ESCOLA NAVAL

talant de bien faire

Tiago Miguel Fonseca Paiva de Sousa Teles

Cibersegurança

Deteção de outliers

Dissertação para obtenção do grau de Mestre em Ciências Militares Navais, na especialidade de Fuzileiros

Alfeite
2015

ESCOLA NAVAL

talant de bien faire

Tiago Miguel Fonseca Paiva de Sousa Teles

Cibersegurança

Deteção de outliers

Dissertação para obtenção do grau de Mestre em Ciências Militares Navais, na especialidade de Fuzileiros

Orientação de:

Coorientação de:

O Aluno Mestrando	O Coorientador
Tiago Sousa Teles	CFR M Pereira Simões
O Coorientador	O Orientador
CFR EN-AEL Ribeiro Correia	Doutor Victor Lobo

Alfeite

2015

Epígrafe

"To every action there is always opposed an equal reaction"

Sir Isaac Newton, 1643-1727

Dedicatória

À minha família, pela colaboração, apoio e por todo o esforço despendido na minha formação…

Agradecimentos

Antes de iniciar esta dissertação gostaria de agradecer a todas as pessoas que a tornaram possível.

- Ao meu orientador Professor Doutor Vítor José de Almeida e Sousa Lobo, pelos ensinamentos, dedicação e apoio na elaboração desta investigação.
- Ao meu coorientador CFR EN_AEL Ribeiro Correia, pela disponibilidade, dedicação e cedência de conhecimentos neste trabalho.
- Ao Instituto Superior Técnico, nomeadamente ao Doutor André Souto, pela colaboração, disponibilidade e entrega nesta investigação, particularmente no âmbito do *data mining* não supervisionado.
- À Direção de Tecnologias de Informação e Comunicações, nomeadamente ao 1TEN STP Baptista das Neves e ao TEC. INF. Rodrigues Gomes pelas opiniões e recomendações técnicas no âmbito da cibersegurança, que foram efetuando ao longo da investigação e obviamente pela captura e cedência dos dados, fulcrais a esta investigação.
- Ao Centro de Comunicações da Marinha, nomeadamente, ao meu coorientador, CFR Pereira Simões pela prestabilidade e informação que me disponibilizou.
- Ao Serviço de Informática da Escola Naval, nomeadamente ao 2TEN STP Ramos Silveiro, e ao ESP. INF. Branco Carinhas, pelo apoio prestado na utilização do "ciberlab" da Escola Naval.
- À Catarina Santos pelo apoio, colaboração e recomendações que foi efetuando ao longo da dissertação.
- À Escola Naval e à Marinha Portuguesa, pelas condições e financiamento fornecidos a esta investigação.
- À minha família e amigos pela compreensão, apoio e partilha de conhecimentos.
- Ao departamento de fuzileiros, oficiais e camaradas fuzileiros, pela preocupação, disponibilidade e ajuda prestada.
- A todos os meus restantes camaradas, professores, Colégio Militar, ESF, ISCTE-IUL e restantes instituições que ao longo dos anos contribuíram para a minha formação e me auxiliaram sempre que necessário.

Resumo

Com o evoluir da tecnologia, o Homem encontra-se cada vez mais dependente das tecnologias de informação e comunicação, sendo o espaço criado por estas tecnologias conhecido por "ciberespaço".

O ciberespaço está sujeito a ataques disruptivos, com consequências dramáticas para o bom funcionamento da sociedade, pelo que é imprescindível protegê-lo e mantê-lo seguro e funcional.

Este trabalho pretende estudar e implementar diversas técnicas de prospeção de dados para proteção contra ciberameças, com o objetivo de validar e melhorar as barreiras atualmente existentes.

Para tal é dada uma introdução às redes de computadores, nomeadamente ao protocolo TCP, para que quem não tenha conhecimentos em telecomunicações consiga facilmente compreender os protocolos utilizados na rede e a forma como estes podem ser atacados.

É feito um resumo sobre o estado contemporâneo da cibersegurança, onde são explicadas quais as tecnologias de defesa e ataque atualmente existentes.

Elabora-se um estudo teórico sobre quais os métodos baseados em reconhecimento de padrões existentes, como funcionam e de que forma podem ser utilizados para detetar ciberataques.

Por fim, utilizam-se dados obtidos através da *firewall* de perímetro da Marinha Portuguesa para testar os referidos métodos, recorrendo-se tanto a aprendizagem supervisionada como não supervisionada.

Palavras-chave:

Cibersegurança, redes de computadores, prospeção de dados, aprendizagem não supervisionada, aprendizagem supervisionada.

Abstract

As technology evolves, Man becomes increasingly more dependent on information and communication technologies, and therefore on cyberspace, the space created by these technologies.

Cyberspace is subject to disruptive attacks, with dramatic consequences for society's proper functioning. As so, it is essential to protect the cyberspace and keep it safe and functional.

This paper aims to study and implement several data mining techniques for protection against cyber threats, in order to validate and improve the currently existing barriers.

As an introduction to computer networks, it is presented the TCP protocol, so that those without knowledge on telecommunications can easily understand the protocols used in the net, as well as how these can be hacked.

A brief summary is conducted on the contemporary state of cybersecurity, where the currently existent attack and defense technologies are tackled.

More on, the methods based on recognition of existing patterns are addressed, setting about their functioning and how they can be used to detect cyberattacks.

Lastly, data acquired through Marinha Portuguesa's perimeter firewall is used to test the referred methods, resorting to both machine learning and clustering.

Keywords:

Cibersecurity, computer networks, data mining, machine learning, clustering.

Índice

Índice de figuras

Índice de tabelas

Índice de algoritmos

Abreviaturas, siglas e acrónimos

ACK	*Acknowledge packet*
ADAM	*Audit Data Analysis and Mining*
BD	Base de Dados
CRISP-DM	*Cross-Industry Standard Process for Data Mining*
CWR	*Congestion Window Reduced*
DARPA	*Defense Advanced Research Projects Agency*
DDoS	*Distributed Denial of Service*
DITIC	Direção de Tecnologias de Informação e Comunicações
DoS	*Denial-of-Service*
DPI	*Deep Packet Inspection*
ECE	*Explicit Congestion Notification - Echo*
ECN	*Explicit Congestion Notification*
FDDI	*Fiber Distributed Data Interface*
FTP	*File Transfer Protocol*
HTTP	*Hyper Text Transfer Protocol*
IDS	*Intrusion Detection Systems*
IEEE	*Institute of Electrical and Electronic Engineers*
IP	*Internet Protocol*
LAN	*Local Area Network*
MAC	*Media Access Control* *Message Authentication Code*
MAN	*Metropolitan Area Network*
MPS	*Mail Protection System*
NCD	*Normalized Compression Distance*
NID	*Normalized Information Distance*
OS	*Operating System*
OSI	*Open Systems Interconnection Seven-Layer Reference Model*

PCI	*Protocol Control Information*
PCT	*Private Communications Transport*
PDU	*Protocol Data Unit*
SDU	*Service Data Unit*
SIEM	*Security Information and Events Management*
SMTP	*Simple Mail Transfer Protocol*
SNMP	*Simple Network Management Protocol*
SSL	*Secure Socket Layer*
SYN	*Synchronise packet*
TCP	*Transmission Control Protocol*
TCP/IP	*Transmission Control Protocol/Interface Program*
TFTP	*Trivial File Transfer Protocol*
TLS	*Transport Layer Security*
UDP	*User Datagram Protocol*
VPN	*Virtual Private Network*
WAN	*Wide Area Network*
WEB/WWW	*World Wide Web*

Capítulo 1 - Introdução

Ao longo da história da humanidade o homem tem adaptado continuamente o seu modo de vida em função da tecnologia de que dispõe, como definiu Ortega Y Gasset: "O homem é o homem e a sua circunstância" (ORTEGA Y GASSET, 2010, pp. 111-119). O seguimento desta maré arrastou também a sua forma de lutar, levando a que os primeiros instrumentos cortantes depressa se transformassem em legiões e exércitos, as jangadas em fragatas e submarinos e a passarola[1] em caças e bombardeiros. Seguindo a mesma linha de raciocínio, o surgimento do conceito de *ciberespaço* também tem cada vez mais um papel determinante na forma como lidamos com os conflitos internacionais, tendo já dado origem ao surgimento do termo *ciberguerra* (Arquilla & Ronfeldt, 1993; Militão, 2014, p. iv). Nesta linha de raciocínio os ataques à Geórgia, à Estónia e mais recentemente à Ucrânia, são exemplos reais do que pode acontecer neste novo espaço de interesse. Para lidar com este novo tipo de conflitos, em outubro de 2010, o Departamento de Defesa dos Estados Unidos da América, adicionou o ciberespaço aos quatro domínios de defesa até então existentes: mar, terra, ar e espaço (Pellerin, 2010), criando assim o USCYBERCOM, composto pelos cibercomandos da Armada, Fuzileiros, Exército e Força Aérea (IDN12, 2013, p. 37). Esta preocupação não se limitou apenas às fronteiras americanas, tendo no mesmo ano a União Europeia realizado o primeiro exercício de simulação de ciberataques e a NATO já criado a Divisão de Desafios de Segurança Emergentes (Lorena, 2010). Paralelamente a estes casos, temos ainda muitos outros tipos de ameaças aos quais temos necessidade de dar resposta, tais como o *cibercrime*, *ciberespionagem*, *ciberterrorismo* e *ciberativismo* (IDN12, 2013, pp. 22-25). A juntar a estas ameaças, todos os sistemas de defesa ou sistemas críticos para o bom funcionamento da sociedade têm uma grande dependência da tecnologia o que os torna vulneráveis a ataques cibernéticos, podendo constituir ameaças físicas para a sociedade.

Por forma a evitar que se seja atacado deve-se inicialmente compreender o inimigo, identificá-lo e analisar-se a forma como ataca. De seguida diminuir as

[1] A primeira aeronave conhecida no mundo a efetuar um voo foi batizada de Passarola, e antecede 74 anos o famoso balão dos Montgolfier. A Passarola era um aeróstato, cujas características técnicas não são atualmente conhecidas na totalidade, inventado por Bartolomeu de Gusmão, padre e cientista português nascido na colónia do Brasil, tendo voado no ano de 1709 (SIQUEIRA, 1990).

vulnerabilidades existentes, melhorando constantemente as barreiras defensivas e por fim estar sempre pronto a tomar a iniciativa em ações contra-ofensivas (Tzu, 2012).

1.1. Contribuições originais

Este projeto tem como objetivo estudar a parte inicial das ações a tomar, procurando-se identificar aquilo que são e aquilo que não são ameaças. Para tal é pretendido utilizar diferentes técnicas de *data mining*[2] por forma a identificar *outliers*[3], que neste contexto são potenciais ameaças cibernéticas.

É pretendido com este trabalho apresentar um método de classificação que, através da introdução de novos dados de teste, seja capaz de distinguir o tráfego que é uma ameaça, daquele que não o é, por forma a saber as ações que se devem tomar.

Com esta investigação pretende-se melhorar o desempenho dos atuais métodos existentes de deteção e análise de tráfego, contribuindo-se assim para que a liberdade de comunicação seja preservada sem que a privacidade e segurança sejam afetadas.

1.2. Metodologia

Este trabalho consiste em proporcionar o conhecimento das potencialidades de técnicas de tratamento estatístico de dados de diferentes naturezas e proveniências, através do Método de Investigação Estatística Multivariada.

Por forma a se atingirem os objetivos estabelecer-se-á uma lógica experimental, baseada na experimentação de métodos de análise de dados, analisar-se-ão os resultados e efetuar-se-ão pesquisas em fontes abertas (maioritariamente na Internet).

1.3. Estrutura do documento

Apesar da maioria deste estudo ser feita no âmbito da prospeção de dados, existem conceitos base que se crê serem necessários para que o leitor consiga compreender a necessidade, relevância e desenrolar desta investigação. Desta forma, o Capítulo 2 começa por enquadrar o leitor na temática das redes de computadores, explicando o seu

[2] Prospeção de dados (também conhecida pelo termo inglês data mining) é o processo de explorar grandes quantidades de dados à procura de padrões consistentes, como regras de associação ou sequências temporais, para detetar relacionamentos sistemáticos entre variáveis, detetando assim novos subconjuntos de dados.

[3] Em estatística, outlier, valor aberrante ou valor atípico, é uma observação que apresenta um grande afastamento das demais da série (que esta "fora" dela), ou que é inconsistente.

funcionamento e explicitando alguns conceitos base. De seguida introduz ainda a problemática desta investigação, explanando o que é a cibersegurança e resumindo a tecnologia de ataque e defesa atualmente existente.

No Capítulo 3, introduz-se o *data mining*, método que se propõe como solução para a problemática referida no capítulo anterior. Este capítulo encontra-se dividido em duas grandes temáticas que acompanharão o resto da investigação: a aprendizagem supervisionada e a aprendizagem não supervisionada.

Entrando no Capítulo 4 é possível entender que dados foram utilizados e de que forma estes foram tratados.

Os resultados obtidos para os diferentes testes que foram realizados são apresentados no Capítulo 5. Este capítulo é dividido em duas grandes temáticas: os testes realizados com aprendizagem supervisionada e os realizados com aprendizagem não supervisionada.

Por fim, no último capítulo tecem-se alguns comentários finais, onde se abordam as contribuições do trabalho e conclusões. São apresentadas algumas propostas de trabalho futuro.

Capítulo 2 - Enquadramento

2.1. Redes de computadores

Este trabalho procura identificar padrões na forma como os computadores comunicam entre si e com os demais servidores. Assim torna-se fundamental entender em que consistem estas redes, quais os protocolos existentes e como é registada toda a informação, para que seja possível compreender como são estabelecidas estas comunicações e por fim processar os dados recolhidos e analisar os resultados finais.

2.1.1. Introdução

Durante as duas primeiras décadas da sua existência os computadores encontravam-se centralizados, normalmente numa grande sala. O desenvolvimento dos computadores e dos sistemas de comunicações teve uma profunda influência na maneira como estes se encontram organizados. Hoje em dia a sua origem encontra-se totalmente obsoleta e este antigo modelo de um só computador a servir todas as necessidades de uma organização foi substituído por um grande número de computadores separados e interconectados. A este tipo de sistemas chamam-se redes de computadores (Tanenbaum, 2011, p. 2). Na sociedade atual, estas redes tornaram-se estradas da informação para a troca de informação e partilha (L. Li, 2014, p. 3).

Para que se possa compreender o funcionamento de uma rede de computadores, para além de ser importante entender em que é que esta consiste, é também importante saber o que a difere de todos os restantes tipos de redes. A grande generalidade das redes, como as destinadas a transportar dados de voz, vídeo e texto encontram-se geralmente especializadas neste tipo de transporte e tipicamente conectam-se a um aparelho geralmente concebido para receber esses dados. Por outro lado, as redes de computadores são desenhadas tendo por base *hardware* programável e não estão destinadas a um único propósito, como realizar chamadas telefónicas ou receber sinal de televisão. Esta é assim, provavelmente, a caraterística mais importante que as distingue (Peterson & Davie, 2007, p. 2).

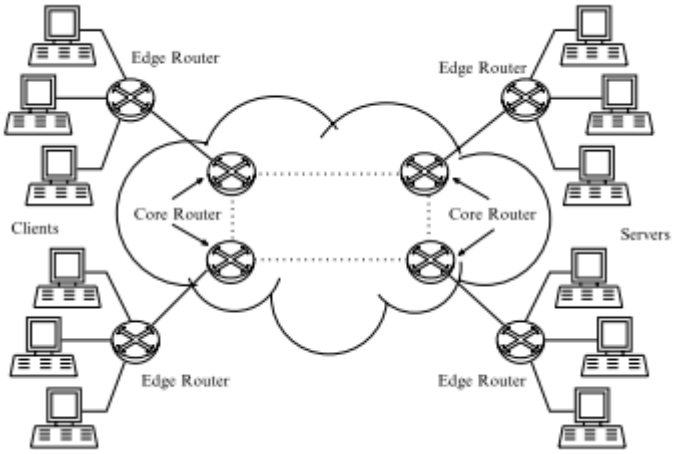

Figura 1 – Rede de computadores (internet) (Pióro & Medhi, 2004, p. 5)

2.1.2. Pilha de protocolos

Para reduzir a complexidade do seu *design*, as redes encontram-se organizadas por camadas, construída cada uma acima da outra. O número, nome, conteúdo e a função de cada camada diferem de rede para rede. O objetivo de cada camada consiste em fornecer serviços à camada acima, sem que esta tenha de receber os detalhes de como os serviços são fornecidos. Assim, quando uma camada *n* numa máquina estabelece uma conversação com a camada *n* na outra máquina, as regras e convenções utilizadas nesta conversação são convencionalmente denominadas por protocolo da camada *n* (Tanenbaum, 2011, p. 29).

Na Figura 2 pode-se observar uma rede composta por cinco camadas.

