INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Sistema inteligente de apoio à deteção e previsão de cyber ataques em sistemas computacionais

JORGE ALEXANDRE NOVO MEIRA
Outubro de 2017

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Sistema inteligente de apoio à deteção e previsão de cyber ataques em sistemas computacionais

Jorge Meira

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas de Informação e Conhecimento

Orientador: Prof. Goreti Marreiros
Co-Orientador: Prof. Isabel Praça

Júri:
Presidente:

Vogais:

Porto, 21 de Outubro de 2017

Resumo

A segurança de informação tornou-se um aspeto muito importante de qualquer sistema de informação organizacional. As ameaças em sistemas computacionais têm-se revelado cada vez mais inteligentes permitindo contornar as soluções básicas de segurança como *firewalls* e antivírus. Lidar com vulnerabilidades de software em sistemas computacionais tornou-se um grande desafio para os administradores de sistemas. Com o crescente número de vulnerabilidades descobertas em cada ano, é impossível para os administradores manter os sistemas livres de ações maliciosas.

Os Sistemas de Deteção de Intrusão (SDI) baseados em anomalias, permitem classificar o tráfego monitorizado de uma rede ou operações de um *host* em atividades normais ou atividades maliciosas. A eficiência da deteção de intrusões irá depender das técnicas utilizadas nestes sistemas.

Este trabalho propõe a aplicação de técnicas *machine learning* num SDI a desenvolver no âmbito do projeto SASSI (ANI | P2020 17775). O objetivo principal do trabalho consiste na exploração e aplicação de metodologias para deteção e previsão de cyber ataques em sistemas computacionais, de modo a oferecer apoio na tomada de decisão a administradores de sistemas garantindo assim estabilidade e segurança nos sistemas de informação organizacionais.

Palavras-chave: anomalias, vulnerabilidades, intrusões, segurança, *machine learning*, classificadores, SDI

Abstract

Information and data security has become a very important part of any organizational information system. Threats in these systems have become increasingly intelligent, so it can deceive the basic security soluions such as firewalls and antivirus. Dealing with software vulnerabilites in computer systems became a major challenge for system administrators. With the increasing number of vulnerabilites discovered each year, it is impossible for administrators to maitain software on their machines, free of malicious actions.

Anomaly based Intrusion Detection Systems (IDS), allows monitorized traffic classification or computer systems classification in normal activity or malicious activity. The efficiency of intrusion detection depends on the techniques used in these systems.

This work proposes the application of machine learning techniques in IDS to be developed in the scope of SASSI project (ANI | P2020 17775). The main goal of this work is to explore and apply different methodologies to detect and predict cyber attacks in computer systems so it can give support in decision-making to system administrators and guarantee stability and security in organizational information systems.

Agradecimentos

O trabalho que é apresentado neste documento é o resultado de um caminho que não seria possível percorrer sozinho. Nesta caminhada houve um conjunto importante de pessoas que se cruzaram comigo e me ajudaram a alcançar os objetivos definidos.

Gostaria de agradecer à professora Goreti Marreiros, pela sua orientação, motivação e pela oportunidade que me proporcionou de ingressar no centro de investigação GECAD para desenvolvimento do projeto SASSI. Agradeço também à professora Isabel Praça (co-orientadora), pelas sugestões e contribuições neste projeto, como também à professora Fátima Rodrigues pela ajuda disponibilizada em problemas relacionados com a área de *machine learning*.

Agradeço a todos os meus colegas do GECAD, à Marta pela partilha do seu conhecimento científico na área de *machine learning*, ao Luís e ao João pelo auxílio na elaboração deste documento. Aos restantes colegas, agradeço pelo ambiente de trabalho agradável que me proporcionaram.

Agradeço a todos os meus familiares, especialmente aos meus avós, pelo apoio que direta ou indiretamente me deram durante a realização deste trabalho.

Agradeço à minha companheira que seguiu todo este processo incentivando-me e transmitindo confiança na realização desta dissertação.

Um muito obrigado aos meus irmãos, a quem me acompanham em todos os momentos importantes da minha vida e aos meus pais pela educação e transmissão de valores, pelo apoio prestado e principalmente pelos grandes sacrifícios realizados para que eu conseguisse atingir este objetivo.

Gostaria de agradecer a todas as pessoas que, de alguma forma, deram o seu contributo para a realização deste trabalho, assim como ao projeto SASSI (POCI-01-0247-FEDER-017775).

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ARP	*Address Resolution Protocol*
ART	*Fuzzy ARTMAP*
AUC	Área Sob a Curva
CIFD	*Common Intrusion Detection Framework*
CISUC	Centro de Informação e Sistemas da Universidade de Coimbra
CRAN	*Comprehensive R Archive Network*
DARPA	*Defense Advanced Research Projects Angency*
DDoS	*Distributed Denial of Service*
DoS	*Denial of Service*
ESA	Agência Espacial Europeia
EUMETSAT	Organização Europeia para Exploração de Satélites Meteorológicos
FAR	Rácio de Falso Alarme
FEEI	Fundos Europeus Estruturais e de Investimento
GAU	Classificador *Cluster Gaussian*
GECAD	*Research Group on Intelligent Engineering and Computing for Advanced Innovation adn Development*
HYP	*Hypersphere*
I&D	Inovação e Desenvolvimento
IDE	*Ambiente de Desenvolvimento Integrado*
IF	Igual Frequência
IRBF	Função Incremental *Radial Basis*
ISEP	Instituto Superior de Engenharia do Porto
K-M	*K-means clustering*
KDD	*Knowledge Discovery and Data Mining*
KNN	K Nearest Neighbor
LEA	Algoritmo *Leader*
MAC	*Media Acess Control*
MLP	Multi-layer Perceptron

MSE	Erro Quadrático Médio
NCD	*New Concept Development*
NEA	Algoritmo *Nearest cluster*
PD	Probabilidade de Deteção
R2L	Remoto para local
RFE	Recursive Feature Elimination
SDI	Sistemas de Deteção de Intrusão
SME	*Small or Medium Enterprise*
SVM	*Support Vector Machine*
TIC	Tecnologias de Informação e Comunicação
U2R	Utilzador para Root
UC	Universidade de Coimbra
VAE	*Autoencoder Varicional*

Capítulo 1

Introdução

1.1 Enquadramento

Os sistemas informáticos assumem um papel preponderante no mundo atual. A existência de uma grande dependência destes sistemas tanto a nível pessoal como empresarial salienta a importância da segurança da informação nos dias de hoje. Garantir a confidencialidade, integridade e disponibilidade dos recursos computacionais, tem sido um grande desafio. Estes sistemas estão sujeitos a vários tipos de ataques que podem ter origem interna ou externa, alguns com o objetivo de paralisar serviços, outros com a intenção de roubar informações (Ferreira et al. 2011).

Uma solução para melhorar a segurança dos sistemas computacionais é a utilização de Sistemas de Deteção de Intrusão (SDI). Um SDI é desenhado para complementar outras medidas de segurança baseadas em prevenção de ataques (Zargar, Baghaie et al. 2012). Alonso-Betanzos et al. (2007) defende que o objetivo de um SDI é informar o administrador de sistemas da existência de atividades suspeitas e recomendar ações específicas para prevenir ou parar a intrusão. Existem dois tipos de SDI, sistemas de deteção de intrusões baseado em anomalias e sistemas de deteção de intrusões por assinaturas que tentam comparar atividades maliciosas anteriormente conhecidas nas bases de dados. Este último contém a desvantagem de só conseguir detetar ataques que são conhecidos (Alonso-Betanzos et al. 2007). SDI baseados em anomalias utilizam técnicas de identificação de padrões para classificar as operações em sistemas computacionais, como por exemplo operação normal ou operação anómala. Maioritariamente são utilizadas abordagens de *machine learning*, ou seja, são utilizados algoritmos que iterativamente aprendem com os dados. Na literatura é possível encontrar abordagens a SDI baseadas em redes neuronais (Mukkamala, Andrew H. Sung e Abraham 2005), *Support Vector Machine* (SVM) e variantes (Fugate e Gattiker 2003).

Neste projeto serão explorados e aplicados vários algoritmos assentes em técnicas de descoberta de conhecimento que permitam detetar e prever comportamentos intrusivos. O objetivo deste estudo permitirá selecionar uma ou várias técnicas a serem implementadas num SDI. Estas tarefa faz parte do projeto SASSI. Trata-se de um projeto de Inovação e Desenvolvimento (I&D) em co-promoção cujo o consórcio é composto por uma *Small or Medium Enterprise* (SME), a VisionTechLab, pelo Instituto Superior de Engenharia do Porto (ISEP), e pela Universidade de Coimbra (UC). O objetivo principal do projeto SASSI consiste no desenvolvimento de um SDI baseado em anomalias com uma plataforma web que centraliza, estrutura e permite a visualização relativa à atividade de redes informáticas e das máquinas que nela estão instaladas (VisionSpace 2015). Para uma melhor perceção e compreensão do modelo de negócio e do valor do projeto SASSI, foi feita uma análise

de valor onde é apresentado e explicito os cinco elementos chave do modelo *New Concept Model* de Koen et al. (2001). Estes são a "Identificação da oportunidade", "análise da oportunidade", "geração de ideias", "seleção da ideia"e "conceito e desenvolvimento". Foi apresentado também um estudo dos conceitos de valor, valor percebido e valor para o cliente, de modo a realizar uma perspetiva longitudinal dos benefícios e sacrifícios do cliente. Por último foi estruturado o modelo de negócio Canvas onde é apresentado a proposta de valor da VisionTechLab.

As abordagens de Sabhnani, Serpen e More (2003), Tavallaee et al. (2009), Jinwon e Sungzoon (2015) e Yassin, Udzir e Muda (2013), apresentam análises realizadas em conjuntos de dados disponíveis na Web aplicando técnicas de *machine learning*. A importância deste estudo consiste na análise da avaliação do desempenho dos algoritmos aplicados nos conjuntos de dados com a presença de anomalias simuladas num sistema. Através desta análise foi permitido selecionar algumas técnicas de *machine learning* como também conjuntos de dados que contêm intrusões para a elaboração de casos de estudo. Os testes realizados permitiram comparar o desempenho dos classificadores para a verificação da existência de um modelo preditivo a ser implementado num SDI.

1.2 Principais Objetivos

No âmbito do projeto SASSI pretende-se criar uma ferramenta que apoie a tomada de decisão para os administradores de redes informáticas que se deparam com problemas de segurança cada vez mais robustos e sofisticados. Será desenvolvido um SDI baseado em anomalias e uma plataforma web que centraliza, estrutura e permite a visualização relativa à atividade de redes informáticas e das máquinas que nela estão instaladas. Este trabalho de tese enquadra-se no projeto SASSI sendo o seu principal objetivo o desenvolvimento de um modelo de deteção e previsão de falhas de sistema, ou seja, no desenvolvimento de um modelo contendo algoritmos assentes em técnicas de descoberta de conhecimento que permitam detetar, de forma autónoma, comportamentos intrusivos ou atividades que comprometam o funcionamento seguro do sistema. Para concretizar este objetivo foi necessário realizar as seguintes tarefas:

- Estudo sobre a segurança da informação, vários tipos de ciberataques existentes e o impacto dos mesmos na sociedade;
- Estudo relacionado com os SDI, componentes que constituem este tipo de sistemas e o seu funcionamento;
- Estudo do tema *machine learning*, exploração e descrição de diferentes técnicas de pré-processamento e algoritmos de classificação;
- Estudo de diferentes abordagens na literatura onde se aplicam várias técnicas de *machine learning* na deteção de intrusões em conjuntos de dados públicos;
- Aplicação de várias modelos preditivos em conjuntos de dados públicos;
- Comparação e avaliação do desempenho dos diferentes modelos preditivos;

1.3 Estrutura

Esta tese é composta por seis capítulos organizados da seguinte forma:

No primeiro capítulo é apresentada uma breve descrição do presente trabalho, incluindo os principais objetivos e estrutura.

Capítulo 2 apresenta uma contextualização do problema, um estudo feito na literatura com os tipos de ataques a sistemas computacionais, tecnologias de deteção de ataques e abordagem ao tema de *Machine Learning*. Foram também apresentados estudos realizados por Sabhnani, Serpen e More (2003), Tavallaee et al. (2009), Jinwon e Sungzoon (2015) e Yassin, Udzir e Muda (2013), onde se aplicaram vários algoritmos para deteção de intrusões em conjuntos de dados públicos. Foi também apresentado o sistema do projeto SASSI, foi descrita a solução proposta a ser desenvolvida no âmbito desta dissertação como também apresentada uma abordagem alternativa. Por fim foi realizado a análise de valor do presente projeto.

No capítulo 3 são descritas todas as técnicas implementadas na deteção de intrusões utilizando os conjuntos de dados públicos NSL-KDD e ISCX 2012.

No capítulo 4 são mencionadas as métricas de avaliação utilizadas para avaliar o desempenho de algoritmos de *Machine Learning*. São também apresentadas os resultados obtidos e analises realizadas ás diferentes técnicas aplicadas nos conjuntos de dados NSL-KDD e ISCX para deteção de intrusões.

O capitulo 5 finaliza com a conclusão do trabalho realizado nesta dissertação.

Capítulo 2

Anomalias e Sistemas de Deteção de Intrusão

2.1 Introdução

A evolução tecnológica e a facilidade de acesso à informação proporcionou uma enorme crescente de ameaças aos sistemas computacionais. Há aproximadamente 40 anos atrás, os computadores funcionavam em ambientes fechados. Não existia um conhecimento público sobre os sistemas e o acesso era restrito o que levava a que o nível de risco fosse baixo (Sutton 2008). Incrementalmente, foi-se transferindo parte da computação para sistemas pessoais como os conhecidos PC's.

Anos mais tarde, os computadores pessoais evoluíram para realizar cada vez mais tarefas de trabalho. Os utilizadores adquiriam conhecimento sobre estas máquinas e sistemas enquanto se vulgarizava o uso do modelo cliente-servidor. Surgiram novos riscos e a proteção era pouca. O uso de redes começou a ser omnipresente e a dependência das tecnologias informáticas tornou-se maior tanto a nível pessoal como empresarial (Cale 2012).

Nos dias de hoje, há cada vez mais tipos de ataques informáticos que são executados em grande número onde a sociedade, as organizações e as nações são afetadas. Um desses exemplos foi a empresa de advogados Mossack Fonseca no Panamá, onde foram divulgados documentos que revelaram a utilização de paraísos fiscais com o objetivo de ocultar os rendimentos de pessoas e empresas em todo o mundo. Conhecido como o escândalo dos "Papéis do Panamá"ou "Panamá *Papers*"resultou na publicação de mais de 11.5 milhões de documentos da base dados "pirateada"da Mossack Fonseca. Estes documentos permitiram "descortinar o mundo das transações e das empresas fachada nos paraísos fiscais, implicando líderes mundiais e personalidades conhecidas"(DN 2016).

Em Outubro de 2016 foram atacados indiretamente vários web-sites de organizações conhecidas como o Twitter, Sotify, Reddit, SoundCloud, Paypall, entre outros. Todas estas empresas são clientes de uma organização designada por Dyn que tem a função de auxiliar os utilizadores a encontrar os web-sites referidos. Dyn foi submetida a dois ataques que fez com que os web-sites dos seus clientes fossem difíceis de serem alcançados por parte dos utilizadores (BBC News 2016). A Dyn acabou por reportar na sua página que foi alvo de um ataque do tipo *Distributed Denial of Service* (DDoS) que será explicado neste capítulo.

Um dos ataques mais falado recentemente influenciou as eleições referentes ao novo presidente dos Estados Unidos. FBI, NSA e CIA assinaram um relatório público que revela, com algum detalhe, o envolvimentos da Rússia nas eleições presidenciais dos Estados Unidos que

tiveram como desfecho a vitória de Donald Trump (ICA 2017). No relatório é apresentada a certeza de que Vladimir Putin, presidente da Rússia, ordenou uma campanha de influência, em 2016, com vista à eleição presidencial dos Estados Unidos com o objetivo de denegrir a imagem de Hillary Clinton fornecendo ao WikiLeaks emails "roubados"da sua conta. Por questões de segurança foram ocultados os detalhes de como ocorreram os ataques e as suas consequências (Lecher 2017).

2.2 Princípios da Segurança de Informação

Para que se possa garantir segurança num sistema de informação de forma a reduzir os riscos de erros, fraudes, vazamento, roubo de informação ou qualquer outra ameaça que possa prejudicar os sistemas e/ou equipamentos de um indivíduo ou organização, será importante abordar os princípios da segurança da informação: (Jawandhiya et al. 2010)

- **Confidencialidade** - Consiste em proteger a informação contra indivíduos, entidades ou processos que não possuem permissão para aceder à mesma.
- **Integridade** - Consiste em impedir que a informação seja modificada ou corrompida por modos não autorizados.
- **Disponibilidade** - Consiste em manter os sistemas em funcionamento impedindo a existência de distúrbios na produtividade dos mesmos. Os serviços têm que se encontrar disponíveis sempre que requisitados.
- **Autenticidade** - Assegurar que uma entidade de interesse ou a origem/destino de uma comunicação é quem afirmar ser. Cada interveniente é autentico para o outro.
- **Não repúdio** - Garantir que cada interveniente não negue ter assinado ou criado informação. O não repúdio fornece provas de que um utilizador realizou uma determinada ação.

2.3 Tipos de Anomalias

Ataques em redes ou sistemas de informação podem ser divididos em duas categorias. **Passivos** quando um intruso interseta dados numa comunicação que estão a ser trocados por entidades numa rede. **Ativos** quando existem tentativas de criação, alteração ou eliminação de dados numa comunicação provocando distúrbios no funcionamento normal de uma rede (Pawar e Anuradha 2015). Dentro destas duas categorias, os ataques podem ser **externos** onde o atacante tem o objetivo de congestionar e propagar informação falsa ou causar distúrbios nas operações da rede, através de nós (pontos de conexão numa rede como por exemplo: *routers*, computadores, *switch*,etc) que não fazem parte da rede. Ataques **internos** onde o intruso pretende ganhar acesso e participar nas atividades da rede através de nós comprometidos que fazem parte da mesma (Jawandhiya et al. 2010). Alguns exemplos dos ataques passivos e ativos mais utilizados por intrusos serão abordados de seguida:

2.3.1 Ataques Passivos

Análise de Tráfego

No ataque de análise de tráfego um intruso tenta identificar o caminho de comunicação entre o emissor e o recetor. Este consegue captar uma quantidade de dados que é trocada na comunicação entre várias entidades. Não existe nenhuma modificação nos dados através deste ataque (Pawar e Anuradha 2015).

Espionagem ou Sniffer

O objetivo deste ataque consiste em procurar informação secreta ou confidencial numa comunicação intersetada. Cada pacote intersetado pode ser analisado *bit* a *bit* com o propósito de detetar informação sensível como por exemplos dados de autenticação. Este ataque ocorre normalmente em redes *ad hoc* que são redes em que todos os terminais funcionam como *routers*, encaminhando de forma comunitária as comunicações advindas dos terminais vizinhos (Jawandhiya et al. 2010).

2.3.2 Ataques Ativos

Spoofing

Quando um intruso personifica ou rouba a identidade de um nó para esconder a sua. Desta forma o intruso consegue espalhar *software* malicioso, roubar informação, ignorar os controladores de acesso, através da sua camuflagem num dispositivo ou utilizador de confiança que se encontra na rede. Os ataques *spoofing* mais utilizados são:

- *IP Spoofing* - O atacante altera o seu IP e utiliza o IP de um nó legítimo numa rede podendo assim enviar mensagens à sua vítima através de uma fonte genuína. Para realizar este ataque o intruso, primeiro inicia uma variedade de técnicas para encontrar um endereço de IP legítimo e de seguida modifica o cabeçalho de modo a que os pacotes indiquem que estão a ser enviados por uma fonte fidedigna (DuPaul 2016).
- *ARP Spoofingl* - *Address Resolution Protocol* (ARP), é um protocolo que é utilizado para associar endereços de IP a endereços *Media Acess Control* (MAC). Neste ataque o intruso envia mensagens ARP para uma rede local, fazendo-se passar por uma fonte fidedigna, com o propósito de ligar o endereço MAC do intruso a um endereço de IP de um membro legítimo na rede. Esta técnica permite que o intruso receba mensagens que eram intencionadas para o membro do endereço de IP roubado (DuPaul 2016).
- *Email Spoofingl* - Neste ataque o intruso altera o cabeçalho de email para que este indique que o email está a ser enviado por uma fonte genuína. É uma técnica utilizada em ataques *phishing* ou campanhas de spam pois a vítima tem tendência a abrir emails de fontes legítimas (Rouse 2016).
- *URL Spoofingl* - É o processo de criar um URL falso que personifica um web-site seguro e legítimo. Na verdade, este URL está a redirecionar o tráfego todo para

um *website* falso com intenção de obter informações confidenciais sobre a vítima (SecuritySupervisor 2014).

Modificação

Os intrusos realizam algumas mudanças aos pacotes que estão a ser partilhadas numa comunicação pondo em causa a sua integridade. Este ataque acontece por exemplo numa das fases do ataque designado por "*Men-in-the-middle*", em que o nó intruso coloca-se no meio da comunicação entre dois nós através da exploração de vulnerabilidades nas relações esporádicas de uma rede. O intruso tenta participar no processo de encaminhamento de pacotes com a finalidade de executar o ataque de modificação de pacotes sem que as vítimas se apercebam (Jawandhiya et al. 2010).

Fabricação

Este ataque utiliza o mesmo método que o ataque de modificação. Em vez de modificar pacotes existentes numa comunicação, os intrusos fabricam e enviam pacotes com fins maliciosos de forma a causar o caos nas operações da rede (Jawandhiya et al. 2010).

Denial of Service (DoS)

Neste ataque um nó é utilizado para "inundar"um servidor com pacotes que utilizam o protocolo TCP/UDP. O objetivo é sobrecarregar a largura de banda do servidor alvo, de modo a que este se torne inacessível a outros utilizadores (Munson 2012).

Distributed Denial of Service (DDoS)

É um ataque semelhante ao *Denial of Service* (DoS) mas em vez de utilizar apenas um nó são utilizados vários nós para sobrecarregar a comunicação do servidor alvo. Este é normalmente um ataque externo em que os nós estão distribuídos por todo o mundo. Este conjunto de nós são conhecidos por "*botnet*"que constituem computadores infetados por *software* malicioso "*bots*"que se espalham de forma autónoma aproveitando vulnerabilidades que podem ser exploraras remotamente, sendo estes controlados por um ou vários atacantes (Munson 2012) como representado na figura 2.1.

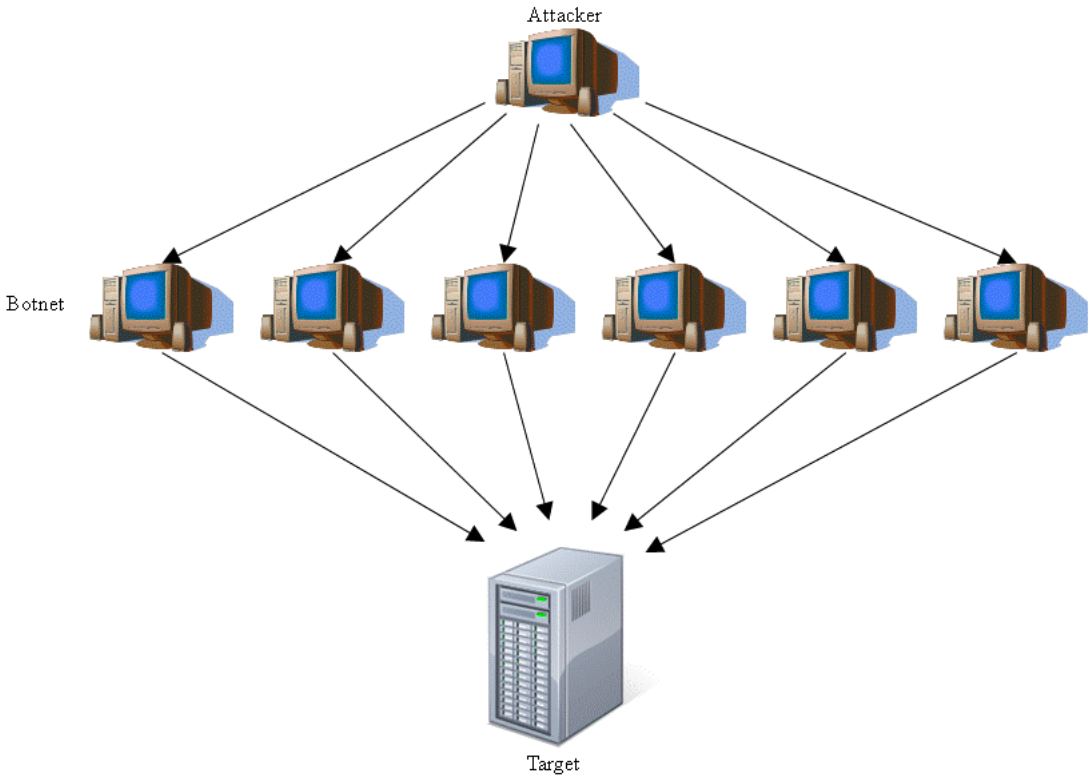

Figura 2.1: Ataque Negação de Serviço Distribuído (Jain's 2011)

Phishing

Consiste no envio de mensagens que não foram solicitadas personificando informação legítima com o intuito de induzir o utilizador a fornecer dados pessoais através do preenchimento de formulários falsos ou na instalação de *software* malicioso (OWASP 2009).

Repudiação

Repudiação refere-se ao ato de negar uma determinada ação. Por exemplo um indivíduo negar a operação de uma compra ou transação online. Este ataque acontece quando existem vulnerabilidades nos controladores de acesso, permitindo assim a manipulação maliciosa da informação como por exemplo adulterar a identificação da execução de novas ações (OWASP 2012).

Backdoor

Através da instalação de um *software* malicioso no dispositivo da vítima, é criado uma porta que permite ao atacante adquirir permissões administrativas para controlar o dispositivo remotamente. O *backdoor* trabalha em segundo plano e esconde-se do utilizador. Desta forma o atacante consegue espiar, gerir ficheiros, obter informação confidencial, controlar todo o sistema operativo da vítima e atacar outros anfitriões (Simsolo 1998).

2.4 Aplicação de Machine Learning em Sistemas de Deteção de Intrusão

Um SDI é uma ferramenta de segurança, que como outras medidas existentes, por exemplo antivírus, *firewall*, controladores de acesso, tem a finalidade de fortalecer a segurança nos sistemas de informação e comunicação (García-Teodoro et al. 2009). Estes sistemas normalmente apresentam-se divididos em duas categorias: os baseados em assinaturas e os baseados em anomalias (Allen et al. 2000).

Os SDI baseados em assinaturas, identificam ataques que seguem padrões previamente reconhecidos e reportados por especialistas. Cada assinatura é utilizada para identificar o respetivo ataque e tem como desvantagem o facto de apenas permitir a identificação de ataques conhecidos, por isso, é necessário uma constante atualização das assinaturas.

Nos SDI baseados em anomalias, estes monitorizam as operações que normalmente ocorrem numa rede e as operações do sistema de cada dispositivo. Quando se verifica a existência de um comportamento que não segue o padrão esperado, uma sinalização de anomalia é gerada no sistema.

Os sistemas de deteção de intrusão podem ser utilizados para monitorizar e detetar intrusões em redes (*Network Intrusion Detection System* - NIDS, sigla em inglês), ou em hosts/anfitriões (*Host Intrusion Detection System* - HIDS, sigla em inglês).

Um trabalho notável foi desenvolvido pelo *Common Intrusion Detection Framework* (CIFD), um grupo criado em 1998 pela *Defense Advanced Research Projects Angency* (DARPA), e que permitia a definição de uma estrutura de uma framework comum no domínio do SDI. O grupo definiu uma arquitetura geral de um SDI baseada em quatro módulos (García-Teodoro et al. 2009) representada na figura 2.2:

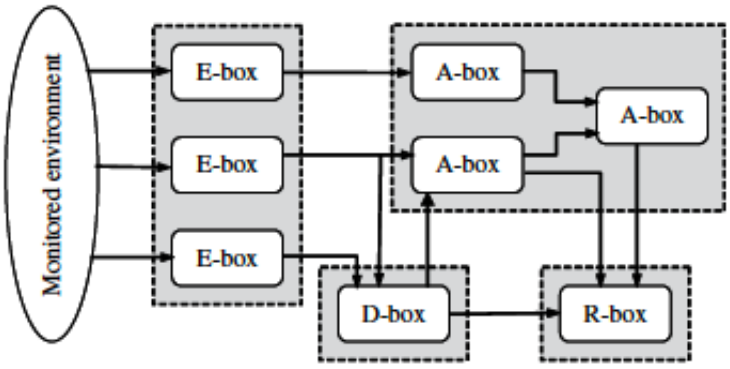

Figura 2.2: Arquitetura geral de um SDI (García-Teodoro et al. 2009)

- Bloco E (Caixa de Eventos) - Este tipo de bloco é composto por sensores que monitorizam o sistema alvo, adquirindo assim eventos de informação que serão analisados por outros blocos.
- Bloco D (Caixa de base de dados) - Estes são elementos destinados a armazenar as informações obtidas do bloco E para mais tarde serem processadas por os Blocos A e R.

- Bloco A (Caixa de análise) - Modulo que processa os dados dos eventos guardados no bloco D identificando possíveis comportamentos anormais de modo a que seja gerado um tipo de alarme para esse evento se necessário.
- Bloco R (Caixa de resposta) - A função principal deste bloco é a execução de uma instrução caso haja a ocorrência de uma intrusão, de moda a prevenir a ação da ameaça detetada.

