesse sentido, e embora fora do âmbito deste trabalho, no cenário com piores resultados - no caso, o cenário 3 - aplicaram-se várias parametrizações dos algoritmos, aferindo, depois, os resultados obtidos. Os testes foram efectuados com recurso à interface gráfica do WEKA, definindo um fluxo para o efeito. O diagrama do fluxo, figura 24, pode ser consultado na secção B.2.

As configurações para cada algoritmo constam nas tabelas 21 (CLONALG), 22 (LVQ) e 23 (MLP).

Tabela 21: Variação de Parâmetros - CLONALG

Parametros	**Início**	**CLG1**	**CLG2**	**CLG3**	**CLG4**	**CLG5**
Antibody Pool Size	120	200	200	120	200	300
Clonal Factor	0,3	0,3	0,3	0,3	0,3	0,3
Number Generations	40	40	40	100	100	100
Remainder Pool Ratio	0,2	0,2	0,2	0,2	0,2	0,4
Selection Pool Size	80	120	120	80	120	240
Total Replacement	10	10	30	10	10	60
Seed	1	1	1	1	1	1

Tabela 22: Variação de Parâmetros - LVQ

Parametros	**Início**	**LVQ1**	**LVQ2**	**LVQ3**	**LVQ4**	**LVQ5**	**LVQ6**
Epsilon (Learning Weight Modifier)	0,1	0,1	0,1	0,1	0,1	0,1	0,1
initMode	1	1	1	1	1	6	5
LearningFunction	1	1	1	1	1	1	1
LearningRate	0,3	0,3	0,3	0,5	0,5	0,5	0,5
totalCBookVectors	90	150	200	90	150	200	200
iterations	4500	4500	4500	4500	4500	8000	8000
useVoting	false	false	false	false	false	true	true
windowSize	0,2	0,2	0,2	0,2	0,2	0,2	0,2
Seed	1	1	1	1	1	1	1

Tabela 23: Variação de Parâmetros - MLP

Parametros	Início	MLP1	MLP2	MLP3	MLP4	MLP5
hiddenLayers	a	t	i	"80,80,10"	"80,40,10"	"100,50,10"
LearningRate	0,1	0,1	0,1	0,1	0,1	0,3
momentum	0,3	0,3	0,3	0,3	0,3	0,3
iterations (epochs)	500	500	500	500	500	500
Seed	1	1	1	1	1	1

Os resultados obtidos para cada algoritmo encontram-se nas tabelas 24 para o CLONALG, 25 no caso do LVQ e 26 para o MLP.

Tabela 24: Variação Parâmetros - Resultados CLONALG

Modelo	TPR	FPR	FNR	TNR	Precision	Recall	Accuracy	F1
Início	0	0	1	1	#DIV/0!	0	0,344689	0
CLG1	0	0	1	1	#DIV/0!	0	0,344681	0
CLG2	0	0	1	1	#DIV/0!	0	0,344681	0
CLG3	0	0	1	1	#DIV/0!	0	0,344681	0
CLG4	0	0	1	1	#DIV/0!	0	0,344681	0
CLG5	0	0	1	1	#DIV/0!	0	0,344681	0

Tabela 25: Variação de Parâmetros - Resultados LVQ

Modelo	TPR	FPR	FNR	TNR	Precision	Recall	Accuracy	F1
Início	0	0	1	1	#DIV/0!	0	0,344689	0
LVQ1	0	0	1	1	#DIV/0!	0	0,344681	0
LVQ2	0	0	1	1	#DIV/0!	0	0,344681	0
LVQ3	0	0	1	1	#DIV/0!	0	0,344681	0
LVQ4	0	0	1	1	#DIV/0!	0	0,344681	0
LVQ5	0	0	1	1	#DIV/0!	0	0,344681	0
LVQ6	0,006107	0,000152	0,993893	0,999848	0,307692	0,006107	0,989004	0,011976

Tabela 26: Variação de Parâmetros - Resultados MLP

Modelo	TPR	FPR	FNR	TNR	Precision	Recall	Accuracy	F1
Início	0	0,000048	1	0,999952	0	0	0,344672	0
MLP1	0	0	1	1	#DIV/0!	0	0,344681	0
MLP2	0	0	1	1	#DIV/0!	0	0,344681	0
MLP3	0	0,000140	1	0,999860	0	0	0,344633	0
MLP4	0	0	1	1	#DIV/0!	0	0,344681	0
MLP5	0	0,000140	1	0,999860	0	0	0,344633	0

Conforme se pode verificar, apenas no caso do algoritmo LVQ, na sexta configuração, houve melhorias significativas. De facto, neste caso específico, o algoritmo reconheceu algum do tráfego malicioso, ainda que muito residual.

O facto de não se conseguir melhorar os resultados para os algoritmos CLONALG e MLP não quer dizer que estes não sejam adequados para detectar este tipo de tráfego malicioso. De facto, pode apenas significar que não foi encontrada a combinação de valores para os diversos parâmetros que permita que estes algoritmos apresentem um melhor desempenho.

5.6.3 *Resultados do Módulo B*

Neste módulo, os ataques DoS foram gerados pelas ferramentas SlowHTTPTest e Hulk, conforme se pode verificar na tabela 3.

Os ataques DDoS são os mesmos que foram utilizados no módulo anterior.

Globalmente, os resultados obtidos neste módulo são francamente melhores do que os obtidos no módulo A. Sendo utilizados os mesmos algoritmos configurados com os mesmos parâmetros, esta situação deve-se, à partida, ao facto de as ferramentas utilizadas nos ataques produzirem padrões com maior semelhança. Seria, assim, possível que um IDS baseado em comportamento e que utilizasse estes algoritmos conseguisse identificar e sinalizar um ataque *zero-day*.

Nos cenários 1 e 2, apesar de apresentarem uma taxa de verdadeiros positivos (TPR) baixa, destaca-se o algoritmo CLONALG, que conseguiu ter a melhor performance, juntamente com o *ensemble*, como se pode verificar pelos valores de F1.

Já o algoritmo MLP mostrou-se incapaz de lidar com este tipo de tráfego, apenas identificando correctamente a esmagadora maioria do tráfego normal.

Por oposição, no cenário 2, o algoritmo LVQ apresentou a maior taxa de verdadeiros positivos (TPR) do cenário, apesar de falhar na identificação do tráfego normal (menor valor de TNR do cenário).

Nos cenários 3 e 4, vemos o domínio do algoritmo MLP, com valores altos de F1 chegando, mesmo, a ser muito próximo de 100% no cenário 3.

No cenário 3, como se pode verificar na tabela 19, todos os algoritmos identificaram correctamente todo o tráfego malicioso (TPR=1), o que poderá estar relacionado com a semelhança dos padrões de tráfego gerados pelas ferramentas respectivas.