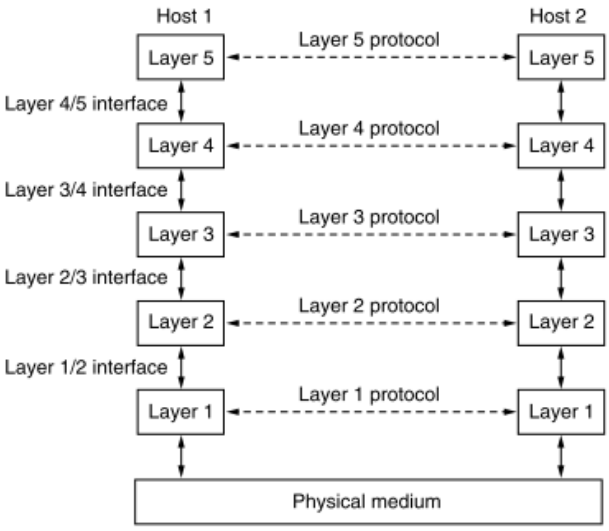

Figura 2 - Camadas e protocolos (Tanenbaum, 2011, p. 30)

2.1.3. Arquiteturas de rede

Para que uma rede consiga operar, esta tem de ter uma conetividade eficaz, correta e robusta entre um grande número de computadores. Como se isto não fosse já extremamente ousado, estas não podem permanecer fixas num único ponto para um dado tempo, isto é, têm que evoluir para conseguirem acomodar alterações. Por forma a lidar com esta complexidade, os *designers* de rede criaram as arquiteturas de rede (Peterson & Davie, 2007, p. 19). As arquiteturas de rede mais não são do que conjuntos de camadas e protocolos. Observe-se o exemplo do fluxo de informação suportado por uma comunicação virtual de camada 5, representado na Figura 3.

O importante a reter nesta figura é a relação entre protocolos e interfaces. Os pares da camada n pensam que estão a comunicar horizontalmente, e como tal utilizam protocolos de camada n, contudo na realidade, esta mensagem chega até ao outro lado através das camadas inferiores, que não fazem ideia de qual o seu significado. A este tipo de comunicação, chama-se comunicação virtual, pois na realidade para a mensagem do par n' chegar a n'', esta tem de passar por várias camadas, que funcionam como mensageiros, isto é, sempre que uma camada n recebe dados da camada $n+1$, é prestado um serviço à camada n, conhecido por *Service Data Unit* (SDU). Para que o protocolo utilizado fique registado é adicionada informação de controlo, conhecida por *Protocol Control Information* (PCI). Após este registo, obtêm-se então o *Protocol Data Unit* (PDU), que é finalmente enviado para a camada $n-1$. A esta operação de adicionar informação de controlo aos dados recebidos das camadas superiores chama-se encapsulamento (Glover & Grant, 2010, p. 757).

Esta abstração no processo é crucial para todo o *design* de rede, permitindo assim que a árdua tarefa de desenhar uma rede, possa ser repartida em várias camadas (Tanenbaum, 2011, p. 33).

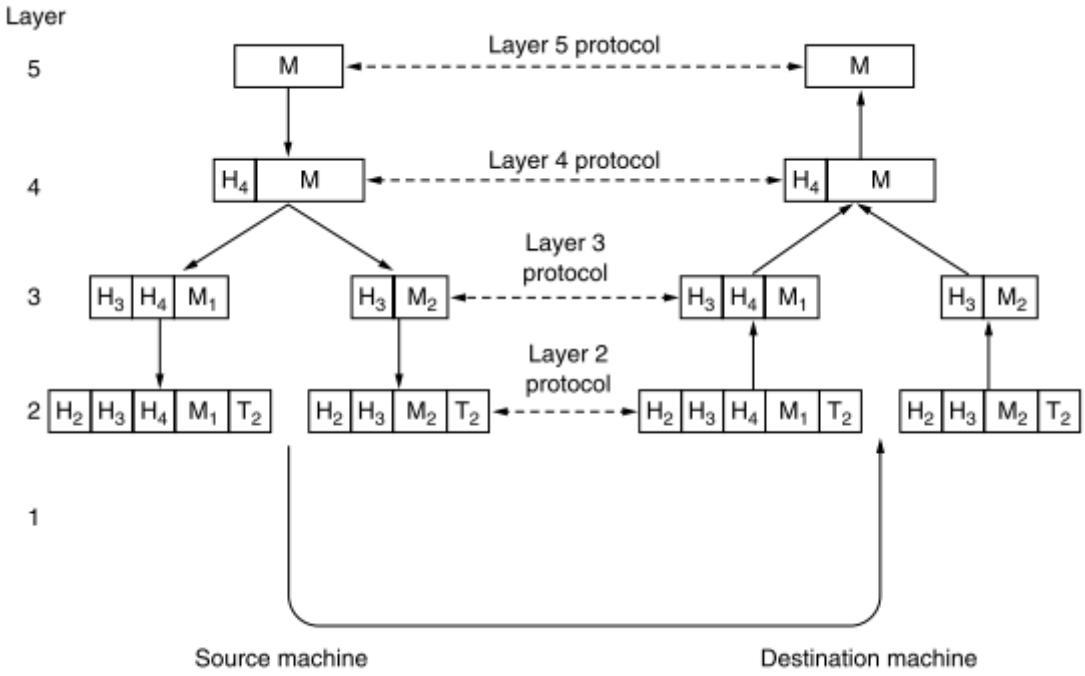

Figura 3 - Exemplo do fluxo de informação suportado por uma comunicação virtual de camada 5 (Tanenbaum, 2011, p. 33)

2.1.4. A arquitetura OSI

Enquanto os protocolos de rede iam sendo desenvolvidos, duas grandes organizações[4] juntaram-se para criar um modelo alternativo de referência. Este modelo de 7 camadas ficou conhecido como "*Open Systems Interconnection Seven-Layer Reference Model (OSI)*" (Comer, 2009, p. 13). A Figura 4 retrata este modelo.

[4] A "*International Organization for Standardization (ISO)*" e a "*International Telecommunications Union, Telecommunication Standardization Sector (ITU-T)*"

Figura 4- Modelo OSI

- **Camada física**

Começando na camada 1, a camada física processa a transmissão dos *bits* em bruto através de um canal de comunicações (*link*) (Peterson & Davie, 2007, p. 26).

- **Camada de ligação de dados**

De seguida a camada de ligação de dados aglomera um *stream*[5] num conjunto maior, chamado *frame*[6].

Os adaptadores de rede, juntamente com os drivers de dispositivo correm no *Operating System* (*OS*), geralmente implementado nesta camada. Desta forma, os *frames* não entregam os bits em bruto diretamente aos *hosts*[7] (Peterson & Davie, 2007, p. 26).

- **Camada de rede**

A camada de rede encarrega-se do encaminhamento entre nós dentro de uma rede de comutação de pacotes. Nesta camada, a unidade de dados trocados entre nós é geralmente denominada por pacote, em vez de *frame*, embora estes sejam fundamentalmente os mesmos. As três camadas inferiores encontram-se implementadas

[5] Fluxo de dados.
[6] Quadro de dados.
[7] Anfitriões/hospedeiros.

em todos os nós da rede, incluindo *switches*[8] dentro da rede e *hosts* ligados ao longo do exterior da rede (Peterson & Davie, 2007, pp. 26-27).

- **Camada de transporte**

De seguida a camada de transporte implementa o que se chama um canal de processo-a-processo. Aqui a unidade de dados trocados é vulgarmente designada por mensagem, em vez de pacote ou *frame*. A camada de transporte e as restantes camadas mais altas normalmente são executadas apenas no final dos *hosts* e não nos *switches* ou *routers* intermediários (Peterson & Davie, 2007, p. 27).

- **Camada de sessão**

Continuando, é esta camada que vai permitir que utilizadores de diferentes máquinas estabeleçam sessões entre si. Estas sessões oferecem vários serviços como: controlo de diálogo[9], gestão de *token*[10] e sincronização[11] (Tanenbaum, 2011, pp. 44-45).

- **Camada de apresentação**

Contrariamente às camadas inferiores, que se preocupam mais com o movimento dos *bits* ao seu redor, a camada de apresentação preocupa-se principalmente com a sintaxe e semântica da informação transmitida. A fim de ser possível que computadores com diferentes representações de dados internos possam comunicar entre si, as estruturas de dados a serem trocados são definidas de uma forma abstrata, juntamente com uma codificação padrão a ser utilizada em comunicações por fio. A camada de apresentação gere essas estruturas de dados abstratas e permite que as estruturas de dados de alto nível sejam definidas e trocadas (Tanenbaum, 2011, p. 45).

- **Camada de aplicação**

Esta camada contém uma grande variedade de protocolos frequentemente necessários aos utilizadores. Um dos mais frequentes é o *Hyper Text Transfer Protocol (HTTP),* que é a base para a *World Wide Web (WEB)*. Quando um browser pretende abrir uma página, este envia o nome da página que pretende para o servidor hospedeiro da

[8] Comutadores.
[9] Mantem o controlo de quem é a vez de transmitir.
[10] Impede que duas partes tentem efetuar simultaneamente um operação critica.
[11] Ponto de verificação para transmissões longas, que permite que se volte ao local onde se perderam e que posteriormente se possam recuperar.

página, usando o protocolo HTTP. De seguida o *host server* envia-lhe a página. São ainda utlizados muitos outros protocolos para transferência de ficheiros, *e-mail* e rede de notícias (Tanenbaum, 2011, p. 45).

2.1.5. Arquitetura da *internet* - Protocolo TCP/IP

A arquitetura da *internet*, vulgarmente conhecida por protocolo TCP/IP encontra-se representada na Figura 5, sendo que à esquerda podemos observar o número, à direita o nome de cada camada e ao centro os respetivos protocolos (Peterson & Davie, 2007, p. 28).

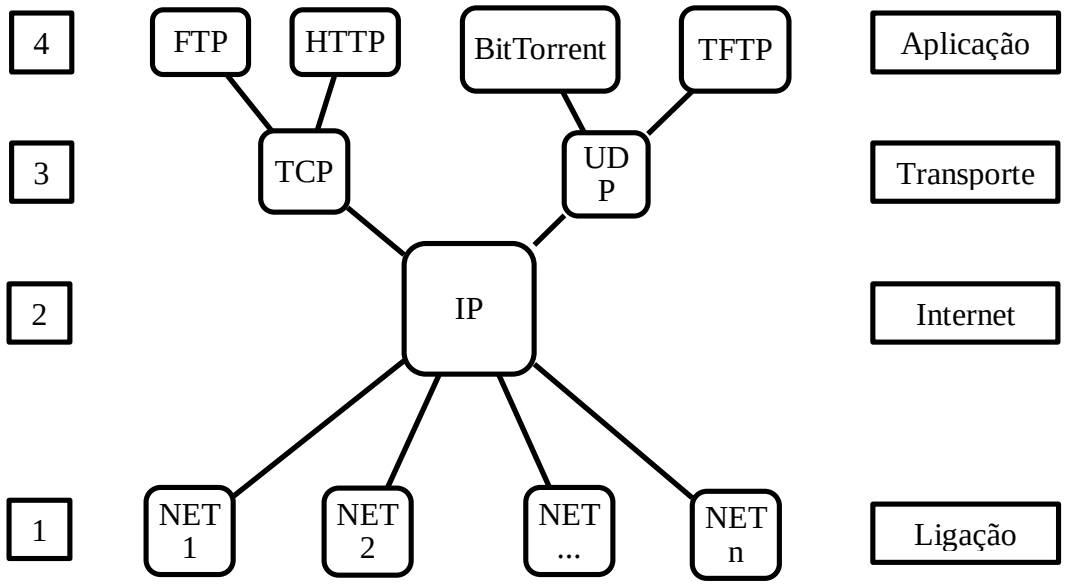

Figura 5- Protocolo TCP/IP

Esta arquitetura foi desenvolvida a partir de experiências com uma rede de troca de pacotes anteriormente utilizada, que se chamava *arpanet* (Peterson & Davie, 2007, p. 28). Esta rede ligava centenas de universidades e instalações governamentais. Contudo com o surgimento das redes satélites e rádio, observou-se que os protocolos até então existentes tinham problemas de interoperabilidade com estas redes, pelo que surgiu a necessidade de se desenvolver um novo protocolo de referência. Dada a preocupação do Departamento da Defesa dos EUA que os seus *hosts*, *routers* e *gateways* pudessem ser danificados a qualquer momento por um ataque da União Soviética, outro grande objetivo era que a rede conseguisse sobreviver a perdas de *hardware* na sub-rede, sem que as conversações existentes fossem destruídas, ou seja, que as conexões se mantivessem

intactas enquanto as máquinas de origem e destino estivessem em funcionamento, mesmo que pelo meio algumas máquinas e linhas de transmissão se perdessem. Para além destes dois fatores, também o surgimento de várias aplicações de transferência de ficheiros e voz em tempo real, com diferentes requisitos, forçava já ao surgimento de uma arquitetura mais flexível. Este fatores conduziram esta arquitetura a uma rede de troca de pacotes, baseada numa camada de conexão que é executada através do cruzamento de redes diferentes. (Tanenbaum, 2011, pp. 45-46). Tanto esta, como a *internet* surgiram antes da arquitetura OSI e a experiência ganha com o seu desenvolvimento foi uma grande mais-valia para a criação do modelo OSI. Apesar do modelo OSI de sete camadas também poder, com alguma imaginação, ser aplicado à internet, geralmente é mais utilizado o modelo de quatro camadas (Peterson & Davie, 2007, p. 28).

- **Camada de ligação**

A camada de ligação descreve o que ligações, como as linhas de série e a clássica *ethernet*, têm de fazer para conhecer as necessidades das *connectionless* da camada *internet*. Não é literalmente uma camada, mas sim uma interface entre os *hosts* e os *links* de transmissão (Tanenbaum, 2011, p. 46). Assim, no nível mais baixo encontra-se uma grande variedade de protocolos[12] de rede, aqui representados por NET 1, NET 2, e por aí adiante. Na prática, estes protocolos são implementados através de uma combinação de *hardware*[13] e *software*[14], que por sua vez ainda podem envolver várias subcamadas, contudo a arquitetura da internet não o prevê (Peterson & Davie, 2007, pp. 28-29).

- **Camada Internet**

A camada *Internet* é o eixo que sustenta toda a arquitetura. O seu trabalho consiste em permitir que os *hosts* injetem pacotes em qualquer rede e que esta os mantenha a deslocarem-se independentemente até ao seu destino. Estes até podem chegar completamente desordenados, cabendo depois às camadas superiores reordená-los. No fundo funciona como um sistema de correio, onde uma pessoa pode deixar uma sequência de cartas numa caixa de correio e elas vão parar ao destino mesmo que se separem ao longo do caminho (Tanenbaum, 2011, pp. 46-47). Esta camada consiste apenas num único

[12] Ex. Ethernet ou Fiber Distributed Data Interface (FDDI).
[13] Ex. Adaptador de rede.
[14] Ex. Driver de dispositivo de rede.

protocolo, o *Internet Protocol (IP)*. Este é o protocolo que suporta a interconexão de várias tecnologias de rede numa única só rede lógica (Peterson & Davie, 2007, p. 29).

- **Camada de transporte**

A terceira camada foi projetada para permitir que entidades pares nos *hosts* de origem e destino possam comunicar entre si, tal como na camada OSI. Esta camada contem dois protocolos principais – o *Transmission Control Protocol* (TCP) e o *User Datagram Protocol (UDP)*. O TCP e o UDP fornecem canais lógicos alternativos às aplicações. O TCP fornece o canal fiável[15] *byte-stream*[16] e o UDP o canal não fiável de entrega de mensagens (Peterson & Davie, 2007, p. 29). O "emissor TCP" segmenta o fluxo de *bytes* que chegam em mensagens discretas e passa cada uma delas para a camada *Internet*. No destino, o "TCP recetor" reorganiza as mensagens recebidas num só fluxo de saída. Este protocolo controla ainda o fluxo de dados e o nível de congestionamento da rede, para garantir que um emissor mais rápido não entope um recetor mais lento com mais mensagens do que este consegue manusear, nem vai sobrecarregar a rede com tráfego que não consegue ser escoado. Já o UDP consiste num protocolo desenhado para aplicações que não pretendem utilizar sequências ou controlo de fluxo TCP. É também frequentemente utilizado para pedidos e respostas entre o cliente e o servidor e aplicações em que a entrega rápida da informação é mais importante que a entrega correta, como na transmissão de áudio e vídeo (Tanenbaum, 2011, p. 47). Em linguagem de internet os protocolos TCP e UDP são muitas vezes chamados protocolos *end-to-end*, contudo é igualmente correto referirmo-nos a eles como protocolos de transporte (Peterson & Davie, 2007, p. 29).