Na arquitetura dos SDI definida por CIFD é identificada uma componente de análise e processamento de dados para deteção de anomalias. Para o processamento desses dados normalmente são utilizadas abordagens com técnicas de *machine learning*. O processo de *machine Learning* baseia-se no estabelecimento de um modelo implícito ou explícito que permite a análise e classificação de padrões. São utilizados algoritmos que iterativamente aprendem com os dados. O aspeto iterativo é importante na medida em que os modelos estão expostas a novos dados e são capazes se adaptar aos mesmos de forma independente. As principais vantagens destes métodos estão na flexibilidade, adaptabilidade e capacidade de detetar interdependências desconhecidas nos dados (Perlin, Nunes e Kozakevicius 2011).

Dois dos métodos adotados em *machine learning* no âmbito geral são aprendizagem supervisionada e aprendizagem não supervisionada. 70% dos métodos utilizados são supervisionadas enquanto que 10% a 20% referem-se a aprendizagem não supervisionada. Em comparação com as técnicas de aprendizagem semi-supervisionada e de reforço, estas são menos utilizadas (SAS 2016):

2.4.1 Aprendizagem Supervisionada

Algoritmos de aprendizagem supervisionada são treinados utilizando uma legenda ou rótulo nos dados de *input* onde o *output* desejado é conhecido. Por exemplo, um conjunto de dados pode ter dados legendados que identifiquem as operações/atividades de um sistema computacional, como M (operação maliciosa no sistema) e N (Operação normal no sistema). O algoritmo recebe um conjunto de *inputs* juntamente com os seus *outputs* corretos e este aprende comparando o *output* atual com o *output* correto de forma a identificar erros. De seguida o modelo é modificado de acordo com os resultados. Através de métodos como classificação, regressão, predição e aumento de gradiente, a aprendizagem supervisionada utiliza padrões para prever os valores dos dados que não se encontram legendados. Aprendizagem supervisionada é normalmente utilizada em aplicações que utilizam dados históricos para prever eventos futuros (SAS 2016).

2.4.2 Aprendizagem Não-Supervisionada

Aprendizagem não-supervisionada é normalmente utilizada para analisar dados históricos que não estão legendados. Não é dito qual é a "resposta certa"ao sistema. O algoritmo tenta decifrar o que lhe é mostrado. O objetivo é explorar dados e encontrar um padrão nos mesmos. Este tipo de aprendizagem tem um bom funcionamento com dados de transações. Por exemplo, consegue identificar segmentos de clientes com atributos semelhantes para serem utilizados em campanhas de marketing. Ou consegue identificar os atributos principais que separam estes segmentos uns dos outros. As técnicas ou algoritmos mais aplicados são, *self-organization maps, mapping, k-means clustering e singular value decompositon*.

Estas técnicas não-supervisionadas são também utilizados para segmentar tópicos de textos, recomendação de artigos e identificação de *outliers* (SAS 2016).

2.4.3 Aprendizagem Semi-Supervisionada

Aprendizagem semi-supervisionada é utilizada para as mesmas aplicações que a aprendizagem supervisionada. Mas utiliza dados legendados e não legendados na fase de treino, normalmente uma pequena quantidade de dados legendados e uma grande quantidade de dados não legendados (pois dados não legendados requerem um menor esforço para serem adquiridos). Este tipo de aprendizagem utiliza os mesmos métodos que a supervisionada como classificação, regressão e predição. Exemplos recentes utilizados pelo *Snapchat, Facebook*, incluem a identificação de expressões faciais numa câmara (SAS 2016).

2.4.4 Reforço

Aprendizagem de reforço é normalmente utilizada na robótica, em jogos e navegação. O algoritmo identifica quais a ações mais eficientes através de tentativa erro. Este tipo de aprendizagem é constituído por três componentes primárias: o agente (o que aprende e toma decisões), o ambiente (onde o agente interage), ações (como o agente pode agir). O objetivo consiste na escolha das ações do agente de modo a proporcionarem o melhor resultado num determinado espaço de tempo (SAS 2016).

2.5 Avaliação das Diferentes Abordagens

Nesta secção exploram-se diferentes abordagens de técnicas de deteção de intrusões baseadas em *machine learning*, aplicadas em vários conjuntos de dados públicos sendo por fim, feita uma análise comparativa destas mesmas abordagens.

2.5.1 Abordagem de Sabhnani, Serpen e More

Algoritmos de *machine learning*, maioritariamente baseados em aprendizagem supervisionada, foram aplicados num conjunto de dados lançado em 1999 pela terceira competição internacional da conferencia *Knowledge Discovery and Data Mining* (KDD) (UCI Machine Learning Repository 2015). O objetivo desta competição consistia na construção de um modelo preditivo de deteção de intrusões numa rede. Este conjunto de dados inclui uma ampla variedade de intrusões simuladas numa rede de um ambiente militar. Os registos estão separados em duas tabelas, uma com dados legendados (conjunto de dados de treino) e outra com não-legendados (conjunto de dados de teste). As intrusões simuladas encontram-se separados em cinco categorias (UCI Machine Learning Repository 2015):

- **DoS**: Um intruso tenta impedir os utilizadores de usar um determinado serviço;
- **Remoto para local (R2L)**: Um intruso tenta adquirir remotamente o acesso local da máquina da vítima;
- **Utilzador para Root (U2R)**: Um intruso consegue adquirir o acesso local da máquina da vítima e tenta adquiri acesso de super-utilizador;

- **Probe**: Um intruso tenta obter informação sobre a máquina da vítima;
- **Normal**: Constitui as operações ou atividades normais na rede;

Este conjunto de dados é o mais utilizado e estudado na literatura, para avaliar SDI.

Muitos dos estudos encontrados na literatura, relativos à deteção de intrusões, indicam que os autores aplicam apenas um algoritmo para detetar ataques de várias categorias, em alguns casos com um fraco desempenho. Sabhnani, Serpen e More (2003) realizaram um estudo, onde aplicaram vários algoritmos de *machine learning* e avaliaram o seu desempenho na deteção de intrusões. Cada registo legendado no conjunto de dados KDD, contem 42 atributos, e existem perto de 5 milhões de registos legendados, que foram utilizados para treinar os algoritmos de classificação de Sabhnani, Serpen e More (2003). Uma outra tabela do conjunto de dados contém os registos não-legendados. Estes registos foram utilizados para testar os algoritmos. No seu trabalho Sabhnani, Serpen e More (2003) aplicaram nove algoritmos para comparação e avaliação do seu desempenho. Estes são Multi-layer Perceptron (MLP), Algoritmo *Nearest cluster* (NEA), Classificador *Cluster Gaussian* (GAU), Função Incremental *Radial Basis* (IRBF), *K-means clustering* (K-M), Algoritmo *Leader* (LEA), *Hypersphere* (HYP), *Fuzzy ARTMAP* (ART) e C4.5.

Tabela 2.1: Resultados do desempenho dos algoritmos (Sabhnani, Serpen e More 2003)

		Probe	**DoS**	**U2R**	**R2L**
MLP	*PD*	0.887	0.972	0.132	0.056
	FAR	0.004	0.003	5E-4	1E-4
GAU	*PD*	0.902	0.824	0.228	0.096
	FAR	0.113	0.009	0.005	0.001
K-M	*PD*	0.876	0.973	0.298	0.064
	FAR	0.026	0.004	0.004	0.001
NEA	*PD*	0.888	0.971	0.022	0.034
	FAR	0.005	0.003	6E-6	1E-4
RBF	*PD*	0.932	0.730	0.061	0.059
	FAR	0.188	0.002	4E-4	0.003
LEA	*PD*	0.838	0.972	0.066	0.001
	FAR	0.003	0.003	3E-4	3E-5
HYP	*PD*	0.848	0.972	0.083	0.010
	FAR	0.004	0.003	9E-5	5E-5
ART	*PD*	0.772	0.970	0.061	0.037
	FAR	0.002	0.003	1E-5	4E-5
C4.5	*PD*	0.808	0.970	0.018	0.046
	FAR	0.007	0.003	2E-5	5E-5

Na tabela 2.1 é possível ver a Probabilidade de Deteção (PD) e o Rácio de Falso Alarme (FAR) de cada algoritmo em cada categoria de ataque. Os resultados evidenciam que para cada categoria, certos algoritmos obtiveram um desempenho de deteção superior quando comparados com outros. MLP, NEA, GAU, K-M e IRBF detetaram mais de 85% dos ataques da categoria Probe. Para ataques em DoS os algoritmos MLP, NEA, LEA, K-M, e HYP atingiram os 97% no rácio de deteção. GAU e K-M, foram os melhores a detetar a categoria U2R com mais de 22% dos ataques registados. No caso da categoria R2L,

só o GAU conseguiu detetar por volta dos 10% dos ataques. Considerando a grandeza FAR, o MLP obteve melhor desempenho na categoria Probe, K-M na categoria DoS e U2R, e o GAU na categoria R2L. Pode-se afirmar que os algoritmos de *machine learning* testados neste conjunto de dados, oferecem um nível de desempenho aceitável para apenas duas categorias, nomeadamente a *Probe* e DoS. Por outro lado todos os 9 algoritmos de classificação falharam nos resultados que obtiveram nas outras duas categorias (U2R, R2L) (Sabhnani, Serpen e More 2003). Para a obtenção de um melhor desempenho na deteção de intrusões, Sabhnani, Serpen e More (2003), desenvolveram um multi-classificador composto por três algoritmos em que cada um deles irá detetar intrusões de uma/duas classes. Como se pode ver na figura 2.3 o algoritmo MLP foi aplicado para detetar ataques da categoria *Probe*, o *K-means* foi aplicado para a categoria DoS e U2R, enquanto que o GAU foi aplicado para a deteção de intrusões na categoria R2L.

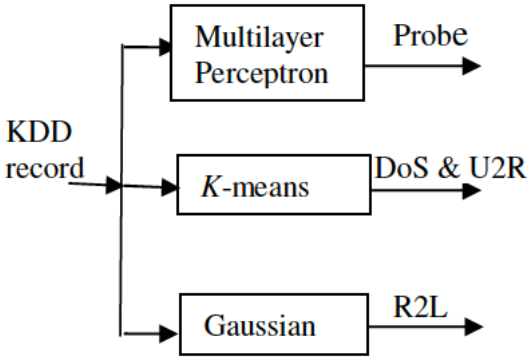

Figura 2.3: Modelo do multi-classificador (Sabhnani, Serpen e More 2003)

Na tabela 2.2, verifica-se a existência de um progresso significante na deteção de intrusões nas categorias *Probe* e U2R. Quanto ao FAR, a melhoria foi razoavelmente pequena em todas as categorias. Este classificador conseguiu obter melhores resultados que o algoritmo utilizado pelo vencedor da competição KDD e por Agarwal e Joshi (2001).

Tabela 2.2: Comparação do modelo multi-classificador com outros na literatura. Adaptado de (Sabhnani, Serpen e More 2003)

		Probe	DoS	U2R	R2L
KDD Cup Winner	*PD*	0.833	0.971	0.132	0.084
	FAR	0.006	0.003	3E-5	5E-5
Agarwal and Joshi	*PD*	0.73	0.969	0.066	0.107
	FAR	8E-5	0.001	4E-5	8E-4
Multi-Classifier	*PD*	0.887	0.973	0.298	0.096
	FAR	0.004	0.004	0.004	0.001

2.5.2 Abordagem de Tavallaee, Mahbod, Bagheri, Wei Lu e Ali A.

Tavallaee et al. (2009) Desenvolveram um novo conjunto de dados designado por NSL-KDD que resolveu os problemas que o conjunto de dados KDD 1999 continha. O conjunto de dados NLS-KDD difere do KDD 1999 nos seguintes sentidos:

- Não incluiu registos redundantes no conjunto de dados de treino, para que os algoritmos não sejam tendenciosos nos registos mais frequentes;
- Não existem registos duplicados no conjunto de dados de teste;
- Os registos foram divididos em níveis de dificuldade baseados no número de algoritmos de *machine learning* que conseguem classificar corretamente os registos. Depois foram selecionados registos aleatórios de cada nível de dificuldade numa fração que é inversamente proporcional à fração de registos distintos do conjunto de dados KDD. Desta forma, os resultados da classificação dos distintos métodos de *machine learning*, variam num intervalo maior, o que faz com que seja mais eficiente, e precisa a avaliação desses mesmos métodos;
- O número de registos no conjunto de dados de treino e de teste são razoáveis, ou seja, será mais acessível realizar experiências no conjunto completo sem a necessidade de selecionar aleatoriamente uma quantidade de registos mais pequena. Consequentemente, a avaliação de resultados de diferentes trabalhos de pesquisa será mais consistente e comparável.

Tavallaee et al. (2009) para cada conjunto de dados de teste, utilizaram os seguintes algoritmos *machine learning*: SVM, J48 (implementação do algoritmo C4.5 em *java*), *Naive-bayes, NB Tree, random forest, random Tree, MLP* e compararam a sua taxa de exatidão na deteção das anomalias:

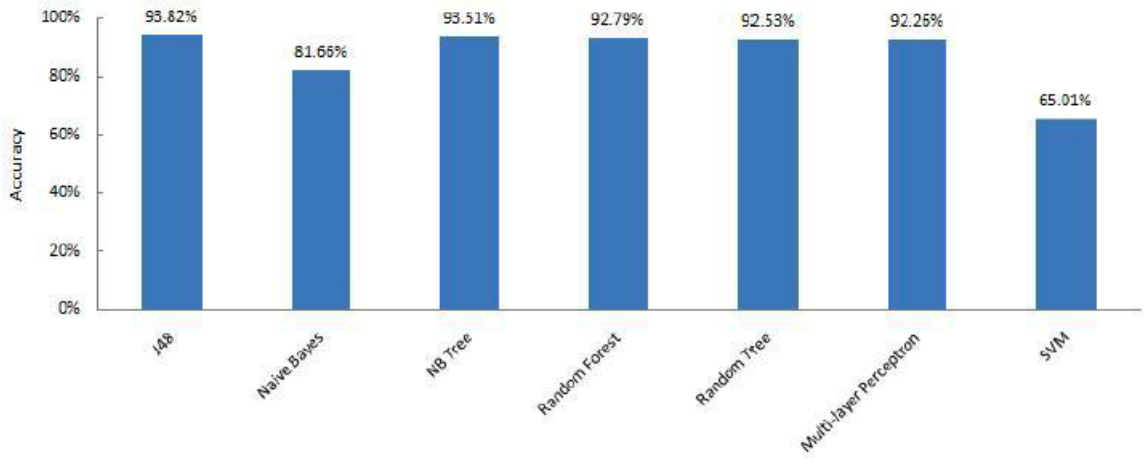

Figura 2.4: Desempenho dos algoritmos selecionados no conjunto de dados KDDTest (Tavallaee et al. 2009)

Como se pode verificar nas figuras 2.4, 2.5, os algoritmos foram aplicados no ficheiro KDDTest (conjunto de dados de teste original relativo ao KDD 1999), e no ficheiro KDDTest$^+$ (conjunto de dados de teste relativo ao NSL-KDD). Ao comparar estes dois conjuntos de dados pode-se observar que a precisão de todos os algoritmos diminui ao ser aplicado no conjunto de dados NSL-KDD de teste, com a exceção do SVM que teve uma melhoria de 4.51%.

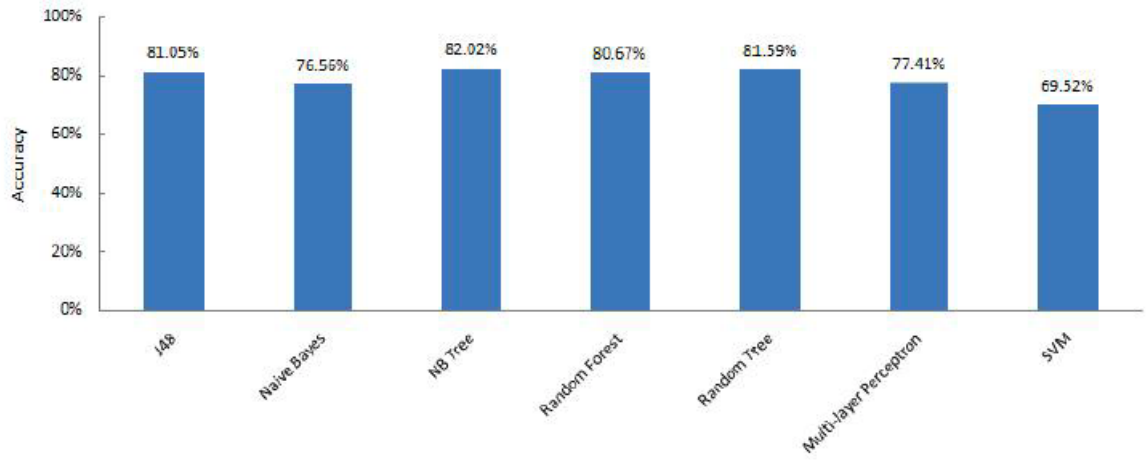

Figura 2.5: Desempenho dos algoritmos selecionados no conjunto de dados KDDTest$^+$ (Tavallaee et al. 2009)

2.5.3 Abordagem de Yassin , Warusia, Nur Izura e Zaiton

Nesta abordagem, Yassin, Udzir e Muda (2013) aplicaram um algoritmo de linguagem não supervisionada (K-Means) em conjunto com um algoritmo de aprendizagem supervisionada (Naive Bayes) num conjunto de dados desenvolvido por Shiravi et al. (2012). Este conjunto de dados designado por ISCX 2012, contém 7 dias de tráfego capturado com 4 tipos diferentes de ataques:

- Sexta (11/06/2010) – Atividade normal;
- Sábado (12/06/2010) - Atividade Normal;
- Domingo (13/06/2010) – Atividade Normal + infiltração interna na rede;
- Segunda (14/06/2010) – Atividade Normal + *HTTP Denial of Service*;
- Terça (15/06/2010) – Atividade Normal + *Distributed Denial of Service* utilizando um *Botnet IRC*;
- Quarta (16/06/2010) – Atividade Normal;
- Quinta (17/06/2010) – Atividade Normal + *Brute Force SSH*;

Como o tamanho e o conteúdo do *dataset* é demasiado grande, Yassin, Udzir e Muda (2013) selecionaram registos de pacotes de comunicação recebidos por um determinado *host* (IP - 192.168.5.122) e determinados dias como representado na tabela 2.3:

Após a criação do *dataset* de treino e de teste, Yassin, Udzir e Muda (2013) aplicaram dois métodos diferentes para a deteção de intrusões. Num dos métodos foi criado um algoritmo que consiste na junção do KMC (*K-Means Clustering*) com o algoritmo NBC (*Naive Bayes Classifier*). Inicialmente os dados são repartidos em 3 *clusters* (K = 3) e de seguida são classificados utilizando *Naive Bayes*. No outro método é apenas aplicado o classificador *Naive Bayes* para classificar as operações de rede como atividade normal ou atividade anómala. A tabela 2.4 apresenta os resultados obtidos com estes dois métodos.

Tabela 2.3: Distribuição do conjunto de dados de treino e de teste de (Yassin, Udzir e Muda 2013)

Date	Training Data		Testing Data	
	Normal	Attack	Normal	Attack
11th	0	0	147	0
12th	22612	0	0	0
14th	16260	1973	0	0
15th	0	0	19115	37159
16th	22879	0	0	0
17th	13621	181	0	0
Total	**77526**		**56421**	

Tabela 2.4: Comparação de resultados com *Naive Bayes* Vs *K-Means* + *Naive Bayes* de (Yassin, Udzir e Muda 2013)

Method	Training Data			Testing Data		
	accuracy	detection rate	false alarm	accuracy	detection rate	false alarm
NBC	82.8	13.8	17.6	88.2	85.0	33.7
KMC+NBC	99.8	95.4	0.13	99.0	98.8	2.2

Como se pode verificar na tabela 2.4, o algoritmo KMC+NBC obteve melhores resultados em todas as medidas comparado com o algoritmo NBC. Ester conseguiu um aumento de 10.8% na *accuracy*, 13.8% na medida *detection rate* e uma diminuição de 17.47% na medida *false alarm rate*.

2.5.4 Abordagem de Jinwon An e Sungzoon Cho

Jinwon e Sungzoon (2015) Introduziram um método de deteção de anomalias utilizando uma rede neuronal de aprendizagem não supervisionada designada por *Autoencoder Varicional* (VAE). Este classificar é uma variante do algoritmo *Autoencoder* cujo a sua função consiste na reconstrução de um determinado *input* e no uso de probabilidades para determinar se o erro de reconstrução se trata ou não de uma anomalia. Para a deteção de intrusões Jinwon e Sungzoon (2015) utilizaram o conjunto de dados KDD 99.

Tabela 2.5: Resultados de VAE no conjunto de dados KDD 99 de (Jinwon e Sungzoon 2015)

Anomaly	AUC ROC	AUC PRC	f1 score
DoS	0.795	0.944	0.981
R2L	0.777	0.17	0.406
U2R	0.782	0.084	0.324
Probe	0.944	0.751	0.791

Na tabela 2.5 estão representados os resultados obtidos na deteção de intrusões. Jinwon e Sungzoon (2015) utilizaram as métricas AUC ROC (área abaixo a curva *Receiver Operating*

Characteristic), AUC PRC (área abaixo da curva *precision recall*) e F_1 para avaliar o desempenho do algoritmo VAE. Na tabela 2.5 pode-se verificar que os resultados das métrica AUC ROC e AUC PRC são altos para o tipo de ataque DoS, mas para as classes de ataque R2L e U2R as métrica AUC PRC e F_1 apresentam resultados baixos devido ao tamanho das classes. AUC ROC não tem em conta o número de registos, permitindo que classes de anomalias com poucos registos obtenham melhor desempenho na métrica AUC ROC. No entanto as métricas AUC PRC e F_1 têm em conta esta situação, justificando então o valor baixo da taxa de deteção para os tipos de ataques R2L e U2R.

2.5.5 Análise das abordagens

As abordagens apresentadas são importantes para a tomada de decisão das diferentes técnicas de *machine learning* a serem aplicadas neste projeto. Decidiu-se utilizar os dois conjuntos de dados referidos na abordagem de Tavallaee et al. (2009) e Yassin, Udzir e Muda (2013) por serem bastante completos, contendo inúmeras observações de atividade normal numa rede, como diferentes tipos de atividade maliciosa e por serem muito utilizados na literatura para este tipo de problemas. Verificou-se que os algoritmos baseados em árvores de decisão como o caso do C4.5 e *Random Forest*, ou redes neuronais como o caso do MLP, são técnicas que obtiveram bons resultados na deteção de intrusões sendo estas técnicas umas das escolhidas para testes. O algoritmo *autoencoder* foi também selecionado, pois poderá ser uma possível solução para deteção de novos ataques ou ataques do tipo zero. Para além destas, foram selecionadas outras técnicas de *machine learning*, como o caso do *Naive Bayes*, *K-Nearest Neighbor* e *Support Vector Machine*.

2.6 Projeto SASSI

O objetivo principal deste projeto passa por desenvolver uma metodologia para deteção, previsão e prevenção de ciberataques, numa rede onde estão ligados vários dispositivos. Esta metodologia consiste primeiramente na aplicação de técnicas de *machine learning* para a deteção e previsão de anomalias ou ataques, numa rede de um ambiente empresarial. Depois de aplicadas as referidas técnicas, serão tomadas ações para prevenção, ou seja, na deteção de um ataque, será enviado um alerta para o painel de monitorização de atividades na rede, com a informação da ocorrência de uma possível anomalia. Este mesmo alerta, contém a indicação de uma ação a tomar para previr que o intruso na rede consiga afetar o funcionamento da mesma (por exemplo: reiniciar o *router*, ou desligar o dispositivo "X"da rede).

Este projeto está a ser desenvolvido pelo GECAD integrado num projeto aprovado pelo Fundos Europeus Estruturais e de Investimento (FEEI) designado por SASSI. O projeto SASSI foi proposto pela empresa VisionTechLab, cujo ramo de atividades está ligado à industria aeroespacial. Com a sua experiência e ao longo das suas atividade nesta área, a VisionTechLab identificou a necessidade da existência de um produto que fornece a máxima segurança, uma vez que o risco envolvido nas operações desta industria é elevado. Desta forma a VisionTechLab irá desenvolver um produto que seja capaz de garantir essa segurança (VisionSpace 2015).

No âmbito deste projeto pretende-se criar uma ferramenta que apoie os administradores de redes informáticas no processo de tomada de decisão relacionado com problemas de

segurança cada vez mais robustos e sofisticados. O sistema a ser desenvolvido será composto por quatro componentes principais (VisionSpace 2015):

- uma plataforma web onde o utilizador final pode visualizar a informação recolhida e processada, através de diversos gráficos ou sumários. O objetivo desta componente é facilitar a visualização e identificação de anomalias e colocar a informação na posse do utilizador por forma a que este consiga resolver os problemas da forma mais fácil e intuitiva possível.
- O segundo componente é o desenvolvimento de um sistema de monitorização de máquinas virtuais sem que haja a necessidade da instalação no sistema operativo da máquina a monitorizar. Esta componente deverá ser capaz de gravar toda a atividade da máquina a um baixo nível e posteriormente será feita uma construção semântica para se perceber a mais alto nível o tipo de ações que foram perpetradas.
- O terceiro componente será a integração de sensores *open source* que ofereçam capacidade de monitorização de outros serviços, como por exemplo o da atividade de rede, ou o do estado dos componentes de rede (*routers, switches*, serviços). Desta forma será possível aglomerar no primeiro componente (a página web) toda a atividade e estado dos sistemas a serem monitorizados
- O quarto componente será a investigação e implementação de algoritmos inteligentes que recebam os dados recolhidos pelos sensores e que sejam capazes de identificar e prever comportamentos anómalos e prevenir a ocorrência de potenciais problemas.

Para uma melhor compreensão na figura 2.6 é apresentada a arquitetura do sistema SASSI:

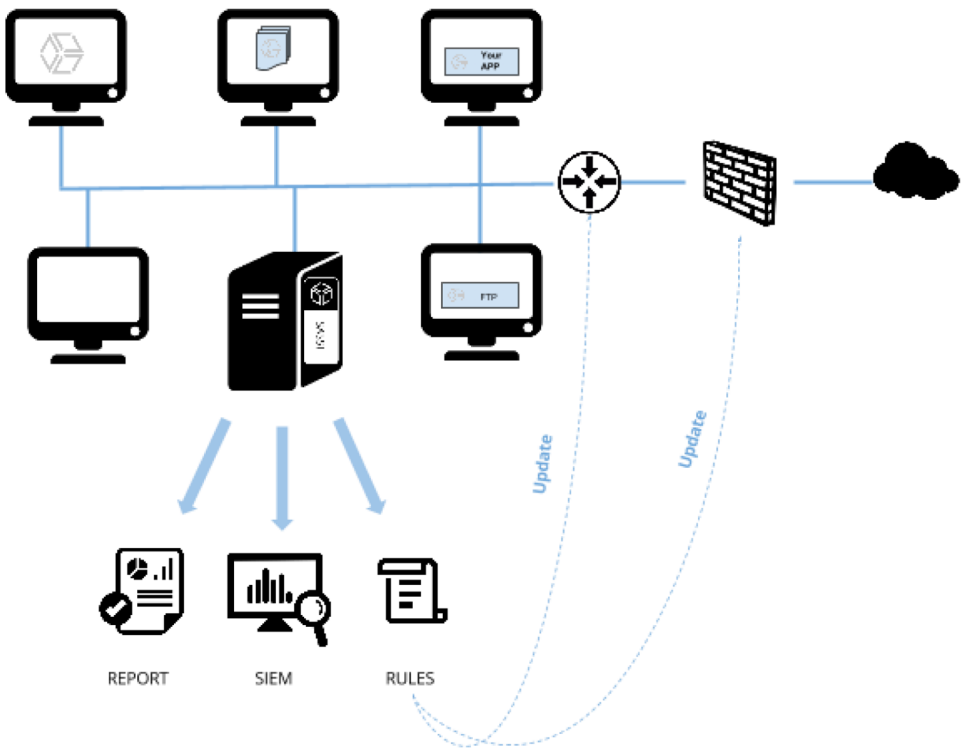

Figura 2.6: Arquitetura do sistema SASSI (VisionSpace Technologies 2016)

2.7 Visão da Solução

O tema desta tese foca-se em parte no desenvolvimento do quarto componente do projeto SASSI, ou seja, na exploração e aplicação de técnicas de *machine learning* de modo a identificar o potencial classificador ou classificadores a serem implementados no sistema SASSI, com o objetivo de conseguirem detetar e prever ataques numa rede de um ambiente empresarial. A figura 2.7 exemplifica a visão da solução a desenvolver para o componente 4:

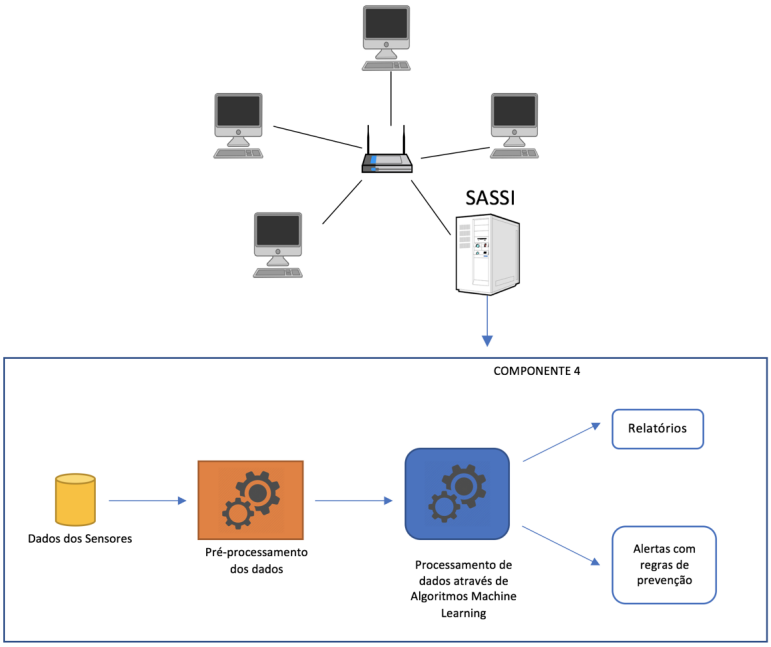

Figura 2.7: Visão da solução do projeto SASSI

Após a captação e armazenamento dos dados do tráfego da rede pelos sensores, estes serão preparados através de técnicas de pré-processamento, ou seja, será realizada a limpeza e tratamento dos dados aplicando técnicas de normalização, discretização e/ou redução de dimensionalidade se necessárias. Depois da fase de pré-processamento, o conjunto de dados será analisado por algoritmos de *machine learning* de forma a classificá-los. O resultado obtido por estes algoritmos, será disponibilizado aos utilizador em forma de relatórios com a informação do comportamento das atividades na rede. Caso seja detetada alguma anomalia, serão enviados alertas ao utilizador com regras de atuação contra as mesmas. Deste modo, o utilizador consegue analisar o evento considerado anómalo e decidir se executa a respetiva regra sugerida pelo sistema.