No que toca ao tráfego normal, apenas o MLP tem um bom desempenho (TNR muito próximo de 100%), ao passo que os restantes algoritmos apresentam uma identificação muito residual.

O cenário 4, que apresenta uma maior diversidade no que toca ao tráfego malicioso - engloba, quer na fase de treino, quer na fase de testes, o tráfego gerado pelas duas ferramentas em cada tipo de ataque - não traz vantagem significativa. De facto, a performance do MLP, traduzida no valor de F1, baixa cerca de 10 pontos percentuais, ao passo que, nos restantes algoritmos, há uma melhoria pouco significativa.

6

CONCLUSÕES E TRABALHO FUTURO

Neste capítulo serão apresentadas as conclusões deste trabalho, fazendo um paralelo entre os objectivos traçados inicialmente e os resultados obtidos.

Serão, também, apresentadas algumas sugestões de possíveis evoluções deste trabalho.

6.1 CONCLUSÕES

No final deste trabalho, algumas palavras em jeito de conclusão.

Primeiramente, uma pequena referência ao *dataset* utilizado, o CSE-CIC IDS-2018. É um *dataset* recente, especialmente vocacionado para teste de sistemas de detecção de intrusões, e que está a ganhar o seu espaço junto da comunidade académica.

Trata-se de um *dataset* bastante extenso, quer em número de atributos, quer no volume de dados, sendo que os dados são apresentados em dois formatos distintos: ficheiros .pcap (pacotes de rede capturados) e ficheiros CSV, já preparados para tratamento por algoritmos de aprendizagem automática.

Quanto aos ficheiros PCAP, a sua nomenclatura não permite, de forma fácil, identificar o respectivo conteúdo. Para além disso, como se trata da captura directa dos pacotes de rede, não é fácil identificar quais os que correspondem a tráfego malicioso ou normal.

Nos ficheiros CSV, cada linha corresponde a um fluxo de pacotes de dados de rede (e não a um pacote de rede), o que, só por si, inviabiliza a determinação de alguns parâmetros, como sejam a cadência de pacotes ou o intervalo de tempo entre pacotes. Para além disso, em alguns ficheiros há algumas linhas erradas, devido à metodologia de construção do próprio *dataset*. Um exemplo disto é aparecer uma linha idêntica à linha de cabeçalho do ficheiro mas no meio do mesmo, causando o surgimento de uma nova classe - Label, o nome do campo que determina a classe - e de erros, pois os restantes valores dessa linha são assumidos como valores "válidos".

Apesar de constituir um conjunto massivo de dados, em alguns tipos de ataque a percentagem de linhas classificadas como ataque é tão baixa que pode, eventualmente, passar despercebido e ser classificado como tráfego normal. Posso citar o exemplo do ficheiro Dados_06 (ver secção A.2), em que, para um total de 1048575 linhas, apenas 362 correspondem a ataques.

Apesar destes problemas, em minha opinião, o *dataset* é bastante útil, pois dispõe de um vasto conjunto de dados abrangendo diversos tipos de ataques, constituindo uma boa ferramenta de trabalho.

Quanto aos objectivos deste trabalho, pode dizer-se que foram atingidos. De facto, foi criada uma arquitectura de testes, baseada essencialmente em ferramentas *open-source*. De salientar, aqui, as aplicações de *data-mining* utilizadas, o WEKA e o Orange.

Em termos de interface com o utilizador e funcionalidades disponibilizadas, o Orange é muito mais *user-friendly*, quer no uso da interface quer mesmo nos componentes disponíveis. Um caso simples é o componente que selecciona aleatoriamente um determinado número de amostras a partir de um *dataset*; o WEKA apenas permite especificar uma percentagem do total de linhas do *dataset*.

Por outro lado, o sistema de *plugins* do WEKA inclui os algoritmos bioinspirados utilizados neste trabalho, ao passo que no Orange esses algoritmos não existem. O WEKA permite, também, a repetibilidade da experiência, garantindo os mesmos resultados, através do uso, em cada componente, de uma variável que inicializa o gerador de números aleatórios.

O uso das duas ferramentas, em complementaridade, foi, de facto, uma mais-valia para a realização deste trabalho, pois permitiu aproveitar e combinar as melhores características das duas para a obtenção dos resultados.

Foi, também, desenvolvida uma aplicação de suporte a este trabalho. A aplicação, desenvolvida com resurso a ferramentas *open-source* - nas quais se incluem, obviamente, a API do WEKA - permitiu automatizar toda a fase de testes do trabalho, permitindo uma customização das operações que não seria possível com recurso à interface gráfica do WEKA. Esta ferramenta de suporte, apesar da sua especificidade, é um bom exemplo de aplicação da API do WEKA.

No que aos testes diz respeito, procurou-se obter significância estatística, executando 10 vezes os testes para cada algoritmo. A parametrização de cada algoritmo foi obtida empiricamente, conjugando os requisitos do próprio algoritmo e os dados a analisar.

Foram definidos 4 cenários de trabalho e considerados dois módulos de teste, cada um contemplando os 4 cenários.

No modulo A os resultados não foram os melhores, com os algoritmos a identificarem, na sua maioria, apenas o tráfego normal.

Numa tentativa de identificar se o problema residiria na parametrização dos algoritmos, foram efectuados testes, com recurso à interface gráfica do WEKA, em que seriam alterados, em cada teste, os parâmetros dos algoritmos. Verificou-se que apenas no caso do algoritmo LVQ foi possível melhorar os resultados.

De realçar que, conforme se mencionou, o facto de não se conseguir melhorar os resultados para os outros dois algoritmos não quer dizer que os mesmos não sejam adequados para a detecção de intrusões. De facto, isto pode apenas significar que, apesar dos esforços desenvolvidos, ainda não terão sido encontrados os parâmetros que produzam os melhores resultados

Os resultados do módulo B são mais simpáticos e com algumas surpresas. Nos dois primeiros cenários, o algoritmo em destaque é o CLONALG, apesar da taxa de verdadeiros positivos ser bastante baixa. A surpresa pela negativa foi o desempenho, nestes cenários, do MLP. Apesar de identificar correctamente a esmagadora maioria do tráfego normal, falha redondamente na identificação do tráfego malicioso.

Já nos dois últimos cenários, o domínio do MLP é absoluto, com valores de F1 e da taxa de verdadeiros positivos acima dos 89%.

Para além dos algoritmos usados individualmente, foi, também, implementado um classificador *ensemble* que, através de uma estratégia de *majority voting*, iria definir se cada linha do *dataset* de teste seria tráfego malicioso ou não. Todos os algoritmos usados na votação tinham igual ponderação. Pode-se verificar, pelas tabelas 13 a 20, que, no que toca ao valor de F1, o *ensemble* está sempre em segundo lugar, com valores marginalmente diferentes do valor F1 do algoritmo que está em terceiro lugar em cada tabela.