- **Camada de aplicação**

O modelo TCP/IP não tem camadas de sessão e apresentação. Em vez disso, as aplicações simplesmente incluem as funções de sessão ou apresentação de que necessitem (Tanenbaum, 2011, p. 47). Acima da camada de transporte existe uma grande gama de protocolos de aplicação tais como o *File Transfer Protocol (FTP)*, o *Trivial File Transfer Protocol (TFTP)*, a *telnet*, o *Simple Mail Transfer Protocol (SMTP) e o BitTorrent* que possibilitam a interoperabilidade das aplicações. Para facilmente se entender a diferença

[15] Os *bytes* chegam exatamente na mesma ordem ao outro lado.
[16] Canal de comunicações onde uma entidade pode enviar uma sequência de *bytes* para a outra, sem erros.

entre um protocolo da camada de aplicação e uma aplicação, considerem-se todos os *browsers*[17] e servidores atualmente disponíveis. A única razão pela qual se pode usar qualquer uma destas aplicações para aceder a um local específico na web reside no fato de todas estas aplicações se encontrem a utilizar o protocolo da camada de aplicação *HTTP* (Peterson & Davie, 2007, p. 29).

2.2. Cibersegurança

Nas primeiras décadas da sua existência, as redes de computadores eram utilizadas principalmente por investigadores universitários para enviar *e-mails* e por empregados de algumas empresas, que as utilizavam para partilhar impressoras. Nestas condições a segurança nunca foi uma grande preocupação. Contudo atualmente existem milhões de utilizadores que as estão a utilizar para realizar operações bancárias, fazerem compras, apresentarem as suas declarações de impostos e, vulnerabilidade atrás de vulnerabilidade, os exemplos repetem-se indefinidamente (Tanenbaum, 2011, p. 763). A isto acresce o fato da internet ser amplamente partilhada por empresas concorrentes, governos antagónicos e oportunistas criminosos (Peterson & Davie, 2007, p. 586). Desta forma a segurança das redes tornou-se um problema de enormes proporções (Tanenbaum, 2011, p. 763).

2.2.1. Introdução

Por forma a permitir que os direitos, liberdades e garantias constitucionalizados sejam respeitados nas plataformas digitais, um número crescente de pessoas tem vindo a trabalhar no sentido de travar a expansão do nível de ameaças. Desta forma criou-se um conceito de segurança do *ciberespaço*, habitualmente conhecido por "*cibersegurança*".

A *cibersegurança* preocupa-se em garantir que terceiros não consigam ler, ou modificar mensagens destinadas a outros recetores. Preocupa-se em evitar que pessoas tentem aceder remotamente a serviços a que não estão autorizadas a usar. Preocupa-se também em arranjar maneiras de verificar se uma mensagem alegadamente das finanças é realmente das finanças e não de uma organização criminosa. Lida também com problemas de captura e reprodução de mensagens legítimas, sem autorização e ainda com

[17] Ex. Firefox, Safari, Internet Explorer, Lynx e Chrome.

pessoas que mais tarde tentam negar o envio dessas mesmas mensagens (Tanenbaum, 2011, p. 763).

A maioria dos problemas de segurança são intencionalmente criados por pessoas maliciosas que tentam ganhar algum benefício com isso (Tanenbaum, 2011, p. 763). Algumas das perversidades mais comuns encontram-se listadas na Tabela 1.

Hacker	**Motivo**
Estudante	Para se divertir a bisbilhotar *e-mails*
Cracker	Para testar o sistema de segurança de alguém ou roubar dados
Empresa	Para descobrir o plano estratégico de uma empresa concorrente
Contabilista	Para desviar dinheiro de uma empresa
Ladrão de identidade	Para roubar números de cartões de crédito e depois vender
Governo	Para descobrir segredos militares ou industriais de um inimigo
Terrorista	Para roubar projetos de armas biológicas

Tabela 1 - Motivos que podem levar as pessoas a causar problemas de segurança (Tanenbaum, 2011, p. 764)

Ao observarmos esta tabela facilmente compreendemos que tornar uma rede segura envolve bem mais do que mantê-la apenas livre de erros informáticos. Para tal é necessário ter vantagem sobre pessoas dedicadas, inteligentes e por vezes bem financiadas. Também deve ficar claro que as medidas tomadas, que vão frustrando os *hackers* ocasionais, terão sempre pouco impacto nos *hackers* profissionais. Os registos policiais mostram que a maioria dos ataques prejudiciais não são efetuados por *outsiders*[18], mas sim por *insiders*[19] que guardam rancor às pessoas/organizações que atacam. Desta forma os sistemas de segurança devem ser desenhados em concordância com este fator (Tanenbaum, 2011, p. 764).

Para além dos já mencionados casos de roubo de bens e serviços, existe ainda uma outra grande ameaça associada a estes perigos, que é a imagem e confiança que as pessoas têm nas empresas e organizações governamentais. A Tabela 2 resume alguns dos maiores problemas ao nível de segurança que existem atualmente na *internet* (Comer, 2009, p. 510).

[18] Pessoas de fora, que são externas a uma organização.
[19] Pessoas de dentro, que pertencem a uma organização.

Problema	Descrição
Phishing	Mascaramento igual ao de um local conhecido, com a finalidade de roubar dados pessoais
Deturpação	Fazer afirmações falsas ou exageradas de bens ou serviços, assim como simular vendas ou vender produtos de qualidade inferior ao descrito
Fraude	Levar utilizadores ingénuos a investir dinheiro ou serem cúmplices de crimes
Negação de serviço	Bloquear intencionalmente um local
Perda de controlo	Intruso obtém o controlo de um sistema de computadores
Perda de dados	Perda de propriedade intelectual ou de outra informação de valor

Tabela 2 - Principais problemas de segurança (Comer, 2009, p. 510)

São inúmeros os casos e problemas diferentes com que o Direito atualmente se depara, e certamente que se poderia elaborar um estudo referente apenas a estas questões, contudo aquilo que se pretende nesta investigação é saber de que forma os criminosos exploram a tecnologia e que tecnologias têm sido criadas para travar o crescimento dos *cibercrimes.*

2.2.2. Tecnologia de ataque

A Tabela 3 lista algumas técnicas conhecidas que os *hackers* habitualmente utilizam.

Técnica	Descrição
Wiretapping	Fazer uma cópia dos pacotes, enquanto estes atravessam a rede
Replay	Enviar pacotes capturados de uma sessão anterior[20]
Buffer overflow	Enviar mais dados do que o recetor está à espera, a fim de armazenar os valores das variáveis para além do *buffer*
Address Spoofing	Fingir o endereço de origem num pacote, para enganar o recetor no processamento dos pacotes

[20] Ex. Reenviar um pacote com a *password* capturada a partir de um *login* anterior.

Name Spoofing	Usar nomes bem conhecidos, com apenas um erro de ortografia, com ligações incorretas[21]
DoS e *DDoS*	Encher um local de pacotes para o impedir de continuar o seu normal funcionamento[22]
SYN flood	Enviar um fluxo aleatório de pacotes SYN[23], para esgotar a capacidade do recetor a nível de conexões TCP
Key breaking	Adivinhar automaticamente uma chave cripto ou uma *password* para ter acesso a dados não autorizados
Port Scanning	Tentar conectar-se a cada porto (existente no protocolo), num *host*, para tentar encontrar vulnerabilidades
Packet interception	Remover um pacote da *internet*, o que permite substitui-lo e efetuar ataques *man-in-the-middle*

Tabela 3 - Técnicas conhecidas utilizadas em ataques (Comer, 2009, p. 511)

De seguida ir-se-á analisar mais detalhadamente os principais ataques conhecidos às camadas de rede, transporte e multicamada.

- **Ataques na camada de rede**

Têm sido detetados e estudados a fundo vários ataques à camada de rede em artigos científicos. Ao atacarem os protocolos de roteamento, os atacantes conseguem absorver o tráfico da rede, colocar-se no caminho entre a origem e o destino e então controlar o fluxo de tráfego da rede, como demonstra a Figura 6 (a) e (b), onde um nó malicioso M se injeta no caminho de passagem entre o remetente S e o destinatário D (Wu, Chen, Wu, & Cardei, 2007, p. 110).

[21] Ex. www.micr0s0ft.com (Kim, Jeong, Kim, & So, 2011, p. 692). Esta técnica é bastante utilizada para ações de *spam*.
[22] A diferença entre ser *Denial of Service (DoS)* ou *Distributed Denial of Service (DDoS)* é que o DoS é feito por apenas um invasor que envia vários pacotes, enquanto o DDoS é distribuído por várias máquinas (Duke, 2002).
[23] Pacote TCP de sincronização.

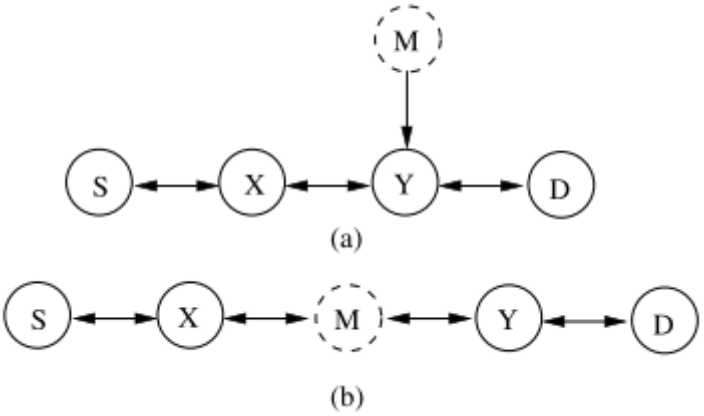

Figura 6 - Ataque de roteamento (Wu et al., 2007, p. 111)

Desta forma, os pacotes de tráfego podem ser encaminhados por uma via não-ótima, o que pode introduzir um atraso significativo. Os pacotes podem também ser desviados por um caminho inexistente e perderem-se. Os atacantes conseguem ainda criar ciclos de roteamento, criando congestionamentos na rede e contenção de canais em certas áreas. Se a estas situações acrescentarmos ainda a possibilidade dos atacantes atuarem em conjunto, as sinergias geradas conseguem inclusive impedir que o nó de origem encontre uma rota para o destino, criando um partição de rede, o que provoca controlo excessivo de trafego na rede e intensifica ainda mais o congestionamento da rede e a degradação do seu desempenho (Wu et al., 2007, p. 110).

Os ataques na camada de rede dividem-se essencialmente em três tipos: ataques na fase de descoberta de rota, na fase de manutenção de rota e na fase de transmissão de dados.

- **Ataques na fase da descoberta**

Existem ataques maliciosos que têm como alvo a fase da descoberta de rotas através do não seguimento dos protocolos de roteamento. Alguns desses ataques encontram-se listados abaixo.

- ***Routing table overflow attack***

Um nó malicioso alerta as rotas para que estas vão de nós não existentes para os nós autorizados na rede. O atacante tenta criar rotas suficientes para evitar que novas rotas sejam criadas. Este tipo de ataque é recorrente em algoritmos de roteamento proactivos que atualizam a informação periodicamente. Assim um atacante pode simplesmente

enviar alertas excessivos para sobrecarregar a tabela de roteamento da vítima (Wu et al., 2007, p. 111).

- ***Routing cache poisoning attack***

Neste caso o atacante tira vantagem do modo aleatório de atualização da tabela de encaminhamento, onde um nó que esteja à escuta pode retirar as informações de roteamento contidas no cabeçalho de um pacote para a sua própria *cache* de rota, mesmo que esse nó não esteja no seu caminho. Assim suponha-se que um nó M quer envenenar as rotas até ao nó X. O M pode enviar pacotes falsos, com rota de origem para o X, através do próprio M. Desta forma os nós vizinhos que ouvirem o pacote vão adicionar esta rota à sua *cache* de rotas (Wu et al., 2007, p. 111).

- **Ataques na fase de manutenção**

Existem ataques que têm como alvo a fase de manutenção da rota, que através do envio de mensagens falsas, de controlo e de erros de quebra de ligação, podem provocar a iniciação de operações de manutenção e reparação (Wu et al., 2007, p. 112).

- **Ataques na fase de transmissão dos dados**

Existem também ataques que têm como alvo a fase de transmissão dos pacotes de dados. Neste cenário os nós maliciosos participam cooperativamente no protocolo de roteamento tanto na fase de descoberta como na de manutenção, contudo na fase de transmissão dos dados, estes não enviam os pacotes de dados de acordo com a tabela de roteamento (Wu et al., 2007, p. 112).

- **Outros ataques conhecidos**

Recentemente foram também identificados ataques mais sofisticados em alguns artigos científicos.

- ***Wormhole* e *Grayhole attack***

Este método aproveita-se de uma infraestrutura preexistente no cenário, para forçar que as rotas usem sempre o nó atacante. O objetivo do ataque *wormhole* é forçar uma grande quantidade de mensagens a passar necessariamente pelos dispositivos atacantes (Bastos & Albini, 2012). As intenções deste nó malicioso podem ser travar o processo de descoberta ou intercetar todos os pacotes que estão a ser enviados para o nó

de destino (Domingos, 2012, p. 18). Desta maneira este pode intercetar mensagens ou simplesmente neutralizar a rede, enviando informações desconexas para os nós comunicantes. Para tal dois nós atacantes distantes estabelecem uma ligação de comunicação a alta velocidade entre eles e comunicam a uma taxa de transferência acima do padrão da rede. Desta forma se dois nós íntegros pretenderem comunicar um com o outro utilizarão a rota atacante, pois esta é mais rápida (Bastos & Albini, 2012). Existe ainda uma forma mais delicada deste ataque, chamada *grayhole attack*, onde o nó malicioso de vez em quando solta os pacotes de dados, tornando a sua deteção ainda mais difícil (Domingos, 2012, p. 18).

- ***Byzantine attack***

Neste, um nó comprometido ou conjunto de nós comprometidos trabalham em colisão e realizam ataques como criação de ciclos de roteamento, envio de pacotes em rotas não ótimas e seleção de pacotes a soltar. Estes ataques são bastante difíceis de detetar, uma vez que a rede geralmente não exibe um comportamento estranho (Domingos, 2012, p. 18).

- ***Resource depletion attack***

Neste tipo de ataque, um nó malicioso tenta esgotar os recursos de outros nós na mesma rede. Os recursos alvo típicos são normalmente a bateria, a largura de banda e o poder computacional. Os ataques podem ser feitos na forma de pedidos desnecessários para rotas, geração de pacotes balizadores ou envio de pacotes obsoletos para outros nós (Domingos, 2012, p. 19).

- ***Information disclosure***

Um nó comprometido pode vazar informações importantes ou confidenciais para nós não autorizados na rede. Isto inclui informação em relação à topologia da rede, localização geográfica dos nós e rotas ótimas para nós autorizados na rede (Domingos, 2012, p. 19).

- **Ataques na camada de transporte**

Conforme já foi referido, os objetivos do TCP incluem estabelecer uma conexão *end-to-end*, entregar os pacotes através do canal fiável *end-to-end*, controlo de fluxo e de

congestão e manter a conexão end-to-end limpa. Assim o TCP é vulnerável aos clássicos SYN *flooding attacks* e aos *session hijacking attacks*.

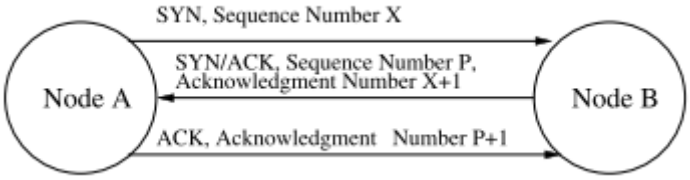

Figura 7 - TCP Three-way handshake (Wu et al., 2007, p. 114)

Uma vez que o TCP não tem nenhum mecanismo que permita distinguir se uma perda foi causada por congestão, erro aleatório ou ataque malicioso, muitas vezes este diminui a sua janela de congestão devido à existência de perdas, o que degrada significativamente a performance da rede (Wu et al., 2007, p. 114).

- **SYN *flooding attack***

O SYN flooding attack é um ataque de negação de serviço em que o atacante estabelece um grande número de pedidos de conexões TCP com um nó vítima (através do envio de vários pacotes SYN), mas nunca completa o procedimento *handshake*, que se pode observar na Figura 7, para estabelecer completamente a conexão. Geralmente existe um tempo limite associado a cada conexão pendente, logo as conexões meio-abertas eventualmente acabam por expirar, contudo os nós maliciosos podem continuar a enviar pacotes a requerer conexões, mais rápido do que o período de expiração de cada conexão pendente (Wu et al., 2007, p. 114).

- ***Session hijacking*[24]**

Este tipo de ataque aproveita-se do fato de várias comunicações estarem protegidas através da configuração de sessões[25]. Numa sessão TCP o atacante finge ser o IP da vítima, determina a sequência correta que é esperada pelo alvo e depois inicia um ataque DoS à mesma. De seguida este assume o nó da vítima e continua a sessão com o alvo (Wu et al., 2007, p. 114).