Uma vez que ainda não existem dados recolhidos pelos sensores do sistema SASSI, serão utilizados dois conjuntos de dados NSL-KDD e ISCX referidos na secção 2.5.2 e 2.5.3 para experimentação de várias técnicas de pré-processamento de dados e algoritmos de *machine learning*. Os resultados obtidos serão analisados e comparados entre as diferentes técnicas utilizadas de modo a verificar a existência de um modelo preditivo a ser selecionado para implementação no sistema SASSI.

2.8 Abordagem alternativa

Uma outra alternativa para o desenvolvimento do componente 4 poderá ser a de uma abordagem híbrida que combina um SDI baseado em assinaturas com um SDI baseado em anomalias. Numa primeira fase os dados dos sensores são processados para deteção de intrusões através de assinaturas ou regras associadas a um determinado tipo de ataque. Se forem detetadas intrusões então são gerados alertas com regras de prevenção, caso não sejam detetadas anomalias, então os dados serão novamente analisados por algoritmos *machine learning* após a fase de pré-processamento. Se forem identificadas intrusões através das técnicas de *machine learning* então serão gerados alertas com regras de prevenção dos ataques e criadas assinaturas para o novo ataque identificado. A imagem 2.8 representa a abordagem descrita. Com esta abordagem o número de falsos positivos (atividade de rede classificada como atividade normal quando na realidade se trata de atividade maliciosa) será menor, por outro lado é computacionalmente mais dispendiosa o que leva a uma análise de dados mais demorada e por consequência uma deteção de intrusões tardia. Esta abordagem poderá ser considerada como uma solução alternativa, caso os algoritmos utilizados na abordagem anterior apresentem dificuldades na deteção de ataques que são conhecidos. Quanto aos ataques do tipo zero se as duas abordagem utilizarem os mesmos algoritmos de previsão então a primeira abordagem será melhor devido ao tempo de resposta.

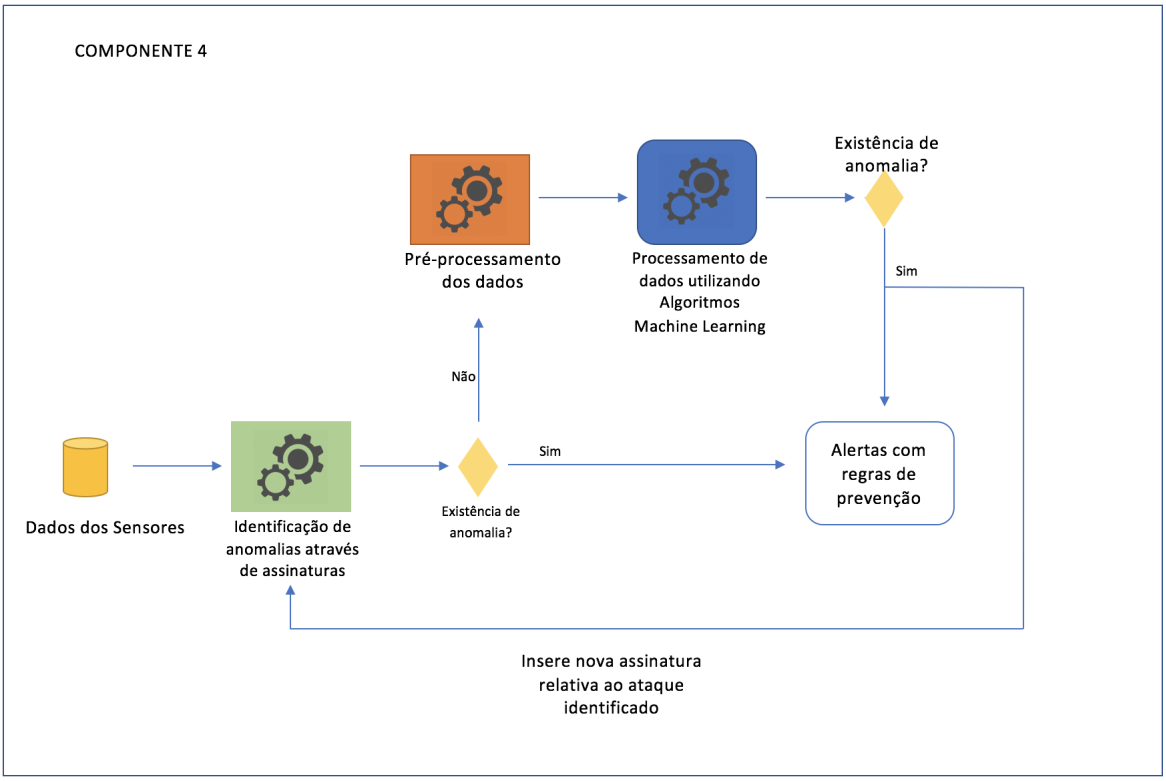

Figura 2.8: Abordagem alternativa ao projeto SASSI

2.9 Análise de Valor

O tema deste projeto consiste na exploração e aplicação de algoritmos assentes em técnicas de *machine learning* que permitam detetar, de forma autónoma, comportamentos intrusivos ou atividades que comprometam o funcionamento seguro do sistema. O estudo destas técnicas permitirá avaliar o desempenho de cada uma para posterior seleção e implementação destes algoritmos no projeto SASSI. Trata-se de um projeto de I&D em co-promoção cujo o consórcio é composto por um SME, a VisionTechLab, pelo ISEP, e pela UC:

> *O projecto SASSI tem como base o desenvolvimento de uma plataforma web que centraliza, estrutura e permite a visualização de informação relativa à atividade de redes informáticas e das máquinas que nela estão instaladas permitindo a deteção automática de anomalias. A informação é recolhida utilizando técnicas que registam e analisam as atividades de um sistema operativo virtualizado ou instalado numa máquina física. A análise será feita de forma furtiva para garantir elevada imunidade às tentativas de intrusão utilizadas por software ou utilizadores maliciosos.*
>
> *No âmbito deste projeto está também prevista a componente de deteção automática de anomalias. Os registos de atividade dos sistemas operativos serão processados com recurso a algoritmos de inteligência artificial por forma a detetar vulnerabilidades do tipo zero day exploits, ou seja: vulnerabilidades que ainda não são conhecidas mas que estão a ser exploradas para atacar sistemas computacionais.* (VisionSpace 2015).

2.9.1 Modelo New Concept Development

A utilização do modelo *New Concept Development* (NCD) é importante para compreender o processo de negócio da VisionTechLab, empresa que propôs o projeto SASSI, e o que levou ao desenvolvimento deste projeto. O modelo NCD encontra-se dividido em três partes: A parte frontal da inovação que é constituída por cinco elementos, o motor que alimenta os elementos, e os fatores de influência externos. Define-se por parte frontal da inovação todas as atividades que são realizadas antes do processo de desenvolvimento do produto. O motor refere-se ao nível executivo, aos gestores, são estes que suportam os cinco elementos do modelo NCD. Os fatores de influencia externos remetem para as capacidades organizacionais e estratégias de negócio (canais de distribuição, clientes, concorrência) (Koen et al. 2001). De seguida serão identificados e descritos os cinco elementos do modelo que constituem a parte frontal da inovação deste projeto.

Identificação da Oportunidade

Nesta secção a organização identifica a oportunidade que pretende perseguir. No caso da VisionTechLab, a organização trabalha na área da indústria aeroespacial. Ao longo do tempo, no desenvolvimento de produtos para esta indústria, em que se lida com informação altamente sensível, identificou-se a necessidade de criar um produto que forneça segurança a estas organizações. Esta seria uma clara oportunidade a explorar no mercado da Segurança Cibernética. (VisionSpace 2015).

Análise da Oportunidade

Após uma análise aprofundada ao mercado pela VisionTechLab, concluiu-se que há de facto uma evidente procura por parte do mercado de uma ferramenta capaz de reconhecer e impedir a ocorrência de anomalias nas Tecnologias de Informação e Comunicação (TIC) e que protegesse os sistemas de informação de possíveis ciberataques. Perante o caráter abrangente da temática da Cibersegurança, a VisionTechLab decidiu focar-se inicialmente na indústria aeroespacial, uma vez que já contém ativos intangíveis nesta área, nomeadamente, experiência e *know-how*. A industria aeroespacial é uma área bastante conservadora, dado o elevado risco envolvido nas suas operações. A criação de um produto que forneça segurança (oportunidade identificada), será uma mais valia para esta indústria. Após a VisionTechLab adquirir maturidade suficiente na área da segurança, o produto será direcionado para clientes de organizações com uso intensivo de TIC, nomeadamente empresas de *Cloud servers e Hosting* (VisionSpace 2015).

Geração de ideias

Nesta fase, surgiram ideias relativas à arquitetura e funcionalidades do produto a ser desenvolvido. A geração de ideias levantou uma série de questões tais como: Será criado um software de deteção de intrusões por assinaturas? Ou baseado em anomalias? Caso seja por anomalias, quais serão os métodos mais eficazes a aplicar? já existem soluções no mercado? Estas questões auxiliaram no processo de decisão sobre o rumo a tomar para o desenvolvimento do produto.

Seleção da ideia

Dada a resposta às questões colocadas no elemento "Geração de Ideias", a VisionTechLab decidiu optar pelo:

> *Desenvolvimento de um sistema que será distribuído em forma de uma máquina virtual ou equipamento físico (*appliance *por* software*: a ser instalada num sistema de virtualização ou máquina local ou então uma* appliance *física) tal como acontece normalmente com este tipo de produtos de segurança. O equipamento terá as especificações necessárias para executar de forma correta os sistemas desenvolvidos no âmbito deste projeto. A montagem do produto final pronto para venda/divulgação será portanto um computador (personalizado com o* branding *do projeto) com o software desenvolvido instalado.* (VisionSpace 2015).

Conceito e desenvolvimento

Depois de delineada a arquitetura e funcionalidades do produto, sendo este uma *appliance* por software a ser instalada num sistema de virtualização ou máquina local ou então uma *appliance* física contendo a plataforma de monitorização para deteção, previsão e prevenção de anomalias numa rede interna. Foi elaborado por parte da VisionTechLab o plano de negócio do projeto. Definiu-se as atividades necessários para o desenvolvimento do produto, custos de desenvolvimento, segmentação de clientes, concorrência no mercado, e todas as entidades inerentes ao mesmo.

2.9.2 Conceito de Valor

Existem inúmeros significados para a designação de "valor". O conceito de valor segundo The Institute of Value Management (2001), é : *"Baseado na relação entre a satisfação das necessidades/expetativas e os recursos requeridos para os atingir"*. Nicola, E. P. Ferreira e J. J. P. Ferreira (2014) definem valor como "foco em crenças, vantagens competitivas, preferências, atitudes e no alcance de objetivos". Estes autores indicam também que "A criação de valor é a chave de qualquer negócio, e qualquer atividade de negócio refere-se à troca de serviços ou bens tangíveis e/ou intangíveis em que o seu valor deverá ser aceite ou recompensado por clientes, tanto dentro de uma empresa ou fora de uma rede colaborativa".

O foco principal deste projeto remete para a segurança de sistemas computacionais. O resultado final, ou seja, a construção e implementação de algoritmos para deteção de anomalias e a criação de metodologias inteligentes para apoio à decisão, acarreta valor para o projeto SASSI, na medida em que proporcionará o desenvolvimento e divulgação do produto proposto pela empresa VisionTechLab. Pode-se também identificar como valor, a aquisição de conhecimento proveniente da realização desta tese.

2.9.3 Valor Percebido / Valor para o Cliente

"O conceito de 'valor percebido' emergiu como a definição de problemas nos negócios em 1990, e tem continuado a receber uma extensa investigação no século atual. The Marketing Science Institute (2006) incluiu a definição de 'valor percebido' na lista de prioridades da investigação em 2006-2007. Estes desenvolvimentos refletiram o grande interesse que se gerou pelo fenómeno de 'criação de valor' entre os investigadores de Marketing." (Sánchez-Fernández e Iniesta-Bonillo 2007). Notou-se um aumento das organizações no reconhecimento de que o 'valor percebido' é o fator chave para a gestão estratégica (Mizik e Jacobson 2003). Várias definições para 'valor percebido' são encontradas na literatura, vários autores sugerem 'valor' como uma troca entre benefício e sacrifício. Estes autores definem 'valor percebido' como uma variedade de noções (como preço percebido, qualidade, benefícios e sacrifícios) em que se encontram interligadas. (Babin, Darden e Griffin (1994), Holbrook (1999), Mathwick, Malhotra e Rigdon (2002), Sinha e DeSarbo (1998), Sweeney et al. (1996)). Slater (1997) observou que *"... a criação do valor para o cliente pode ser a razão para a existência da empresa, e de certo, do o seu sucesso"*. Estas palavras indicam que a criação de valor para o cliente, tornou-se uma estratégia imperativa para construir e sustentar uma vantagem competitiva (Wang et al. 2004). Foi estabelecido que os ativos intangíveis, lealdade e lucro estão forçosamente ligados ao valor que é criado para os clientes (Salem Khalifa 2004). O conceito de 'valor para o cliente' tornou-se num problema fundamental a ser encaminhado para todas as atividades de marketing (Holbrook 1999). Woodall (2003) Diz que: *"O valor para o cliente é uma perceção pessoal das vantagens advindas da associação do cliente a uma oferta da organização, e pode ocorrer como redução no sacrifício, presença do benefício; o resultado do peso da combinação de sacrifício e beneficio; ou uma agregação, ao longo do tempo, de uma ou de todas estas"*. Woodall (2003) apresenta também um modelo de uma perspetivava longitudinal composta por quatro fases, nomeadamente a pré-compra, a transação da compra, pós-compra e depois do uso (produto/serviço).

Parte deste projeto, incide no desenvolvimento do sistema de monitorização e deteção de intrusões. Este produto não apresenta sacrifícios, na primeira fase da perspetiva longitudinal

(pré-compra). Como benefícios, teremos a deteção, previsão e prevenção de ataques numa rede em que estão ligados vários dispositivos, ou seja, a garantia de um produto que fornece segurança e estabilidade numa rede interna. Para as seguintes fases da perspetiva longitudinal, nomeadamente, transação, pós-compra e pós-utilização do produto, não existem dados que permitam analisar a perceção do cliente. No entanto, espera-se o seguinte: Os benefícios associados serão os mesmos para as restantes fases, ou seja, o auxilio na tomada de decisão dos administradores de redes proporcionando maior segurança na rede interna de uma organização. Quanto aos sacrifícios, na fase de transação estará associado, o custo de aquisição do produto, na fase pós-compra, prevê-se como sacrifício o tempo despendido no processo de aprendizagem do funcionamento do software e na fase pós-utilização não se prevê qualquer sacrifício.

2.9.4 Modelo de Negócio Canvas

O modelo de negócio Canvas é uma ferramenta prática que permite abordar todos os aspetos fundamentais de um modelo de negócio. Este é composto por nove elementos que constituem as quatro principais áreas de um negócio nomeadamente a parte financeira, a oferta, a infraestrutura e os clientes (Greenwald 2012). Para uma melhor compreensão do negocio onde este projeto se insere, foi elaborado o modelo Canvas 2.9 do ponto de vista da organização VisionTechLab.

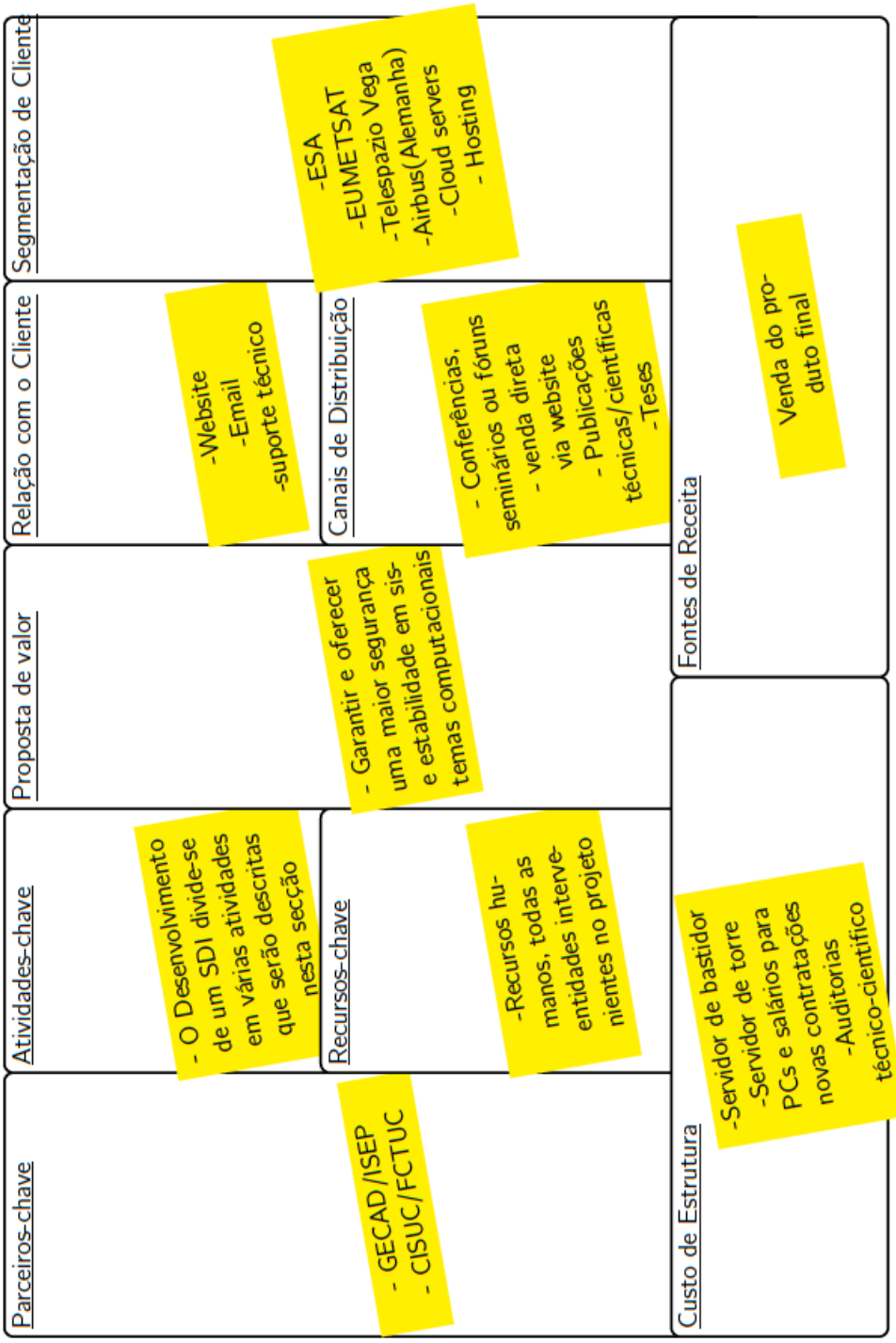

Figura 2.9: Modelo Canvas

Proposta de Valor

Como proposta de valor, no âmbito do projeto SASSI, será desenvolvido um produto que monitoriza as atividades e operações numa rede onde se encontram ligados vários dispositivos de modo a detetar, prever e prevenir o acontecimento de ataques ao sistema. O principal valor desta solução foca-se no fornecimento de segurança e estabilidade da rede interna do cliente. Este produto diferencia-se dos SDI atuais pois será uma *appliance* física ou uma máquina virtual evitando desta forma a sua instalação num sistema operativo. O SDI será capaz de gravar toda a atividade da maquina a um baixo nível, permitindo a não deteção da monitorização por parte de algum atacante que tenha conseguido acesso ao sistema.

Canais de Distribuição

Quanto aos canais de distribuição, o produto será divulgado em seminários, *foruns*, conferencias técnicas/cientificas, através de teses de mestrado ou doutoramento. Poderá ser comprado via *webiste* e será distribuído por empresas de distribuição.

Relação com o Cliente

O cliente poderá requisitar apoio técnico através do *website*, email ou requerer serviços de manutenção de um assistente técnico.

Segmentação de Cliente

A VisionTechLab é uma empresa ligada à indústria aeroespacial e tem como clientes a Agência Espacial Europeia (ESA), a Organização Europeia para Exploração de Satélites Meteorológicos (EUMETSAT) e outros grandes operadores europeus privados desta indústria como o *Airbus* e a *Telespazio Vega* (VisionSpace 2015). No decorrer da sua atividade identificou-se a necessidade de criar uma ferramenta capaz de reconhecer potenciais anomalias no processo produtivo e que fortalecesse simultaneamente a cibersegurança dos sistemas de informação. Esta necessidade levou a expansão para novos mercados e a VisionTechLab delineou como novos clientes-alvo organizações com uso intensivo das TIC, nomeadamente empresas de *Cloud servers e Hosting*.

Parceiros Chave

Como indicado no modelo Canvas 2.9, os parceiros da VisionTechLab que prestarão auxilio no desenvolvimento do projeto SASSI são o *Research Group on Intelligent Engineering and Computing for Advanced Innovation adn Development* (GECAD) com sede no ISEP e o Centro de Informação e Sistemas da Universidade de Coimbra (CISUC).

Atividades chave

Quanto às atividades chaves para o desenvolvimento do referido SDI serão divididas da seguinte forma (VisionSpace 2015):

- Modelação e Desenvolvimento da página web de monitorização;
- Desenvolvimento do sistema de monitorização;
- Geração dos dados de anomalia;
- Desenvolvimento do algoritmo de deteção e previsão de falhas de sistema;
- Desenvolvimento de metodologias para Apoio à Decisão Inteligente;
- Desenvolvimento da hardware *appliance* e integração do produto;
- Gestão e divulgação do projeto;

Recursos Chave

Todos as entidades intervenientes no projeto SASSI como programadores, analistas, administradores, recursos computacionais (servidores, computadores, etc), fazem parte dos recursos chaves.

Custo de Estrutura e Fontes de Receita

Quanto ao custo de estruturas estão associados os salários dos recursos humanos, tecnologias de informação e comunicação, subcontratação de trabalhos especializados e auditorias. A principal fonte de receita será a venda do produto final.

2.10 Conclusão

Ao longo deste capítulo foi apresentada uma contextualização do problema, foram descritos os princípios da segurança de informação e os tipos de anomalias que normalmente se encontram em sistemas computacionais numa rede. De seguida realizou-se um estudo da arquitetura dos SDI e fez-se um enquadramento da aplicação de técnicas de *machine learning* nestes sistemas. Foram também apresentadas várias abordagens ao problema de deteção de intrusões utilizando diferentes algoritmos preditivos. A análise destas abordagens permitiu identificar algumas técnicas a serem exploradas nesta dissertação. Foi também apresentado o projeto SASSI, a solução proposta para desenvolvimento e uma abordagem alternativa. Por fim foi realizado a análise de valor do projeto SASSI.

Capítulo 3

Deteção de intrusões nos conjuntos de dados NSL-KDD e ISCX 2012

Neste capítulo serão descritas todas as técnicas utilizadas na deteção de intrusões utilizando os conjuntos de dados públicos NSL-KDD e ISCX 2012. Primeiro fez-se uma análise exploratória aos dados destes dois conjuntos de modo a compreender a constituição e estrutura dos mesmos. Conhecer os atributos, os seus valores, a correlação entre eles, os tipos e quantidade de intrusões existentes, são fatores importantes a ter em conta antes de aplicar qualquer tratamento ou modelação nos dados. Após esta análise, deu-se inicio ao pré-processamento dos dados. Nesta fase os dados são tratados, através de várias técnicas de modo a serem alimentados pelos algoritmos de previsão. Por fim, são descritos todos os tipos de algoritmos utilizados, o seu funcionamento e configuração na deteção de intrusões.

Para a aplicação destes modelos preditivos utilizou-se a linguagem R através do *Ambiente de Desenvolvimento Integrado* (IDE) *RStudio*. O R é uma linguagem utilizada para estatística computacional e gráfica. Fornece uma grande variedade de técnicas gráficas e de estatística(modelação linear e não-linear, testes clássicos de estatística, análise de séries temporais, *clustering*,...). Contém um repositório designado por *Comprehensive R Archive Network* (CRAN) com uma grande quantidade de pacotes que fornecem funções de estatística, criação de gráficos e algoritmos de *machine learning*. Escolheu-se esta linguagem pelo facto de ser *open source* e ser muito utilizada para este tipo de aplicações, levando à existência de uma enorme comunidade de suporte, com inúmeros fóruns que auxiliam na resolução de problemas deste tipo.

3.1 Descrição e Análise Exploratória dos Dados

3.1.1 NSL-KDD

Este conjunto de dados foi desenvolvido por Tavallaee et al. (2009) com o propósito de resolver os problemas existentes no conjunto de dados KDD 99 referido na secção 2.5.2. NSL-KDD encontra-se dividido em dois subconjuntos de dados, um para treinar o(s) algoritmo(s) contendo 125.973 observações e outro para testar o(s) algoritmo(s) contendo 22.544 observações. Cada um destes subconjuntos apresenta 43 atributos. Na tabela 3.1 podem visualizar-se os atributos agregados pelo seu tipo de dados, sendo estes nominais, binários ou numéricos (Stolfo et al. n.d.).

É importante realçar que o subconjunto de teste não tem a mesma distribuição de probabilidade que o subconjunto de treino, e inclui novos tipos de ataques. Desta forma a tarefa de

Tabela 3.1: Atributos presentes no conjunto de dados NSL-KDD (Stolfo et al. n.d.)

Tipo	Atributos
Nominal	Protocol_type (2), Service (3), Flag (4), attack_type (43)
Binário	Land (7), logged_in (12), root_shell(14), su_attempted(15), is_host_login(21), is_guest_login(22)
Numérico	Duration(1), src_bytes(5), dst_bytes(6), wrong_fragment(8), urgent(9), hot(10), num_failed_logins(11), num_compromised(13), num_root(16), num_file_creations(17), num_shells(18), num_access_files(19), num_outbound_cmds(20), count(23) srv_count(24), serror_rate(25), srv_serror_rate(26), rerror_rate(27), srv_rerror_rate(28), same_srv_rate(29) diff_srv_rate(30), srv_diff_host_rate(31), dst_host_count(32), dst_host_srv_count(33), dst_host_same_srv_rate(34), dst_host_diff_srv_rate(35), dst_host_same_src_port_rate(36), dst_host_srv_diff_host_rate(37), dst_host_serror_rate(38), dst_host_srv_serror_rate(39), dst_host_rerror_rate(40), dst_host_srv_rerror_rate(41), difficulty_level(42)

deteção de intrusões torna-se mais realista desafiando os classificadores a detetarem novos tipos de ataques. O subconjunto de treino contém 24 tipos de ataques e no subconjunto de teste podem-se encontram mais 14 tipos de ataques para além dos 24 tipos presentes no subconjunto de treino, o que faz um total de 38 tipos de ataques agrupados nas 4 categorias referidas na secção 2.5.1. Na tabela 3.2 é apresentada a distribuição destes ataques pelas 4 categorias nos subconjuntos de dados de treino e de teste.

Tabela 3.2: Distribuição dos ataques presentes no conjunto de dados NSL-KDD, adaptado de Natesan e Balasubramanie (2012)

Categoria de Ataque	Ataques no conjunto de treino NSL-KDD	Ataques adicionais no conjunto de teste de NSL-KDD
DoS	back, neptune, smurf, teardrop, land, pod.	apache2, mailbomb, processtable.
Probe	satan, portsweep, ipsweep, nmap	mscan, saint
R2L	warezmaster, warezclient, ftpwrite, guesspassword, imap, multihop, phf, spy	sendmail, named, snmpgetattack, snmpguess, xlock, xsnoop, worm
U2R	rootkit, bufferoverflow, loadmodule, perl.	httptunnel, ps, sqlattack, xterm

Para uma melhor compreensão da distribuição das 4 categorias de ataques pelo conjunto de

dados, uniu-se o conjunto de dados de treino com o de teste e foram criados vários gráficos de barras para cada um dos atributos nominais, como se pode ver nas figuras 3.1, 3.2, 3.3 e 3.4. Foram substituídos os valores do atributo attack_type (tipos de ataques) pelas categorias de ataque correspondente aos vários ataques referidos na tabela 3.2. As cores representadas nos gráficos referem-se às 5 categorias do atributo attack_type. O valor "0" corresponde à categoria de atividade normal na rede, os valores "1, 2, 3 e 4" às categorias de ataque "DoS, Probe, R2L e U2R". O ataque mais comum é o "DoS" representado pela cor laranja utilizando na maioria dos seus ataques o protocolo "TCP", a *flag* "S0", e o serviço "*private*". A categoria de ataque "Probe" é a segunda com maior presença no conjunto de dados NSL-KDD, seguido da categoria de ataque "R2L" e por último "U2R". A distribuição destas duas ultimas categorias é de difícil perceção nos gráficos dos atributos protocol_type, *service, flag* devido ao número reduzido de observações para estas categorias. Quanto à categoria de ataque *Probe*, utiliza o protocolo "TCP" e "ICMP" para a maior parte dos seus ataques. É possível verificar também que esta categoria de ataque utiliza maioritariamente as *flags* "REJ", "RSTR" e "SF" e os serviços "eco_i", "*private*" e "*other*" (este serviço não se encontram legível devido á dimensão reduzida no eixo horizontal do gráfico da figura 3.3).