Globalmente, podemos verificar que, tendo em conta a estratégia adoptada - *majority voting* - e considerando que dois dos algoritmos apresentam, na sua maioria, resultados menos significativos, o recurso ao classificador *ensemble* não traz mais-valias significativas.

6.2 TRABALHO FUTURO

Conforme já foi mencionado, está fora do âmbito deste trabalho a optimização dos parâmetros de cada algoritmo bem como a redução do número de atributos do *dataset*. Sendo assim, em termos de trabalho futuro, pode-se sugerir:

- Determinar a configuração óptima para cada algoritmo, obtendo os valores para cada parâmetro com recurso a técnicas adequadas ao efeito, como sejam as técnicas de *hill-climbing*;
- com vista a diminuir o esforço computacional necessário, efectuar uma redução do número de atributos do dataset, recorrendo a métodos adequados, como sejam o *Principal Components Analysis* (PCA) ou métodos baseados em algoritmos genéticos.

Para além disso, pode-se considerar o recurso a outros algoritmos bioinspirados, sejam imuno-inspirados (AIRS, Immunos), neurológicos (*deep learning*) ou outros, como seja o caso de *swarm intelligence* ou sistemas autonómicos.

Também a aplicação desenvolvida pode ser alvo de uma reestruturação, reformulando, por exemplo, as opções da linha de comando, uma vez que estas foram implementadas para dar resposta a necessidades específicas deste trabalho. Também a localização dos ficheiros de entrada e saída, que neste momento é *hard-coded*, poderia ser fornecida dinamicamente, quer através de um ficheiro de configuração quer por parâmetros na linha de comando.

BIBLIOGRAFIA

[1] SYMANTEC CORPORATION. *ISTR - Internet Security Threat Report*. SYMANTEC CORPORATION, fev. de 2019. URL: https://www.symantec.com/security-center/threat-report.

[2] Eibe Frank, Mark A. Hall e Ian H. Witten. «The WEKA Workbench». Em: *Data Mining: Practical Machine Learning Tools and Techniques*. Ed. por Morgan Kaufmann. 4ª ed. 2016. Cap. Online Appendix. URL: https://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf.

[3] L. N. de Castro e F. J. Von Zuben. «Learning and optimization using the clonal selection principle». Em: *IEEE Transactions on Evolutionary Computation* 6.3 (jun. de 2002), pp. 239–251. ISSN: 1941-0026. DOI: 10.1109/TEVC.2002.1011539.

[4] L. N. de Castro e F. J. Von Zuben. «The clonal selection algorithm with engineering applications». Em: *Proceedings of GECCO*. Ed. por editor. 2000, pp. 36–39.

[5] Teuvo Kohonen. *Self-Organizing Maps*. Vol. 30. Springer Series in Information Sciences. Springer Science & Business Media, 2001. DOI: 10.1007/978-3-642-56927-2.

[6] The Genius Blog, ed. *Basics of Multilayer Perceptron – A Simple Explanation of Multilayer Perceptron*. Jan. de 2018. URL: https://kindsonthegenius.com/blog/2018/01/basics-of-multilayer-perceptron-a-simple-explanation-of-multilayer-perceptron.html (acedido em 11/02/2020).

[7] Judith Hurwitz e Daniel Kirsch. *Machine Learning for dummies*. IBM Limited Edition. John Wiley & Sons, Inc, 2018.

[8] IBM, ed. *What is machine learning?* URL: https://www.ibm.com/topics/machine-learning (acedido em 19/02/2020).

[9] M. Tim Jones. «Supervised learning models. Explore some supervised learning approaches such as support vector machines and probabilistic classifiers». Em: (26 de fev. de 2018). Ed. por IBM. URL: https://developer.ibm.com/articles/cc-supervised-learning-models/ (acedido em 19/02/2020).

[10] M. Tim Jones. «Unsupervised learning for data classification. Discover the theory and ideas behind unsupervised learning». Em: (4 de dez. de 2017). Ed. por IBM. URL: https://developer.ibm.com/articles/cc-unsupervised-learning-data-classification/.

[11] Andreas Fuchsberger. «Intrusion Detection Systems and Intrusion Prevention Systems». Em: *Information Security Technical Report* 10 (3 2005), pp. 134–139. DOI: 10.16/j.istr.2005.08.001.

[12] Karen Scarfone e Peter Mell. *Guide to Intrusion Detection and Prevention Systems (IDPS) (Draft). Recommendations of the National Institute of Standards and Technology*. NIST - National Institute of Standards e Technology. NIST - National Institute of Standards e Technology, 2012.

[13] Vinod Kumar e Om Prakash Sangwan. «Signature based intrusion detection system using SNORT». Em: *International Journal of Computer Applications & Information Technology* 1.3 (2012), pp. 35–41.

[14] P. García-Teodoro et al. «Anomaly-based network intrusion detection: Techniques, systems and challenges». Em: *Computers and Security* 28 (1-2 mar. de 2009), pp. 18–28. DOI: https://doi.org/10.1016/j.cose.2008.08.003.

[15] Monowar Bhuyan, Dhruba K Bhattacharyya e Jugal Kalita. «Towards Generating Real-life Datasets for Network Intrusion Detection». Em: *International Journal of Network Security* 17 (nov. de 2015), pp. 683–701.

[16] Markus Ring et al. «A Survey of Network-based Intrusion Detection Datasets». Em: (2019).

[17] Amjad Al-Tobi e Ishbel Duncan. «KDD 1999 generation faults: a review and analysis». Em: *Journal of Cyber Security Technology* (set. de 2018), pp. 1–37. DOI: 10.1080/23742917.2018.1518061.

[18] John McHugh. «Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory». Em: *ACM Trans. Inf. Syst. Secur.* 3.4 (nov. de 2000), pp. 262–294. ISSN: 1094-9224. DOI: 10.1145/382912.382923. URL: https://doi.org/10.1145/382912.382923.

[19] Nour Moustafa e Jill Slay. «UNSW-NB15: A Comprehensive Dataset for Network Intrusion Detection systems. (UNSW-NB15 Network Data Set)». Em: ().

[20] Iman Sharafaldin., Arash Habibi Lashkari. e Ali A. Ghorbani. «Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization». Em: *Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,* INSTICC. SciTePress, 2018, pp. 108–116. ISBN: 978-989-758-282-0. DOI: 10.5220/0006639801080116.

[21] Mohammad Mahboubian e Nor Asilah Wati Abdul Hamid. «A Machine Learning Based AIS IDS». Em: *International Journal of Machine Learning and Computing* 3.3 (jun. de 2013), pp. 259–262. DOI: 10.7763/IJMLC.2013.V3.315.