O *TCP ACK storm*, pode ser criado quando um atacante inicia um TCP *session hijacking*. Conforme podemos observar na Figura 8, o atacante envia dados para a sessão,

[24] Roubo de sessão.
[25] Fornecimento de credenciais.

fazendo-se passar pelo nó B, ao que o nó A irá dar o ACK dos pacotes ao nó B. Este pacote não irá conter os número de sequência que o nó B esperava pelo que quando este recebe o pacote, tenta ressincronizar a sessão TCP com o nó A, enviando-lhe o pacote de ACK que este estava à espera de receber. O ciclo continua e os pacotes andam para a frente e para trás, criando um *ACK storm.* Caso a sessão seja UDP, os atacantes nem têm de se preocupar com a gestão de números de sequência e outros mecanismos TCP. Uma vez que o UDP é *connectionless*, moldar uma sessão sem ser detetado é bastante mais fácil que numa sessão TCP (Wu et al., 2007, p. 115).

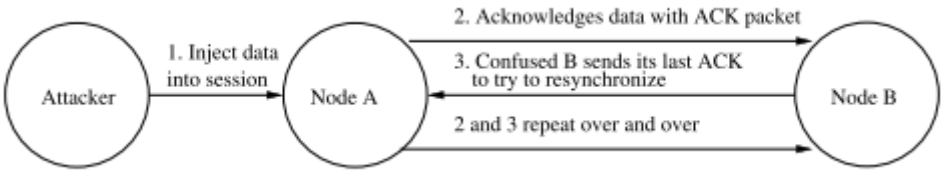

Figura 8 - TCP ACK Storm (Wu et al., 2007, p. 115)

- **Ataques multicamada**

Alguns *ciberataques* podem ser desencadeados a partir de múltiplas camadas, em vez de uma só. Como exemplo desses ataques temos o DoS, *man-in-the-middl*e e os *impersonation attacks*.

- **DoS**

Os ataques DoS podem ser desencadeados em várias camadas. Na camada física, o intruso pode fazer *jamming* a um sinal o que interrompe as comunicações. Na camada de ligação, os nós maliciosos podem ocupar canais através do efeito de captura, aproveitando-se do esquema exponencial binário dos protocolos *Message Authentication Code* (MAC), impedindo assim que outros nós acedam ao canal. Na camada de rede, conforme anteriormente referido, o processo de roteamento pode ser interrompido através do controlo de pacotes, *table overflow* ou envenenamento (*poisoning)*. Na camada de transporte e aplicação o *SYN flooding* e o *session hijacking* pode provocar ataques DoS (Wu et al., 2007, p. 116).

- **Ataques de personificação**

Utilizam a identidade de outros nós, como o endereço MAC ou IP. Muitas vezes são o primeiro de uma sequência de muitos ataques e utilizados para iniciar ataques mais sofisticados (Wu et al., 2007, p. 116).

- **Ataques *man-in-the-middle***

O invasor coloca-se entre o emissor e o recetor e vê a informação que está a ser enviada entre os dois extremos. Em alguns casos o invasor consegue persuadir o emissor a comunicar com o recetor e vice-versa. (Wu et al., 2007, p. 116).

2.2.3. Tecnologia de defesa

Para tentar minorar as perturbações causadas pelos *hackers*, têm-se desenvolvido bastantes produtos a nível de segurança que executam uma grande variedade de funções (Comer, 2009, p. 515).

A Tabela 4 resume a tecnologia que esses produtos utilizam.

Técnica	**Objetivo**
Hashing	Integridade dos dados
Encriptação	Privacidade
Assinaturas digitais	Autenticação de mensagens
Certificados digitais	Enviar autenticação
Firewalls	Integridade do local
Intrusion Detection Systems (IDS)	Integridade do local
Content analysis	Integridade do local
VPN	Privacidade dos dados

Tabela 4 - Tecnologia de segurança (Comer, 2009, p. 515)

De seguida ir-se-á aprofundar em que consiste a tecnologia de que dispomos.

- ***Hashing***

O método de *hashing* fornece um código de autenticação de mensagens (MAC), que um intruso não consegue quebrar ou forjar. Este código baseia-se numa chave secreta conhecida apenas pelo emissor e recetor. O remetente pega numa mensagem como *input*

e utiliza a chave para gerar um *hash* H e transmite o H com a mensagem. O H é um pequeno conjunto de bits e o comprimento de H é independente do tamanho da mensagem. Ao chegar ao destino o recetor utiliza a chave para gerar um *hash* da mensagem e compara o *hash* com H. Se os dois combinarem, a mensagem chegou intata. Desta forma um atacante que não tenha a chave, não vai ser capaz de modificar a mensagem sem introduzir um erro (Comer, 2009, p. 515).

- **Controlo de acesso e *passwords***

Existem mecanismos de controlo de acesso que controlam quais os utilizadores e aplicações que podem aceder a determinados dados. Este controlo pode ser feito através do uso de *passwords* ou através da implementação de listas de controlo de acesso, que especificam detalhadamente quem pode aceder aos objetos. Quando as listas de controlo de acesso e *passwords* percorrem redes de dados (filares ou sem fios) corre-se o risco de estas serem escutadas e copiadas, por qualquer pessoa que se encontre ao alcance dos pacotes transmitidos (Comer, 2009, p. 516).

Como exemplo de aplicações de controlo de acesso numa LAN, pode-se considerar a rede interna da marinha portuguesa, onde é necessário haver um utilizador registado, com a respetiva *password*, para se poder aceder à rede *wireless* e o computador estar registado na lista de controlo de acesso para se poder conectar através de ligação *ethernet*.

- **Cifra**

A criptografia é fundamental, pois a encriptação permite confidencialidade dos dados, autenticação de mensagens, integridade dos dados e evita que os ataques se repitam. Fundamentalmente um remetente cifra os dados para baralhar os *bits* da mensagem de tal forma que apena o recetor os consiga ordenar novamente (Comer, 2009, p. 516).

Conforme podemos observar na Tabela 5 a tecnologia de encriptação divide-se em quatro partes.

Terminologia	Significado
Plaintext	Mensagem antes de ser cifrada
Cyphertext	Mensagem após ser cifrada
Encryption key	Pequeno conjunto de bits usados para cifrar uma mensagem
Decryption key	Pequeno conjunto de bits usados para decifrar uma mensagem

Tabela 5 - Partes da tecnologia de encriptação (Comer, 2009, p. 516)

Matematicamente, pensa-se na cifra como uma função, que tem dois argumentos, uma chave K_1 e a mensagem original para ser cifrada, M. O algoritmo produz uma versão cifrada da mensagem, C. Assim, podemos representar a função *cyphertext* como (Comer, 2009, p. 517):

$$C = encrypt\ (K_1, M)\ . \tag{1}$$

Para decifrar a mensagem, basta efetuar a inversa da função usada no processo de cifra, logo (Comer, 2009, p. 517):

$$M = decrypt\ \big(K_2, encrypt(K_1, M)\big)\ . \tag{2}$$

- **Autenticação com assinaturas digitais**

Para autenticar o remetente de uma mensagem podem-se também utilizar mecanismos de cifra. Esta técnica é conhecida por assinatura digital. Para assinar uma mensagem o remetente cifra uma mensagem utilizando uma chave conhecida apenas por este. De seguida o recetor utiliza a função inversa para decifrar a mensagem e sabe quem enviou a mensagem, pois apenas este emissor tem a chave para efetuar a cifra (Comer, 2009, p. 518).

- **Certificados digitais**

Um dos principais problemas a nível de segurança que rodeiam as chaves públicas é a forma como estas são obtidas. Assim é possível disponibilizarem-se publicações convencionais[26] para obrigar as pessoas a introduzi-las manualmente nos computadores. Contudo continua sempre a existir a possibilidade de serem desenvolvidos sistemas automáticos que distribuam as chaves públicas, este problema é conhecido como *key*

[26] Semelhantes a listas telefónicas.

distribution problem. Para o resolver têm sido desenvolvidos vários mecanismos que assentam num princípio: sabendo uma chave (a chave pública de uma chave de autoridades), é possível obterem-se chaves de forma segura. Assim um administrador apenas precisa de configurar uma chave pública (Comer, 2009, p. 520).

- ***Firewalls***

Apesar da tecnologia de encriptação ajudar a resolver muitos problemas, existe outra tecnologia que também é importante. Conhecida como *Internet Firewall*, esta tecnologia ajuda a proteger os computadores pertencentes a uma organização e as redes de tráfego não pretendido. A *firewall* encontra-se entre a organização e o resto da *internet* e todos os pacotes que entrem ou saiam da organização passam pela *firewall* (Comer, 2009, p. 521). A Figura 9 ilustra o funcionamento de uma *firewall*.

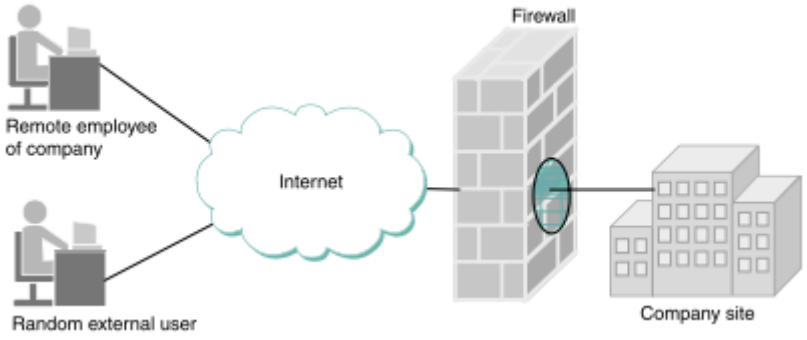

Figura 9 - Firewall entre um local e o resto da internet (Peterson & Davie, 2007, p. 627)

A *firewall* é a ferramenta de segurança mais importante utilizada para gerir conexões entre duas organizações que não se conhecem uma à outra. De forma resumida, pode-se então afirmar que (Comer, 2009, p. 521):

- Todo o tráfico que entra e sai da organização passa pela *firewall*;
- A *firewall* implementa uma política de segurança e descarta pacotes que não satisfazem as políticas estabelecidas;
- A *firewall* em si é imune a *ciberataques.*

- ***Intrusion Detection Systems* (IDS)**

Um IDS monitoriza todos os pacotes que chegam a um local e notifica o administrador do local se forem detetadas violações de segurança. O IDS funciona como

uma camada de segurança extra e mesmo que a *firewall* detete o ataque, o IDS pode notificar o administrador do local de que ocorreu um problema (Comer, 2009, p. 524).

A maioria dos IDS podem ser configurados para detetar um tipo específico de ataque. Assim, por exemplo, o IDS pode ser configurado para detetar *port scanning attacks* onde o atacante envia *datagramas* UDP para sucessivas portas UDP ou tenta iniciar conexões TCP em várias portas TCP. Nalguns casos o IDS e a *firewall* estão interconectados para providenciar filtros automáticos, porém, é necessário ter sempre presente que o IDS e a firewall funcionam em níveis diferentes da pilha de protocolos. Desta forma, em vez de meramente notificar o administrador do local acerca do problema, o IDS cria uma regra na *firewall* que bloqueia pacotes que estão a causar problemas. O motivo por que se utiliza este sistema é a rapidez, um humano demora muitos segundos a responder após ser notificado de um problema e numa rede de 1 *gigabit*[27], podem chegar mais de 50.000 pacotes por segundo (Comer, 2009, p. 524).

A grande diferença entre o IDS e a *firewall* é que o IDS inclui o estado da informação. Isto é, enquanto a *firewall* aplica regras a cada pacote, o IDS mantem o histórico dos pacotes. Assim, enquanto a *firewall* determina se deve admitir um pacote SYN, o IDS consegue observar que estão a chegar muitos SYNs de um só remetente (Comer, 2009, p. 524).

Uma vez que exige mais cálculo e memória de acesso, o IDS não consegue lidar com muitos pacotes por segundo (Comer, 2009, p. 524).

- ***Content analysis***

Apesar de evitar muitos problemas de segurança, a *firewall* de rede tem uma grande limitação, esta apenas examina campos nos cabeçalhos IP dos pacotes. Isto é, a *firewall* não consegue testar o conteúdo de um pacote. Uma das maneiras mais comuns de introduzir vírus numa organização é através de anexos de *e-mail*[28]. Como é que esta pode então prevenir este tipo de problemas? Recorrendo-se à análise de conteúdos. Esta divide-se em dois tipos: *file sacanning* e *Deep Packet Inspection* (DPI) (Comer, 2009, p. 524).

[27] 1 Gb = 10^9 bits.
[28] Se o utilizador abrir o anexo, o programa pode instalar *software* no computador, incluindo *malware* e *vírus* (Comer, 2009, p. 524).

- ***File Scanning***

Este método de análise de ficheiros inteiros, consiste numa técnica bastante conhecida e utilizada pelos *softwares* de segurança instalados nos computadores. Para tal o *file scanner* adquire um ficheiro como *input* e procura padrões de *bytes* que indiquem problemas. Por exemplo, muitos *scanners* de vírus procuram sequências de bytes conhecidas como *fingerprint*[29] (Comer, 2009, p. 525).

- ***Deep Packet Inspection* (DPI)**

O segundo método da análise de conteúdos opera em pacotes em vez de ficheiros. Isto é, em vez de meramente examinar os cabeçalhos dos pacotes que passam para um local, o DPI também examina os dados contidos dentro de cada pacote (Comer, 2009, p. 525).

- ***Virtual Private Networks* (VPN)**

Uma das mais importantes e amplamente utilizadas tecnologias de segurança utiliza encriptação para fornecer um acesso seguro à *intranet* de uma organização a partir de locais remotos. Esta tecnologia foi originalmente desenhada para fornecer uma interconexão com baixos custos entre múltiplos locais numa organização. Numa VPN uma organização aluga circuitos para conectar os locais. Cada conexão alugada estende-se de um *router* de um lado até a um router noutro lado. A informação passa diretamente de um *router* para o outro. A Figura 10 ilustra as duas possibilidades de uma organização ligar três locais. (Comer, 2009, p. 525).

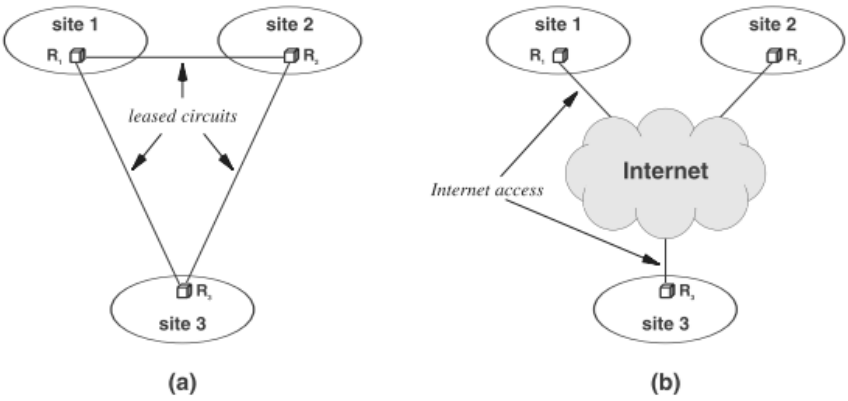

Figura 10 - Locais conectados através de VPN (a) e Internet (b) (Comer, 2009, p. 525)

[29] Impressões digitais.

A principal vantagem entre utilizar circuitos alugados para conectar locais reside no fato de a rede ser completamente privada e portanto confidencial. A principal vantagem de utilizar a *internet* é o seu baixo custo, em vez de se pagar circuitos para conectar locais, a organização apenas precisa de pagar pelo serviço de *internet* de cada lugar (Comer, 2009, p. 526).

A VPN combina o melhor de dois mundos ao usar a internet para transferir dados entre locais e efetuar passos adicionais para garantir que os dados não podem ser acedidos por *outsiders*[30]. Ou seja, em vez de conexões alugadas muito dispendiosas, a VPN utiliza cifra. Contudo, caso uma organização pretenda estar ainda mais imune a ataques, esta pode sempre dedicar servidores para o serviço de VPN e utilizar a *firewall* para proibir os *routers* da VPN de aceitarem pacotes não autorizados (Comer, 2009, p. 527).

- **Defesa na camada de rede**

Alguns ataques ligados à modificação de mensagens de roteamento podem ser previstos por mecanismos como a autenticação da origem e a integridade da mensagem. Os ataques de DoS podem ser evitados através da limitação ao atacante de inserir ciclos de roteamento (Tseng, Ni, Chen, & Sheu, 2002, p. 122).

- **Defesa contra *wormhole attacks***

Num ataque *wormhole* o atacante recebe pacotes a partir de um ponto da rede, encaminha-os para outro ponto e de seguida reenvia-os novamente para a rede a partir desse ponto. Assim, em teoria, se os *bits* de dados forem transferidos numa modelação especial conhecida apenas pelos seus nós vizinhos, são resistentes a *wormholes.* Uma outra possibilidade é utilizar métodos de prevenção através de sistemas de deteção de intrusões (Tseng et al., 2002, p. 122). A Tabela 6 lista algumas das principais técnicas de defesa contra *wormhole attacks*.