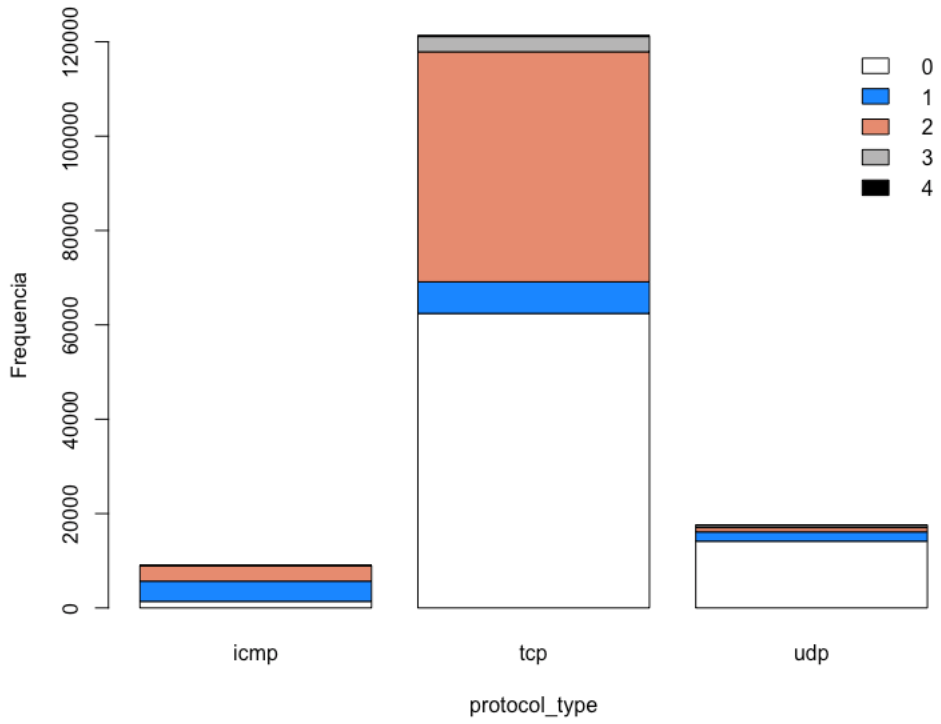

Figura 3.1: Gráficos de barros do atributo protocol_type com a distribuição dos tipos de ataques

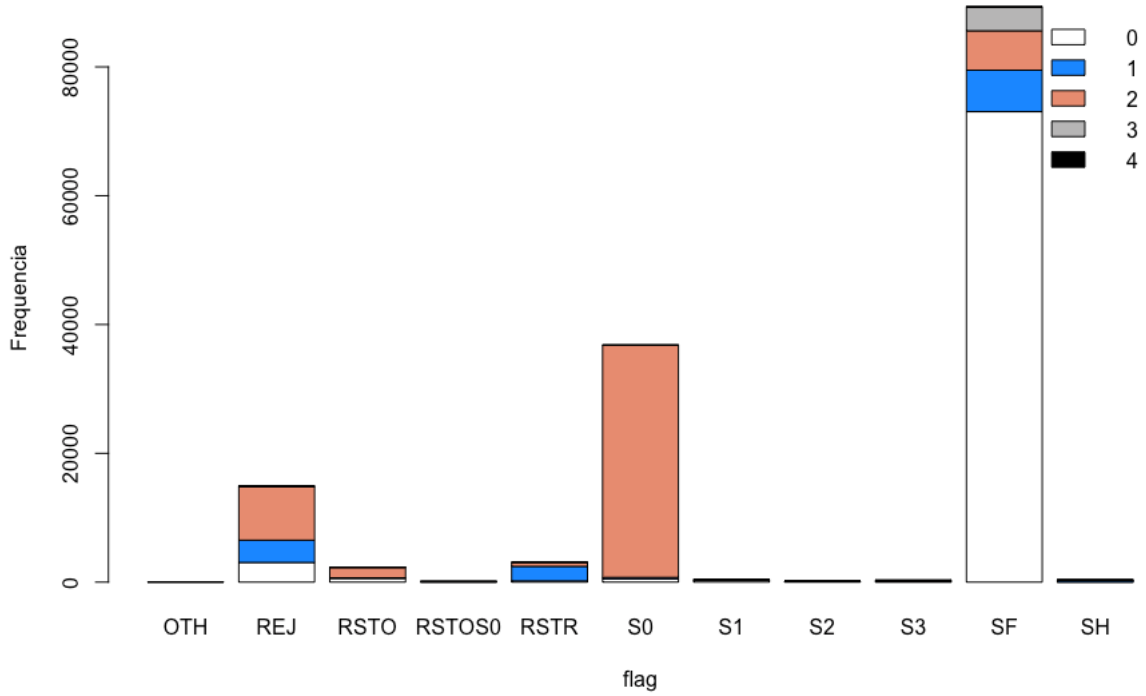

Figura 3.2: Gráficos de barros do atributo flag com a distribuição dos tipos de ataques

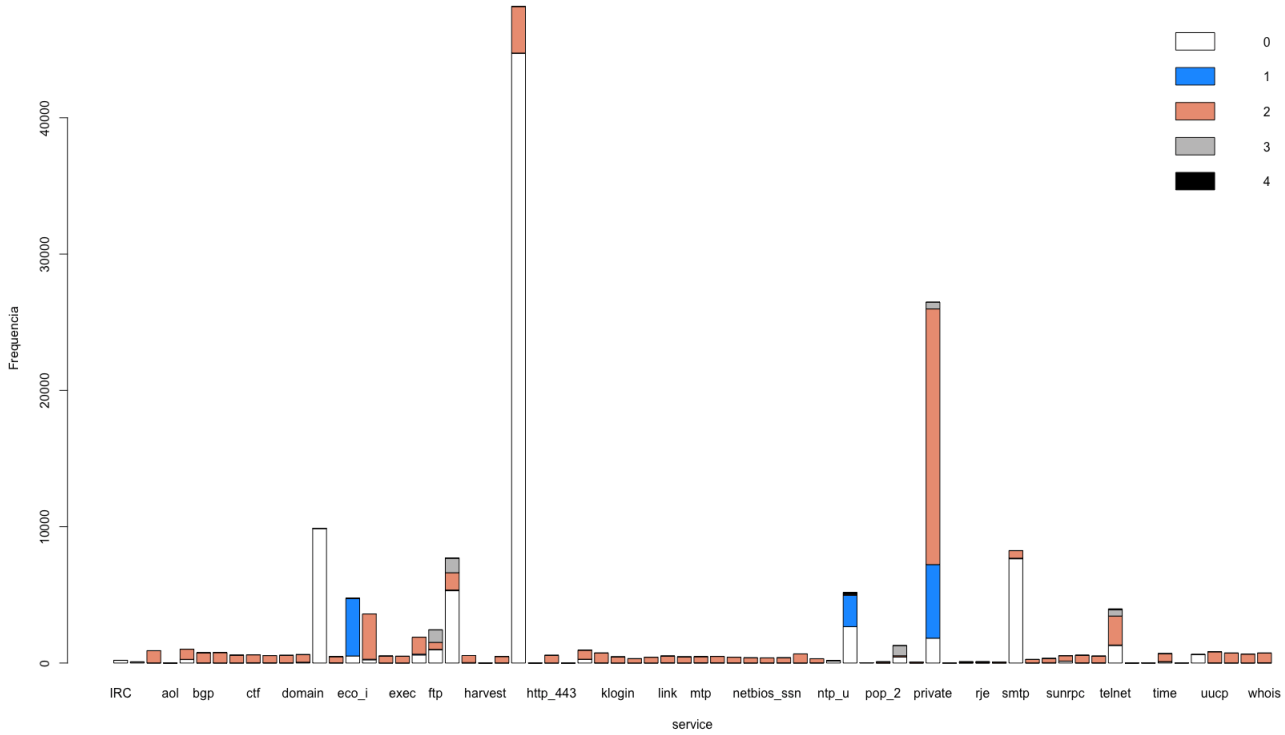

Figura 3.3: Gráficos de barros do atributo service com a distribuição dos tipos de ataques

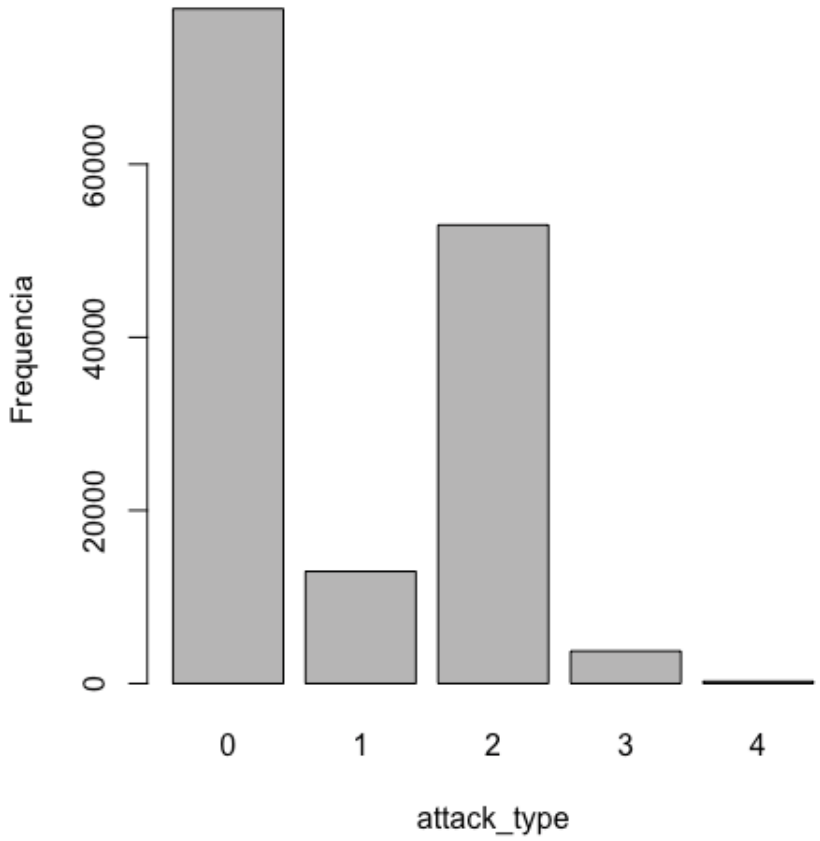

Figura 3.4: Gráficos de barros do atributo attack_type

É também importante determinar se o NSL-KDD se inclui num problema de dados linearmente separáveis de modo a auxiliar na compreensão do comportamento de determinados algoritmos. Para verificar a linearidade dos dados será necessário recorrer e satisfazer todos os pressupostos do modelo de regressão linear (Matos 1995):

1. O erros E_i (resíduos), são variáveis aleatórias de média próxima de zero e distribuição normal;
2. Os erros E_i têm variância constante (σ^2) - Hipótese de homocedasticidade;
3. As variáveis aleatórias $E_1, E_2, ..., E_n$ são independentes;
4. As variáveis explicativas $X_1, X_2, ..., X_n$ não estão correlacionadas - Hipótese de ausência de multicolinearidade;

Após o cálculo da média dos resíduos, chegou-se ao resultado $-2.133132e^{-18}$. Sendo este valor muito próximo de zero, então pode-se concluir que o primeiro pressuposto se encontra satisfeito no conjunto de dados. Quanto ao segundo pressuposto referente à homocedasticidade foi criado um gráfico, visível na figura 3.5, com a distribuição dos resíduos em relação aos valores previstos. Observa-se que os resíduos mostram um certo padrão de distribuição em relação aos valores previstos. Pode-se então descartar a hipótese de homocedasticidade e concluir que a variância dos resíduos não é constante. Como não

é possível verificar os pressupostos dos resíduos, conclui-se que os dados representam um problema não linearmente separável.

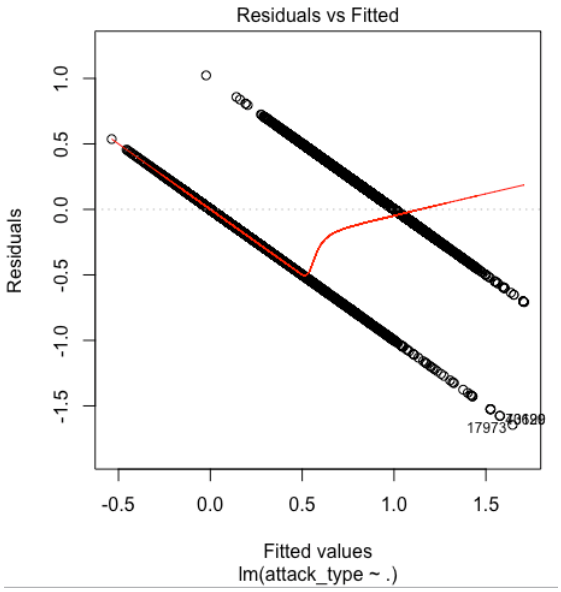

Figura 3.5: Gráficos de dispersão dos resíduos em relação aos valores previstos no conjunto de dados NSL-KDD

3.1.2 ISCX-2012

O conjunto de dados ISCX foi desenvolvido por Shiravi et al. (2012) no instituto de cibersegurança do Canadá. Este conjunto de dados baseia-se no conceito de perfis que contém descrições detalhadas de intrusões e distribuições abstratas para aplicações, protocolos, serviços e entidades de rede de baixo nível. Foram analisadas interações reais de comunicação de rede de modo a criar perfis para agentes que geram tráfego real para protocolos HTTP, SMTP, SSH, IMAP, POP3 e FTP. A este respeito, foram estabelecidas um conjunto de diretrizes para delinear um conjunto de dados válido que estabelecem a base para a geração de perfis. Estas diretrizes são vitais para a eficácia do conjunto de dados em termos de realismo, captura total, integridade e atividade maliciosa (Shiravi et al. 2012). O ISCX contém 21 atributos que se apresentam descritos na tabela 3.3. Contém também 7 dias de tráfego de rede capturado com 4 tipos de ataques diferentes como descrito na secção 2.5.3.

Será interessante perceber qual é o nível de correlação que os atributos apresentam entre si, o que poderá facultar uma indicação da existência de atributos dependentes (colineares) nos dados. Na figura 3.6 é apresentada a correlação entre os atributos de uma amostra de um determinado endereço de IP destino do conjunto de dados ISCX. Para a criação deste gráfico foram realizadas algumas modificações:

- Todos os atributos do tipo nominais foram convertidos para numéricos;
- O atributo *destination* foi eliminado pois apresenta o mesmo valor, uma vez que foi filtrado para um endereço de IP específico;

Tabela 3.3: Descrição dos atributos presentes no conjunto de dados ISCX 2012

Atributos	Descrição	Tipo
X	Índice do conjunto de dados	Numérico
appName	Nome da aplicação	Nominal
TotalSourceBytes	Quantidade total de bytes enviado pela máquina fonte	Numérico
TotalDestinationBytes	Quantidade total de Bytes recebido pela máquina destino	Numérico
totalSourcePackets	Quantidade total de pacotes enviado pela máquina fonte	Numérico
totalDestinationPackets	Quantidade total de pacotes recebido pela máquina destino	Numérico
sourcePayloadAsBase64	PayLoad da máquina fonte codificado em Base64	Nominal
sourcePayloadAsUTF	PayLoad da máquina fonte codificado em UTF	Nominal
destinationPayloadAsBase64	PayLoad da máquina destino codificado em Base64	Nominal
destinationPayloadAsUTF	PayLoad da máquina destino codificado em UTF	Nominal
direction	Direção do fluxo, exemplo: remote to local	Nominal
sourceTCPFlagsDescription	Solicitações da máquina fonte de ação TCP, exemplo: Push, Synchronize, Finish, Acknowledge, Reset	Nominal
destinationTCPFlagsDescription	Ação TCP da máquina destino	Nominal
source	Endereço IP da máquina fonte	Nominal
protocolName	Nome do protocol utilizado pelo fluxo, exemplo: TCP-IP	Nominal
sourcePort	Número da porta da máquina fonte	Numérico
Destination	Endereço de IP da máquina destino	Nominal
destinationPort	Número da porta da máquina destino	Numérico
startDateTime	Tempo de iniciação do fluxo	Numérico
stopDateTime	Tempo de paragem do fluxo	Numérico
Tag	Legenda do fluxo, sendo este normal ou de ataque	Nominal

- Todos os atributos *Payload* (contém informações dos pacotes como cabeçalho ou meta-dados) foram removidos;
- O atributo *Tag* foi removido;

Na figura 3.6 o tamanho do círculo é proporcional ao valor absoluto da correlação e a cor indica a direção e a grandeza dessa correlação. Tal como seria de esperar, todos os valores da diagonal a que corresponde a correlação do atributo com ele próprio contêm um valor máximo de correlação. É possível observar uma correlação elevada entre os atributos totalSourceBytes, totalDestinationPackets e totalSourcePackets, a qual também se verifica mas a um baixo nível entre o atributo totalDestinationBytes e os atributos totalDestinationPackets e totalSourcePackets. Observa-se também uma anticorrelação no valor de -0,6 entre o atributo source e direction.

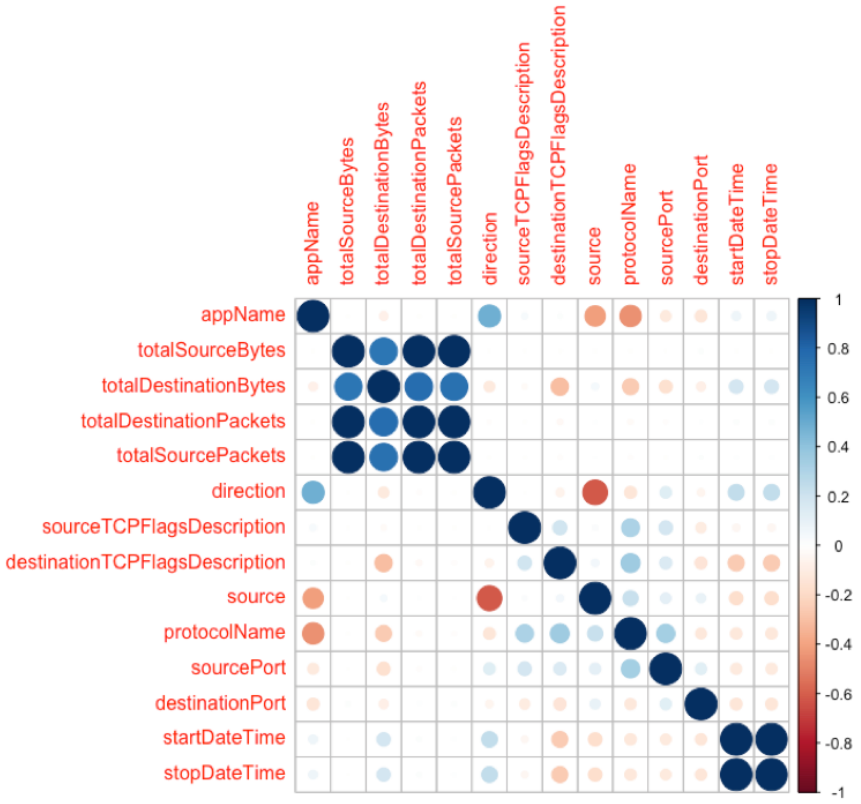

Figura 3.6: Gráfico que representa a correlação dos atributos do conjunto de dados ISCX 2012

Em relação à análise deste conjunto de dados corresponder a um problema de dados linearmente separáveis, repetiu-se o mesmo processo realizado no conjunto de dados NSL-KDD e foi criado um gráfico da distribuição dos resíduos em relação aos valores previstos representado na figura 3.7. Pode-se concluir que devido ao facto da existência de um padrão na distribuição dos pontos dos resíduos, o pressuposto da homocedasticidade não é aplicável no conjunto de dados ISCX, pois os erros não têm variância constante. Pode-se então concluir que os dados não são linearmente separáveis. Esta análise da natureza dos dados, auxilia na escolha dos algoritmos como também na parametrização dos mesmos, pois existem algoritmos que estão preparados para a deteção de padrões em dados linearmente ou não linearmente separáveis. Existem também classificadores como o caso do algoritmo SVM preparado para a classificação de dados de natureza linear ou não-linear dependendo da sua parametrização, por exemplo na escolha da função *kernel* a aplicar. O objetivo desta função é descrito com mais detalhe neste capítulo.

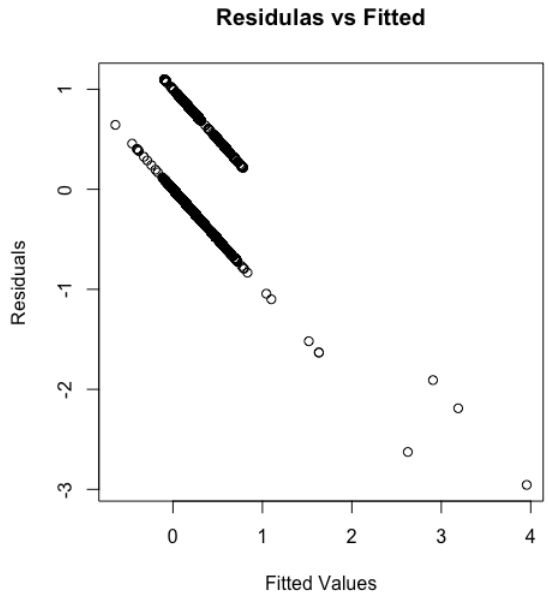

Figura 3.7: Gráficos de dispersão dos resíduos em relação aos valores previstos no conjunto de dados ISCX

3.2 Pré-Processamento

Para os conjuntos de dados NSL-KDD e ISCX foram aplicadas várias técnicas de pré-processamento, de modo a comparar o impacto destas após a aplicação dos algoritmos preditivos nestes conjuntos de dados. Como se pode ver na figura 3.8, o pré processamento encontra-se dividido em 4 etapas, nomeadamente separação e tratamento, discretização, normalização e redução de dimensionalidade. O fluxo 3.8 representa apenas uma das sequências da fase pré-processamento, foram aplicadas várias sequências combinando as técnicas de cada etapa do pré-processamento. As combinações testadas são as seguintes:

- Separação e tratamento + Discretização + Normalização + Redução de Dimensionalidade + Algoritmos preditivos;
  - IF + Z-Score + RFE + Algoritmos preditivos;
  - IF + MinMax + RFE + Algoritmos preditivos;
- Separação e tratamento + Normalização + Redução de Dimensionalidade + Algoritmos preditivos;
  - Z-Score + RFE + Algoritmos preditivos;
  - MinMax + RFE + Algoritmos preditivos;
- Separação e tratamento + Discretização + Redução de Dimensionalidade + Algoritmos preditivos;
  - IF + RFE + Algoritmos preditivos;
- Separação e tratamento + Redução de Dimensionalidade + Algoritmos preditivos;
  - RFE + Algoritmos preditivos;

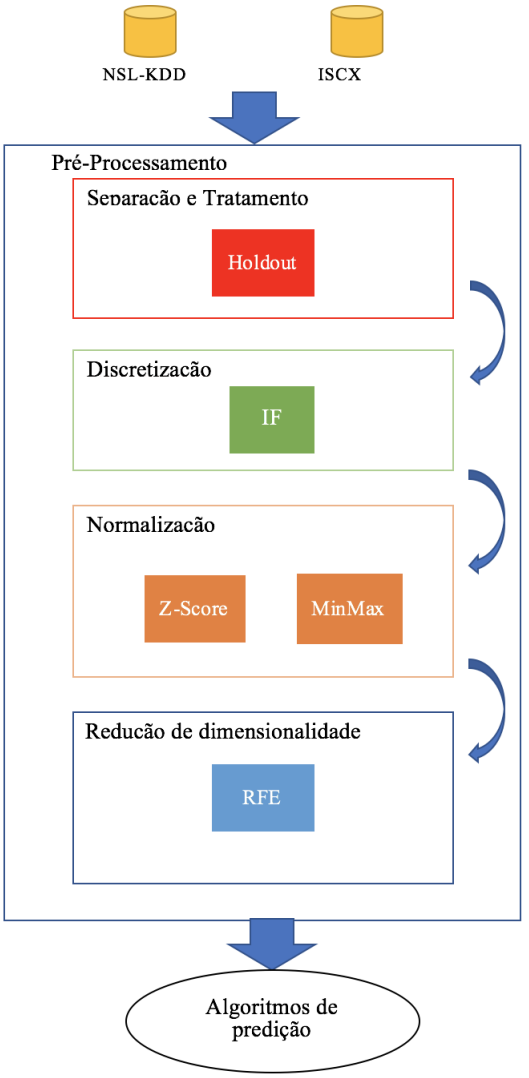

Figura 3.8: Fases do pré-processamento de dados a aplicar em NSL-KDD e ISCX

3.2.1 Separação e Tratamento dos dados

Nesta etapa, foi utilizado o método *Holdout* estratificado. Este método divide um conjunto de dados de forma aleatória em duas partes. Uma das partes representa aproximadamente 2/3 do conjunto de dados e será utilizado para treino dos classificadores. A outra parte (aproximadamente 1/3) será utilizada para testar os classificadores (Han, Pei e Kamber 2011). No conjunto de dados NSL-KDD, este já se encontra dividido desta forma, mas como tanto o conjunto de dados de treino como o de teste contêm demasiadas observações, o que levaria a um custo demasiado elevado de recursos computacionais, então selecionou-se uma amostra menor nas mesmas proporções e contendo em cada uma delas 10% de registos rotulados como "Ataque". Após a seleção destas amostragens, procedeu-se ao tratamento dos dados:

- Os atributos nominais "Protocol_type", "Service", "Flag", "attack_type" foram convertidos para numéricos;

- Remoção dos atributos com valores constantes;

Para o conjunto de dados ISCX, foram selecionados registos de pacotes de comunicação recebidos por um determinado host (IP -192.168.5.122) em cada um dos dias de monitorização de tráfego. De seguida procedeu-se à divisão do conjunto de dados, aplicando também o método *Holdout* estratificado. O conjunto de treino contém um total de 35.000 registos e o de teste 15.000, em cada um destes conjuntos de dados existem 10% de registos contendo pacotes de atividade maliciosa. De notar que os tipos de ataques presentes no conjunto de dados de teste e de treino do ISCX são os mesmos, não existindo ataques novos. Quanto ao tratamento dos dados foram realizadas as seguintes operações:

- Foi removido o atributo "X" que representa o índice do conjunto de dados;
- Todos os atributos nominais foram convertidos para numéricos como o caso dos atributos "appName", "direction", "sourceTCPFlagsDescription", "destinationTCPFlagsDescription", "source", "protocolName", "Tag";
- Remoção do atributo "destination", uma vez que foi selecionado o endereço de IP destino 192.168.5.122, então este atributo terá um valor constante sendo portanto irrelevante para análise;
- Foram removidos os atributos "sourcePayloadAsBase64", "sourcePayloadAsUTF", "destinationPayloadAsBase64", "destinationPayloadAsUTF". Estes atributos contêm dados não numéricos, sendo estes inputs incompatíveis para os algoritmos de classificação utilizados;
- Foi calculada a diferença de tempo entre o atributo "startDateTime" e "StopDateTime", dando origem a um novo atributo designado por "diffTime". De seguido procedeu-se á remoção dos atributos "startDateTime" e "StopDateTime".

3.2.2 Normalização

Para ser possível comparar variáveis, é necessário proceder à normalização dos dados para que estes estejam dentro da mesma escala. A normalização procede à redução dos dados a uma determinada escala, desta forma, impede que alguns algoritmos de classificação deem mais importância a atributos com grandes valores numéricos. Uma vez que os atributos se encontram todos na mesma escala, então os classificadores atribuem o mesmo peso a cada atributo. Para cada um dos conjuntos de dados (NSL-KDD e ISCX) foram aplicadas duas técnicas de normalização:

MinMax

A normalização Min-Max, também designada normalização por escala, realiza uma transformação do conjunto de entrada original para um novo conjunto específico numa escala em que os valores variam entre 0 e 1 (Y. K. Jain e Bhandare 2011). São inicialmente definidos os valores mínimo (min) e máxmimo (max) para os novos valores de cada atributo. De seguida, as seguintes operações são realizadas para cada atributo. Primeiro, o menor valor do atributo, (menor), é subtraído a cada valor. O resultado de cada valor, é depois, dividido pela diferença entre o maior e o menor valores originais do atributo (maior-menor). Cada novo valor é depois multiplicado pela diferença entre os valores limites da nova escala,

max-min. No final o valor min é somado a cada valor produzido. A equação 3.1 ilustra as operações descritas.

$$VNovo = min + \frac{(VAtual - menor)}{maior - menor}(max - min) \tag{3.1}$$

Z-Score

Para normalização por padronização Z-Score, a cada valor do atributo a ser normalizado é adicionada ou subtraída uma medida de localização, sendo o valor resultante multiplicado ou dividido por uma medida de escala. Com esta operação, diferentes atributos podem apresentar limites inferiores e superiores diferentes, mas terão os mesmos valores para as medidas de escala e dispersão (Gama et al. 2015). Esta técnica de normalização, referida também como Média-Zero ou Normalização Unidade-Variante, transforma os dados de variáveis de entrada de tal forma que a média é zero, o desvio padrão é um. A equação 3.2 resumo esta transformação.

$$VNovo = \frac{VAtual - \mu}{\sigma} \tag{3.2}$$

3.2.3 Discretização

Uma das operações potencialmente úteis de transformação de variáveis é a discretização de variáveis numéricas, que pode ser usada quando algum tipo de análise apenas pode ser aplicada a variáveis nominais ou quando se quer simplificar a análise. Esta operação implica dividir o intervalo de valores possíveis em sub-intervalos e considerar cada um deles como uma categoria (Rocha e G.Ferreira 2017). Desta forma, alguns classificadores ou métodos de redução de dimensionalidade lidam melhor com estes dados porque o intervalo de valores é menor, levando a uma aprendizagem mais precisa e rápida em determinados classificadores (H. Liu et al. 2002). Basicamente, o processo de discretização passa pelo agrupamento de valores contínuos num determinado número de intervalos discretos (Janssens et al. 2006). No entanto, a determinação de quais os valores contínuos a serem agrupados, quantos intervalos se devem gerar e quais são os pontes de corte desses intervalos, não é sempre idêntico, dependendo da técnica de discretização a utilizar. Para os conjuntos de dados NSL-KDD e ISCX foi utilizada a técnica de discretização de intervalos Igual Frequência (IF). Esta técnica divide os valores dos atributos a serem discretizados em k intervalos, de modo a que cada intervalo contenha aproximadamente o mesmo número de observações. Assim cada intervalo contêm $\frac{n}{k}$ valores adjacentes. O valor de k é um parâmetro definido pelo utilizador e para a obtenção deste valor utilizou-se a heurística $\sqrt{n}$ em que n representa o número de amostras.