[22] Nasir Rashid et al. «Artificial Immune System–Negative Selection Classification Algorithm (NSCA) for Four Class Electroencephalogram (EEG) Signals». Em: *Frontiers in Human Neuroscience* 12 (2018), p. 439. ISSN: 1662-5161. DOI: 10.3389/fnhum.2018.00439. URL: https://www.frontiersin.org/article/10.3389/fnhum.2018.00439.

[23] Neuroscience News, ed. *Brain Like Activity in Immune System Promises Better Disease Treatments*. Australian National University. 13 de jul. de 2017. URL: https://neurosciencenews.com/immune-system-brain-activity-7074/ (acedido em 18/02/2020).

[24] Sagar Aryal. *Difference between Innate and Adaptive Immunity*. Ed. por MicrobiologyInfo.com. 23 de jun. de 2018. URL: https://microbiologyinfo.com/difference-between-innate-and-adaptive-immunity/ (acedido em 18/02/2020).

[25] Donald R Forsdyke. «The origins of the clonal selection theory of immunity as a case study for evaluation in science.» Em: *The FASEB Journal* 9.2 (1995), pp. 164–166.

[26] F. M. Burnet. «A modification of jerne's theory of antibody production using the concept of clonal selection». Em: *CA: A Cancer Journal for Clinicians* 26.2 (1976), pp. 119–121. DOI: 10.3322/canjclin.26.2.119. eprint: https://acsjournals.onlinelibrary.wiley.com/doi/pdf/10.3322/canjclin.26.2.119. URL: https://acsjournals.onlinelibrary.wiley.com/doi/abs/10.3322/canjclin.26.2.119.

[27] Sir Frank Macfarlane Burnet et al. *The clonal selection theory of acquired immunity*. Vol. 3. Vanderbilt University Press Nashville, 1959.

[28] Jason Brownlee. «Clonal selection theory & clonalg-the clonal selection classification algorithm (csca)». Em: *Swinburne University of Technology* (2005), p. 38.

[29] Shangce Gao et al. «An Improved Clonal Selection Algorithm and Its Application to Traveling Salesman Problems». Em: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E90-A (dez. de 2007). DOI: 10.1093/ietfec/e90-a.12.2930.

[30] Stephanie Forrest et al. «Self-nonself discrimination in a computer». Em: *Proceedings of 1994 IEEE computer society symposium on research in security and privacy*. Ieee. 1994, pp. 202–212.

[31] Andrew Watkins, Jon Timmis e Lois Boggess. «Artificial Immune Recognition System (AIRS): An Immune-Inspired Supervised Learning Algorithm». Em: *Genetic Programming and Evolvable Machines* 5.3 (set. de 2004), pp. 291–317. ISSN: 1573-7632. URL: https://doi.org/10.1023/B:GENP.0000030197.83685.94.

[32] Jason Brownlee et al. «Artificial immune recognition system (airs)-a review and analysis». Em: *Center for Intelligent Systems and Complex Processes (CISCP), Faculty of Information and Communication Technologies (ICT), Swinburne University of Technology, Victoria, Australia* (2005).

[33] Daniel Graupe. *Principles of artificial neural networks*. Vol. 7. Advanced Series in Circuits and Systems. World Scientific, 2013.

[34] *Sistema Nervoso Central. Neurônio*. URL: https://sites.google.com/site/anatomiasistemanervosocentral/home/neuronio (acedido em 18/06/2020).

[35] Darrell Whitley. «A genetic algorithm tutorial». Em: *Statistics and computing* 4.2 (1994), pp. 65–85.

[36] Oliver Kramer. *Genetic algorithm essentials*. Vol. 679. Studies in Computational Intelligence. Springer, 2017.

[37] A. G. Ganek e T. A. Corbi. «The dawning of the autonomic computing era». Em: *IBM Systems Journal* 42.1 (2003), pp. 5–18.

[38] Philippe Lalanda, Julie A McCann e Ada Diaconescu. *Autonomic computing: principles, design and implementation*. Undergraduate Topics in Computer Science. Springer Science & Business Media, 2013.

[39] C. Kolias, G. Kambourakis e M. Maragoudakis. «Swarm intelligence in intrusion detection: A survey». Em: *Computers & Security* 30.8 (2011), pp. 625–642. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2011.08.009. URL: http://www.sciencedirect.com/science/article/pii/S016740481100109X.

[40] Saeed Ghorbani, Morteza Barari e Mojtaba Hoseini. «Presenting a new method to improve the detection of micro-seismic events». Em: *Environmental Monitoring and Assessment* 190 (jul. de 2018). DOI: 10.1007/s10661-018-6837-6.

[41] Jason Brownlee. *Machine Learning Mastery. Making developers Awesome at Machine Learning*. URL: https://machinelearningmastery.com/ (acedido em 18/06/2020).

[42] Jason Brownlee. *WEKA Classification Algorithms. A WEKA Plug-in*. URL: https://web.archive.org/web/20181113025835/http://wekaclassalgos.sourceforge.net/ (acedido em 05/07/2020).

[43] Pedro Domingos. *A Revolução do Algoritmo Mestre. Como a aprendizagem automática está a mudar o mundo*. Ed. por Manuscrito Editora. 10-2017. ISBN: 9789898871183.

[44] G. Creech e J. Hu. «Generation of a new IDS test dataset: Time to retire the KDD collection». Em: *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. Abr. de 2013, pp. 4487–4492. DOI: 10.1109/WCNC.2013.6555301.

[45] Jiyeon Kim, Yulim Shin e Eunjung Choi. «An Intrusion Detection Model based on a Convolutional Neural Network». Em: *Journal of Multimedia Information System* 6.4 (2019), pp. 165–172. DOI: 10.33851/JMIS.2019.6.4.165. URL: https://doi.org/10.33851/JMIS.2019.6.4.165.

[46] Mohamed Amine Ferrag et al. «Deep Learning Techniques for Cyber Security Intrusion Detection : A Detailed Analysis». Em: 6th International Symposium for ICS & SCADA Cyber Security Research 2019 (ICS-CSR). URL: https://www.scienceopen.com/hosted-document?doi=10.14236/ewic/icscsr19.16 (acedido em 04/03/2020).

[47] Qianru Zhou e Dimitrios Pezaros. *Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection – An Analysis on CIC-AWS-2018 dataset*. 2019. arXiv: 1905.03685 [cs.CR].

[48] Rama Krushna Das et al. *E-CLONALG: A classifier based on Clonal Selection Algorithm*. Em: *Transactions on Machine Learning and Artificial Intelligence*. Vol. 4. 6. 2016. DOI: https://doi.org/10.14738/tmlai.46.2562.