[30] Pessoas estranhas à organização.

Técnica	Objetivo
Packet leashes	Pacote temporário de *leash*[31] que estabelece um tempo de vida para cada pacote, o que o restringe a viajar longas distâncias
Sector	Técnica de delimitação de distância, baseada em sequências de *hash* (não requer sincronização de tempo ou informação de localização)

Tabela 6 - Técnicas de defesa contra ataques wormhole.

- **Defesa contra ataques de personificação e rejeição**

Existem vários protocolos que utilizam técnicas de autenticação e não repúdio, cada um com as suas fraquezas, para evitar este tipo de ataques (Wu et al., 2007, p. 124). Estes protocolos têm como objetivo certificar que o remetente da mensagem é quem diz ser (Russell & Gangemi, 1991, p. 176) e garantir que o remetente e o recetor não conseguem negar que enviaram/receberam a mensagem (Russell & Gangemi, 1991, p. 195).

- **Defesa na camada de transporte**

Tanto o protocolo TCP na *internet*, como os nós nas redes sem fios, estão vulneráveis a SYN *flooding attacks* ou *session hijacking attacks*. As encriptações ponto-a-ponto ou *end-to-end* mantêm a confidencialidade das mensagens na camada de transporte ou acima desta. Até ao momento já se desenvolveram protocolos de segurança de comunicações baseados em chaves públicas, como o *Secure Socket Layer* (SSL), o *Transport Layer Security* (TLS) e o *Private Communications Transport* (PCT). Estes protocolos ajudam a manter os dados seguros durante a transmissão, evitando ataques de mascaramento, *man-in-the-middle* e de repetição (Wu et al., 2007, p. 126).

- **Defesa contra ataques multicamada**

Conforme já foi referido, existem táticas maliciosas que têm como alvo várias camadas. Assim sendo, as contramedidas também terão de ser implementadas em várias

[31] Informação adicionada em cada pacote para restringir a distância de transmissão.

camadas, podendo ser tomadas através de atuação independente[32] ou conjunta[33] (Wu et al., 2007, p. 127).

2.2.4. Conclusões

As redes de computadores, e a *internet* podem ser utilizadas para atividades criminosas (Comer, 2009, p. 531). A segurança é um aspeto importante que pode determinar o sucesso e implementação de uma rede (Wu et al., 2007, p. 131).

Conforme foi referido existem já diversas tecnologias de defesa contra *ciberataques*. No caso particular da marinha portuguesa, existem também várias em uso, que podemos consultar na Tabela 7.

Plataforma	***Software***
SIEM[34]	*IBM QRADAR 3510*
Firewall Perímetro (IDS e *anti-bot*)	*Checkpoint NG 77.20*
Firewall Interna	*CISCO ASA 5520*
MPS[35]	*Anubisnetwork*
Proxy server	*CISCO IronPort*
Web Publishing	*Microsoft Threat Management Gateway*
Autenticação	*Microsoft Active Directory*
VPN[36] IPSEC	*Cisco ASA 5540*
VPN SSL	*Checkpoint Remote Access*
Antivírus corporativo	*Symantec Endpoint Protection 12.1*

Tabela 7 - Plataformas de segurança atualmente em uso na Marinha Portuguesa

Relativamente a estas plataformas é importante salientar que:

- O SIEM consiste numa plataforma de correlação e gestão de eventos de segurança. Este permite efetuar uma análise em tempo real e simultaneamente uma análise "forense" com base em eventos registados no passado (*content analysis*);

[32] As ações contra o mesmo tipo de ataque são tomadas em cada camada, sem haver qualquer colaboração entre elas.
[33] Existe colaboração entre as várias camadas (ex. Se um nó detetar uma intrusão numa camada mais alta, as mais baixas são alertadas para investigarem.).
[34] *Security Information and Events Management* (SIEM).
[35] *Mail Protection System* (MPS).
[36] *Virtual Private Network* (VPN).

- A *firewall* de perímetro analisa o tráfego que entra na rede de marinha (incluí as funcionalidades de IDS e anti-bot). A *firewall* interna, de proteção aos sistemas, providencia serviços de topo à organização (*e-mail*, DNS e domínio);
- O MPS recorre à análise de conteúdos (*content analysis*);
- O *proxy server* funciona como um intermediário que se encontra entre os clientes e a WWW, o que lhe permite utilizar técnicas como o *file scanning* para detetar ameaças. Este obriga ainda à autenticação do utilizador e efetua o registo de acessos;
- O *web publishing* garante a proteção dos *sites* que são colocados *online* e que de algum modo permitam a interação com os utilizadores nas redes públicas, nomeadamente através da monitorização e autorização prévia (autenticação).
- O *software* "*Microsoft Active Directory"* controla as autenticações de todos os utilizadores na rede;
- Está-se a recorrer a dois tipos de tecnologia VPN. Às VPN IPSEC, que se encontram implementadas na *firewall* de perímetro (permitem ou negam acesso remoto às sub-redes de computadores) e às VPN SSL que se encontram por trás da *firewall* de perímetro (com regras que permitem ou não aceder a aplicações ou dados);
- O antivírus corporativo utiliza algumas técnicas para deteção de ameças como a análise de assinaturas e a deteção de comportamentos suspeitos (tal como no IDS).

Apesar da sua reconhecida eficácia, todas estas plataformas são elaboradas com base em conjuntos de regras que atribuem valores específicos às ameaças já conhecidas, o que as torna numa brecha defensiva possível de ser explorada por novas ameaças.

Capítulo 3 - *Data mining*

Para tentar preencher esta brecha, existem alguns investigadores (Singhal, 2007, pp. 1-3) que acreditam que é necessário desenvolverem-se novas técnicas para detetar ataques e descobrir vulnerabilidades e que para tal existe uma grande necessidade de se conseguir, automaticamente, analisar dados, resumi-los e prever futuras tendências, entrando-se assim no âmbito do *machine learning* e *data mining*[37].

3.1. Definição de *data mining*

De acordo com o *MIT Technology Review* o *data mining* é uma das dez tecnologias emergentes que irá mudar o mundo (Larose, 2005, p. 2). *Data mining* consiste no processo de descobrir significativamente novas correlações, padrões e tendências através do processamento de grandes quantidades de dados, armazenados em repositórios, utilizando tecnologias de reconhecimento de padrões, estatística e técnicas matemáticas (Freitas, 2013, pp. 1-2).

3.1.1. Vantagens

Todos os dias somos inundados com inúmeros dados em diversas áreas (Liu & Motoda, 2012, p. xix). Contudo, não existem analistas humanos suficientes para transformar toda esta informação em conhecimento (Hand, Mannila, & Smyth, 2001, pp. 1-3). Assim o notável crescimento desta área tem sido alimentado por uma confluência de fatores (Larose, 2005, p. 4):

- Crescimento explosivo da recolha de dados;
- Armazenamento dos dados em *data warehouses;*
- Aumento da disponibilidade de acesso dos dados, a partir da *internet*;
- Desenvolvimento de *softwares* de *data mining;*
- Melhoramento da capacidade de computação a nível de processamento e armazenamento.

3.1.2. Aplicação

O *data mining* pode ser aplicado a qualquer tipo de dados, sendo que as formas mais comuns de dados são: as bases de dados, os *data warehouses* e os dados

[37] Prospeção (processamento) de dados;

transacionais. A Tabela 8 resume o significado de cada uma (Han, Kamber, & Pei, 2011, pp. 8-14).

Terminologia	**Significado**
Base de dados	Coleção de dados interrelacionados e respetivo conjunto de *softwares* para gerir e aceder aos mesmos
Data warehouse	Grande repositório de informação (Zhao & Liu, 2011, p. 109) recolhida a partir de múltiplas fontes, armazenada num esquema unificado e habitualmente guardada num único local
Dados transacionais	Dados obtidos a partir de transações. Geralmente incluem um número de transação e uma lista com os itens que compõem a transação

Tabela 8 - Formas básicas de dados em data mining

3.1.3. Fases de um projeto

De acordo com a *Cross-Industry Standard Process for Data Mining* (CRISP-DM), conforme se pode observar na Figura 11, um dado projeto de *data mining* tem um ciclo de vida de seis fases (Liu & Motoda, 2012, pp. 10-12). Uma primeira fase em que se tenta compreender o negócio, isto é, onde se enunciam os objetivos e requisitos do projeto e onde se traça uma estratégia para atingir esses objetivos. Na segunda fase recolhem-se os dados e avalia-se a qualidade dos mesmos. De seguida, na terceira fase preparam-se os dados até se ter um *data set*[38] final que se irá usar nas fases seguintes. Na quarta fase selecionam-se e aplicam-se técnicas de modulação e calibram-se os modelos por forma a se otimizarem os resultados. Já na quinta fase avaliam-se os modelos e a sua qualidade em função dos objetivos definidos na primeira fase, antes de se implementarem no terreno. Por fim, na sexta fase aplicam-se os modelos criados (Larose, 2005, pp. 5-10).

[38] Conjunto de dados.

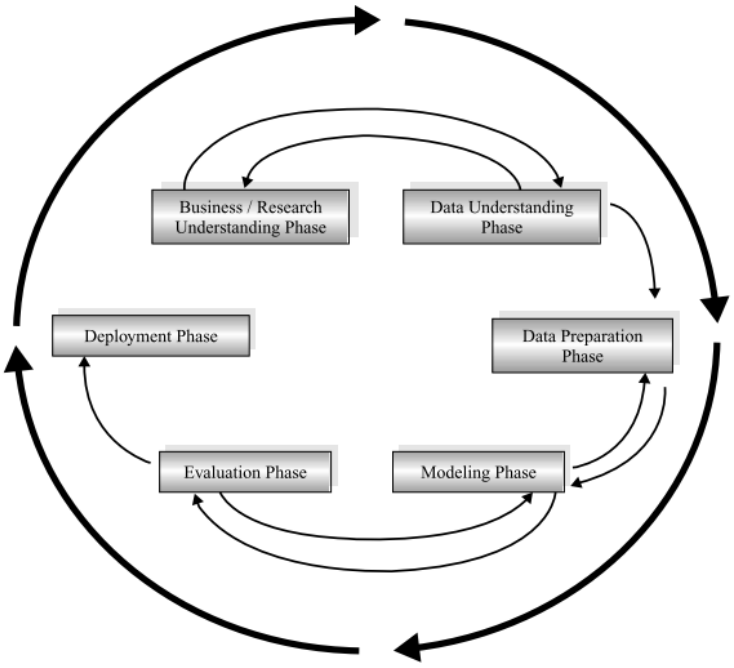

Figura 11 - Ciclo de vida de um projeto de data mining (Larose, 2005, p. 6)

3.1.4. Objetivos

O *data mining* por ser utilizado para vários fins, sendo que os mais comuns são a descrição, estima, predição, classificação, *clustering* e associação (Larose, 2005, pp. 11-17). Na descrição os investigadores utilizam o *data mining* para encontrar formas de descrever padrões e tendências contidos nos dados. Na estima efetua-se uma classificação dos dados, sendo que a variável alvo é sempre numérica. No que toca à predição, também é semelhante à classificação, contudo o objetivo é tentar prever tendências futuras. Na classificação utiliza-se uma variável alvo categórica, que pode ser particionada em várias categorias. Já o *clustering* refere-se a agrupamentos de registos, observações ou casos em classes de objetos semelhantes. Por fim, pode ainda ser utilizado para efetuar associações, que consiste em procurar atributos que estejam relacionados (Hand et al., 2001, pp. 12-15).

3.2. Trabalho relacionado: *Data mining & cyber security*

O *data mining* está já atualmente a ser aplicado no desenvolvimento de modelos preditivos que permitam dar uma resposta em tempo real sempre que ocorra uma dada sequência de processos (Dua & Du, 2011, p. 6). Existem assim já várias áreas de aplicação do mesmo, no que toca à cibersegurança, contudo esta investigação concentra-se apenas

nos problemas relacionados com o trafego de rede, conduzindo-nos assim para uma área mais específica.

Através do *data mining* e *machine learning* é possível traçar perfis que agrupem conexões semelhantes e procurar comportamentos dominantes. A maioria dos investigadores nesta área encontra-se principalmente a estudar as comunicações estabelecidas entre *hosts*. Até ao momento, as dificuldades encontradas resumem-se principalmente a dois fatores: o fluxo de tráfego a circular na rede é enorme e é extremamente difícil identificar todas as regras capazes de descrever o fluxo de tráfego, de forma eficiente e eficaz (Dua & Du, 2011, pp. 160-161).

Em termos de investigação nesta área, por parte de Portugal ainda pouco foi feito. Existem vários documentos e teses de mestrado que alertam para a importância do *ciberespaço*, no âmbito do Direito e da Defesa, contudo em termos de utilização de técnicas para caraterizar *ciberameaças*, nomeadamente de *data mining*, existe muito pouca informação publicada.

A nível internacional (Singhal, 2007) e (Dua & Du, 2011) definem a forma como o *data mining* pode ser aplicado neste âmbito definindo ainda quais os meios disponíveis para o fazer, porém não fazem uma análise detalhada às ameaças propriamente ditas.

Um estudo realizado pela Academia Militar dos EUA (Barbara, Couto, Jajodia, Popyack, & Wu, 2001) aborda bem esta questão, apoiando-se no sistema *Audit Data Analysis and Mining* (ADAM). Este conclui que o comportamento do ADAM nos dá uma boa evidência da utilidade das técnicas de *data mining* em deteção de intrusões, apesar de haver um grande número de problemas por resolver, tais como o ajustamento/sintonização entre o que deve ser considerado uma ligação suspeita ou não, a criação de perfis padrão por forma a se saber aquilo que deve ser considerado um comportamento normal ou não (sugerindo-se uma criação de um modelo padrão para cada dia da semana e parte do dia) e ainda a dependência do conjunto de treino (devido à dificuldade de distinguir ligações normais de ataques, sem bastante trabalho manual). Os autores desta pesquisa referem ainda que seria bastante interessante tentar diminuir a dependência dos conhecimentos prévios na deteção de ataques (utilizando-se técnicas baseadas em estimadores de *pseudo-bayes*), tentar automatizar o processo de obtenção de um perfil de atividade normal (através da eliminação de *outliers*) e detetar outro tipo de

ataques (uma vez que estes apenas se restringiram a ataques PROBE[39] e DoS). Uma ideia sugerida era que se tentasse modelar a regularidade desses ataques através de séries temporais (podendo-se depois usar uma ferramenta preditiva para detetar os ataques).

3.3. *Machine Learning*

Consiste no conjunto de técnicas desenvolvidas para possibilitar que uma dada máquina aprenda. Como saber se uma máquina aprendeu? Esta questão levanta várias outras questões filosóficas, contudo podemos considerar que uma máquina aprendeu sempre que esta altera o seu comportamento para melhor do que no passado (Witten & Frank, 2005, pp. 7-9).

O *machine learning* é também ele uma área do *data mining*, que combina inteligência artificial com estatística e tem por intuito gerar algoritmos que aprendam ao explorar o espaço de *n* dimensões de um determinado conjunto de dados e encontrem generalizações aceitáveis. Uma das suas principais vertentes é o *machine learning* indutivo, em que a generalização é obtida a partir de um conjunto de amostras/instâncias e formalizada recorrendo a diferentes técnicas e modelos. A aprendizagem indutiva pode ser definida como o processo que estima as dependências *input-output* desconhecidas ou a estrutura de um sistema, recorrendo a um número limitado de observações ou a medidas de *inputs* e *outputs* do sistema (Kantardzic, 2011, pp. 89-101). A Figura 12 representa o processo de aprendizagem descrito:

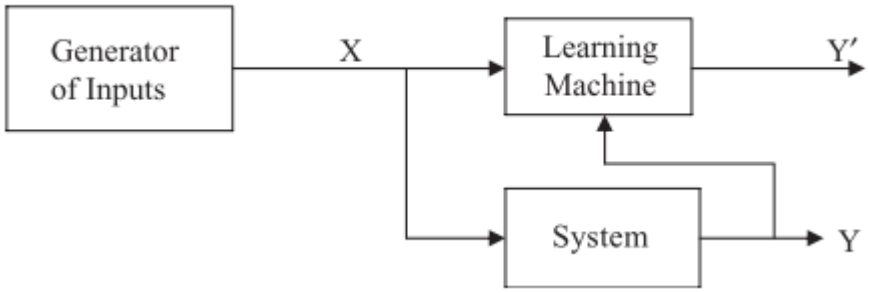

Figura 12 - Processo de machine learning utilizando as informações do sistema para gerar um output Y' para o input X (Kantardzic, 2011, p. 89)

[39] Ação tomada ou objeto usado com o propósito de retirar informações acerca do estado da rede. (Rouse, 2005)

Como exemplos de aprendizagem indutiva destacam-se os problemas de interpolação, regressão, classificação, *clustering* e estima da densidade.