3.2.4 Redução de Dimensionalidade

A elevada dimensionalidade nos conjuntos de dados tem um impacto importante nos classificadores, pois conjuntos de dados com um grande número de atributos irrelevantes ou redundantes provocam uma diminuição no desempenho dos algoritmos de classificação. Este fenómeno é denominado por "maldição de dimensionalidade" (A. Jain e Zongker 1997),

porque os atributos dispensáveis aumentam o tamanho do espaço de pesquisa tornando a generalização mais difícil. Para superar este problema, são utilizadas técnicas de redução de dimensionalidade. Desta forma o conjunto de atributos necessários para descrever o problema é reduzido originando na maioria das vezes um melhor desempenho nos classificadores. Seleção de atributos é a técnica mais conhecida de redução de dimensionalidade. Consiste na deteção de atributos relevantes descartando os irrelevantes. O objetivo consiste em obter um conjunto de atributos que descrevam apropriadamente um determinando problema com o mínimo de redução no desempenho dos classificadores (Guyon et al. 2006), e com os benefícios implícitos de uma melhor compreensão dos dados e redução na necessidade de recursos computacionais. As técnicas de seleção de atributos podem ser divididas em três métodos, *wrappers*, filtragem e embutidos.

Métodos de Filtragem

Figura 3.9: Método de Filtragem

Métodos de filtragem, aplicam medidas estatísticas para atribuir uma pontuação a cada atributo. Os atributos são classificados pela pontuação e selecionados para serem mantidos ou removidos do conjunto de dados. Estes métodos são na maioria univariados considerando cada atributo como sendo independente. Alguns exemplos de métodos de filtragem utilizam vários testes estatísticos como o caso do Chi quadrado, utilizam também ganho de informação e/ou coeficientes de correlação (Kaushik 2016).

Métodos Wrapper

Estes métodos consideram a seleção de um conjunto de atributos como um problema de procura, onde são preparadas diferentes combinações de atributos, avaliadas e comparadas entre si. É utilizado um modelo preditivo para avaliar a combinação de atributos e atribuir uma pontuação baseado no desempenho do modelo. Estes métodos são normalmente bastantes dispendiosos a nível computacional (Brownlee 2014).

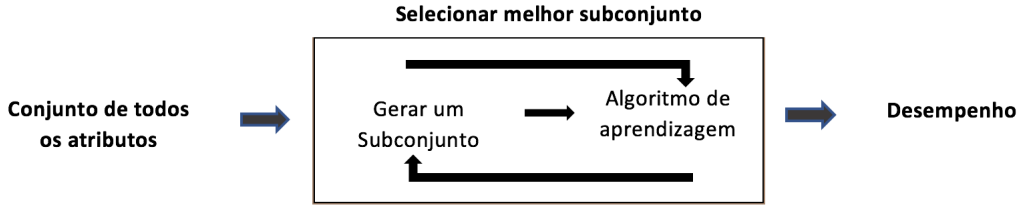

Figura 3.10: Método Wrapper

Alguns exemplos de métodos *wrapper* são:

- **Seleção para a frente** – sendo este um método iterativo no qual o modelo começa sem atributos. Em cada iteração, é adicionado um atributo que melhora o desempenho do modelo. Quando a nova variável a ser adicionada não melhorar o desempenho do modelo, então o método termina (Brownlee 2014).
- **Eliminação para trás** – Neste caso, o modelo inicia com todos os atributos removendo em cada iteração o atributo menos significativo que permite melhorar o desempenho do modelo. Este processo é repetido até não se observarem melhorias no desempenho do modelo (Brownlee 2014).
- **Recursive Feature Elimination (RFE)** – É um algoritmo de otimização que visa encontrar o subconjunto de atributos que oferecem o melhor desempenho ao modelo. Este processo classifica os atributos de acordo com uma medida de importância especifica. Em cada interação são criados repetidamente vários modelos, a importância dos atributos é medida e o(s) atributo(s) com menor valor de importância é/são removido(s). A recursividade é necessária pois em algumas medidas a importância relativa a cada atributo pode alterar significativamente quando avaliados num conjunto diferente de atributos durante o processo de eliminação (em particular para atributos com alto grau de correlação). A ordem (inversa) em que os atributos são eliminados é utilizada para construir o modelo final de classificação dos mesmos (A. Jain e Zongker 1997).

Métodos Embedded

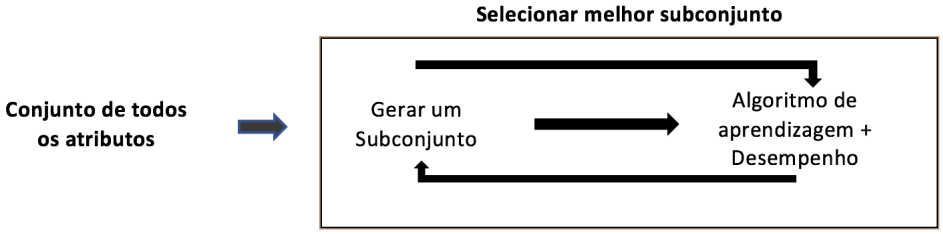

Figura 3.11: Método de Embutido

Métodos embutidos combinam as qualidades dos métodos por filtragem e *wrapper*. São utilizados por algoritmos que tem incorporado o seu próprio método de seleção de atributos. Estes aprendem quais os atributos que melhor contribuem para o desempenho do modelo enquanto este ainda está a ser criado. Os métodos embutidos de seleção de atributos mais comuns são os métodos de regularização, também designados por métodos de penalização em que introduzem restrições adicionais na otimização de um algoritmo preditivo. Exemplos destes algoritmos são LASSO, *Elastic Net e Ridge Regression* (A. Jain e Zongker 1997).

Foi selecionado um dos métodos *wrapper*, RFE para ser aplicado nos conjuntos de dados NSL-KDD e ISCX. Este algoritmo encontra-se disponível no pacote *Caret* da biblioteca (CRAN) do R. Em cada iteração foi utilizado o algoritmo preditivo *Random Forest* para avaliar o modelo nos conjuntos de dados de treino (NSL-KDD e ISCX) através de validação cruzada com 5 *folds*. A metodologia de validação cruzada em k-*folds* divide um conjunto de dados em k partes iguais. Em cada iteração o modelo é treinado nas k-1 partes e testado

numa delas. Este processo é repetido de modo a que a parte k de teste é alterada em cada iteração. Desta forma, a tendência e a variância da previsão do modelo são reduzidas (T. Srivastava 2016). No gráfico esquerdo 3.12 relativo ao conjunto de dados NSL-KDD verifica-se que com 9 atributos o desempenho do algoritmo aproxima-se do seu máximo valor. No caso do conjunto de dados ISCX, pode-se visualizar no gráfico direito da figura 3.12 que o algoritmo atinge quase o seu máximo de desempenho com apenas 6 atributos. Fizeram-se vários testes com os algoritmos preditivos de modo a observar qual o número ótimo de atributos a utilizar em cada um dos conjuntos de dados tendo em conta o desempenho dos algoritmos e o custo computacional. Conclui-se que no conjunto de dados NSL-KDD os algoritmos obtiveram melhores resultados com 14 atributos, exceto num dos casos em que o número ótimo foi de 9 atributos tendo em conta a ordem de classificação fornecida pelo algoritmo de seleção de atributos RFE. Relativamente ao ISCX chegou-se a um número ótimo de 6 atributos a utilizar em todos os algoritmos preditivos.

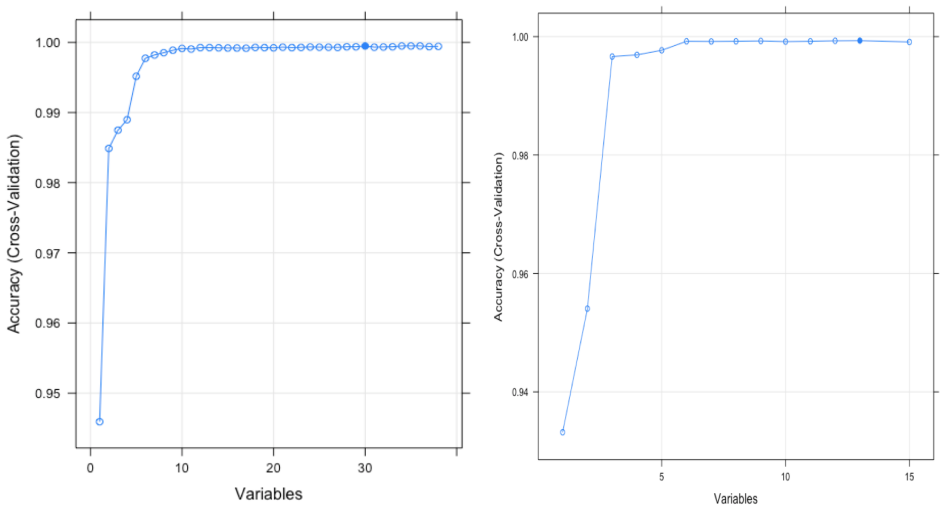

Figura 3.12: Representação da evolução do desempenho do algoritmo RFE VS número de atributos. Gráfico esquerdo diz respeito ao conjunto de dados NSL-KDD e o da direita ao conjunto de dados ISCX

3.3 Modelos Preditivos

Para a predição dos ataques incluídos nos dois conjuntos de dados, foram selecionados 6 algoritmos de classificação de aprendizagem supervisionada e um classificador de aprendizagem não-supervisionada. A escolha destes algoritmos deve-se ao facto de serem bastante mencionados na literatura para problemas relacionados com deteções de anomalias, tratarem-se de algoritmos que lidam com classes desbalanceadas e também por lidarem com problemas multi-classes, como é o caso.

3.3.1 Algoritmos de Aprendizagem Supervisionada

Redes Neuronais

A rede neuronal é uma unidade de processamento que se assemelha aos neurónios do cérebro humano em dois pontos principais (Haykin 1999):

- A rede adquire conhecimento através de um processo de aprendizagem a partir do ambiente envolvente.
- Os nós da rede (neurónios) estão ligados através de interligações com pesos, utilizados para guardar o conhecimento;

Uma rede neuronal é caraterizada por dois aspetos básicos, a sua arquitetura que está relacionada com o tipo e números de unidades de processamento e a forma como os neurónios estão conectados; A aprendizagem que diz respeito às regras utilizadas para o ajuste dos pesos da rede e a informação utilizada pela rede.

MLP é a arquitetura mais utilizada em redes neuronais na aplicação de problemas de reconhecimento de padrões tanto no âmbito geral como na área de deteção de anomalias, ver figura 3.13. Os primeiros métodos foram baseados nesta arquitetura, Cannady (1998) aplicou uma MLP para analisar o tráfego de rede a ataques externos. Em Zhu et al. (2005) a MLP foi utilizada para deteção de comportamentos anormais por parte do utilizador. Também em Mukkamala, Andrew H Sung e Abraham (2005) foram usadas redes MLP para detetar anomalias. As rede MLP têm a vantagem de aprender modelos-não lineares.

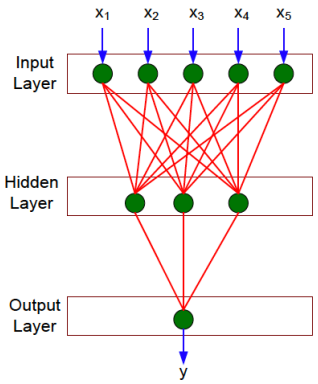

Figura 3.13: Arquitetura MLP (Yong Joseph Bakos 2010)

Uma rede MLP é dividida em três classes uma camada de input que recebe a informação a ser processada, uma camada de output onde são encontrados os dados processados, e uma ou mais camadas ocultas entre estas duas. As unidades de processamento desempenham um papel muito simples. Cada terminal de entrada de um neurónio recebe um valor. Os valores recebidos são ponderados e combinados por uma função de ativação matemática f_a. O *output* da função é a resposta do neurónio para a entrada do próximo. Considerando um objeto x com d atributos representado na forma do vetor $x = [x_1, x_2, x_3, ..., x_d]^t$ e um neurónio com d terminais de entrada cujos pesos são $w_1, w2, w_3, ..., w_n$ sendo representados na forma vetorial $w = [w_1, w2, w_3, ..., w_n]$. O *input* total recebido pelo neurónio u é definido pela equação 3.3 (Gama et al. 2015).

$$u = \sum_{j=1}^{d} x_j w_j \tag{3.3}$$

As Redes neuronais podem apresentar conexões de retropropagação ou *feedback*. Estas conexões permitem que um neurónio receba no seu terminal de entrada o *output* de um neurónio da mesma camada ou de uma camada posterior. Redes sem conexões de retro-propagação, são as mais utilizadas e denominadas por *feedforward*) (Gama et al. 2015).

Para aplicação da rede neuronal *feedforward* no conjunto de dados NSL-KDD e ISCX foi utilizado a função *train* cujo o parâmetro *Method=nnet* .Esta função está disponível no pacote *Caret* do repositório CRAN do R. Através desta função o classificador é treinado com o método de validação cruzada com 5 *folds*. No final da fase de treino, o modelo apresenta os resultados de classificação obtidos para os vários parâmetros utilizados na rede neuronal. Por fim, é escolhido o modelo com o melhor resultado e aplicado no conjunto de dados de teste. Os seguintes parâmetros obtiveram os resultados mais altos na classificação das observações, tanto no conjunto de dados NSL-KDD como no ISCX:

- **Size = 5** - O parâmetro *Size* indica o número de unidades de neurónios na camada oculta. O modelo obteve o resultado mais alto com 5 neurónios na camada oculta. a Figura 3.14 representa a estrutura da rede neuronal utilizada no conjunto de dados NSL-KDD, constituída por 13 neurónios na camada *input* (número de atributos do conjunto de dados NSL-KDD após aplicar a técnica de redução de dimensionalidade), 5 neurónios na camada oculta e 1 neurónio na camada *output*;
- **linout = FALSE** - Ao colocar este parâmetro como falso a rede neuronal aplica a função de ativação sigmoide;
- **Decay = 0,01** Em cada atualização dos pesos, estes são depois multiplicados pelo o parâmetro *Decay*. Isto previne que os pesos atinjam um valor demasiado alto;

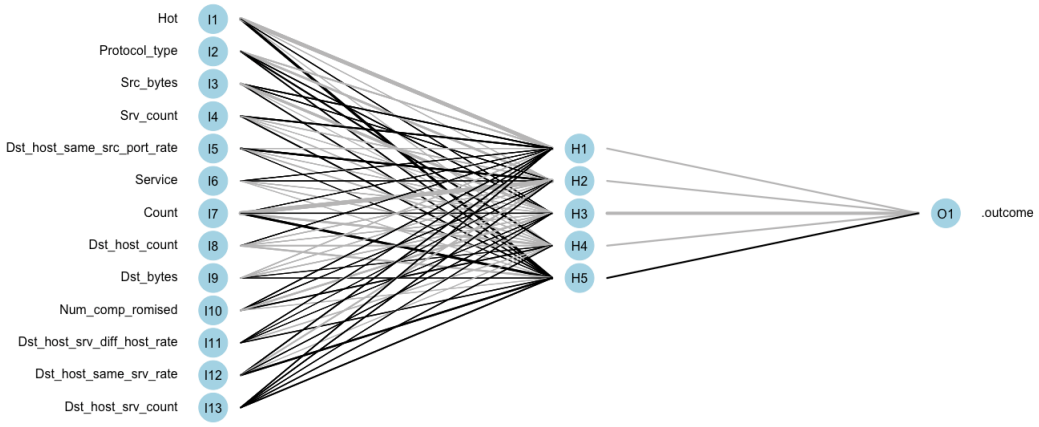

Figura 3.14: Rede neuronal utilizado no conjunto de dados NSL-KDD

Support Vector Machine

É uma técnica de aprendizagem supervisionada que tem a capacidade de resolver problemas de classificação e regressão desenvolvida por Cortes e Vapnik (1995). Este tipo de algoritmo foca-se na procura de um hiperplano (generalização de um plano em diferentes dimensões, por exemplo num plano bidimensional é uma linha que separa e classifica dados) que melhor divide um conjunto de dados em duas classes. Os vetores de suporte são os pontos que se encontram perto do hiperplano, ver na figura 3.15. Estes são considerados os elementos críticos do conjunto de dados pois se removidos, a posição do hiperplano iria alterar-se.

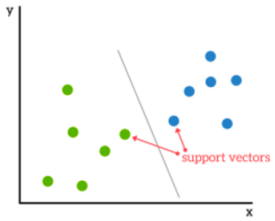

Figura 3.15: Representação de um plano bidimensional com vetores de suporte e hiperplano (Kdnuggets n.d.)

SVM são eficazes na classificação dos dados linearmente separáveis ou que possuam uma distribuição aproximadamente linear. Porém existem muitos casos em que não é possível dividir satisfatoriamente os dados de treino num hiperplano. As SVM com problemas deste tipo mapeando o conjunto de treino do seu espaço original para um novo espaço de maior dimensão, denominado por espaço de características *(feature space)* (Osuna e Platt 1998). Para calcular os produtos escalares entre objetos mapeados no novo espaço, são utilizadas funções denominadas por *Kernels*. A utilidade dos *kernels* está portanto, na simplicidade do seu cálculo e na sua capacidade de representar espaços abstratos. Alguns dos *kernels* mais utilizados na prática, são os polinomiais, os de função base radial e os sigmoidais. Cada um deles apresenta parâmetros que devem ser determinados pelo utilizador (Gama et al. 2015).

Este algoritmo não está preparado para grandes conjuntos de dados porque o tempo de treino dos dados pode ser elevado. Também é pouco eficiente em conjuntos de dados com ruído, com classes sobrepostas.

Huang e Chen (2014) integraram este algoritmo num SDI. A abordagem proposta consegue evitar o problema do SVM com outliers. O método utilizado tem uma grande precisão comparado com os modelos convencionais.

No âmbito do trabalho aqui descrito, realizaram-se testes nos dois conjuntos de dados (NSL-KDD e ISCX) para otimização dos parâmetros do algoritmo SVM. Para a realização dos referidos testes foi utilizado o pacote *Caret* do repositório CRAN. Através da função *train* procedeu-se à aplicação da técnica de validação cruzada com 10 *folds* de forma a obter os melhores resultados para os parâmetros das SVM. Por fim, foi aplicado este classificador utilizando o pacote *e1071* do repositório CRAN com os seguintes parâmetros obtidos nos testes de otimização:

- Utilizou-se a função kernel base radial;
- O valor do coeficiente $C = 1$ - Este parâmetro refere-se à constante de penalização que tem como objetivo controlar o tamanho da margem do hiperplano. Quanto maior for o valor de C, menor será o tamanho da margem e vice-versa (Cournapeau 2007);

- O valor gamma $\gamma = 0.01$ - O parâmetro gamma define a distancia que a influencia de um simples exemplo de treino pode atingir. Com valores baixos de γ correspondem a uma influência "alargada" e valores altos a uma influência "próxima" (Cournapeau 2007);

C4.5

Desenvolvido por Quinlan (1993), o algoritmo C4.5, dado um conjunto de dados já classificados constrói um classificador na forma de uma árvore de decisão. Estas árvores de decisão são um método de aprendizagem supervisionada muito eficaz. Tem como objetivo a partição de um conjunto de dados em grupos de dados homogéneos para a previsão de uma determinada variável. Recebe como input um conjunto de dados classificados e retorna uma árvore que se assemelha a um diagrama em que cada nó (folha) da extremidade da árvore é uma classe e cada nó interno representa um teste. Cada folha representa a decisão de se incluir numa classe, verificando o caminho desde a raiz, pelos nós de teste até às extremidades (Hssina et al. 2014). A teoria de Shannon é a base dos algoritmos que utilizam árvores de decisão. Entropia de Shannon é a mais aplicada em que define a quantidade de informação fornecida por um evento. Dado uma probabilidade de distribuição P = (p_1, p_2,...,p_n) e uma amostra **S**, então a quantidade de informação desta distribuição, designada por Entropia de P, é dada pela expressão (Hssina et al. 2014) 3.4:

$$Entropia(P) = -\sum_{i=1}^{n} p_i \times log(p_i) \tag{3.4}$$

C4.5 aplica o ganho de informação, utilizando a entropia como medida de impureza para gerar a árvore de decisão. Para determinar quão boa é uma condição de teste realizada, terá que se comparar o grau de entropia do nó-pai (antes da divisão) com o grau de entropia dos nós-filhos (após divisão). O atributo que gerar maior diferença é escolhido como condição de teste. Define o ganho de um teste **T** e a posição **p** como representado na equação (Hssina et al. 2014) 3.5.

$$Ganho(p, T) = Entropia(p) - \sum_{j=i}^{n}(p_j \times Entropia(p_j)) \tag{3.5}$$

Os valores p_j representados na equação 3.5 são o conjunto de todos os valores possíveis para o atributo T. Esta medida é utilizada para classificação de atributos e construção da árvore de decisão onde em cada nó está localizado o atributo com maior ganho de informação. Na existência de domínios com ruído que possam causar *overfiting*, o C4.5 aplica a técnica da poda (*prunning*) após a construção da árvore, sendo esta uma fase importante no processo de construção pois permite uma nova generalização para novos exemplos de classificação. Na fase da poda, o algoritmo C4.5 retrocede pela árvore quando esta é criada e tenta remover ramificações que não ajudam no processo de decisão substituindo esses ramos por nós folha. O algoritmo C4.5 foi utilizado nos conjuntos de dados NSL-KDD e ISCX, através do pacote *Rweka* do repositório CRAN do R.

Random Forest

Random Forest (floresta aleatória) é um algoritmo que como o nome indica, cria uma floresta com um determinado número de árvores de decisão. Este é um método de conjuntos cujo objetivo é criar um "*strong learner*"através de um grupo de "*weak learners*". Cada classificador, neste caso, cada árvore de decisão é um "*week learner*", enquanto que a junção de todos as árvores de decisão são consideradas um "*strong learner*"(Benyamin 2012). O algoritmo *Random Forest*, como referido, utiliza classificadores em forma de árvores de decisão $\{h(x, \Theta_k), k = 1, ..., \}$ onde $\{\Theta_k\}$ são vetores aleatórios independentes distribuídos de forma idêntica e x é um padrão de entrada (Breiman 2001). No processo de treino, os algoritmos *Random Forest* criam múltiplas árvores de decisão do tipo *CART* (Breiman et al. 1984), onde cada uma delas treinada com uma amostra de dados retirada do conjunto de dados treino, e procuram apenas por um subconjunto de dados aleatório das variáveis de entrada para determinar uma divisão para cada nó da árvore. Na classificação, cada árvore do algoritmo *Random Forest* apresenta um voto para a classe mais popular da variável de entrada x. O resultado do algoritmo é determinado pela maioria dos votos dados pelas árvores de decisão.

O número de variáveis é um parâmetro definido pelo utilizador (muitas das vezes dito que é o único parâmetro ajustável numa *Random Forest*), mas o algoritmo não é sensível a este parâmetro. Muitas das vezes, um valor de atributos é selecionado e colocado na raiz quadrada do número de *inputs*, para limitar o número de variáveis utilizadas para divisão, a complexidade computacional do algoritmo é reduzida como também a correlação entre árvores. As árvores de decisão não são podadas contribuindo também para a redução computacional. Este algoritmo, consegue suportar conjuntos de dados de alta dimensionalidade e utilizar conjuntos grandes de árvores de decisão. Isto combinado com o facto de que a seleção aleatória de variáveis para divisão procura minimizar a correlação entre os conjuntos de árvores, resultando em taxas de erro reduzidas.

O pacote *randomForest* do repositório CRAN do R fornece o algoritmo mencionado para aplicação do mesmo nos conjuntos de dados NSL-KDD e ISCX. Utilizou-se 500 árvores (número que vêm por defeito no parâmetro *n_tree*) a serem criadas neste modelo para predição dos dados.

Naive Bayes

A Teoria Bayesiana faz duas assunções (Bayesianos 2011):

- Os atributos são de igual importância;
- Os atributos são estatisticamente independentes, ou seja, conhecer o valor de um atributo não diz nada acerca do valor de outro atributo.

Assunções de independência normalmente não são corretas, no entanto, funcionam bem no esquema de aprendizagem Bayesiana. As relações entre os eventos dependentes podem ser descritas por meio do **Teorema de Bayes**, como apresentado na expressão 3.6:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(B \cap A)}{P(B)} \tag{3.6}$$

A notação P(A|B) pode ser lido como a probabilidade do evento A, dado que o evento B aconteceu. Também conhecida como probabilidade condicional uma vez que a probabilidade de A é condicionada pelo evento B (Hajek 2003). Um exemplo para melhor compreensão pode ser a verificação de emails *Spam*. Ao selecionar aleatoriamente uma mensagem da caixa de correio e caso essa mensagem contenha a palavra Ganhou (evento B), qual é a probabilidade da mensagem ser *Spam* (evento A). Aplicando o teorema de *Bayes* calcula-se a probabilidade posterior que mede a probabilidade da a mensagem ser *Spam*, apresentado na equação 3.7

$$P(Spam|Ganhou) = \frac{P(Ganhou|Spam)P(Spam)}{P(Ganhou)} \tag{3.7}$$

Supondo que na caixa de correio encontram-se 50 mensagens, em que 4 mensagens contem a palavra Ganhou, e se destas mensagens 3 forem Spam e o total das mensagens Spam for igual 10 então P(Ganhou|Spam) = 3/10 = 0.3. Aplicando na formula 3.7 então o resultado será apresentado na equação 3.8.

$$P(Spam|Ganhou) = \frac{0.3 \times (\frac{10}{50})}{\frac{4}{50}} = 0.75 \tag{3.8}$$

Portanto **75%** é a probabilidade de uma mensagem ser Spam, caso contenha a palavra Ganhou. A aplicação do teorema de Bayes requer o conhecimento de duas probabilidades a priori - P(decisão$_i$) e uma probabilidade condicional - P(x|decisão$_i$). Se forem adicionados mais termos (W) ao filtro de Spam então a equação seria 3.9

$$P(Spam|W_1 \cap W_2 \cap W_n) = \frac{P(W_1|Spam)P(W_2|Spam)P(W_n|Spam)}{P(W1)P(W2)P(Wn)} \tag{3.9}$$

Naive Bayes foi aplicado ao conjunto de dados NSL-KDD e ISCX utilizando o pacote *e1071* do repositório CRAN do R.

K Nearest Neighbor (KNN)

O algoritmo K Nearest Neighbor (KNN) tem variações definidas pelo número de k vizinhos considerados. Dessas variações a mais simples é o algoritmo 1- vizinho mais próximo (1-NN, do inglês Nearest Neighbor) onde $k = 1$. Neste algoritmo cada objeto representa um ponto no espaço definido pelos atributos. Selecionando uma métrica nesse espaço é possível calcular as distancias entre esses dois pontos. A métrica mais utilizada é a distancia euclidiana, dada pela equação 3.10, em que x_i^l e x_j^l são dois objetos representados por vetores no espaço $\mathbb{R}$d, e x_i^l e x_j^l são elementos desses vetores, que correspondem aos valores da coordenada *l* (atributos) (Gama et al. 2015).

$$d(X_i, X_j) = \sqrt{\sum_{i=1}^{d}(x_i^l - x_j^l)^2} \tag{3.10}$$

O algoritmo KNN em que $K = 1$ ou também designado por 1-NN, é bastante simples, como se pode visualizar no algoritmo 3.1. Na fase de treino, o algoritmo memoriza os exemplos

rotulados do conjunto de treino. Para classificar um exemplo não rotulado, ou seja, cuja classe não é conhecida, é calculada a distância entre o vetor de valores de atributos e cada exemplo rotulado em memória. O rótula da classe associada ao exemplo de treino mais próximo de teste é utilizado para classificar o novo exemplo (Gama et al. 2015).

Algoritmo 3.1 1 - Vizinho mais próximo

Entrada: Um conjunto de treino: $D = (x_i, y_i, i = 1, ..., n)$
Um objeto de teste a ser classificado: $t = x_t, y_t =?$
A função de distância entre objetos: $d(x_a, x_b)$
Saída: y_t : Classe atribuída ao exemplo t
$d_{min} \leftarrow +\infty$
  para cada $i \in 1, ..., n$ **faça**
    se $d(x_i, x_t) < d_{min}$ **então**
      $d_{min} \leftarrow d(x_i, x_t)$
      $idx \leftarrow i$
    fim
fim
$y_t = y_{idx}$
Retorna: y_t

A figura 3.16 apresenta um exemplo do funcionamento deste algoritmo para diferentes valores de k num problema de duas classes. No espaço definido pelos atributos, e utilizando a distancia euclidiana, o objeto de treino mais próximo do objeto de teste pertence á classe com o símbolo "−"para k=1 e k=2 e com o símbolo "+"para k=3.