[49] H. M. Elshafie, T. M. Mahmoud e A. A. Ali. «Improving the Performance of the Snort Intrusion Detection Using Clonal Selection». Em: *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*. Fev. de 2019, pp. 104–110. DOI: 10.1109/ITCE.2019.8646601.

[50] Ali Asghar Heidari et al. «An efficient hybrid multilayer perceptron neural network with grasshopper optimization». Em: *Soft Computing* 23.17 (2019), pp. 7941–7958.

[51] Zakaria Alameer et al. «Forecasting gold price fluctuations using improved multilayer perceptron neural network and whale optimization algorithm». Em: *Resources Policy* 61 (2019), pp. 250–260. ISSN: 0301-4207. DOI: https://doi.org/10.1016/j.resourpol.2019.02.014. URL: http://www.sciencedirect.com/science/article/pii/S0301420718304926%22.

[52] T Mahrina et al. «Comparative Analysis of Backpropagation With Learning Vector Quantization (LVQ) to Predict Rainfall in Medan City». Em: *Journal of Physics: Conference Series* 1235 (jun. de 2019), p. 012083. DOI: 10.1088/1742-6596/1235/1/012083. URL: https://doi.org/10.1088%2F1742-6596%2F1235%2F1%2F012083.

[53] Poltak Sihombing et al. «The Student Attendance Controlling Based on Face Recognition by using Learning Vectorization Quantization (LVQ) Algorithm». Em: *Journal of Physics: Conference Series* 1235 (jun. de 2019), p. 012069. DOI: 10.1088/1742-6596/1235/1/012069. URL: https://doi.org/10.1088%2F1742-6596%2F1235%2F1%2F012069.

[54] R. Arifando et al. «Hybrid Genetic Algorithm Learning Vector Quantization for Classification of Social Assistance Recipients». Em: *2019 International Conference on Sustainable Information Engineering and Technology (SIET)*. 2019, pp. 316–321.

[55] Pande Nyoman Ariyuda Semadi e Reza Pulungan. «Improving learning vector quantization using data reduction». Em: *International Journal of Advances in Intelligent Informatics* 5.3 (2019), pp. 218–229.

[56] Janez Demšar et al. «Orange: Data Mining Toolbox in Python». Em: *Journal of Machine Learning Research* 14 (2013), pp. 2349–2353. URL: http://jmlr.org/papers/v14/demsar13a.html.

[57] *CSE-CIC-IDS2018 on AWS. A collaborative project between the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC)*. CIC - Canadian Institute for Cibersecurity. 2018. URL: https://www.unb.ca/cic/datasets/ids-2018.html (acedido em 01/12/2019).

[58] Tanishka Shorey et al. «Performance comparison and analysis of slowloris, goldeneye and xerxes ddos attack tools». Em: *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE. 2018, pp. 318–322.

[59] Thanachon Cheepborisuttikul e Yunyong Teng-amnuay. «Using Low Orbit Ion Cannon for Denial of Service Attack Based on CVE». Em: *Proceedings of the Second International Conference on Advances in Information Technology - AIT 2013*. Institute of Research Engineers and Doctors. 2013, pp. 145–149. ISBN: 978-981-07-5939-1.

[60] Shashank Kyatam, Abdullah Alhayajneh e Thaier Hayajneh. «Heartbleed attacks implementation and vulnerability». Em: *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. IEEE. 2017, pp. 1–6.

[61] Claude Sammut e Geoffrey I. Webb. *Encyclopedia of Machine Learning and Data Mining*. 2nd. Springer Publishing Company, Incorporated, 2017. ISBN: 148997685X.

[62] Markus Ringnér. «What is principal component analysis?» Em: *Nature Biotechnology* 26.3 (mar. de 2008), pp. 303–304. ISSN: 1546-1696. URL: https://doi.org/10.1038/nbt0308-303.

[63] J Shlens. «A tutorial on principal component analysis. arXiv». Em: *arXiv preprint arXiv:1404.1100* (2014).

APÊNDICES

A

DATASET CSE-CIC-IDS2018

A.1 LISTAGEM DE ATRIBUTOS DO DATASET

Tabela 27: Características do dataset CIC-IDS2018[57]

Feature Name	**Description**
DstPort	Destination Port
Protocol	Protocol
Timestamp	Timestamp
Flow Feduration	Duration of the flow in Microsecond
total FWwd Packet	Total packets in the forward direction
total Bwd packets	Total packets in the backward direction
total Length of Fwd Packet	Total size of packet in forward direction
total Length of Bwd Packet	Total size of packet in backward direction
Fwd Packet Length Min	Minimum size of packet in forward direction
Fwd Packet Length Max	Maximum size of packet in forward direction
Fwd Packet Length Mean	Mean size of packet in forward direction
Fwd Packet Length Std	Standard deviation size of packet in forward direction
Bwd Packet Length Min	Minimum size of packet in backward direction
Bwd Packet Length Max	Maximum size of packet in backward direction
Bwd Packet Length Mean	Mean size of packet in backward direction
Bwd Packet Length Std	Standard deviation size of packet in backward direction
Flow Byte/s	Number of flow packets per second
Flow Packets/s	Number of flow bytes per second

Tabela 27: Continuação

Feature Name	**Description**
Flow IAT Mean	Mean time between two packets sent in the flow
Flow IAT Std	Standard deviation time between two packets sent in the flow
Flow IAT Max	Maximum time between two packets sent in the flow
Flow IAT Min	Minimum time between two packets sent in the flow
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Fwd IAT Max	Maximum time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Total	Total time between two packets sent in the forward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Total	Total time between two packets sent in the backward direction
Fwd PSH flag	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
Bwd PSH Flag	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
Fwd URG Flag	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)

Tabela 27: Continuação

Feature Name	**Description**
Bwd URG Flag	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
Fwd Header Length	Length of header for forward packet
Bwd Header Length	Length of header for backward packet
FWD Packets/s	Number of forward packets per second
Bwd Packets/s	Number of backward packets per second
Min Packet Length	Minimum length of a packet
Max Packet Length	Maximum length of a packet
Packet Length Mean	Mean length of a packet
Packet Length Std	Standard deviation length of a packet
Packet Length Variance	Variance length of a packet
FIN Flag Count	Number of packets with FIN
SYN Flag Count	Number of packets with SYN
RST Flag Count	Number of packets with RST
PSH Flag Count	Number of packets with PUSH
ACK Flag Count	Number of packets with ACK
URG Flag Count	Number of packets with URG
CWR Flag Count	Number of packets with CWE
ECE Flag Count	Number of packets with ECE
down/Up Ratio	Download and upload ratio
Average Packet Size	Average size of packet
Avg Fwd Segment Size	Average size observed in the forward direction
AVG Bwd Segment Size	Average number of bytes bulk rate in the forward direction
Fwd Avg Bytes/Bulk	Average number of bytes bulk rate in the forward direction
Fwd AVG Packet/Bulk	Average number of packets bulk rate in the forward direction
Fwd AVG Bulk Rate	Average number of bulk rate in the forward direction
Bwd Avg Bytes/Bulk	Average number of bytes bulk rate in the backward direction