Ainda dentro da aprendizagem indutiva, existem dois tipos de métodos: aprendizagem supervisionada e aprendizagem não supervisionada. A aprendizagem supervisionada está associada ao processo de classificação, em que a supervisão do sistema passa por aprender com os exemplos identificados do conjunto de treino. A aprendizagem não supervisionada está basicamente associada ao *clustering*. Neste método, são descobertas classes de dados dentro do conjunto de treino, os exemplos não estão previamente classificados, assim sendo, não existe a supervisão do sistema (Han et al., 2011, pp. 24-25).

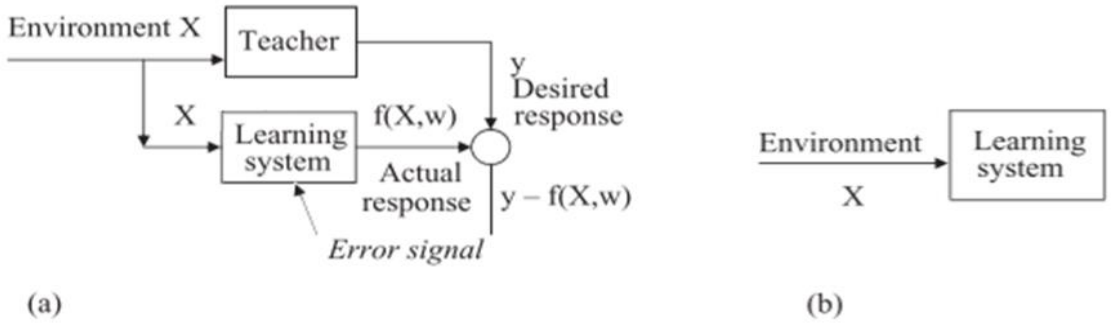

Figura 13 - Dois tipos de aprendizagem indutiva: Aprendizagem supervisionada a) e aprendizagem não supervisionada b) (Kantardzic, 2011, p. 100)

3.4. Aprendizagem supervisionada

Neste método de aprendizagem indutiva, como referido anteriormente, o algoritmo utilizado aprende com os exemplos identificados do conjunto de treino e associa cada novo caso aos que já conhece (Witten & Frank, 2005, p. 43). No âmbito desta dissertação, foram selecionados três classificadores distintos baseados em aprendizagem supervisionada, que apesar da dificuldade que existe *a priori* em definir qual o algoritmo que tem melhores resultados para um dado conjunto de dados, se consideram eficazes, dado o seu já reconhecido bom desempenho (Gama & Brazdil, 1995). Assim utilizar-se-á o naive de Bayes, o classificador vizinho mais próximo (IBK) e as árvores de decisão (C4.5).

3.4.1. Métodos de classificação *Bayesiana*

Os métodos de classificação *Bayesiana* consistem em métodos de predição de classes baseados no teorema de *Bayes*, que será explicado de seguida. Estudos nesta área revelam que existe um classificador, conhecido por *naive de Bayes* baseado neste teorema que é comparado às redes neuronais e árvores de decisão, tendo excelentes desempenhos em termos de eficácia e velocidade, quando aplicado em grandes quantidades de dados (Han et al., 2011, p. 350)

- **Teorema de *Bayes***

Considere-se uma instância B e uma hipótese A. Se estivermos perante um problema de classificação, aquilo que pretendemos calcular é $P(A|B)$, ou seja a probabilidade de *A* ocorrer, sabendo que ocorreu *B*. Por exemplo, se se pretender decidir se uma determinada ligação com um conjunto de pacotes *tcp* deve ser rejeitada (hipótese *A*), sabendo-se que ocorreu ao instante t_6 e utiliza o porto *https* (*B*), calcula-se a probabilidade de esta ser rejeitada sabendo-se que ocorreu ao instante t_6 e utiliza o porto *https*.

Considere-se agora *P(A)* como sendo a probabilidade de a ligação ser rejeitada, sem se saber nenhuma informação sobre esta.

Considere-se também *P(B)* como sendo a probabilidade de uma ligação ocorrer ao instante t_6 e utilizar o porto *https*.

Considere-se ainda *P(B|A),* isto é, a probabilidade da ligação ocorrer ao instante t_6 e utilizar o porto *https*, sabendo que foi rejeitada.

Pretende-se então calcular *P(A|B)* utilizando a informação anterior.

Pela regra de La Place (Hartigan, 1983, p. 1),

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \tag{3}$$

multiplicando e dividindo por P(A),

$$\frac{P(A|B)\,P(A)}{P(A)} = \frac{P(A \cap B)}{P(B)}, \tag{4}$$

isolando P(A|B),

$$P(A|B) = \frac{P(A \cap B)\, P(A)}{P(A)\, P(B)}, \tag{5}$$

Simplificando, obtém-se o teorema de *Bayes* (Hartigan, 1983, pp. 29-31):

$$P(A|B) = \frac{P(B|A)\, P(A)}{P(B)}. \tag{6}$$

Este teorema foi idealizado para tentar contornar o problema de não se poder saber *a priori* uma classe dado um dado. Assim, com esta expressão consegue-se saber *a priori* como são os dados de uma classe dada a própria classe.

- ***Naive Bayes***

Considere-se um conjunto de treino T, composto por um conjunto de instâncias B_i com dimensão n, sendo que cada uma é representada pelo vetor de atributos $B_i = (b_i^1, b_i^2, \dots, b_i^j \dots b_{in}^{jn}, C_i \dots C_{in})$, que descreve n avaliações feitas a outros tantos atributos mais uma classe C. Os atributos são denominados $X^1, X^2, \dots, X^n$.

Assim, se por exemplo considerarmos um conjunto de treino T^{ex}, composto por 5 instâncias, com 3 dimensões, em que $j = 1$ corresponde ao atributo tempo e $j = 2$ ao atributo número do porto de origem utilizado, então cada instância pode ser representada por um vetor B_i. Assumindo-se que a instância B_1 foi registada ao instante t_1, com porto de origem HTTP e foi rejeitada ($C_1 = Rejeitada$), a instância B_2 ao instante t_2 com porto HTTPS e foi aceite ($C_2 = Aceite$), a instância B_3 ao instante t_3 com porto FTP e rejeitada ($C_3 = Rejeitada$), a instância B_4 ao instante t_4 com porto SMTP, também rejeitada ($C_4 = Rejeitada$), e a instância B_5 ao instante t_5 com porto HTTPS, aceite ($C_5 = Aceite$), temos $B_1(1, HTTP, Rejeitada)$, $B_2(2, HTTPS, Aceite)$, $B_3(3, FTP, Rejeitada)$, $B_4(4, SMTP, Rejeitada)$ e $B_5(5, HTTPS, Aceite)$.

Desta forma, podemos dizer que o atributo tempo, correspondente a X^1 tem os valores $(1, 2, 3, 4, 5)$ e o atributo porto de origem, X^2 tem os valores (HTTP,HTTPS,FTP, SMTP,HTTPS). A seguinte matriz e a Figura 14 ilustram o conjunto de treino T^{ex}, onde as instâncias aceites estão representadas a verde e as rejeitadas a vermelho.

	X^1	X^2	C
B_1	1	HTTP	Rejeitada
B_2	2	HTTPS	Aceite
B_3	3	FTP	Rejeitada
B_4	4	SMTP	Rejeitada
B_5	5	HTTPS	Aceite

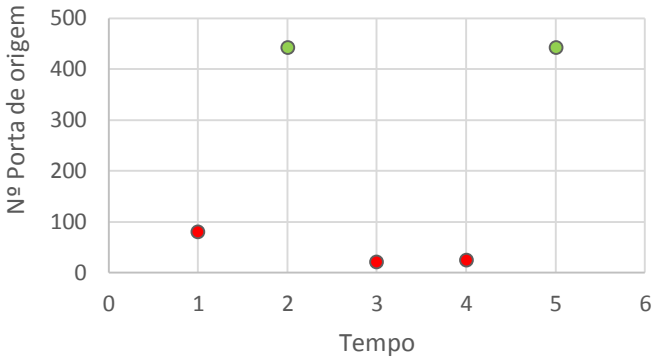

Figura 14 - Gráfico de dispersão do conjunto de treino T^{ex}

Note-se que cada ponto representa uma instância representada pelo vetor B e cada eixo um atributo representado por X.

Considere-se agora que para uma dada classificação existem m classes, $C^1, C^2, \ldots, C^m$. O classificador irá classificar B como pertencente à classe que tenha a maior probabilidade condicionada em B, ou seja, a classificação dada a B é C^i, em que C^i é tal que:

$$P(C^i|B) > P(C^j|B)\ para\ 1 \leq j \leq m, j \neq i\ . \tag{7}$$

Para tal é necessário maximizar $P(C^i|B)$. Pelo teorema de Bayes temos:

$$P(C^i|B) = \frac{P(B|C^i)\,P(C^i)}{P(B)}. \tag{8}$$

Uma vez que $P(B)$ é constante para todas as classes, apenas é necessário maximizar $P(B|C^i)\,P(C^i)$. Sempre que a probabilidade de cada classe não é conhecida, assume-se que estas são equiprováveis, ou seja, no caso considerado em que há m classes, $P(C^i) = 1/m$, onde m é o número de classes.

Assim, regressando ao exemplo mencionado, considere-se $P(C^1)$ a probabilidade da classe "Aceite" e $P(C^2)$ a probabilidade da classe "Rejeitada". Desta forma, $P(C^1) = \#C^{1,T_{ex}}/\#T^{ex} = 2/5$ e $P(C^2) = \#C^{2,T_{ex}}/\#T^{ex} = 3/5$, onde $\#C^{1,T_{ex}}$ é o número de elementos de T^{ex} que têm a classe C^1.

Caso o conjunto T tenha muitos atributos (dimensões), torna-se bastante difícil calcular $P(B|C^i)$, pois esse número pode ser muito pequeno, sendo mesmo 0 num conjunto em concreto, caso esse conjunto seja finito (Hand et al., 2001). Assim, para

reduzir a dificuldade de processamento, admite-se que os atributos são independentes entre si (Hristea, 2012, pp. 10-12), logo sendo b^k o valor do atributo X^k por instância B, tem-se:

$$P(B|C^i) = \prod_{k=1}^{n} P(b^k|C^i) = P(b^1|C^i) \times P(b^2|C^i) \times \ldots \times P(b^n|C^i)\,. \quad (9)$$

Para se calcular a probabilidade de $P(b^k|C^i)$ é necessário ter em conta se os atributos X^k são categóricos ou contínuos.

Se X^ké uma variável categórica, então $P(b^k|C^i)$ corresponde ao número de instâncias da classe C^i no conjunto de treino T, que tenham o valor b^k para o atributo X^k, dividido pelo número total de instâncias da classe C^i em T.

Se X^k é uma variável contínua, pode-se recorrer, por exemplo, a uma distribuição normal com uma média μ e desvio padrão σ, definida por:

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\,, \quad (10)$$

portanto

$$P(b^k|C^i) \cong g(b^k, \mu^{C^i}, \sigma^{C^i})\,. \quad (11)$$

Para se estimar a classe de B, calcula-se então $P(B|C^i)\,P(C^i)$ para cada classe C^i e escolhe-se a classe com maior probabilidade, ou seja onde $P(B|C^i)\,P(C^i)$ é máximo. Assim, a classe de B é i, se e somente se:

$$P(B|C^i)\,P(C^i) > P(B|C^j)\,P(C^j), \forall\, 1 \leq j \leq m, j \neq i\,. \quad (12)$$

Regressando novamente ao exemplo mencionado, imagine-se que se pretende saber se uma dada ligação registada ao segundo 6, no porto HTTPS deve ser aceite $B_6(6, HTTPS, Aceite)$ ou rejeitada $B_6(6, HTTPS, Rejeitada)$. Considere-se X^1 o atributo tempo, como variável contínua e X^2 o atributo porto de origem, como variável categórica. Assim, para $B_6(6, HTTPS, Aceite)$ no atributo X^1, temos:

$$\mu^{C^1} = \frac{(2+5)}{2} = \frac{7}{2}\,,\ \sigma^{C^1} = \sqrt{\frac{(2-3.5)^2+(5-3.5)^2}{2-1}} = \sqrt{2(\tfrac{3}{2})^2} = \sqrt{\tfrac{9}{2}} \cong 2.12\,, \quad (13)$$

$$P(6|C^1) = g\left(6, \mu^{C^1}, \sigma^{C^1}\right) = \frac{1}{\sqrt{2\pi}\sqrt{\frac{9}{2}}} e^{-\frac{(6-3.5)^2}{2\times\sqrt{\frac{9}{2}}^2}} \cong 0.094,$$

de forma semelhante, para $B_6(6, HTTPS, Rejeitada)$

$$\mu^{C^2} = \frac{(1+3+4)}{3} = \frac{8}{3}\ ,\ \sigma^{C^2} = \sqrt{\frac{\left(1-\frac{8}{3}\right)^2+\left(3-\frac{8}{3}\right)^2+\left(4-\frac{8}{3}\right)^2}{3-1}} = \sqrt{\frac{\left(-\frac{5}{3}\right)^2+\left(\frac{1}{3}\right)^2+\left(\frac{4}{3}\right)^2}{3-1}} = \sqrt{\frac{\frac{14}{3}}{2}} = \sqrt{\frac{7}{3}} \cong 1.528\,, \tag{14}$$

$$P(6|\,C^2) = g\left(6, \mu^{C^2}, \sigma^{C^2}\right) = \frac{1}{\sqrt{2\pi}\sqrt{\frac{7}{3}}} e^{-\frac{\left(6-\frac{8}{3}\right)^2}{2\times\sqrt{\frac{7}{3}}^2}} \cong 0.0241\,.$$

Para X^2, temos:

$$P(HTTPS|C^1) = 1, P(HTTPS|C^2) = 0\,. \tag{15}$$

Desta forma,

$$P(B|C^1) = P(6|C^1) \times P(HTTPS|C^1) \cong 0.094\,, \tag{16}$$
$$P(B|C^2) = P(6|C^2) \times P(HTTPS|C^2) \cong 0\,.$$

Logo,

$$P(B|C^1) \times P(C^1) = 0.094 \times \frac{2}{5} = 0.0376\,, \tag{17}$$
$$P(B|C^2) \times P(C^2) = 0\,.$$

Assim sendo, pelo *naive de Bayes*, o classificador classificaria a ligação como "Aceite", pois $P(B|C^1)\,P(C^1) > P(B|C^2)\,P(C^2)$. Por forma a se conferir este resultado, recorreu-se ao *software weka*. A Figura 15 mostra os registos obtidos. É de notar que a estimação do desvio padrão está errada, pois este *software* faz $\sigma = \sqrt{\frac{\sum_i (x_i - y_i)^2}{n}}$ em vez de $\sigma = \sqrt{\frac{\sum_i (x_i - y_i)^2}{n-1}}$. Para n grande, caso normal em data mining, obviamente que os valores

são praticamente iguais, no nosso caso, como $n = 2$ o resultado difere $(\sqrt{\frac{(2-3.5)^2+(5-3.5)^2}{2}} = 1.5, \sqrt{\frac{\left(1-\frac{8}{3}\right)^2+\left(3-\frac{8}{3}\right)^2+\left(4-\frac{8}{3}\right)^2}{3}} \cong 1.2472)$.

```
=== Classifier model (full training set) ===

Naive Bayes Classifier

                    Class
Attribute         Aceite Rejeitada
                  (0.43)     (0.57)
=====================================
Tempo
  mean                3.5     2.6667
  std. dev.           1.5     1.2472
  weight sum            2          3
  precision             1          1

Porta
  HTTP                1.0        2.0
  HTTPS               3.0        1.0
  FTP                 1.0        2.0
  SMTP                1.0        2.0
  [total]             6.0        7.0


Time taken to build model: 0 seconds

=== Predictions on test set ===

inst#,actual,predicted,error,prediction
1,1:Aceite,1:Aceite,,0.945

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds
```

Figura 15 - Resultados obtidos no software weka para o exemplo estudado

3.4.2. Eficácia dos classificadores *Bayesianos*

Em teoria estes classificadores têm a menor taxa de erro quando comparados com todos os outros classificadores (Lewis, 1998, p. 1). Contudo, na prática nem sempre é assim, devido às suposições que são feitas para possibilitar o seu uso, como por exemplo a independência de atributos. Estes classificadores são ainda bastante úteis para ajudar a entender outros classificadores, por exemplo, sob algumas suposições, pode mostrar que muitas redes neuronais originam resultados baseados nas máximas probabilidades condicionadas, tal como este (Amor, Benferhat, & Elouedi, 2004, pp. 420-421).