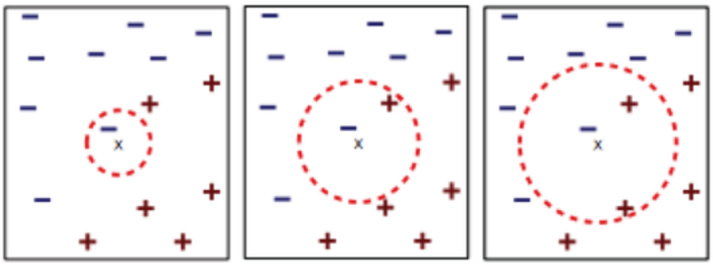

Figura 3.16: K-Nearest Neighbors para k=1 na imagem da esquerda, k=2 na imagem do meio e k=3 na imagem da direita (Song et al. 2014)

Na literatura este algoritmo foi bastante utilizado para a deteção de anomalias em redes. Sultani (2015) utilizou o algoritmo KNN para deteção de ataques no conjunto de dados NSL-KDD. Xiang et al. (2009) aplicaram-no no conjunto de dados KDD 99. Li e Guo (2007) propuseram um novo método supervisionado de deteção de intrusões em rede baseado no TCM-KNN (Transductive Condidence Machines - K-Nearest Neigbors) e aplicaram no KDD 99.

Para a deteção de intrusões nos conjuntos de dados NSL-KDD e ICSCX, foi escolhido o algoritmo KNN em que o valor de k é igual a 1 pois foi o valor que obteve o desempenho mais alto na classificação dos dados. Este classificador encontra-se no pacote *class* do repositório CRAN do R.

3.3.2 Algoritmo de Aprendizagem Não-Supervisionada

Autoencoder

Este é um algoritmo de classificação de uma classe (*one-class classification*), ou seja, uma classe (com dados normais ou positivos) terá que ser distinguida de outras classes (com dados anómalos). É uma abordagem em que o algoritmo é treinado apenas com um tipo de classe, no caso deste projeto, será a classe que contém dados de atividade normal na rede e testado num conjunto de dados que contenham outras classes como dados de atividade normal e dos vários tipos de ataques. Este método de classificação de uma classe é muito utilizado na deteção de *outliers* e/ou deteção de novas observações (Chandola, Banerjee e Kumar 2007). De acordo com Chandola, Banerjee e Kumar (2007) a deteção de *outliers* refere-se à tarefa de encontrar padrões que não apresentam o comportamento esperado (observações anómalas) enquanto que a deteção de novas observações identificam padrões não conhecidos que geralmente se encontram integrados nos registos de atividade normal após serem descobertos. Este é um cenário típico numa ampla variedade de ambientes reais e consequentemente esta disciplina tem ganho muita atenção ao longo dos anos (Francos 2017).

O *Autoencoder* é uma rede neuronal que faz parte de uma sub-área de *machine learning* designada por *deep learning* (conjuntos de algoritmos com várias camadas de processamento que são utilizados para modelar abstrações de alto nível de dados (Deng e Yu 2013)). As redes neuronais são redes de unidades de processamento interligadas que se encontram organizadas por uma ou mais camadas, que podem ser utilizadas na implementação de um mapeamento funcional complexo entre variáveis de *input* e *output*. Podem ser realizadas transformações lineares ou não-lineares através do processamento das unidades nas diferentes camadas. Os parâmetros destas unidades (pesos) são ajustados através do uso de dados de treino de forma a que a função de erro seja minimizada sobre o conjunto de treino (Mazhelis e Review 2016).

O algoritmo *Autoencoder*, também referido como *autoassociator*, é uma espécie de rede neuronal que é treinada para que os atributos de entrada sejam iguais ou muito semelhantes aos atributos de saída (Japkowicz 1999). Assume-se que o *autoencoder* aprende a estrutura interna dos dados. Na classificação, apenas os vetores cuja estrutura é semelhante à estrutura conhecida pela rede neuronal, é reproduzido pelo *autoencoder* com precisão.

Sendo uma rede neuronal, os *autoencoders* são sensíveis a *outliers* (Bishop 1995), pois contribuem para a minimização da função de erro. A desvantagem dos *autoencoders* é a necessidade de utilizar um determinado número de parâmetros que têm que ser especificados pelo utilizador (Delft e Magnificus 2001). Estes incluem a seleção de um número de camadas ocultas N_{h1}, de um número de unidades ocultas N_{hu} em cada camada, o tipo de função de transformação, a taxa de aprendizagem e a regra de paragem. Para além destes parâmetros, será necessário estimar um número de pesos (normalmente igual ao número de unidades ocultas e de entrada) para o conjunto de treino. É essencial uma grande quantidade de dados para uma estimação precisa dos pesos. Os recursos computacionais para este algoritmo são considerados elevados, uma vez que, o processo de aprendizagem é iterativo, sendo este repetido vários vezes ao longo do conjunto de dados de treino, até a regra de paragem ser satisfeita. Hinton (1989) estima que a complexidade de aprendizagem para redes neuronais é aproximadamente $O(N_w^3)$ onde N_w é o número de pesos na rede.

No domínio das deteções de intrusões, os *autoencoders* foram aplicados com êxito na deteção de anomalias de tráfego de rede com o protocolo TCP/IP (B 2005).

Para aplicação deste algoritmo foi utilizado o pacote H2O do repositório CRAN. Este pacote, é um motor matemático *open source* para *big data* que processa algoritmos de *machine learning* paralelamente distribuídos como modelos lineares generalizados, *gradient boosting*, *Random Forests* e redes neuronais (*deep learning*) em vários ambientes de *cluster* (Kraljevic 2017). Após carregar os conjuntos de dados de treino e teste respetivamente do NSL-KDD e ISCX, para este motor, utilizou-se a função *h2o.deeplearning* para treinar o algoritmo *autoencoder*. Relativamente aos parâmetros a utilizar, seguiu-se a abordagem de Glander (2017) que consiste na deteção de fraude em transações de cartões de crédito utilizando *autoencoders*:

- **autoencoder = TRUE** - ativar este parâmetro para funcionar como *autoencoder*;
- **hidden = c(50,5,50)** - define o número de camadas ocultas e unidades da rede neuronal, neste caso o vetor c(50, 5, 50) contém 3 valores e cada valor corresponde ao número de neurónios por camada. O número de unidades deste parâmetro é diferente da abordagem de Glander (2017) pois após se realizarem vários testes, verificou-se que os resultados da classificação são superiores com esta estrutura;
- **activation = Tanh** - definir qual a função de ativação, neste caso escolheu-se a função tangente hiperbólica, pois obteve os resultados mais altos na deteção de anomalias;

Depois de o modelo finalizar o seu processo de treino, utilizou-se a função *h2o.anomaly*. Esta função tem como objetivo detetar anomalias num conjunto de dados. A função reconstrói o conjunto de dados original utilizando o modelo de treino e calcula o Erro Quadrático Médio (MSE) para cada ponto do conjunto de dados de teste.

O MSE é uma métrica que incorpora o bias e a variância de um modelo preditivo. É uma das métricas de desempenho de algoritmos de classificação mais prevalentes. O MSE de um algoritmo de classificação calcula-se como apresentado na equação 3.11 (Mitchell et al. 2016):

$$MSE(\hat{H}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{3.11}$$

O erro quadrático médio é uma medida de desempenho bastante utilizada precisamente por combinar estas entidades estatísticas e permitir saber o quão tendencioso e o quão preciso é um classificador (Mitchell et al. 2016).

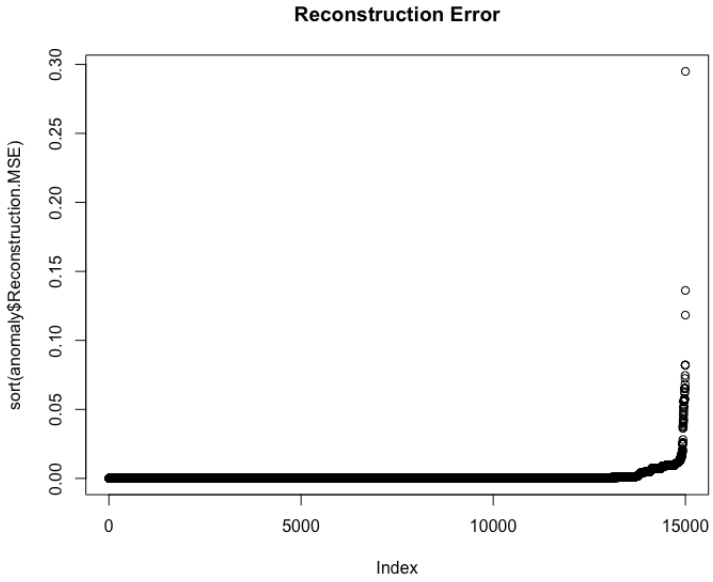

Figura 3.17: Gráfico de reconstrução do erro quadrático médio

De seguida é construido um gráfico da reconstrução do erro quadrático médio como apresentado na figura 3.17. Este gráfico representa um exemplo de um teste executado a uma amostra de teste do conjunto de dados ISCX. Verifica-se que a uma certa altura o erro quadrático médio aumenta. Isto significa que o modelo não conseguiu identificar esses registos de dados corretamente, o que poderá ser considerado uma anomalia. Então delineou-se um limite no gráfico, neste caso igual a 0,002 sendo todos os registos superiores a esse erro considerados como anomalias.

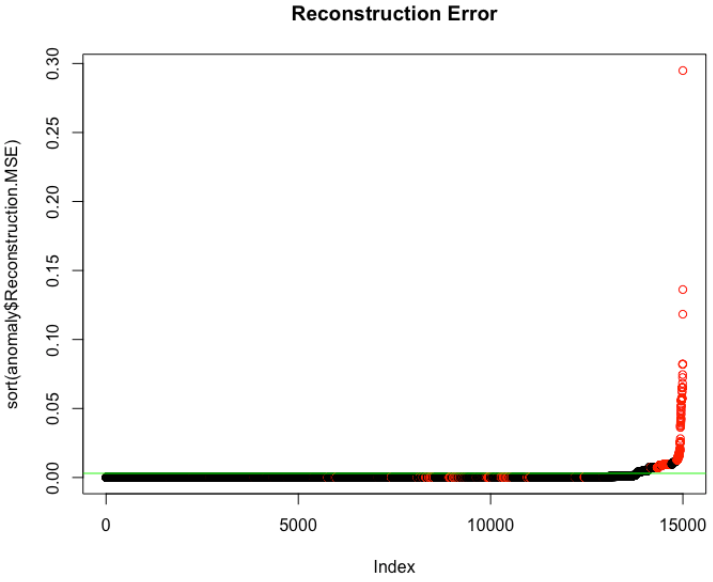

Figura 3.18: Gráfico de reconstrução do erro quadrático médio

Para uma melhor perceção coloriram-se os registos, os de cor vermelha correspondem à classe ataque enquanto que os registos de cor preta correspondem à classe normal. Como

se pode ver no gráfico 3.18 a maior parte dos registos da classe ataque encontram-se acima do limite traçado a verde.

3.4 Conclusão

Antes de aplicar os algoritmos preditivos ao conjunto de dados NSL-KDD e ISCX, é importante que os dados sejam analisados. Essa análise, que pode ser realizada por técnicas de estatística e de visualização, permite uma melhor compreensão da distribuição dos dados e pode suportar a escolha da melhor forma para modelar o problema. A fase de pré-processamento, pode facilitar o processo de aprendizagem e melhorar o desempenho dos algoritmos preditivos, uma vez que estas técnicas podem eliminar ou reduzir problemas presentes nos dados.

Neste Capitulo foram apresentadas diversas técnicas utilizadas no pré-processamento como a utilização de uma técnica de amostragem para selecionar subconjuntos representativos de dados. Foram apresentadas alternativas para a transformação de dados, com o intuito de facilitar o seu uso por diferentes algoritmos preditivos, tais como a normalização e discretização de dados numéricos. Foi descrita a importância da redução da dimensionalidade dos dados. E por fim apresentados os algoritmos preditivos escolhidos a serem testados na deteção de intrusões nos conjuntos de dados mencionados.

Capítulo 4

Avaliação e Comparação dos resultados obtidos pelos Classificadores

Neste capítulo abordam-se as métricas de avaliação a utilizar nas técnicas de Machine Learning aplicadas nos conjuntos de dados NSL-KDD e ISCX. De seguida são apresentados os resultados dos classificadores com as diferentes técnicas de pré-processamento utilizadas, tanto em problemas binários como em problemas multi-classe. Para finalizar são comparados os resultados das melhores técnicas de pré-processamento aplicadas em conjunto com um determinado classificador de modo a determinar qual o modelo preditivo mais adequado para problemas de deteção de intrusões.

4.1 Métricas de avaliação

As métricas de avaliação mais utilizadas no desempenho dos algoritmos podem ser divididas em três categorias principais (Mitchell et al. 2016):

- **Métricas baseadas em limites**: os algoritmos de classificação produzem frequentemente um output entre 0 e 1, sendo que este valor não reflete necessariamente a probabilidade do output pertencer a uma ou outra classe. Para discernir a que classe o output pertence é necessário definir um limite, normalmente 0.5, que discrimine entre as classes. As métricas baseadas em limites medem a taxa de exemplos corretamente classificados tendo em conta um determinado limite. Incluem-se neste tipo métricas como a *accuracy* e *F1-score*. As métricas baseadas em limites são adequadas quando é necessário que um classificador tenha uma determinada taxa de casos corretos. Estas métricas são fortemente afetadas por desequilíbrios de classe, devendo a distribuição das classes ser semelhante em dados de treino e dados operacionais (Mitchell et al. 2016).
- **Métricas de ranking**: uma métrica de *ranking* avalia a capacidade de um classificador ordenar os seus outputs com respeito a uma classe. Os valores preditos não têm interesse para as métricas de *ranking*, o que interessa é o quão corretamente os outputs refletem a ordenação das duas classes. Uma das métricas deste tipo mais utilizadas é a Área Sob a Curva (AUC) (Mitchell et al. 2016).
- **Métricas baseadas em probabilidade**: as métricas deste tipo definem a probabilidade esperada do output de um classificador estar corretamente classificado. As métricas

baseadas em probabilidade são apropriadas para avaliar o desempenho geral de um algoritmo de aprendizagem, uma vez que um algoritmo que tenha um bom desempenho de acordo com uma métrica de probabilidade também terá um bom desempenho com base numa métrica de ranking. No entanto, o oposto não se verifica necessariamente. O MSE inclui-se neste tipo de métrica (Mitchell et al. 2016).

4.1.1 Matriz de confusão

A matriz de confusão é uma matriz de N x N em que N representa o número de classes que um modelo pode prever. No caso dum problema binário, $N = 2$ o que dá origem a uma matriz de 2 x 2. Cada linha da matriz representa as instâncias preditas como membros de uma classe e cada coluna representa as instâncias que são de facto membros da classe (ou vice-versa). O nome "matriz de confusão" advém do facto de tornar mais fácil perceber se o sistema está a confundir duas classes (atribuir a etiqueta de uma às instâncias da outra). Trata-se de um tipo especial de tabela de contingência 4.1 (Souza 2009).

Tabela 4.1: Matriz de confusão (Souza 2009)

Um exemplo de uma matriz de confusão relativa ao tema do projeto pode ser apresentada como na tabela 4.2:

Tabela 4.2: Exemplo de Matriz confusão a utilizar no projeto

Prever classe Ataque		
	Ataque	Normal
Ataque	VP	FP
Normal	FN	VN

4.1.2 Accuracy

A *accuracy* é uma métrica que avalia a proporção de resultados verdadeiros face ao número total de amostras classificadas em problemas de classificação binários. A *accuracy* varia entre 0 e 1. O valor zero significa que nenhuma instância foi corretamente classificada e o valor um significa que todas as instâncias foram corretamente classificadas. A *accuracy* depende dos seguintes parâmetros (Mitchell et al. 2016):

- VP: Verdadeiros Positivos, número de instâncias no conjunto de dados positivos que foram corretamente classificadas.
- VN: Verdadeiros Negativos, número de instâncias no conjunto de dados negativos que foram corretamente classificadas.
- FP: Falsos Positivos, número de instâncias no conjunto de dados positivos que foram incorretamente classificadas.
- FN: Falsos Negativos, número de instâncias no conjunto de dados negativos que foram incorretamente classificadas.

A formula geral da *accuracy* é calculada usando a equação 4.1:

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN} \tag{4.1}$$

A *accuracy* é uma medida baseada em limites que tem em consideração um único limite, normalmente 0.5. Consequentemente, a *accuracy* pode ser enganadora se o algoritmo de aprendizagem for treinado com dados cuja distribuição de classes não é equilibrada (Mitchell et al. 2016). Nesses casos, o algoritmo pode ter uma *accuracy* bastante elevada sem ter qualquer sensibilidade (taxa de verdadeiros positivos ou *recall* 4.3). A associação de pesos aos falsos negativos e aos falsos positivos permite ajustar o impacto causado por diferentes tipos de erros de acordo com a gravidade dos mesmos (Mitchell et al. 2016).

4.1.3 Precision e Recall

Esta duas métricas são normalmente utilizadas em conjunto. A *precison* responde à questão: "Quantas observações são realmente intrusões de todas as observações preditas como intrusões pelo classificador". A *precison* quantifica o desempenho de um classificador de não classificar uma observação da classe normal como intrusão. É o rácio de detetar corretamente observações positivas do total de todas as observações detetadas como positivas (Turi Platform 2016). O valor desta métrica varia entre 0 e 1 e valores altos significam um baixo rácio de falsos positivos. A equação da *precision* é dada pela formula 4.2:

$$Precision = \frac{VP}{VP + FP} \tag{4.2}$$

A métrica *recall* responde à questão: "Quantas observações foram identificadas como intrusões, de todas as observações da classe intrusão". O *Recall* é o rácio de detetar corretamente observações positivas, de todas as observações da classe positiva, neste caso da classe ataque. O resultado desta métrica encontra-se no intervalo entre 0 e 1. O calculo da métrica *recall* é representado pela equação 4.3:

$$Recall = \frac{VP}{VP + FN} \tag{4.3}$$

4.1.4 F1-Score

A métrica F_1 inicialmente introduzida por Rijsbergen (1979) é uma média harmónica que combina as medidas *recall* (r) e *precision* (p) com um peso igual na forma da equação 4.4:

$$F_1(r, p) = \frac{2rp}{r + p} \tag{4.4}$$

Esta medida tem em conta os falsos negativos e os falsos positivos. Não é tão intuitivo de perceber como a *accuracy*, mas é especialmente útil se existir uma distribuição desequilibrada das classes (Yang e X. Liu 1999). F_1 pode ser interpretada como uma média de pesos entre a *precision* e o *recall*, onde o valor de F_1 atingi o seu máximo em 1 e mínimo em 0 (Yedidia 2016).

4.2 Avaliação dos algoritmos

Foram efetuadas experiências para dois problemas diferentes, problema binário (2 classes) e problema multi-classes (5-classes). Para cada um dos algoritmos selecionados foram realizadas 10 experiências. Em cada experiência foram aplicadas as 6 combinações das técnicas de pré-processamento referidas no capítulo 3.2 e selecionadas as amostragens referidas no capítulo 3.2.1 com registos aleatórios para os conjuntos de dados de treino e de teste. Registou-se o valor das métricas *accuracy* e F_1 e por fim calculou-se a média das 10 experiências para estas métricas. Como existe um desequilibro de classes nestes conjuntos de dados, sendo a classe "Normal"composta por 90% dos dados, então a análise de comparação de desempenho dos classificadores será focada apenas na métrica F_1 para a classe "Ataque"sendo a *accuracy* meramente informativa. De seguida serão apresentadas as diferentes técnicas de pré-processamento e o resultado obtido pelos algoritmos de classificação num teste de 2 classes (classe normal e classe de ataque). Para finalizar será apresentado o resultado do desempenho dos algoritmos num teste multi-classes (classe normal + classes dos vários tipos de ataques) em que cada algoritmo utiliza a melhor combinação de técnicas de pré-processamento para identificação destas classes.

4.2.1 IF + ZScore + RFE

Nesta combinação, as técnicas de pré-processamento foram aplicadas da seguinte forma nos conjuntos de dados NSL-KDD e ISCX:

- Os dados foram discretizados através da técnica Igual frequência (IF);
- Os dados foram normalizados através da técnica ZScore;
- Procedeu-se à redução de atributos através da técnica Recursive Feature Elimination (RFE);

Na figura 4.1 estão representados a percentagem de *accuracy* e F_1 para cada algoritmo no conjunto de dados NSL-KDD. Verifica-se que para esta combinação o classificador SVM foi o que obteve melhor resultado com uma *accuracy* de 93,75% e F_1 igual a 70,73%. A árvore de decisão C4.5 não conseguiu criar um modelo preditivo com estas técnicas de pré-processamento de dados.

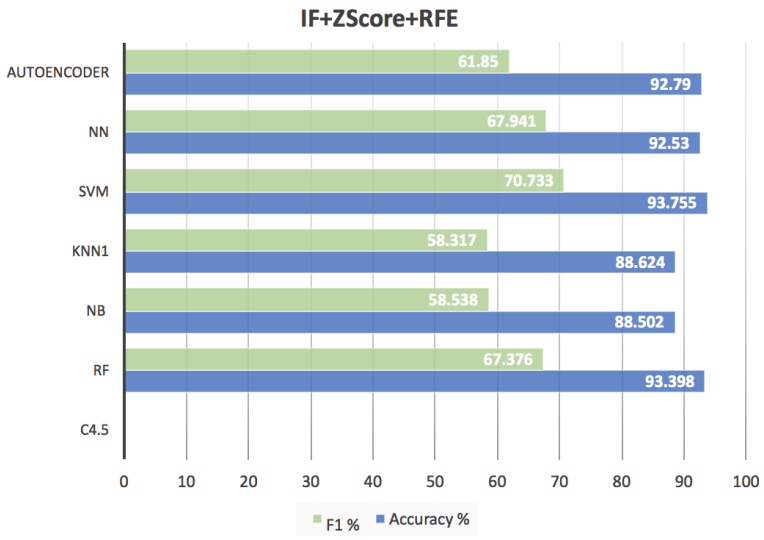

Figura 4.1: Desempenho dos classificadores utilizando as técnicas IF+ZScore+RFE no conjunto de dados NSL-KDD

O gráfico de barras presente na figura 4.2 diz respeito ao conjunto de dados ISCX. Pode-se visualizar na imagem que o algoritmo KNN e as árvores de decisão Random Forest (RF) obtiveram melhores resultados que os restantes classificadores, sendo o KNN o vencedor atingindo o máximo de valor na *accuracy* e F_1, respetivamente 99,75% e 99,74%. O algoritmo Naive Bayes (NB) foi o que obteve pior classificação cujo valor de $F_1 = 47,03\%$.

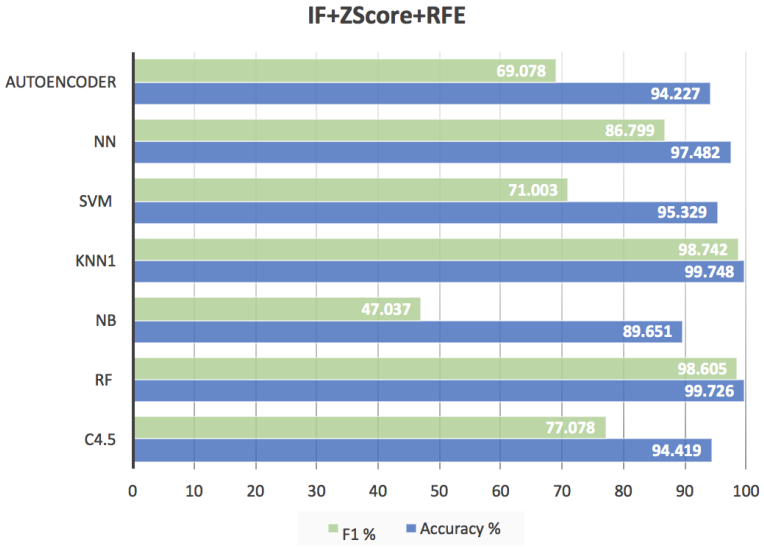

Figura 4.2: Desempenho dos classificadores utilizando as técnicas de pré-processamento EF+ZScore+RFE no conjunto de dados ISCX

4.2.2 IF + MinMax + RFE

Na seguinte combinação, os dados foram discretizados com a técnica Igual frequência (IF) e normalizados com a técnica MinMax. Por fim realizou-se a operação de redução de dimensionalidade dos conjuntos de dados NSL-KDD e ISCX através da técnica Recursive Feature Elimination (RFE). Em NSL-KDD com esta combinação de técnicas de pré-processamento, é possível visualizar na figura 4.3 que os algoritmos de redes neuronais (NN), as SVM e RF apresentaram o resultado mais elevado com o valor de F_1 a variar entre os 65% e 70% e uma *accuracy* aproximada de 93%. O classificador NB e KNN1 obtiveram os resultados mais baixos com F_1 igual a 58,6% e 59,2% respetivamente.

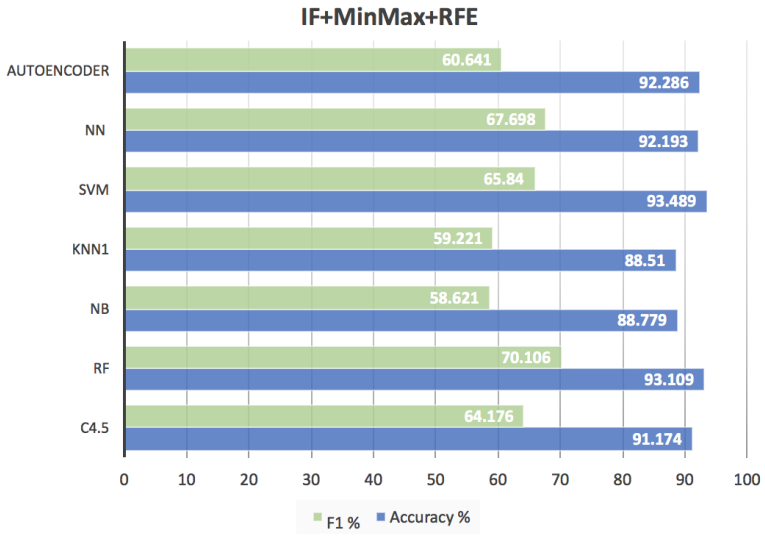

Figura 4.3: Desempenho dos classificadores com as técnicas IF+MinMax+RFE no conjunto de dados NSL-KDD

Quanto ao conjunto de dados ISCX, a figura 4.4 indica que o algoritmo KNN1 foi o melhor $F_1 = 98,05\%$ seguida do algoritmo RF $F_1 = 90,1\%$. Pode-se verificar também que as SVM não conseguiram criar um modelo de deteção de intrusões utilizando as técnicas de pré-processamento indicadas.

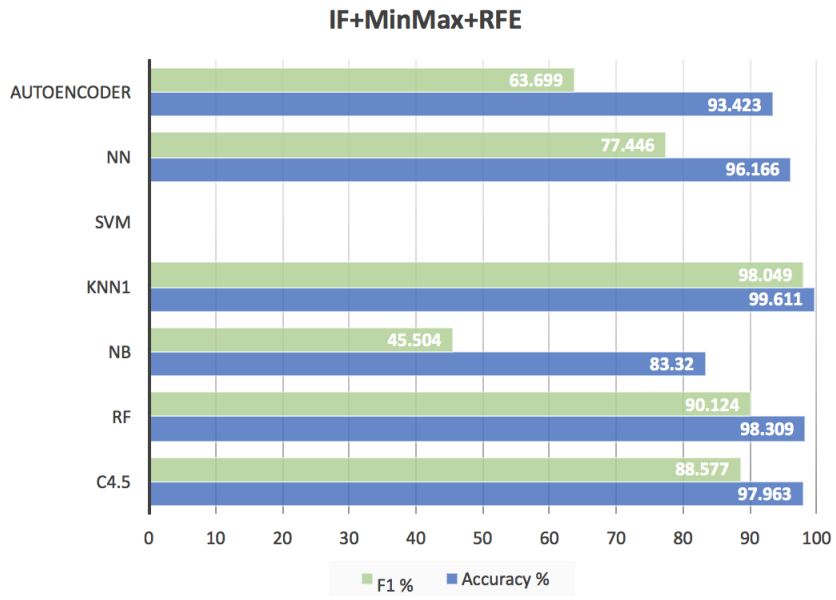

Figura 4.4: Desempenho dos classificadores com as técnicas IF+MinMax+RFE no conjunto de dados ISCX

4.2.3 MinMax + RFE

Foram utilizadas duas técnicas de pré-processamento aos conjuntos dados NSL-KDD e ISCX, normalização com MinMax e redução de dimensionalidade através de RFE. Neste caso os dados não foram discretizados. A imagem 4.5 representa o gráfico de barras do desempenho dos algoritmos na deteção de intrusões aplicados em NSL-KDD. A árvore de decisão C4.5 e as NN atingiram o maior valor de F_1 aproximadamente 70% e *accuracy* 93%. Com estas técnicas de pré-processamento o algoritmo NB foi considerado o pior na classificação dos dados obtendo 44,2% em F_1, seguido do *Autoencoder* com um valor baixo de $F_1 = 54,2\%$.