Tabela 27: Continuação

Feature Name	**Description**
Bwd AVG Packet/Bulk	Average number of packets bulk rate in the backward direction
Bwd AVG Bulk Rate	Average number of bulk rate in the backward direction
Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction
Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
Init_Win_bytes_forward	The total number of bytes sent in initial window in the forward direction
Init_Win_bytes_backward	The total number of bytes sent in initial window in the backward direction
Act_data_pkt_forward	Count of packets with at least 1 byte of TCP data payload in the forward direction
min_seg_size_forward	Minimum segment size observed in the forward direction
Active Min	Minimum time a flow was active before becoming idle
Active Mean	Mean time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Idle Min	Minimum time a flow was idle before becoming active
Idle Mean	Mean time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active

A.2 SUMÁRIO DO DATASET

Recorrendo à API do Weka, produziu-se um sumário das características do dataset, nomeadamente, para cada dia, o número de instâncias, o número de classes e o número de instâncias classificadas em cada classe. Devido à sua dimensão, não foi possível efectuar a contabilização para o ficheiro Dados_04.

```
====================================
Ficheiro : Dados01
Valores distintos da classe: 3
Numero de linhas do dataset: 1048575
Benign => 667626
FTP-BruteForce => 193360
SSH-Bruteforce => 187589
====================================

====================================
Ficheiro : Dados02
Valores distintos da classe: 3
Numero de linhas do dataset: 1048575
Benign => 996077
DoS attacks-GoldenEye => 41508
DoS attacks-Slowloris => 10990
====================================

====================================
Ficheiro : Dados03
Valores distintos da classe: 4
Numero de linhas do dataset: 1048575
Benign => 446772
DoS attacks-SlowHTTPTest => 139890
DoS attacks-Hulk => 461912
Label => 1
====================================

====================================
Ficheiro : Dados05
Valores distintos da classe: 3
```

```
Numero de linhas do dataset: 1048575
Benign => 360833
DDOS attack-LOIC-UDP => 1730
DDOS attack-HOIC => 686012
====================================

====================================
Ficheiro : Dados06
Valores distintos da classe: 4
Numero de linhas do dataset: 1048575
Benign => 1048213
Brute Force -Web => 249
Brute Force -XSS => 79
SQL Injection => 34
====================================

====================================
Ficheiro : Dados07
Valores distintos da classe: 4
Numero de linhas do dataset: 1048575
Benign => 1048009
Brute Force -Web => 362
Brute Force -XSS => 151
SQL Injection => 53
====================================

====================================
Ficheiro : Dados08
Valores distintos da classe: 3
Numero de linhas do dataset: 613104
Benign => 544200
Label => 33
Infilteration => 68871
====================================

====================================
Ficheiro : Dados09
Valores distintos da classe: 3
```

```
Numero de linhas do dataset: 331125
Benign => 238037
Label => 25
Infilteration => 93063
====================================

====================================
Ficheiro : Dados10
Valores distintos da classe: 2
Numero de linhas do dataset: 1048575
Benign => 762384
Bot => 286191
====================================
```