3.4.3. Método dos k-vizinhos

O método do vizinho mais próximo ou *k*-vizinhos tem vindo a ser amplamente utilizado na área de reconhecimento de padrões, apesar de estar associado a alguma demora na fase de teste e nenhum processamento prévio, motivo pelo qual ganharam a

alcunha de *lazzy*[40]. De entre os vários métodos de aprendizagem supervisionada que existem, este método destaca-se pelo seu alto desempenho consistente (Freitas, 2013, pp. 34-35). Contrariamente a alguns métodos, este algoritmo na fase de treino não constrói nenhum modelo de classificação ajustado aos dados, assim sendo, o k-vizinhos insere-se no grupo da aprendizagem baseada em instâncias (Lobo, 2002, pp. 80-82), em que o conjunto de treino é simplesmente armazenado e o processo de classificação passa por comparar um novo conjunto de dados (conjunto de teste) com os dados de treino devidamente classificados (Han et al., 2011, p. 423).

As instâncias de treino são descritas por n atributos e cada uma representa um ponto num espaço de dimensão n. Para um dado conjunto de teste, este classificador identifica o padrão de distância para as k instâncias de treino consideradas mais próximas da instância desconhecida (Han et al., 2011, p. 423).

O grau de semelhança ou proximidade entre os dados determina-se com recurso a uma função de distância $d(x, y)$, isto é, uma função com as seguintes propriedades (Larose, 2005, p. 99):

1. $d(x, y) \geq 0 \; e \; d(x, y) = 0 \; se \; e \; só \; se \; x = y$, (18)
2. $d(x, y) = d(y, x)$,
3. $d(x, z) \leq d(x, y) + d(y, z)$.

Algumas das distâncias mais utilizadas são as distâncias de *Minkowski* (Zezula, Amato, Dohnal, & Batko, 2006, p. 10), nomeadamente a de *Manhattan*[41] e a *Euclidiana*[42] (Hand et al., 2001, p. 25):

$$dMinkowski(x, y) = \sqrt[p]{\sum_{i=1}^{n} |x^i - y^i|^p} \; , \tag{19}$$

$$dManhattan(x, y) = \sum_{i=1}^{n} \left| x^i - y^i \right| , \tag{20}$$

[40] Preguiçoso.
[41] Distância de *Minkowski* de ordem 1.
[42] Distância de *Minkowski* de ordem 2, a distância Euclidiana torna-se menos discriminante com o aumento do número de atributos, devendo considerar-se a utilização de outra função para estes casos (Kantardzic, 2011, p. 120).

$$dEuclidiana(x,y) = \sqrt{\sum_{i=1}^{n}(x^i - y^i)^2}, \tag{21}$$

Sendo $x = x^1, x^2, \ldots, x^n$ *e* $y = y^1, y^2, \ldots, y^n$ os valores dos n atributos de dois conjuntos de dados que se pretendem comparar. Antes de se proceder ao cálculo das distâncias, é importante normalizar[43] os dados, para que os atributos tenham igual peso na classificação (Han et al., 2011, p. 113).

Após normalizados, é então necessário prever qual a classe de uma determinada instância $(x, c(x))$. Para tal o algoritmo determina os k vizinhos, pertencentes ao conjunto de treino, que distam menos desta, classificando-a segundo a expressão:

$$\hat{c}(x) \leftarrow arg\max_{v \epsilon V} \sum_{i=1}^{k} \delta(v, c(y_i)) \,, \tag{22}$$

em que $y_1, \ldots, y_k$ correspondem aos k vizinhos mais próximos de x, pertencentes ao conjunto de treino, V é o conjunto de classes e

$$\delta(x,y) = \begin{cases} 0, & x \neq y \\ 1, & x = y \end{cases}. \tag{23}$$

A escolha do parâmetro k não é um processo trivial. Normalmente é selecionado por experimentação ou partindo do conhecimento *à priori* do problema de classificação considerado. Para vários valores de k, o que tiver menor taxa de erro deve ser o escolhido. Por um lado, quanto maior o valor de k, maior será o tempo de processamento na fase de teste do algoritmo mas mais suave será a fronteira entre diferentes classes, reduzindo a influência do ruído. Contudo o tempo de teste é independente do número de classes, daqui a vantagem de utilização deste algoritmo em problemas com múltiplas classes. Por outro lado, um k mais pequeno irá preservar o comportamento local de interesse. De uma forma geral, são utilizados k na ordem das unidades e dezenas, mais do que nas centenas e milhares (Kantardzic, 2011, p. 120). Outra perspetiva a considerar na fase de seleção do k, é que este deverá ser preferencialmente ímpar, para que o classificador não tenha dúvidas em relação à classe dominante, evitando a situação de ter o mesmo número de vizinhos de duas classes distintas (Larose, 2005, p. 98).

[43] Na normalização a escala dos dados é reduzida para valores entre -1,0 e 1,0 ou 0,0 a 1,0, por exemplo.

O processo de classificação do vizinho mais próximo pode-se resumir nos seguintes passos:

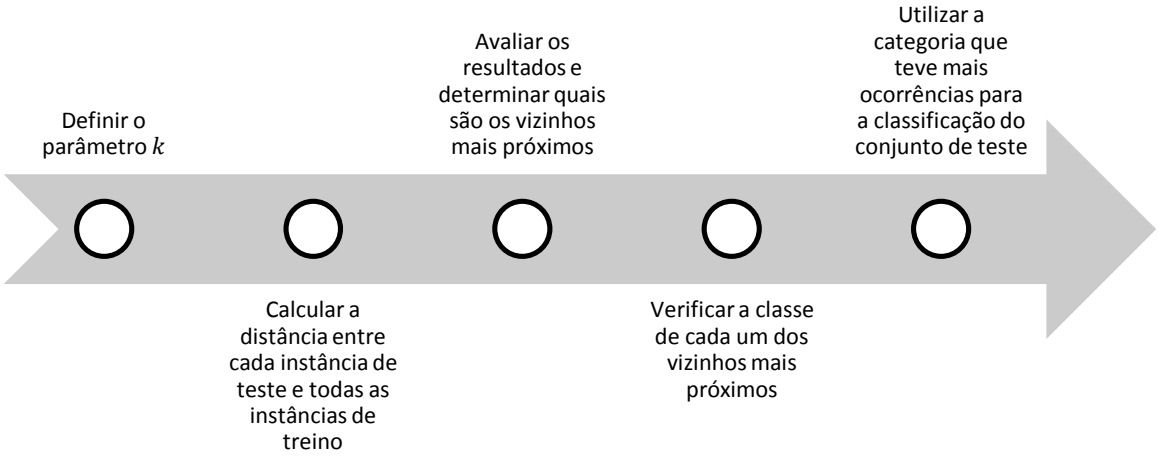

Figura 16 - Processo de classificação do método k-vizinhos

Aplicando este classificador ao exemplo já anteriormente estudado para o método do *naive* de *Bayes* considere-se que dado o conjunto de treino utilizado na elaboração da Figura 14, se pretende novamente saber se uma dada ligação registada ao segundo 6, no porto HTTPS deve ser aceite $B_6(6, HTTPS, Aceite)$ ou rejeitada $B_6(6, HTTPS, Rejeitada)$.A Figura 17 ilustra esta situação.

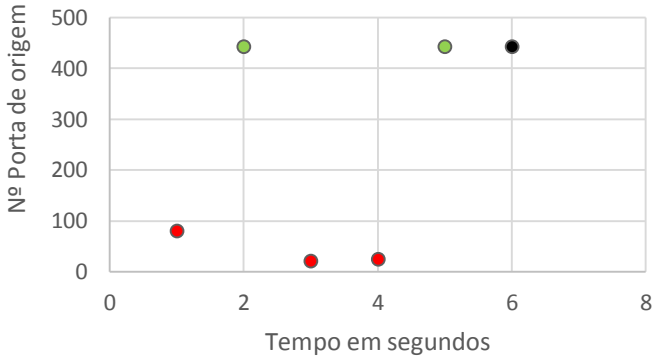

Figura 17 - Situação de estudo

Agora observe-se a classificação efetuada para $k = 1$, $k = 3$ e $k = 5$.

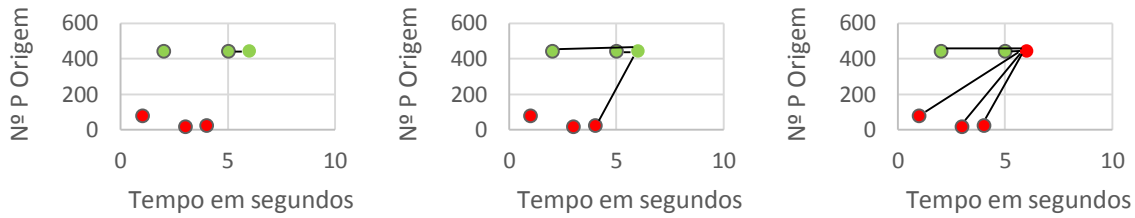

Figura 18 - Situação de estudo para k=1, k=3 e k=5

No primeiro caso, o algoritmo procura apenas o vizinho mais próximo e dá-lhe a mesma classificação que tem o seu vizinho. No segundo caso, este procura os 3 vizinhos mais próximos e observa que 2 têm a classificação "Aceite" e 1 tem a classificação "Rejeitada". Logo, pelo critério de maioria, atribui a classificação de "Aceite". No último exemplo o classificador segue o mesmo raciocínio e pelo critério de maioria atribui a classificação "Rejeitada".

Assim, para $k = 1$ e $k = 3$ classificou-se a instância B_6 como "Aceite" e para $k = 5$ como "Rejeitada". Esta situação demonstra que o método de k-vizinhos é bastante bom, porém é necessário ter atenção à escolha do k. Assim, se se pretender detetar *outliers*, o melhor é escolher-se um k baixo, se possível 1, caso se pretenda optar pelo critério da maioria, deve-se escolher um k alto.

É ainda necessário não esquecer que quanto maior o k, maior será o tempo de classificação. Esta situação observou-se experimentalmente e confirmou-se de seguida no código JAVA utilizado no *software* weka (anexo ANX 1.2). Aqui pode-se observar que o *software* weka ordena a distância a cada vizinho, por validação cruzada. De seguida seleciona apenas os k mais próximos (mantendo todos os que têm a mesma distância, caso o número de k-vizinhos seja ultrapassado). Por fim, este calcula a média das classificações para cada vizinho, em cada instância, aumentado assim o tempo de processamento, para k muito grandes.

Note-se que este exemplo serve apenas para que se consiga compreender de forma intuitiva o funcionamento do método em estudo. Para se conseguir fazer um classificador semelhante com o devido rigor científico seria necessário recorrer-se a bastantes mais dados e a métodos de normalização de dados para se poderem comparar distâncias.

3.4.4. Árvores de decisão

Uma árvore de decisão consiste num grafo com estrutura em árvore, onde cada nó representa uma opção para um dado atributo, cada ramo a opção escolhida e onde no final existe um nó que atribui uma classe às opções que foram sendo tomadas (Han et al., 2011, p. 330). A Figura 19 representa uma árvore de decisão, baseada no exemplo de estudo da Figura 14.

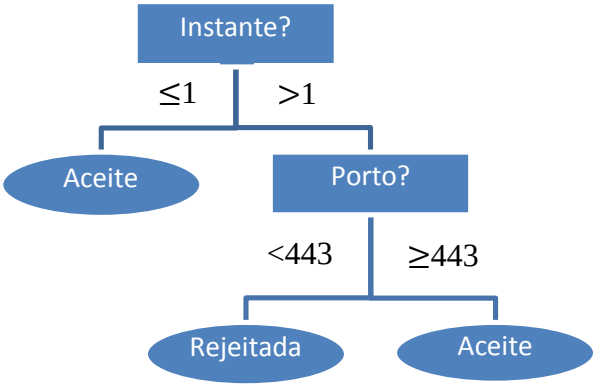

Figura 19- Exemplo de uma árvore de decisão

Nas árvores de decisão os nós internos são por vezes representados por retângulos e os nós externos de forma oval (Han et al., 2011, p. 331).

Por norma as técnicas utilizadas para criar árvores de decisão têm como objetivo reduzir ao máximo o número de nós necessários, assim para o exemplo acima mencionado, bastaria questionar qual o porto, contudo para melhor se representar a estrutura e funcionamento de uma árvore de decisão, optou-se neste exemplo por se questionar primeiro o instante a que ocorreu a ligação.

- **Utilização de árvores de decisão em problemas de classificação**

Para se obter uma dada classificação para uma instância B, basta observar os valores dos atributos e compará-los com os nós de decisão, até se chegar a uma dada classificação. As árvores de decisão podem facilmente ser convertidas para regras de classificação (Freitas, 2013, pp. 45-46).

Assim, se se pretender aplicar este método para saber se uma dada ligação registada ao segundo 6, no porto HTTPS deve ser aceite $B_6(6, HTTPS, Aceite)$ ou rejeitada $B_6(6, HTTPS, Rejeitada)$, basta seguir a árvore de decisão e rapidamente observamos que esta é classificada como "Aceite".

- **Vantagens em utilizar árvores de decisão**

As árvores de decisão são bastante fáceis de se construir. Estas suportam dados multidimensionais e a sua representação é intuitiva e fácil de assimilar por humanos. Os processos de aprendizagem e classificação são simples e rápidos. Geralmente estas têm uma boa eficácia, contudo dependendo dos dados e da sua sensibilidade a perturbações no conjunto de treino podem ter maiores ou menores taxas de erros (J. Ross Quinlan, 1986, pp. 82-83).

- **Algoritmo ID3 e C4.5**

O algoritmo ID3 é um algoritmo utilizado na criação de árvores de decisão, onde inicialmente se seleciona um atributo do conjunto de treino para se dividir o mesmo. De seguida cria-se um ramo para cada valor do atributo escolhido e movem-se as instâncias correspondentes ao mesmo para este. O algoritmo é aplicado recursivamente até todas as instâncias num nó terem apenas uma classificação.(Kantardzic, 2011, p. 171)

Na construção de árvores de decisão, o objetivo consiste geralmente em construir uma árvore de decisão com o menor número de nós possível. Para tal é necessário selecionar o atributo que tenha maior ganho de informação, ou seja, o que minimiza a informação necessária na subárvore, para classificar a instância (Freitas, 2013, pp. 47-49).

Existe ainda uma extensão do ID3, chamada algoritmo C4.5 (também conhecida por J48), que para além da classificação de atributos categóricos, permite também a classificação de atributos numéricos. Este método favorece atributos que dividam os dados em subconjuntos com uma baixa entropia, isto é, com a maioria das instâncias pertencentes apenas a uma classe. Desta forma, o algoritmo escolhe os atributos que proporcionam maior grau de descriminação entre as classes (J Ross Quinlan, 2014, p. 25).

3.4.5. Avaliação dos dados classificados

Para se poder determinar qual é definitivamente o melhor método a utilizar para cada situação, é necessário comparar os classificadores. Existem basicamente três formas simples de o fazer: através do método *holdout*, em que se utiliza uma parte dos dados como conjunto de treino e outra como conjunto de teste, o método de validação cruzada e o método de *bootstrap* (Leite, 2007, pp. 53-56).

- **Método de *holdhout***

Neste método dividem-se os dados em dois conjuntos, o conjunto de treino e o conjunto de teste. Ao fazer-se esta divisão é necessário ter em conta que a escolha da percentagem destinada a treino afetará a de teste e vice-versa, isto é, se selecionarmos uma grande percentagem para treino, o classificador à partida será mais fiável, contudo não podemos garantir que a taxa de acerto do mesmo seja muito fiável, pois foram utilizados poucas instâncias para teste. Se por outro lado optarmos por escolher um conjunto de teste demasiado grande, a taxa de acerto será fiável, contudo muito menor, pois o classificador criado utilizou poucas instâncias para treino. Assim sendo, segundo (Gupta, 2011, p. 136), este método apenas se deve utilizar quando é possível preservar 5000 instâncias para treino, com uma percentagem de conjunto de treino a variar entre os 70% e os 80%.

- **Método de validação cruzada**

Neste método dividem-se os dados em n amostras. De seguida treinam-se os classificadores com as amostras a_2, a_3 ... a_n, testam-se com a amostra a_1, é registada a taxa de erro e junta-se a amostra a_1 ao conjunto de treino. Após a amostra a_1 ser colocada novamente no conjunto de treino, retira-se a amostra a_2, testam-se os dados com esta, regista-se novamente a taxa de erro e volta-se a juntar a amostra a_2 no conjunto de treino. Este processo é repetido até todas as amostras serem testadas e no final calcula-se a média aritmética de cada taxa de erro (nenhum dos classificadores é melhor que o outro). Assim a taxa de erro deste método pode ser definida por:

$$\overline{Erro} = \frac{e_1 + e_2 + \ldots + e_n}{n} = \frac{1}{n}\sum_{i=1}^{n} e_i \,. \tag{24}$$

O método de validação cruzada utiliza-se particularmente para conjuntos de dados muito pequenos, em que não se pode utilizar o método de treino-teste, podendo no limite o número de n amostras ser igual ao número de i instâncias, sendo neste caso chamado *leave-one-out* (Freitas, 2013, pp. 86-87). Este método é particularmente bom para comparar as taxas de erro dos classificadores, pois a taxa de erro é bastante fiável, dado o número de dados utilizados para teste.