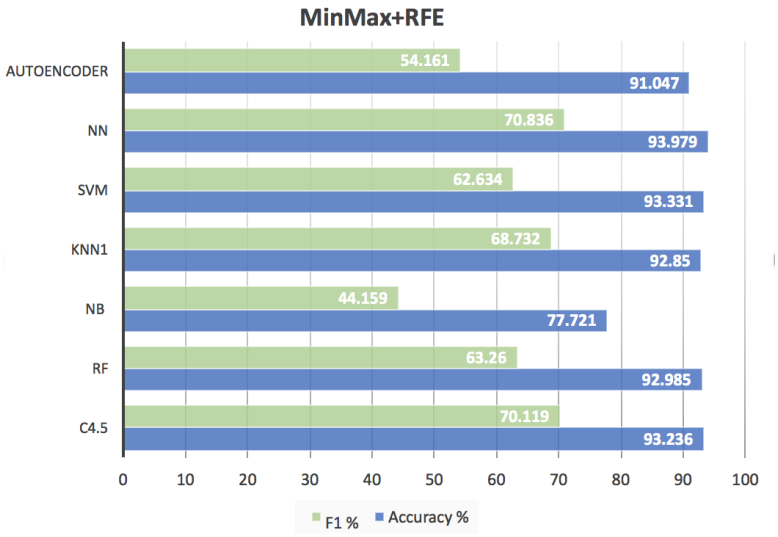

Figura 4.5: Desempenho dos classificadores com as técnicas MinMax+RFE no conjunto de dados NSL-KDD

Na figura 4.6 referente ao conjunto de dados ISCX, verifica-se que o classificador SVM não conseguiu criar um modelo para classificar as observações. Pode-se afirmar que a técnica de normalização MinMax não permite que este algoritmo consiga realizar uma separação correta aos dados no hiperplano através dos vetores de suporte. Chega-se a esta conclusão pois na combinação IF+MinMax+RFE as SVM também não conseguiram criar um modelo preditivo para o conjunto de dados ISCX enquanto que com a combinação IF+ZScore+RFE o algoritmo conseguiu criar um modelo com um valor razoável de F_1 de 71%. A diferença destas técnicas de combinação encontram-se na técnica de normalização, sendo então o responsável por este problema a técnica de normalização MinMax. Pode-se visualizar também no gráfico que o algoritmo KNN atingiu um excelente resultado com $F_1 = 98\%$ e $accuracy = 99,6\%$

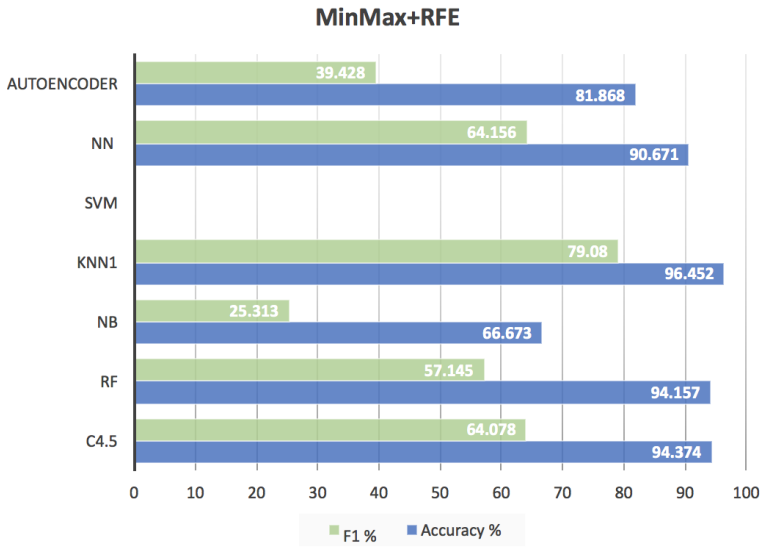

Figura 4.6: Desempenho dos classificadores com as técnicas MinMax+RFE no conjunto de dados ISCX

4.2.4 ZScore + RFE

Na combinação ZScore + RFE os dados foram normalizados com a técnica ZScore e os atributos dos conjuntos de dados reduzidos com a técnica RFE. Não foram aplicadas técnicas de discretização aos dados. Os algoritmos que obtiveram melhores resultados no conjunto de dados NSL-KDD (figura 4.7) foram as SVM e NN cujo valor de F_1 é aproximadamente 70%, por outro lado, as árvores de decisão RF e C4.5, a rede neuronal *Autoencoder* e o algoritmo Naive Bayes não conseguiram criar um bom modelo preditivo pois o resultado de F_1 é inferior a 50%.

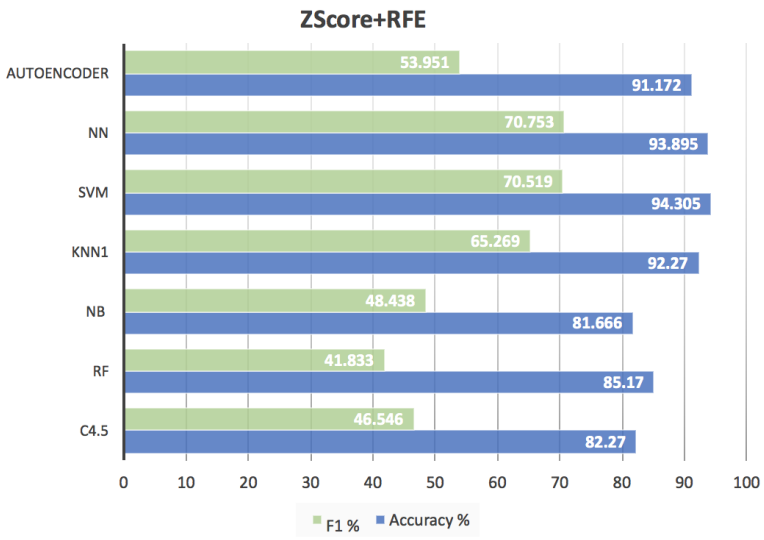

Figura 4.7: Desempenho dos classificadores com as técnicas ZScore+RFE no conjunto de dados NSL-KDD

Relativamente à predição de intrusões no conjunto de dados ISCX representado na figura 4.8, conclui-se que as RF e o classificador KNN1 atingiram os melhores resultados, enquanto que o algoritmo Naive Bayes só conseguiu atingir os 23% em F_1.

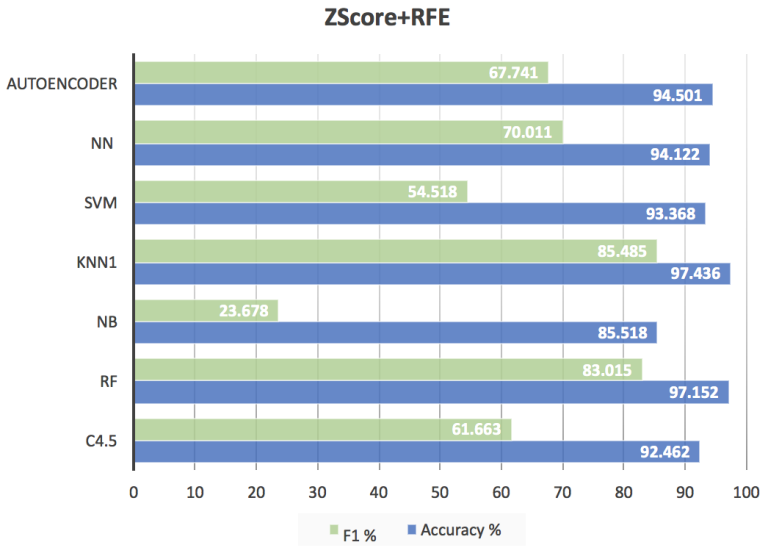

Figura 4.8: Desempenho dos classificadores com as técnicas ZScore+RFE no conjunto de dados ISCX

4.2.5 IF + RFE

Para esta combinação, não foram aplicadas técnicas de normalização nos conjuntos de dados NSL-KDD e ISCX. Os dados foram apenas discretizados com a técnica IF e de seguida procedeu-se à redução de dimensionalidade através da técnica RFE. Na figura 4.9, relativa ao conjunto de dados NSL-KDD, verifica-se que os classificadores NN, SVM e RF

obtiveram a taxa mais alta de F_1 com esta combinação a rondar os 69% e uma *accuracy* próxima dos 93%. O classificador *Autoencoder* não conseguiu criar um modelo preditivo com esta combinação devido à ausência da normalização dos dados.

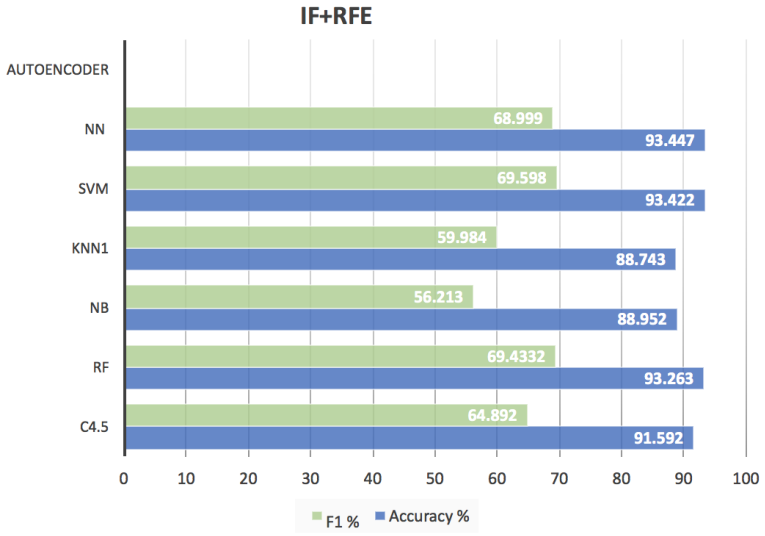

Figura 4.9: Desempenho dos classificadores com as técnicas IF+RFE no conjunto de dados NSL-KDD

Através da figura 4.10 referente ao conjunto de dados ISCX observa-se que para esta combinação de técnicas os classificadores RF e KNN1 atingiram o valor mais alto de F_1 e *accuracy*. Enquanto que o modelo criado pelo algoritmo Naive Bayes obteve apenas 46,5% de F_1. Neste conjunto de dados o *Autoencoder* também não conseguir criar um modelo preditivo com a combinação IF+RFE.

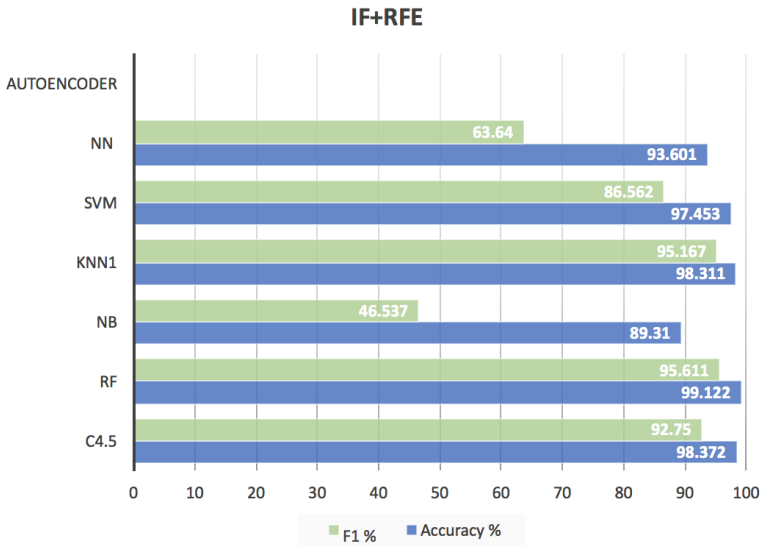

Figura 4.10: Desempenho dos classificadores com as técnicas IF+RFE no conjunto de dados ISCX

4.2.6 RFE

Nesta secção pretende-se analisar os resultados dos algoritmos nos conjuntos de dados NSL-KDD e ISCX, com a aplicação de apenas uma técnica de pré-processamento de dados, a técnica RFE relativa à redução de atributos, ou seja, não foi aplicada qualquer técnica de normalização e discretização aos dados. Como se pode ver na figura 4.11, referente ao conjunto de dados NSL-KDD, as SVM e o *Autoencoder* não conseguirem criar um modelo preditivo devido à não normalização e/ou discretização dos dados. Verifica-se também que as RF e o algoritmo KNN1 atingiram um valor aproximado de 70% de F_1. Apenas com esta técnica no conjunto de dados ISCX visível na figura 4.12, as árvores de decisão RF e C4.5 obtiveram excelentes valores tanto de F_1 como de *accuracy* a rondar os 99% nestas duas métricas.

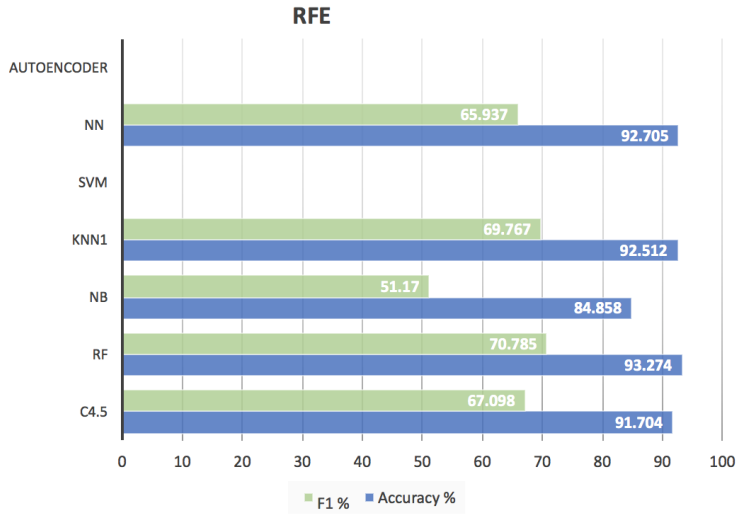

Figura 4.11: Desempenho dos classificadores com a técnica RFE no conjunto de dados NSL-KDD

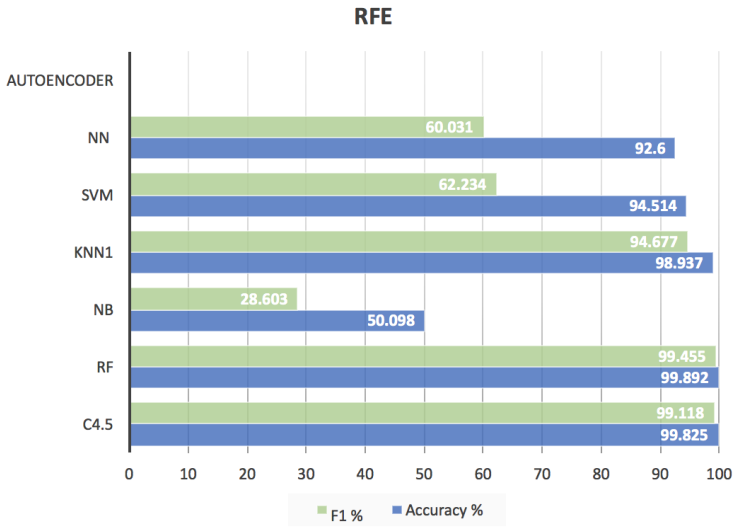

Figura 4.12: Desempenho dos classificadores com a técnica RFE no conjunto de dados ISCX

4.2.7 Resultados das Melhores Combinações de Técnicas de Pré-processamento de Dados

Para uma melhor comparação dos algoritmos procedeu-se á construção de um gráfico para cada conjunto de dados, representando o desempenho dos algoritmos com a sua melhor combinação de técnicas de pré-processamento. Na figura 4.13 relativa ao conjunto de dados NSL-KDD, verifica-se que a combinação IF+MinMax+RFE foi a melhor combinação para os algoritmos NB e RF. O classificador NB, mesmo com a melhor combinação de técnicas de pré-processamento, obteve o valor mais baixo de F_1 em comparação com todos os outros algoritmos. Por sua vez, o algoritmo RF obteve um dos valores mais altos de de F_1 com esta combinação. O melhor resultado obtido, foi do algoritmo NN com a combinação MinMax+RFE, não sendo este resultado muito disperso de outros obtidos pelos classificadores C4.5, KNN, SVM e RF, em que estes valores rondam os 70% de F_1.

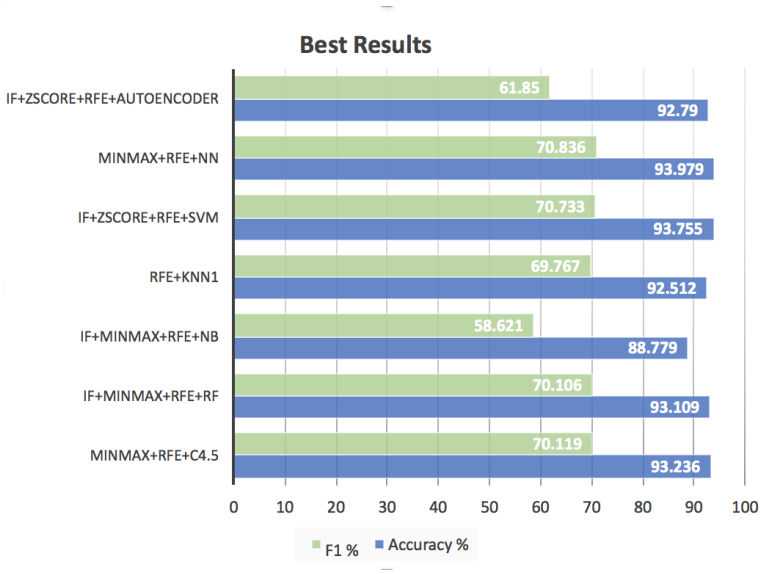

Figura 4.13: Desempenho dos classificadores combinado com as melhores técnicas de pré-processamento no conjunto de dados NSL-KDD

No conjuntos de dados ISCX, representado na figura 4.14 , observa-se que os valores mais altos de F_1 corresponde aos algoritmos C4.5 e RF a rondar os 99% utilizando a técnica de pré-processamento RFE. A combinação IF+MinMax+RFE foi a melhor combinação para os algoritmos NN, KNN1 e NB. O classificador NB, mesmo com a melhor combinação de técnicas de pré-processamento, obteve o valor mais baixo de F_1 em comparação com todos os outros algoritmos, enquanto que os outros dois classificadores (NN e KNN1) atingiram bons resultados, $F_1 = 88,8\%$ para NN e $F_1 = 98,7\%$ para KNN1. Quanto á combinação IF+RFE foi a melhor para os algoritmos SVM e *Autoencoder*, sendo este ultimo o segundo classificador com o pior resultado de F_1.

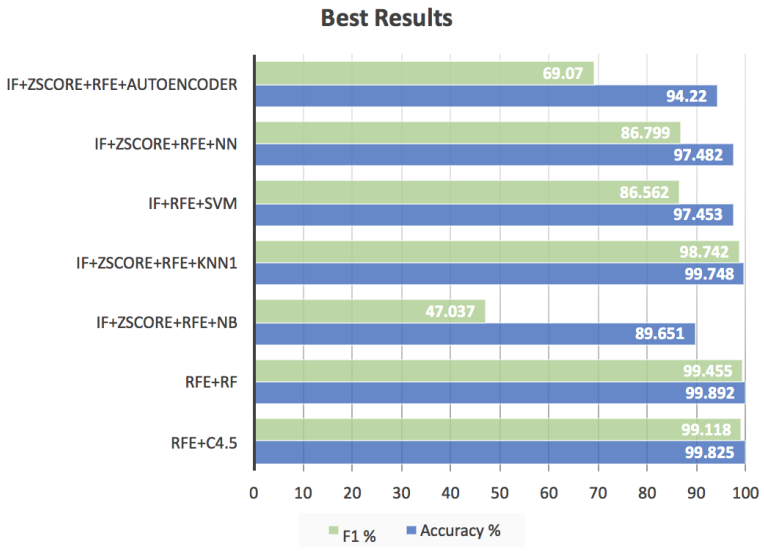

Figura 4.14: Desempenho dos classificadores combinado com as melhores técnicas de pré-processamento no conjunto de dados ISCX

4.2.8 Resultados dos Algoritmos na Deteção de Intrusões Multi-Classe

Para deteção de intrusões em multi-classes foi excluído o algoritmo *Autoencoder*, pois não se encontra preparado para este tipo de problemas. A classificação multi-classes, permite averiguar o desempenho dos algoritmos na deteção dos vários tipos de intrusões existentes. No gráfico da figura 4.15 referente ao conjunto de dados NSL-KDD, está representado a *accuracy* de cada algoritmo e o valor de F_1 obtido na classificação de cada classe. As classes apresentadas nesta figura, encontram-se descritas no capítulo 3.1.1. No referido gráfico, é de salientar a fraca deteção das classes 3 e 4 relativas à categoria de ataques R2L e U2R. Isto acontece devido ao facto de as observações destas classes representarem uma percentagem muito pequena tanto no conjunto de treino (0,29% para R2L e 0,15% para U2R) como o de teste (1,76% para R2L e 0,44% para U2R). O algoritmo SVM foi o que obteve melhor resultado de F_1 na classificação do ataque do tipo *Probe* (classe 2), seguido das árvores de decisão RF e das redes neuronais. Estes dois últimos algoritmos mencionados foram os que obtiveram resultados de F_1 mais altos relativamente ao ataque do tipo DoS (classe 1).

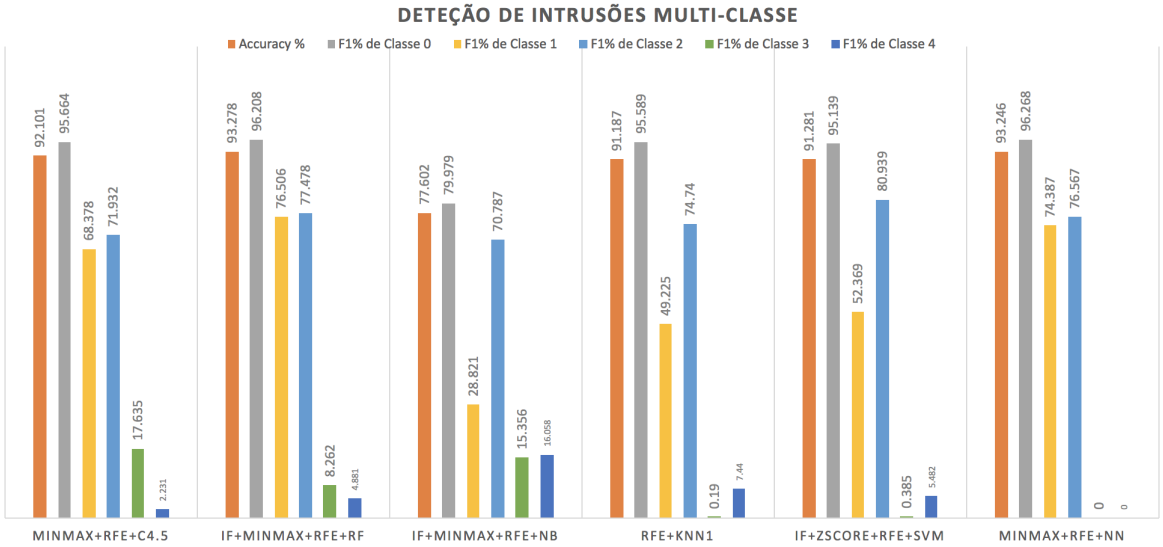

Figura 4.15: Desempenho dos classificadores na deteção de intrusões multi-classe, no conjunto de dados NSL-KDD

A figura 4.16 apresenta os resultados dos classificadores no conjunto de dados ISCX na deteção dos ataques descritos em 2.5.3. Neste gráfico a Classe 0 corresponde à classe Normal e as classes 1, 2, 3 e 4 correspondem aos ataques capturados no dia 13, 14, 15 e 17 de Julho de 2010. Verifica-se que todas as classes foram detetas com sucesso pelas árvores de decisão C4.5 e RF, com valores de *accuracy* e F_1 a rondar os 99%. O algoritmo KNN foi o 3º melhor classificador na deteção de intrusões com resultados muito próximos das árvores de decisão. Quanto aos algoritmos SVM e NN, conseguirem bons resultados expecto na deteção da classe 3. Por último, o algoritmo NB foi o que obteve piores resultados nos 2 conjuntos de dados, isto por não ser um bom algoritmo em problemas de dados não linearmente separáveis.

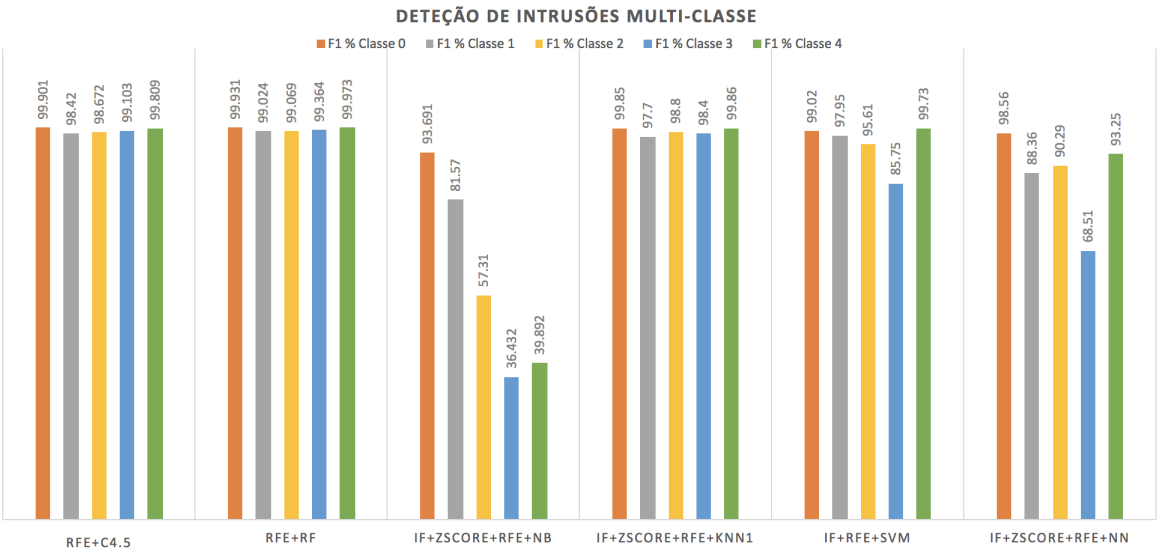

Figura 4.16: Desempenho dos classificadores na deteção de intrusões multi-classe, no conjunto de dados ISCX

4.3 Conclusão

Foram apresentadas ao longo deste capítulo várias análises aos resultados obtidos na classificação dos conjuntos de dados NSL-KDD e ISCX. Após a observação dos resultados, conclui-se que não existe uma combinação de técnicas de pré-processamento corretas a utilizar com um determinado algoritmo, pois as técnicas variam consoante a estrutura dos dados. Quanto aos classificadores, verificou-se que obtiveram quase todos uma boa classificação no conjunto de dados ISCX, isto porque não foram testados com ataques novos, apenas ataques conhecidos pelos classificadores. Relativamente ao conjunto de dados NSL-KDD os resultados da classificação são mais baixos, pois neste conjunto de dados os algoritmos são testados com ataques novos, mesmo assim, estes resultados não diferem muito dos resultados encontrados na literatura utilizando este mesmo conjunto de dados. No geral verificou-se que os algoritmos C4.5, *Random Forest* e *K-Nearest Neighbor* obtiveram praticamente os mesmos resultados de classificação nos dois conjuntos de dados, sendo estes considerados mais consistentes na deteção e previsão de intrusões. Tendo em conta o objetivo principal desta dissertação, ou seja, o desenvolvimento de um modelo preditivo a implementar no SDI do projeto SASSI, pode-se concluir que a aplicação do classificador C4.5 poderá ser a solução mais viável para deteção de ataques conhecidos, pois obteve uma taxa de erro inferior a 1% em quase todas as intrusões contidas no conjunto de dados ISCX, deste modo pode-se também descartar a solução alternativa descrita na secção 2.8. Descarta-se também o uso dos classificadores *Random Forest* e *K-Nearest Neighbor* que obtiveram bons resultados na deteção de intrusões mas são computacionalmente mais dispendiosos que o C4.5. No entanto nenhum dos modelos utilizados nesta dissertação conseguiu resultados satisfatórios na deteção de ataques não-conhecidos, também designados por ataques do tipo zero. Para combater o problema identificado será necessário explorar novas abordagens, focadas principalmente em algoritmos de aprendizagem não-supervisionada, pois estes são treinados com observações sem rótulos criando um modelo preditivo que classifica as observações através dos diferentes padrões extraídos de um conjunto de dados.

Capítulo 5

Conclusão

5.1 Resumo

Este trabalho propõe a análise de diferentes técnicas de *machine learning*, de modo a selecionar a melhor técnica a ser implementada num SDI para deteção e previsão de ataques numa rede interna de uma organização. Para uma melhor compreensão do tema, foi elaborado uma recolha de informação relacionada com os princípios da segurança de informação, os vários tipos de anomalias existentes nas comunicações de um sistema computacional, como uma análise arquitetónica dos sistemas de deteção de intrusões.

De seguida foram apresentadas vários abordagens relacionadas com a deteção de intrusões aplicando diferentes técnicas de *machine learning* em conjuntos de dados públicos. Foi também descrito a arquitetura do projeto SASSI, cujo o seu objetivo foca-se na criação de uma ferramenta que apoie os administradores de redes informáticas no processo de tomada de decisão relacionado com problemas de segurança. Após a apresentação do modelo do projeto SASSI, foi elaborado a visão da solução da componente a desenvolver pelo GECAD e a importância da contribuição desta dissertação nesse mesmo desenvolvimento. Utilizaram-se os conjuntos de dados NSL-KDD e ISCX por serem bastante completos, contendo inúmeras observações de atividade normal numa rede como diferentes tipos de atividade maliciosa e serem muito utilizados na literatura para este tipo de problemas.

Foram realizadas análises aos conjuntos de dados NSL-KDD e ISCX, de modo a perceber como os dados se encontram estruturados, a correlação entre cada atributo, e a distribuição das classes. Depois procedeu-se ao tratamento dos dados aplicando várias técnicas de pré-processamento, separação, normalização, discretização e redução de dimensionalidade. Com os dados tratados, selecionaram-se 7 algoritmos preditivos de aprendizagem supervisionada e não-supervisionada para processarem os dados em NSL-KDD e ISCX com a finalidade de detetar as intrusões contidas nestes conjuntos de dados.