B

DIAGRAMAS DE FLUXOS

B.1 ORANGE

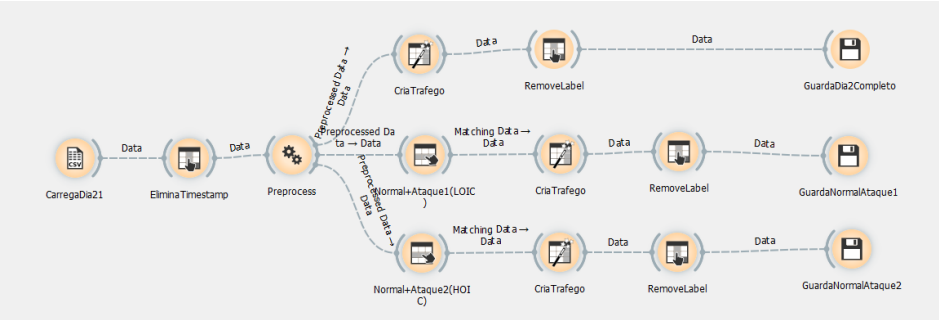

Figura 19: Orange - Fluxo para o dia 21

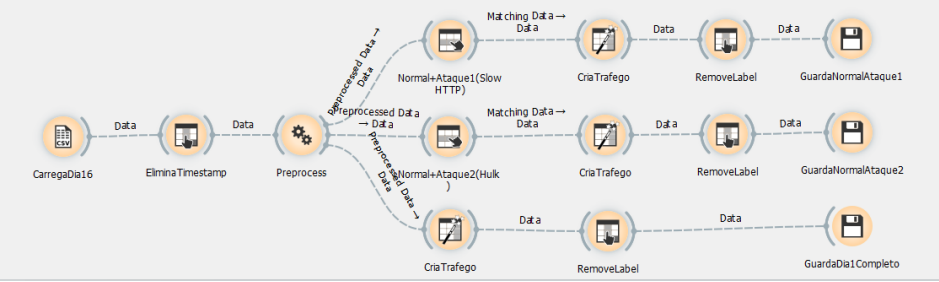

Figura 20: Orange - Fluxo para o dia 16

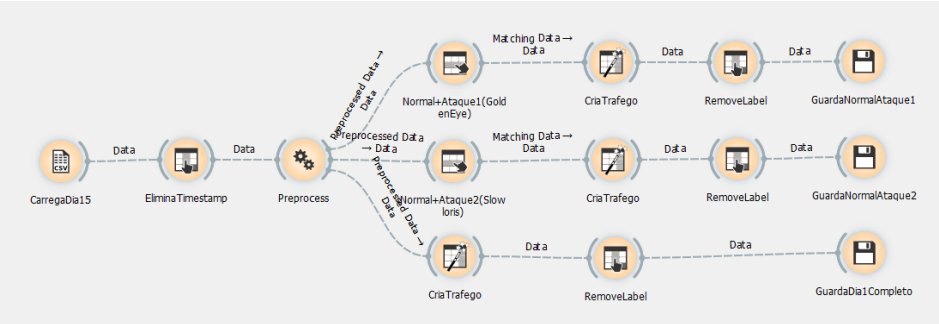

Figura 21: Orange - Fluxo para o dia 15

B.2 WEKA

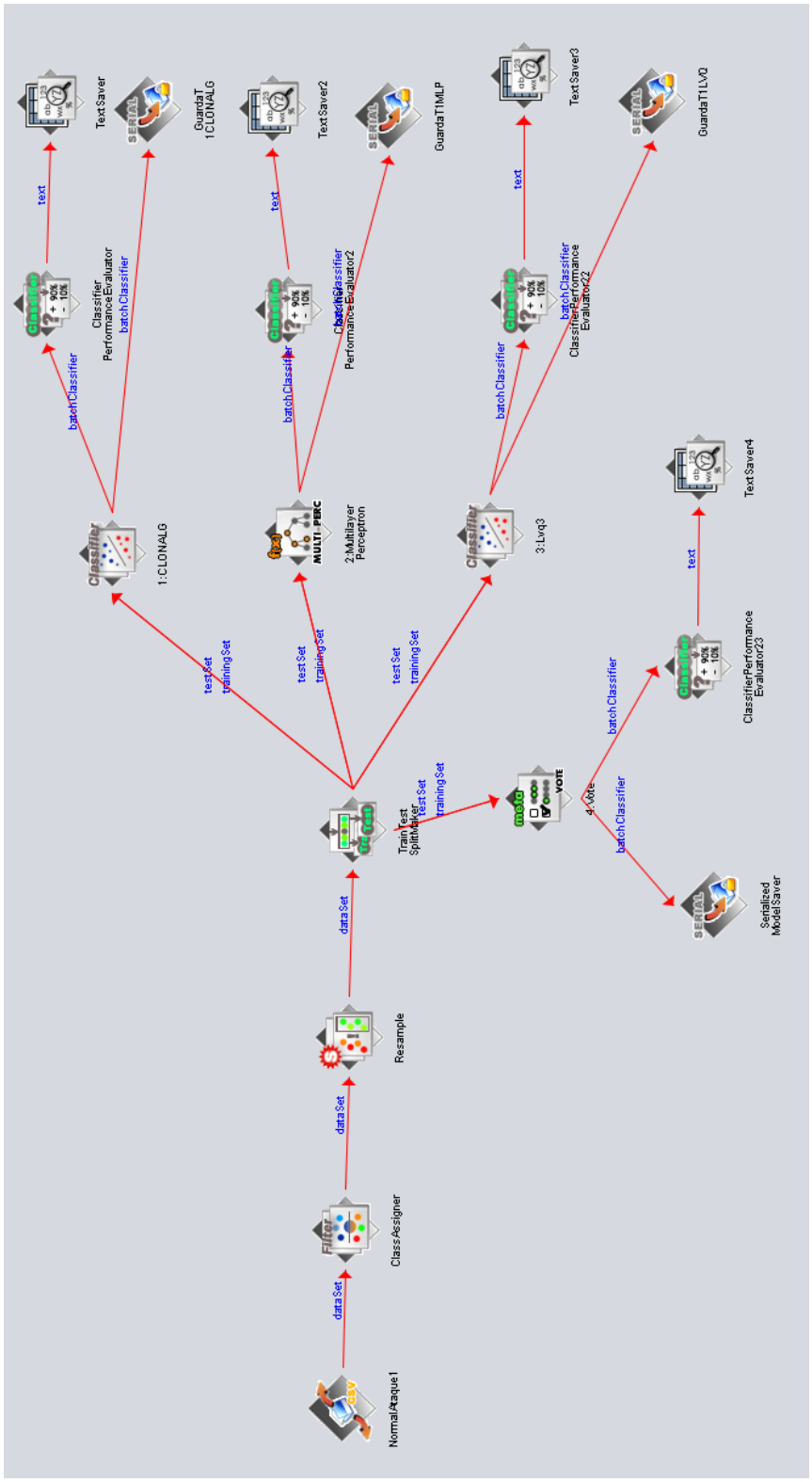

Figura 22: Weka - Fluxo de treino dos modelos

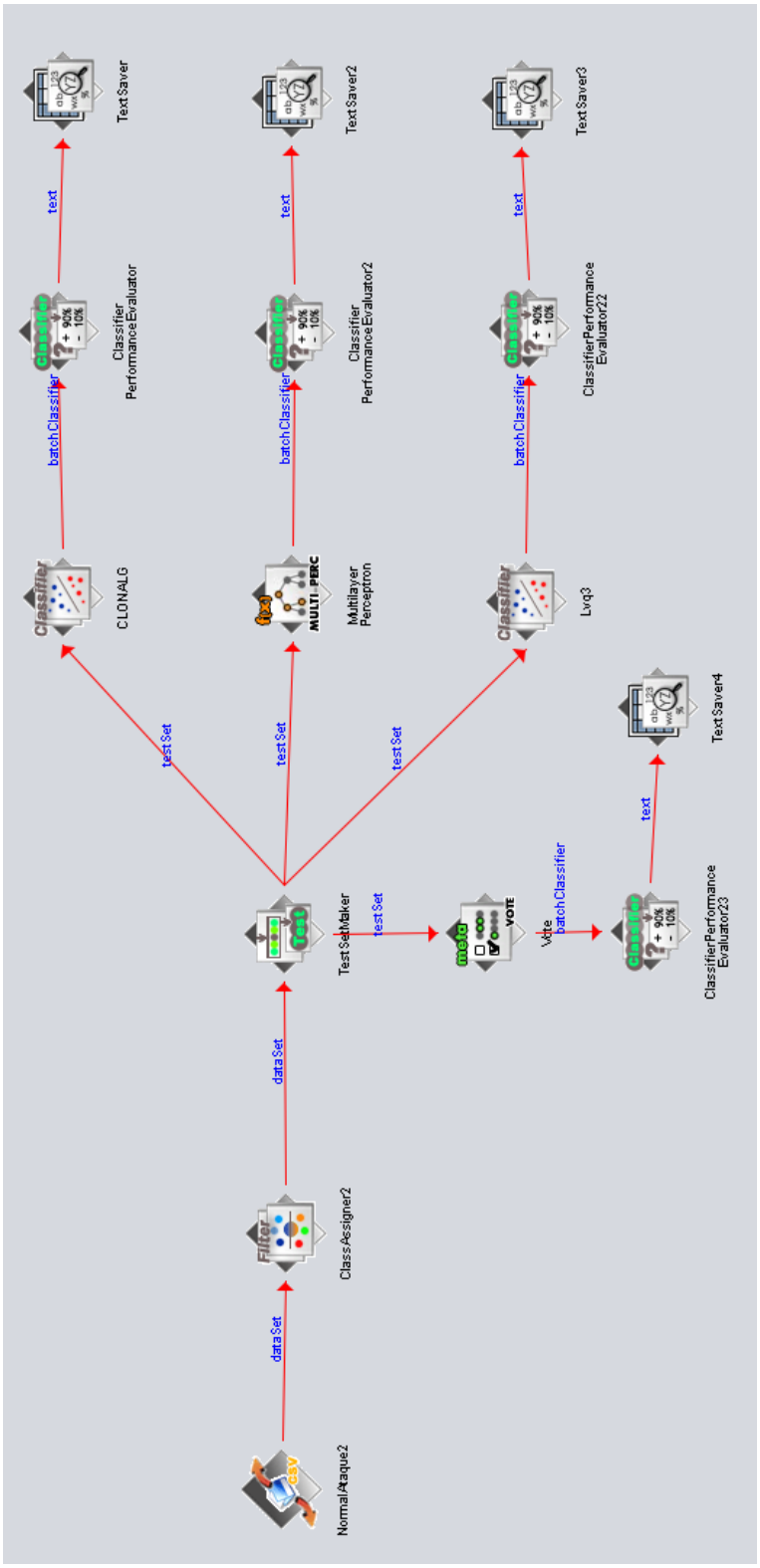

Figura 23: Weka - Fluxo de teste dos modelos

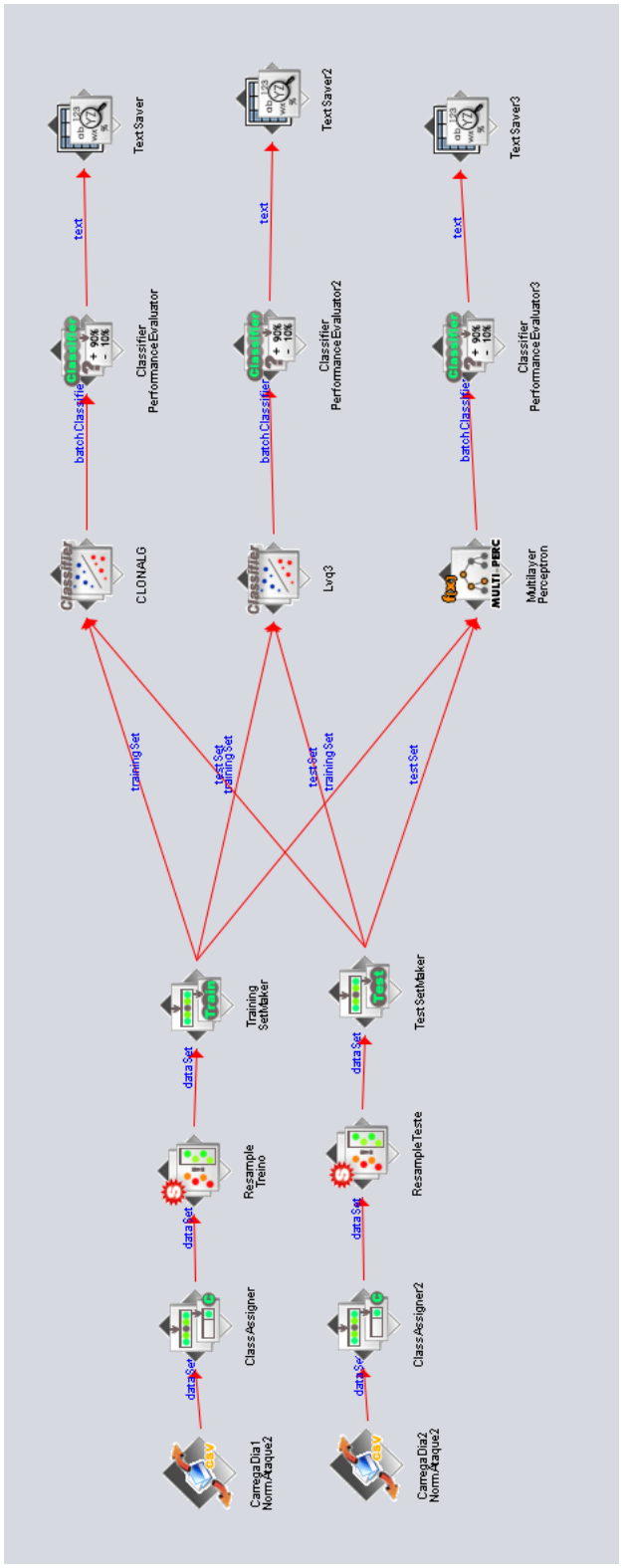

Figura 24: Weka - Fluxo para teste de variação de parâmetros

C

TABELAS DE RESULTADOS

C.1 PARAMETRIZAÇÃO INICIAL - RESULTADOS

Zero-Day Attack

	True Positive Rate	False Positive Rate	True Negative Rate	False Negative Rate	Precision	Accuracy	Recall	F1
CLONALG1	0,0240	0,0001	0,9999	0,9760	0,9957	0,4398	0,0240	0,0468
CLONALG2	0,0235	0,0002	0,9998	0,9765	0,9946	0,4395	0,0235	0,0458
CLONALG3	0,0235	0,0001	0,9999	0,9765	0,9959	0,4395	0,0235	0,0458
Ensemble	0,0000	0,0000	1,0000	1,0000	#DIV/0!	0,4261	0,0000	0,0000
BackMLP1	0,9979	0,0002	0,9998	0,0021	0,9998	0,9987	0,9979	0,9988