Para finalizar criaram-se gráficos com os resultados obtidos na deteção das intrusões através das métricas *accuracy* e *F1* de cada algoritmo para cada combinação de técnicas de pré-processamento em problemas binários e multi-classes. Os gráficos auxiliam na comparação dos algoritmos preditivos para seleção do melhor classificador a ser testado e/ou implemento no sistema SASSI.

5.2 Contributos do trabalho

Em termos científicos, a dissertação contribuiu para o aprofundamento de conhecimentos na área de *machine learning* aplicada ao contexto da segurança de informação de sistemas computacionais em rede. Neste tema foi possível identificar e compreender as diferentes abordagens utilizadas na deteção e previsão de intrusões. A presente dissertação contribui também para autores que pretendam realizar um estudo comparativo com a abordagem adotada na mesma.

Este projeto visa desenvolver um modelo de deteção e previsão de intrusões através do estudo e aplicação de várias técnicas no âmbito do *machine learning*. O principal contributo desta dissertação é direcionado para o projeto SASSI pois corresponde a uma das tarefas a desenvolver nesse projeto. Para concretização deste contributo, realizaram-se as seguintes tarefas que permitiram identificar diferentes técnicas de *machine learning* a serem testadas, algumas lacunas nos conjuntos de dados e perceber quais os melhores métodos estudados a aplicar em problemas relacionados com a deteção e previsão de intrusões em redes:

- A realização do estado da arte permitiu identificar um conjunto de técnicas e abordagens a aplicar em problemas de deteção de intrusões:
  - Decidiu-se utilizar os dois conjuntos de dados NSL-KDD e ISCX por serem bastante completos, contendo inúmeras observações de atividade normal numa rede, como diferentes tipos de atividade maliciosa e por serem muito utilizados na literatura para este tipo de problemas;
  - Exploraram-se diferentes técnicas de pré-processamento dos dados nomeadamente técnicas de normalização, discretização e redução de dimensionalidade;
  - Selecionaram-se vários algoritmos de classificação, sendo estes o C4.5, *Random Forest, Autoencoder, Naive Bayes*, redes Neuronais, SVM e KNN para elaboração dos casos de estudo devido aos bons resultados obtidos nas abordagens estudadas;
- Através da analise exploratória aos conjuntos de dados observou-se o seguinte:
  - O conjunto de dados NSL-KDD contém poucas observações de duas categorias de ataque dificultando aos algoritmos a sua identificação;
  - Tanto o conjunto de dados NSL-KDD como o ISCX apresentam atributos irrelevantes e/ou redundantes;
  - Os dados dos conjuntos NSL-KDD e ISCX não são linearmente separáveis;
  - O conjunto de dados ISCX possui registos de atividade de rede bem simulada, mas perde na falta da diversidade de intrusões;
- Após a aplicação dos algoritmos nos conjuntos de dados verificou-se que:
  - Os resultados das melhores combinações de técnicas de pré-processamento utilizadas em conjunto com os algoritmos diferem nos dois conjuntos de dados, isto porque as estrutura de dados NSL-KDD não é a mesma que a de ISCX;
  - As SVM e redes neuronais obtiveram os melhores resultados na classificação binária do conjunto de dados NSL-KDD, no entanto estes resultados não foram

significativamente superiores em relação aos classificadores C4.5, *Random Forest* e KNN;

- Os classificadores C4.5 e *Random Forest* obtiveram os melhores resultados na deteção de intrusões em problemas multi-classes no conjunto de dados ISCX. Em NSL-KDD, estes algoritmos obtiveram bons resultados na deteção de intrusões em todas as classes exceto nas classes U2R e R2L;
- A abordagem aplicada com o *autoencoder* não obteve resultados satisfatórios;
- O classificador C4.5 foi considerado a melhor escolha para uma possível implementação no sistema SASSI por ser o algoritmo com menores custos computacionais e maior taxa de deteção de intrusões;
- As abordagens apresentadas não foram suficientemente boas na deteção de novas intrusões, pelo que será necessário estudar e aplicar novos métodos relacionados com este tipo de problemas;

Mesmo não obtendo um modelo preditivo completo que detete intrusões conhecidas e desconhecidas, as técnicas estudadas serão novamente aplicadas e testadas num conjunto de dados real desenvolvido pela *VisionTechLab*.

5.3 Trabalho Futuro

No que diz respeito ao trabalho futuro, será criado um conjunto de dados que irá conter registos de atividade normal e um leque de vários tipos de atividade intrusiva. Estes dados serão obtidos pelos sensores do sistema SASSI. Serão aplicadas técnicas de *machine learning* ao conjunto de dados criado como as que foram apresentadas nesta dissertação. Serão aplicadas outras abordagens com algoritmos de aprendizagem não-supervisionada como o caso do *Autoencoder*. A abordagem apresentada para este classificador não obteve bons resultados mas existem mais abordagens a serem exploradas neste algoritmo devido à sua complexidade e vasta quantidade de parâmetros a afinar. As técnicas de aprendizagem não-supervisionadas são uma possível solução para o aumento da taxa de deteção de ataques não conhecidos. Será também realizado um levantamento da taxionomia detalhada dos vários ataques existentes em redes, como as consequências e respetivo impacto associado à ocorrência dos diversos tipos de intrusão e falhas do sistema. A informação recolhida será necessária para desenvolver metodologias inteligentes de prevenção, para que seja possível, ser identificadas as regras de atuação mais adequadas a cada circunstância. Será elaborado um sistema de apoio à decisão capaz de indicar ao gestor do sistema as ações que deverá tomar. Esta metodologia será capaz de avaliar o contexto e sugerir, mediante o tipo de evento e o estado atual do sistema, as ações adequadas. Ao serem detetadas anomalias ainda não identificadas e categorizadas, o sistema de apoio à decisão será capaz de adicionar as mesmas e identificar as regras de atuação adequadas, através de mecanismos de aprendizagem automática com as decisões tomadas pelo gestor do sistema. Por fim a componente de deteção, previsão e prevenção será integrada no sistema SASSI e testada de modo a verificar que o seu comportamento segue as funcionalidades previstas.

Bibliografia

Agarwal, Ramesh e Mahesh V Joshi (2001). «PNrule: A new framework for learning classifier models in data mining (a case-study in network intrusion detection)». Em: *Proceedings of the 2001 SIAM International Conference on Data Mining*. SIAM, pp. 1–17.

Allen, Julia, Alan Christie, William Fithen, John McHugh, Jed Pickel e Ed Stoner (2000). «State of the Practice of Intrusion Detection Technologies». Em: January, pp. 1–242.

Alonso-Betanzos, A., N. Sánchez-Maroño, F. M. Carballal-Fortes, J. Suárez-Romero e B. Pérez-Sánchez (2007). «Classification of computer intrusions using functional networks. A comparative study». Em: January, pp. 579–584. url: https://www.researchgate.net/profile/Amparo%7B%5C_%7DAlonso-Betanzos/publication/221165974%7B%5C_%7DClassification%7B%5C_%7Dof%7B%5C_%7Dcomputer%7B%5C_%7Dintrusions%7B%5C_%7Dusing%7B%5C_%7Dfunctional%7B%5C_%7Dnetworks%7B%5C_%7DA%7B%5C_%7Dcomparative%7B%5C_%7Dstudy/links/0deec525c1b74e677e000000.pdf.

Arora, Sandeep Kumar e ayushrees@gmail com Ayushree (2016). «Detection and performance analysis of wormhole attack in MANET using DELPHI technique». Em: *International Journal of Security and its Applications* 10.10, pp. 321–330. issn: 17389976. doi: 10.14257/ijsia.2016.10.10.28.

B, Zhang (2005). «Internet intrusion detection by autoassociative neural network». Em:

Babin, Barry J, William R Darden e Mitch Griffin (1994). «Work and/or fun: measuring hedonic and utilitarian shopping value». Em: *Journal of consumer research* 20.4, pp. 644–656.

Bayesianos, Classificadores (2011). «Classificadores Bayesianos». Em: pp. 1–7. url: http://www.inf.ufrgs.br/%7B~%7Dalvares/CMP259DCBD/classificadores-bayseanos.pdf.

BBC News (2016). *Cyber attacks briefly knock out top sites*. url: http://www.bbc.co.uk/news/technology-37728015.

Benyamin, Dan (2012). *A Gentle Introduction to Random Forests, Ensembles, and Performance Metrics in a Commercial System*. url: http://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics.

Bishop, Christopher M (1995). «Neural Networks for Pattern Recognition». Em: url: http://cs.du.edu/%7B~%7Dmitchell/mario%7B%5C_%7Dbooks/Neural%7B%5C_%7DNetworks%7B%5C_%7Dfor%7B%5C_%7DPattern%7B%5C_%7DRecognition%7B%5C_%7D-%7B%5C_%7DChristopher%7B%5C_%7DBishop.pdf.

Breiman, Leo (2001). «Random Forests». Em: pp. 1–33. url: http://www.math.univ-toulouse.fr/%7B~%7Dagarivie/Telecom/apprentissage/articles/randomforest2001.pdf.

Breiman, Leo, J.H Friedman, R.A Olshen e C.J Stone (1984). *Classification and Regression Trees*.

Brownlee, Jason (2014). *An Introduction to Feature Selection*.

Cale, Emma (2012). *advantages-technology-nowadays-45910 @ smallbusiness.chron.com*. url: http://smallbusiness.chron.com/advantages-technology-nowadays-45910.html.

Cannady, James (1998). «Artificial neural networks for misuse detection». Em: *National information systems security conference*, pp. 368–381.

Chandola, Varun, Arindam Banerjee e Vipin Kumar (2007). «Anomaly Detection: A Survey». Em: url: https://pdfs.semanticscholar.org/7b5a/c1fb5627addf92ad5804a6569a6cfa9385ac.pdf.

Cortes, Corinna e Vladimir Vapnik (1995). «Support-Vector Networks». Em: *Machine Learning* 20.3, pp. 273–297. issn: 15730565. doi: 10.1023/A:1022627411411. arXiv: arXiv:1011.1669v3.

Cournapeau, David (2007). *RBF SVM parameters*. url: http://scikit-learn.org/stable/auto%7B%5C_%7Dexamples/svm/plot%7B%5C_%7Drbf%7B%5C_%7Dparameters.html.

Delft, Technische Universiteit e Rector Magnificus (2001). «One-class classification». Tese de doutoramento. url: http://homepage.tudelft.nl/n9d04/thesis.pdf.

Deng, Li e Dong Yu (2013). «Deep Learning». Em: pp. 3–4. url: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/DeepLearning-NowPublishing-Vol7-SIG-039.pdf.

DN (2016). *Enorme base de dados dos Papéis do Panamá publicada online*. url: http://www.dn.pt/mundo/interior/base-de-dados-gigante-dos-papeis-do-panama-publicada-online-5165749.html.

DuPaul, Neil (2016). *spoofing-attack @ www.veracode.com*. url: https://www.veracode.com/security/spoofing-attack.

Ferreira, G. A. Carrijo, R. De Oliveira e N. V S Araujo (2011). «Intrusion detection system with wavelet and neural artifical network approach for networks computers». Em: *IEEE Latin America Transactions* 9.5, pp. 832–837. issn: 15480992. doi: 10.1109/TLA.2011.6030997.

Francos, Diego Fernández (2017). «Computational learning algorithms for large-scale datasets». Tese de doutoramento.

Fugate, Mike e James R Gattiker (2003). «Computer intrusion detection with classification and anomaly detection, using SVMs». Em: *International Journal of Pattern Recognition and Artificial Intelligence* 17.03, pp. 441–458.

Gama, João, André Ponce de Leon Carvalho, Katti Faceli, Ana Carolina Lorena e Márcia Oliveira (2015). *Extração de Conhecimento de dados*.

García-Teodoro, P, J Díaz-Verdejo, G Maciá-Ferná Ndez e E Vá Zquez (2009). «Anomaly-based network intrusion detection: Techniques, systems and challenges». Em: *Computers & Security* 28.February, pp. 18–28. issn: 01674048. doi: 10.1016/j.cose.2008.08.003.

Glander, Shirin (2017). *Update to autoencoders and anomaly detection with machine learning in fraud analytics*. url: https://shiring.github.io/machine%7B%5C_%7Dlearning/2017/05/02/fraud%7B%5C_%7D2.

Greenwald, Ted (2012). *Business Model Canvas: A Simple Tool For Designing Innovative Business Models*. url: http://www.forbes.com/sites/tedgreenwald/2012/01/31/business-model-canvas-a-simple-tool-for-designing-innovative-business-models/%7B%5C#%7D1f90fa5d77c8.

Guyon, Isabelle, Masoud Nikravesh, Steve Gunn e Lotfi A.Zadeh (2006). *Feature Extraction. Foundations and Applications*.

Hajek, Alan (2003). «What Conditional Probability Could Not Be». Em: *Synthese* 137.3, pp. 273–323. issn: 00397857. doi: 10.1023/B:SYNT.0000004904.91112.16.

Han, Jiawei, Jian Pei e Micheline Kamber (2011). *Data mining: concepts and techniques*. Elsevier.

Haykin, S. (1999). *Neural networks: A comprehensive foundation (2nd ed.)*

Hinton, Geoffrey E (1989). «Connectionist Learning Procedures». Em: 40.1989, pp. 185–234. url: http://www.cs.toronto.edu/%7E~%7Dfritz/absps/clp.pdf.

Holbrook, Morris B (1999). *Consumer value: a framework for analysis and research*. Psychology Press.

Hssina, Badr, Abdelkarim Merbouha, Hanane Ezzikouri e Mohammed Erritali (2014). «A comparative study of decision tree ID3 and C4.5». Em: *International Journal of Advanced Computer Science and Applications* 2, pp. 13–19. issn: 2158107X. doi: 10.14569/SpecialIssue.2014.040203.

Huang, J.-J. e C.-Y. Chen (2014). «Integration of rough sets and support vector machines for network intrusion detection». Em: *Journal of Industrial and Production Engineering* 31.7, pp. 425–432. issn: 21681015. doi: 10.1080/21681015.2014.975163. url: http://www.scopus.com/inward/record.url?eid=2-s2.0-84919682525%7B%5C&%7DpartnerID=40%7B%5C&%7Dmd5=0513beaf69009c34b1bc04487b00c918.

ICA (2017). «Background to "Assessing Russian Activities and Intentions in Recent US Elections ": The Analytic Process and Cyber Incident Attribution». Em: January.

Jain, Anil e Douglas Zongker (1997). «Feature selection: evaluation, application, and small sample performance». Em:

Jain, Yogendra Kumar e Santosh Kumar Bhandare (2011). «Min Max Normalization Based Data Perturbation Method for Privacy Protection». Em: Viii, pp. 45–50. url: https://pdfs.semanticscholar.org/855a/ec7e4697dabc2f8e7c77e307256d651886ce.pdf.

Jain's, Raj (2011). *Cyber Warfare: The Newest Battlefield*. url: http://www.cs.wustl.edu/%7E~%7Djain/cse571-11/ftp/cyberwar/.

Janssens, Davy, Tom Brijs, Koen Vanhoof e Geert Wets (2006). «Evaluating the performance of Cost-based Discretization versus Entropy- and Error-based Discretization». Em:

Japkowicz, B Y Nathalie (1999). «CONCEPT-LEARNING IN THE ABSENCE OF COUNTER-EXAMPLES : AN AUTOASSOCIATION-BASED APPROACH TO CLASSIFICATION». Em:

Jawandhiya, Pradip M, Mangesh M Ghonge, M S Ali e Prof J S Deshpande (2010). «A Survey of Mobile Ad Hoc Network Attacks». Em: 2.9, pp. 4063–4071.

Jinwon, An e Cho Sungzoon (2015). «Variational Autoencoder based Anomaly Detection using Reconstruction Probability». Em: *Technical Report*, pp. 1–18. url: http://dm.snu.ac.kr/static/docs/TR/SNUDM-TR-2015-03.pdf.

Kaushik, Saurav (2016). *Introduction to Feature Selection methods with an example*.

Kdnuggets. *support-vector-machines-simple-explanation @ www.kdnuggets.com*. url: http://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html.

Koen, Peter, Greg Ajamian, Robert Burkart, Allen Clamen, Jeffrey Davidson, Robb D'Amore, Claudia Elkins, Kathy Herald, Michael Incorvia, Albert Johnson et al. (2001). «Providing clarity and a common language to the "fuzzy front end"». Em: *Research-Technology Management* 44.2, pp. 46–55.

Kraljevic, Tom (2017). «Package 'h2o'». Em: url: https://cran.r-project.org/web/packages/h2o/h2o.pdf.

Lecher, Colin (2017). *US releases declassified report on Russian hacking*. url: http://www.theverge.com/2017/1/6/14191002/us-releases-declassified-report-on-russian-hacking.

Li, Yang e Li Guo (2007). «An active learning based TCM-KNN algorithm for supervised network intrusion detection». Em: 26, pp. 459–467. doi: 10.1016/j.cose.2007.10.002. url: http://ai2-s2-pdfs.s3.amazonaws.com/b19b/37a291bffdb9038f06e2bfadf26209c53280.pdf.

Liu, Huan, Farhad Hussain, Chew L I M Tan e Manoranjan Dash (2002). «Discretization : An Enabling Technique». Em: pp. 393–423. url: https://pdfs.semanticscholar.org/2d18/73800b294a104a836168ac5bba11edeadc7f.pdf.

Mathwick, Charla, Naresh K Malhotra e Edward Rigdon (2002). «The effect of dynamic retail experiences on experiential perceptions of value: an Internet and catalog comparison». Em: *Journal of retailing* 78.1, pp. 51–60.

Matos, Manuel António (1995). «Manual Operacional para a Regressão Linear FEUP 1995». Em: url: http://ecocaruaru1.dominiotemporario.com/doc/VALIDACAO.pdf.

Mazhelis, Oleksiy e One-class Classifiers A Review (2016). «One-Class Classifiers : A Review and Analysis of Suitability in the Context of Mobile-Masquerader Detection». Em: url: http://arima.episciences.org/1877/pdf.

Mitchell, Tom M. (Tom Michael), Karl Hansson, Siril Yella, Mark Dougherty, Hasan Fleyeh, D T Pham, A A Afify, Thorsten Wuest, Daniel Weimer, Christopher Irgens e Klaus-Dieter Thoben (2016). «Machine learning in manufacturing: advantages, challenges, and applications». Em: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 4.1, pp. 1–13. issn: 0954-4054. doi: 10.5923/j.ajis.20160601.01. url: http://www.tandfonline.com/doi/full/10.1080/21693277.2016.1192517%7B%5C%%7D5Cnhttp://pib.sagepub.com/lookup/doi/10.1243/095440505X32274%7B%5C%%7D5Cnhttp://sdj.sagepub.com/lookup/10.1243/095440505X32274.

Mizik, Natalie e Robert Jacobson (2003). «Trading off between value creation and value appropriation: The financial implications of shifts in strategic emphasis». Em: *Journal of marketing* 67.1, pp. 63–76.

Mukkamala, Srinivas, Andrew H. Sung e Ajith Abraham (2005). «Intrusion detection using an ensemble of intelligent paradigms». Em: *Journal of Network and Computer Applications* 28.2, pp. 167–182. issn: 10848045. doi: 10.1016/j.jnca.2004.01.003.

Mukkamala, Srinivas, Andrew H Sung e Ajith Abraham (2005). «Intrusion detection using an ensemble of intelligent paradigms». Em: *Journal of network and computer applications* 28.2, pp. 167–182.

Munson, Lee (2012). «DOS VS DDOS – WHAT IS THE DIFFERENCE?» Em: url: http://www.security-faqs.com/dos-vs-ddos-what-is-the-difference.html.

Natesan, P e P Balasubramanie (2012). «Multi Stage Filter Using Enhanced Adaboost for Network Intrusion Detection». Em: 4.3, pp. 121–135. url: http://airccse.org/journal/nsa/0512nsa08.pdf.

Nicola, Susana, Eduarda Pinto Ferreira e João José Pinto Ferreira (2014). «A quantitative model for decomposing & assessing the value for the customer». Em: *Journal of Innovation Management* 2.1, pp. 104–138.

Osuna, Edgar e John Platt (1998). «Support vector machines». Em: url: http://www.cs.cmu.edu/afs/cs.cmu.edu/usr/guestrin/www/Class/10701/readings/hearst98.pdf.

OWASP (2009). *Phishing*. url: https://www.owasp.org/index.php/Phishing.

– (2012). *Repudiation Attack*. url: https://www.google.pt/url?sa=t%7B%5C&%7Drct=j%7B%5C&%7Dq=%7B%5C&%7Desrc=s%7B%5C&%7Dsource=web%7B%5C&%7Dcd=5%7B%5C&%7Dcad=rja%7B%5C&%7Duact=8%7B%5C&%7Dved=0ahUKEwjmormZ5PLNAhUBshQKHSoEBOkQFggzMAQ%7B%5C&%7Durl=http%7B%5C%%7D3A%7B%5C%%7D2F%7B%5C%%7D2Frevistas.iel.unicamp.br%7B%5C%%7D2Findex.php%7B%5C%%7D2Ftla%7B%5C%%7D2Farticle%7B%5C%%7D2Fdownload%7B%5C%%7D2F2034%7B%5C%%7D2F1589%7B%5C&%7Dusg=AFQjCNG7rPAjJOHUIj5dT6T3iZwWcUGETw.

Pawar, Mohan V. e J. Anuradha (2015). «Network Security and Types of Attacks in Network». Em: *Procedia Computer Science* 48.2015, pp. 503–506. issn: 18770509. doi: 10.1016/j.procs.2015.04.126. url: http://linkinghub.elsevier.com/retrieve/pii/S1877050915006353.

Perlin, Tiago, Raul Ceretta Nunes e Alice De Jesus Kozakevicius (2011). «Detecção de Anomalias em Redes de Computadores através de Transformadas Wavelet». Em: 3.1, pp. 2–15. issn: 2176-6649. doi: 10.5335/rbca.2011.002. url: http://www.upf.br/seer/index.php/rbca/article/view/1313/1067.

Quinlan, J. Ross (1993). «C4.5: Programs for Machine Learning». Em:

Rai, Ajay Prakash, Vineet Srivastava e Rinkoo Bhatia (2012). «Wormhole Attack Detection in Mobile Ad Hoc Networks». Em: 2.2, pp. 174–179.

Rijsbergen, C.J. Van (1979). «Information Retrieval». Em:

Rocha, Miguel e Pedro G.Ferreira (2017). *Análise e exploração de dados com R*.

Rouse, Margaret (2016). *email-spoofing @ searchsecurity.techtarget.com*. url: http://searchsecurity.techtarget.com/definition/email-spoofing.

Sabhnani, Maheshkumar, Gursel Serpen e Komal K More (2003). «Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context». Em: *Proceedings of International Conference on Machine Learning: Models, Technologies, and Applications (MLMTA)*, pp. 209–215. doi: citeulike-article-id:9827151. url: http://neuro.bstu.by/ai/To-dom/My%7B%5C_%7Dresearch/Papers-0/For-research/D-mining/Anomaly-D/KDD-cup-99/mlmta03.pdf.

Salem Khalifa, Azaddin (2004). «Customer value: a review of recent literature and an integrative configuration». Em: *Management decision* 42.5, pp. 645–666.

Sánchez-Fernández, Raquel e M Ángeles Iniesta-Bonillo (2007). «The concept of perceived value: a systematic review of the research». Em: *Marketing theory* 7.4, pp. 427–451.

SAS (2016). *machine-learning What is it and Why it matters*. url: http://www.sas.com/it%7B%5C_%7Dit/insights/analytics/machine-learning.html.

SecuritySupervisor (2014). *What is Url Spoofing*. url: http://www.securitysupervisor.com/security-q-a/network-security/262-what-is-url-spoofing.

Shiravi, Ali, Hadi Shiravi, Mahbod Tavallaee e Ali A. Ghorbani (2012). «Toward developing a systematic approach to generate benchmark datasets for intrusion detection». Em: *Computers and Security* 31.3, pp. 357–374. issn: 01674048. doi: 10.1016/j.cose.2011.12.012. arXiv: arXiv:1011.1669v3. url: http://dx.doi.org/10.1016/j.cose.2011.12.012.

Simsolo, Yaniv (1998). «OWASP Top Ten Backdoors». Em: pp. 1–33. url: https://www.owasp.org/images/a/ae/OWASP%7B%5C_%7D10%7B%5C_%7DMost%7B%5C_%7DCommon%7B%5C_%7DBackdoors.pdf.

Sinha, Indrajit Jay e Wayne S DeSarbo (1998). «An integrated approach toward the spatial modeling of perceived customer value». Em:

Slater, Stanley F (1997). «Developing a customer value-based theory of the firm». Em: *Journal of the Academy of marketing Science* 25.2, p. 162.

Song, Lydia, Lauren Steimle, Xiaoxiao Xu e Arye Nehorai (2014). *Email Spam Detection Using Machine Learning*. url: https://ese.wustl.edu/ContentFiles/Research/UndergraduateResearch/CompletedProjects/WebPages/sp14/SongSteimle/WebPage/classifiers.html.

Souza, César (2009). *Análise de Poder Discriminativo Através de Curvas ROC*. url: http://crsouza.com/2009/07/13/analise-de-poder-discriminativo-atraves-de-curvas-roc/.

Srivastava, Tavish (2016). *Important Model Evaluation Error Metrics Everyone should know*. url: https://www.analyticsvidhya.com/blog/2016/02/7-important-model-evaluation-error-metrics/.

Stolfo, Salvatore J, Wei Fan, West Street, New York, Wenke Lee, Andreas Prodromidis e Philip K Chan. «Cost-based Modeling for Fraud and Intrusion Detection : Results from the JAM Project». Em:

Sultani, Zainab (2015). «Learning Vector Quantization (LVQ) and k- Nearest Neighbor for Intrusion Classification Learning Vector Quantization (LVQ) and k-Nearest Neighbor for Intrusion Classification». Em: January 2012. url: https://www.researchgate.net/profile/Zainab%7B%5C_%7DSultani/publication/271193607%7B%5C_%7DLearning%7B%5C_%7DVector%7B%5C_%7DQuantization%7B%5C_%7DLVQ%7B%5C_%7Dand%7B%5C_%7Dk-Nearest%7B%5C_%7DNeighbor%7B%5C_%7Dfor%7B%5C_%7DIntrusion%7B%5C_%7DClassification/links/54c0e5170cf21674cea0765d.pdf.

Sutton, Chris (2008). *Internet Began 35 Years Ago at UCLA with First Message Ever Sent Between Two Computers*. url: https://web.archive.org/web/20080308120314/http://www.engineer.ucla.edu/stories/2004/Internet35.htm.

Sweeney, Jillian, Geoffrey Soutar, Alma Whiteley e Lester Johnson (1996). «Generating consumption value items: a parallel interviewing process approach». Em: *AP-Asia Pacific Advances in Consumer Research Volume 2*.

Tavallaee, Mahbod, Ebrahim Bagheri, Wei Lu e Ali A. Ghorbani (2009). «A detailed analysis of the KDD CUP 99 data set». Em: *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009* Cisda, pp. 1–6. issn: 2329-6267. doi: 10.1109/CISDA.2009.5356528. arXiv: arXiv:1011.1669v3.

The Institute of Value Management (2001). *What-Is-Value-Management @ Ivm.Org.Uk*. url: http://ivm.org.uk/what-is-value-management.

The Marketing Science Institute (2006). *Marketing Science Institute @ www.msi.org*. url: http://www.msi.org/research/.

Turi Platform (2016). *Classification Metrics*. url: https://turi.com/learn/userguide/evaluation/classification.html.

UCI Machine Learning Repository (2015). *KDD Cup 1999 Data*. url: https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data.

VisionSpace (2015). «Parte B (Anexo Técnico) Sistema De Incentivos À Investigação E Desenvolvimento Tecnológico (Si I & Dt)». Em: pp. 1–74.

VisionSpace Technologies (2016). «Manual da base de dados do sensor de sistema». Em:

Wang, Yonggui, Hing Po Lo, Renyong Chi e Yongheng Yang (2004). «An integrated framework for customer value and customer-relationship-management performance: a customer-based perspective from China». Em: *Managing Service Quality: An International Journal* 14.2/3, pp. 169–182.

Woodall, Tony (2003). «Conceptualising'value for the customer': An attributional, structural and dispositional analysis». Em: *Academy of marketing science review* 2003, p. 1.

Xiang, Cheng, Liu Bing-Xiang, Li Ke e Yan Jun (2009). *Intrusion Detection System Based on KNN-MARS*. url: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5319135.

Yang, Yiming e Xin Liu (1999). «A re-examination of text categorization methods». Em: url: http://people.csail.mit.edu/jim/temp/yang.pdf.

Yassin, Warusia, Nur Izura Udzir e Zaiton Muda (2013). «Anomaly-Based Intrusion Detection Through K- Means Clustering and Naives Bayes Classification». Em: *Proceedings of the 4th International Conference on Computing and Informatics, ICOCI 2013* 049, pp. 298–303.

Yedidia, Adam (2016). «Against the F-score». Em: pp. 1–14. url: `https://adamyedidia.files.wordpress.com/2014/11/f%7B%5C_%7Dscore.pdf`.

Yong Joseph Bakos (2010). *Artificial Neural Network (ANN)*. url: `http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio%7B%5C_%7Dexports/lguo/ann.html`.

Zargar, Gholam Reza, Tania Baghaie et al. (2012). «Category-Based Intrusion Detection Using PCA». Em: *Journal of Information Security* 3.04, p. 259.

Zhu, Qifeng, Andreas Stolcke, Barry Y Chen e Nelson Morgan (2005). «Using MLP features in SRI's conversational speech recognition system.» Em: *Interspeech*. Vol. 2005, pp. 2141–2144.