BackMLP2	0,9974	0,0007	0,9993	0,0026	0,9994	0,9982	0,9974	0,9984
BackMLP3	1,0000	0,0002	0,9998	0,0000	0,9998	0,9999	1,0000	0,9999
Ensemble	0,9979	0,0002	0,9998	0,0021	0,9998	0,9987	0,9979	0,9988
LVQ3_1	0,7485	0,0002	0,9998	0,2515	0,9998	0,8556	0,7485	0,8561
LVQ3_2	0,7251	0,0000	1,0000	0,2749	1,0000	0,8422	0,7251	0,8406
LVQ3_3	0,0240	0,0002	0,9998	0,9760	0,9953	0,4398	0,0240	0,0468
Ensemble	0,7485	0,0002	0,9998	0,2515	0,9998	0,8556	0,7485	0,8561

Figura 25: Parametrização inicial - Resultados

D E C L A R A Ç Ã O

Declaro, sob compromisso de honra, que o trabalho apresentado nesta dissertação, com o título *"Benchmark de Sistemas de Detecção de Intrusões Baseados em Comportamento com recurso a algoritmos bioinspirados"*, é original e foi realizado por Paulo Jorge Gomes Ferreira (2180047) sob orientação de Professor Doutor Mário João Gonçalves Antunes.

Leiria, Setembro de 2020

Assinado por : **PAULO JORGE GOMES FERREIRA**
Num. de Identificação: BI089010418
Data: 2020.09.20 18:55:10+01'00'

CARTÃO DE CIDADÃO

Paulo Jorge Gomes Ferreira