- **Método de *bootstrap* ou resubstituição**

Existe ainda um método semelhante a este, conhecido por *bootstrap*, utilizado geralmente para conjuntos de dados pequenos, onde em vez de se irem retirando amostras de dados para teste, copia-se o conjunto de treino completo e testa-se o conjunto de treino com ele próprio (Gupta, 2011, p. 137). Trivialmente se conclui que ao se testarem os dados com eles próprios se obtêm estimativas demasiado otimistas, contudo existem várias técnicas estatísticas que permitem que se obtenham estimativas de erro mais próximas da realidade (Efron & Tibshirani, 1994, p. 6).

3.4.6. Problema de *overfitting*

Apesar de não ser utilizado no âmbito desta dissertação, dada a grande quantidade de dados disponíveis, o método de *bootstrap* evidencia o problema de se testar o classificador com os mesmos dados utilizados no seu treino.

Esta situação levanta então uma questão. Será que se utilizarmos dados de teste "próximos" dos dados utilizados para treino conseguimos garantir que estes não estão correlacionados? Por exemplo, no caso de uma dada ligação com porto de origem x ter sido negada ao segundo 1, será que ela não poderá também ser negada ao segundo 5 por o pedido de ligação se repetir? Este problema é conhecido por *overfitting*, isto é, pode haver situações em que em vez de um classificador classificar as instâncias maliciosas por terem caraterísticas maliciosas, pode simplesmente classificá-las como maliciosas porque simplesmente decorou que antes esta instância, em particular, era maliciosa.

Caso isso aconteça, significa que o classificador se ajustou de tal forma aos dados, que sobre-aprendeu e portanto não é uma boa representação da realidade (Hawkins, 2004, pp. 1-4).

Para evitar que situações como esta sucedam, devem-se sempre recolher novos dados, se possível com algum desfasamento dos anteriores, e testar o classificador com esses dados, por forma a se garantir que as taxas de erro ou de acerto são efetivamente fiáveis.

3.5. Aprendizagem não supervisionada

A aprendizagem não supervisionada consiste num método de aprendizagem em que não se sabe o atributo respeitante à classe de cada instância, dizendo-se habitualmente

por isso que "é como aprender sem um professor", isto é, o sistema tem de ser capaz de aprender sozinho através da observação (Aldrich & Auret, 2013, pp. 9-10; Te Ming Huang & Kopriva, 2006, p. 175). Uma boa analogia a este método é a história de um náufrago que ao chegar sozinho a uma ilha tem de aprender várias técnicas de sobrevivência, através apenas da observação do meio.

Esta forma de aprendizagem é geralmente utilizada para análise exploratória dos dados, onde se utilizam regras de associação ou técnicas de agrupamento. As técnicas de associação visam encontrar associações nos dados (Damasceno, 2005, p. 4), por exemplo, se a ligação A e B são aceites, então a D é aceite. Por outro lado, as técnicas de agrupamento, em inglês *clustering*, procuram caraterísticas nos dados que lhes permitam fazer agrupamentos (Fisher, Pazzani, & Langley, 2014, pp. 331-334).

3.6. C*lustering*

Uma vez que aquilo que se pretende identificar são padrões/caraterísticas estranhas nas conexões, aquilo que será focado neste estudo é a parte do *clustering*.

O *clustering* pode ser definido como sendo uma operação estatística em que os dados são agrupados em pequenos grupos, conhecidos por *clusters*. Os *clusters* têm sempre duas propriedades (Tufféry, 2011, p. 235):

- Contrariamente à classificação, não lhes é atribuída uma classe *à priori;*
- Os *clusters* são sempre combinações de objetos com caraterísticas semelhantes.

Tal como na classificação, o objetivo de efetuar *clusters* é agrupar dados. Contudo, a grande diferença é que neste método o algoritmo não sabe os nomes e nalguns casos nem sequer o número de grupos que existem, motivo pelo qual se diz que este método é descritivo e não preditivo (Vazirgiannis, Halkidi, & Gunopulos, 2012, p. 20).

Para se entender facilmente o funcionamento deste método, considere-se o seguinte exemplo: imagine-se que se pretendia separar homens e mulheres sabendo apenas o seu peso e altura.

Se entregássemos essa informação a um algoritmo de *clustering* e lhe pedíssemos para a agrupar, este iria tentar encontrar padrões e em teoria conseguiria dividir os homens

das mulheres, apesar de não saber o número de classes de cada sexo, o número de elementos que contém cada sexo, nem quais são os indivíduos correspondentes a cada sexo.

Para se efetuar esta experiência reuniram-se, aleatoriamente, as informações de peso e altura de 50 celebridades, onde 25 são homens e 25 são mulheres (ver apêndice APN 5.1).

De seguida, recorreu-se mais uma vez ao *software weka*, desta vez utilizando o algoritmo de *clustering* "*Simple EM (Expectation Maximisation)*". Este método estima as médias de cada uma das k distribuições normais e de seguida procura a hipótese $h = < \mu_1, \ldots, \mu_k >$ que maximiza a verossimilhança dessas médias (Brandt, 2006, pp. 28-32; Witten & Frank, 2011, pp. 287-288). Saliente-se ainda que é utilizado o método da máxima verossimilhança por ser o que produz melhores resultados para esta separação.

Por forma a se verificarem mais facilmente os dados, foi introduzida a classe com a informação do sexo de cada um dos indíviduos. Esta classe foi totalmente ignorada para efeitos de *clustering*. Os resultados obtidos, podem ser visualizados nas figuras abaixo.

```
                Cluster
Attribute             0          1
                    (0.5)      (0.5)
==============================
Altura
  mean            167.2515 179.6304
  std. dev.         6.6603   5.7155

Peso
  mean             52.3886  78.6355
  std. dev.         4.5223  16.9529
```

```
Log likelihood: -7.39437


Class attribute: Class
Classes to Clusters:

  0  1  <-- assigned to cluster
  4 21 | H
 22  3 | M

Cluster 0 <-- M
Cluster 1 <-- H

Incorrectly clustered instances :       7.0      14      %
```

Figuras 20 e 21 - Dados obtidos no software weka para o exemplo de cluster

Nestas figuras, observa-se que as médias de peso e altura para cada *cluster* diferem bastante e que apenas foram classificados incorretamente 4 homens e 3 mulheres.

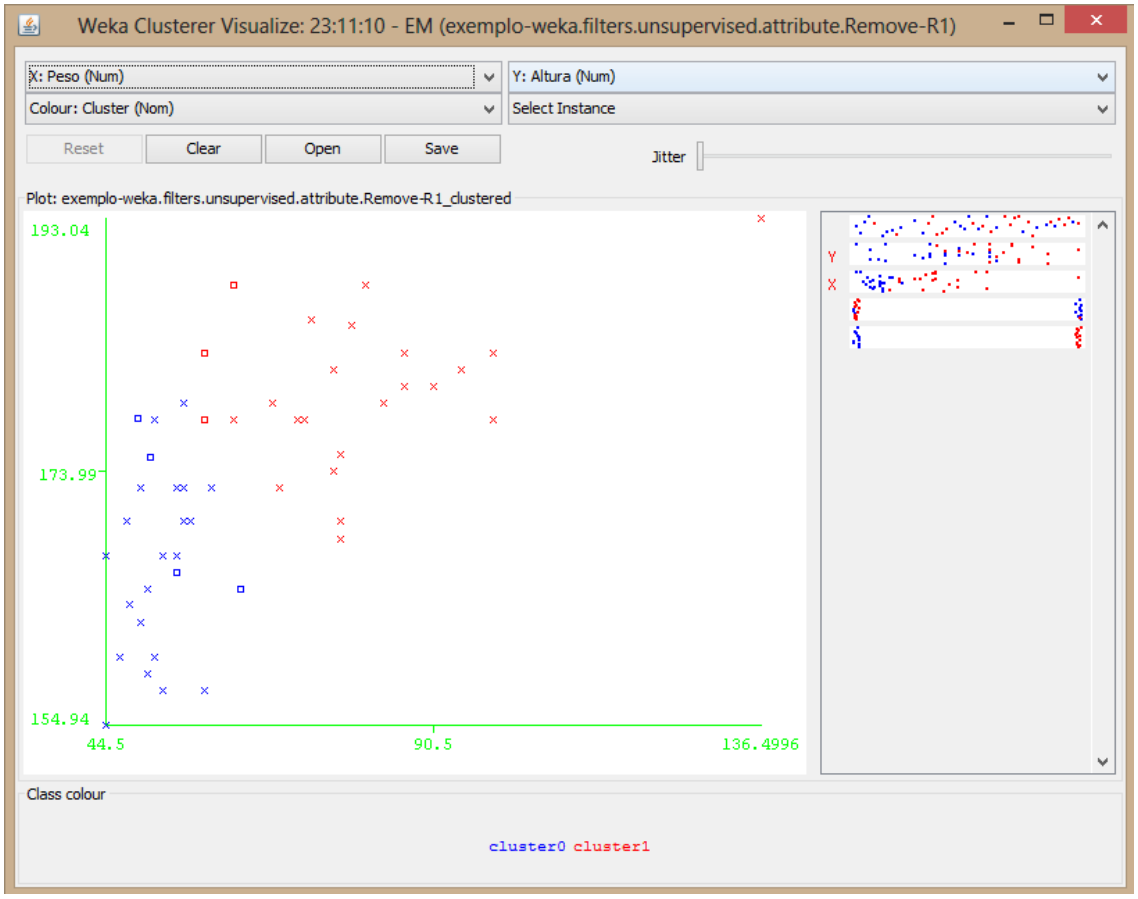

Figura 22 - Gráfico com dispersão dos erros para o exemplo de cluster

Na Figura 22 denota-se que, apesar de alguns *outliers* estarem mal agrupados (representados por quadrados), podem-se claramente visualizar 2 grupos distintos. A azul observa-se um *cluster* tendencialmente, com menor peso e altura, constituído maioritariamente por mulheres e a vermelho o *cluster* oposto.

Repare-se que esta situação serve apenas de exemplo teórico, para que se compreenda o funcionamento do *clustering*, pois para que um algoritmo de aprendizagem não supervisionada obtenha bons resultados são necessários bastantes dados, algo que não foi tido em conta neste exemplo.

3.6.1. Escolha do melhor algoritmo para um caso concreto

Existem vários algoritmos de *clustering*, cada um com as suas particularidades. Para se escolher qual o melhor algoritmo podem-se experimentar vários ou então utilizar algoritmos que já tenham demonstrado bons resultados em estudos semelhantes. No âmbito desta investigação, foi estabelecida uma parceria com o Doutor André Souto, investigador do Instituto de Telecomunicações, Instituto Superior Técnico e bolseiro da

Fundação para a Ciência e a Tecnologia, por forma a se utilizar o algoritmo *Normalized Compression Distance* (NCD), baseado na complexidade de *Kolmogorov*, que já havia demonstrado bons resultados em estudos semelhantes (Souto, 2014), quando aplicado a *logs*[44] de uma *firewall* da comunidade DARPA[45].

3.6.2. Complexidade de *Kolmogorov*

A complexidade de *Kolmogorov* $K(x)$ consiste num valor numérico associado a uma determinada sequência binária finita. O seu valor é igual ao tamanho do menor programa que devolve x (Wehner, 2005, p. 3). Por norma diz-se que um objeto é simples se existe pelo menos uma forma curta de o descrever e complexo se todas as suas descrições são longas. Assim imaginemos que se pretende descrever um dado objeto x. Existem n maneiras de descrever x e x resume todas essas n maneiras de o descrever. Assim para descrever x podemos utilizar o comprimento da maneira mais curta de descrever x, como uma medida da complexidade de x (M Li & Vitányi, 2008, pp. 1-2). Desta forma, podemos afirmar que a complexidade de *Kolmogorov* não depende da máquina utilizada, ou seja o tipo de linguagem de programação utilizado não interfere com a mesma (Wehner, 2005, p. 3) e pode ser aplicada a qualquer máquina de *Turing* (Cilibrasi, Vitányi, & De Wolf, 2004, p. 51).

- **Necessidade de se comprimir x**

Considere-se a versão comprimida de uma dada sequência x como sendo um programa que descreve x ao ser executado num dado descompressor. A complexidade de *Kolmogorov* nunca será maior do que o comprimento da sequência comprimida. Assim, sendo este comprimento é um majorante de $K(x)$, comprime-se x e depois utiliza-se o valor do comprimento de $x_{comprimido}$, como sendo o valor de $K(x)$.

3.6.3. Distâncias usadas para definir agrupamentos

Quando se comparam dois ficheiros é necessário ter uma métrica ou função de distância para determinar se eles são parecidos ou não.

Por forma a se poderem comparar semelhanças entre ficheiros, sem se conhecer nada sobre o significado do conteúdo desses ficheiros ou a área a que estes se aplicam,

[44] Resumo de conexões estabelecidas, após serem processadas pela *firewall* e restantes sistemas.
[45] *Defense Advanced Research Projects Agency* (DARPA)

encontrou-se uma distância universal conhecida por *Normalized Information Distance* (NID) (Ming Li, Chen, Li, Ma, & Vitányi, 2004, p. 1). Segundo os seus autores, para cada par de objetos a, b, esta distância é sempre menor do que a sua distância efetiva (assumindo-se que se conhecia o significado do conteúdo). Para que esta distância pudesse ser utilizada em aplicações reais, os seus autores desenvolveram ainda uma versão que pudesse ser utilizada em compressores reais, conhecida por *Normalized Compression Distance* (NCD) (Ming Li et al., 2004, p. 1).

A NCD de duas sequências x e y é dada por (Wehner, 2005, p. 3):

$$NCD(x,y) = \frac{K(xy) - min\{K(x), K(y)\}}{max\{K(x), K(y)\}}, \tag{25}$$

onde xy representa a junção (concatenação) de x com y e K a complexidade de *Kolmogorov*, estimada pelo método acima descrito.

Assim sendo, transpondo para o estudo em causa, podemos dizer que se duas conexões forem parecidas, isto é, se tiverem valores semelhantes para os diferentes campos analisados (tempo, IP, porto, seq, ack, …), então elas são também metricamente semelhantes, motivo pelo qual é possível agrupá-las de acordo com a sua distância.

3.6.4. Utilização da NCD para aplicações práticas

Uma vez que não é possível computar a complexidade de *Kolmogorov*, também não é possível computar a <aNCD (Souto, 2014, p. 4), logo para que esta seja aplicada em situações práticas, é necessário recorrer a aproximações através de programas de compressão como o *gzip*, *bzip2* e o PPMZ (Cilibrasi et al., 2004, p. 53).

Ao se compararem todos os ficheiros entre si (dois a dois) as distâncias entre todos os pares são armazenadas numa matriz M. Para que se consiga compreender a informação registada na mesma podem-se definir agrupamentos baseados nessas distâncias, ficando-se assim com o conjunto de *clusters* (Cilibrasi et al., 2004, p. 53).

3.6.5. Construção de *clusters*

Devido à sua simplicidade conceptual, o *hierarchical clustering* é considerado um dos melhores métodos não supervisionados para efetuar *clustering* (Duda, Hart, & Stork,

2001, p. 38, Cap. 10) e a forma mais natural de os representar é num dendrograma[46] (Cilibrasi et al., 2004, p. 54).

- ***Hierachical clustering***

O *hierarchical clustering* é um método de *clustering* que produz representações hierárquicas. Considere-se uma dada sequência com n amostras e c *clusters*. Divida-se agora a sequência em n *clusters*, onde cada *cluster* contém uma e uma só amostra. De seguida reparta-se em $n-1, n-2, \ldots, 1$ *clusters*, onde neste último caso, todas as amostras formam um só cluster. Diz-se que estamos num dado nível k quando $c = n - k + 1$. Quaisquer duas amostras x e x' serão a certa algura agrupadas no mesmo *cluster*. Caso a sequência de *clusters* implique que sempre que duas amostras estejam no mesmo *cluster*, ao mesmo nível k, permaneçam juntas nos níveis acima, então pode-se dizer que se está perante uma sequência de *hierarchical clustering* (Duda et al., 2001, p. 37, Cap. 10).

- **Representação de *clusters***

No sistema *CompLearn*, que se irá utilizar na presente investigação, os *clusters* são representados a partir do método do quarteto. Assim, considere-se um dado grupo de 4 elementos $\{a, b, c, d\}$ e para cada grupo, construa-se uma árvore onde cada nó interno tenha 3 vizinhos. A Figura 23 ilustra todas as topologias possíveis (Souto, 2014, p. 5), onde $\{n_0, n_1\}$ corresponde a cada nó interno e $\{a, b, c, d\}$ a cada nó externo.

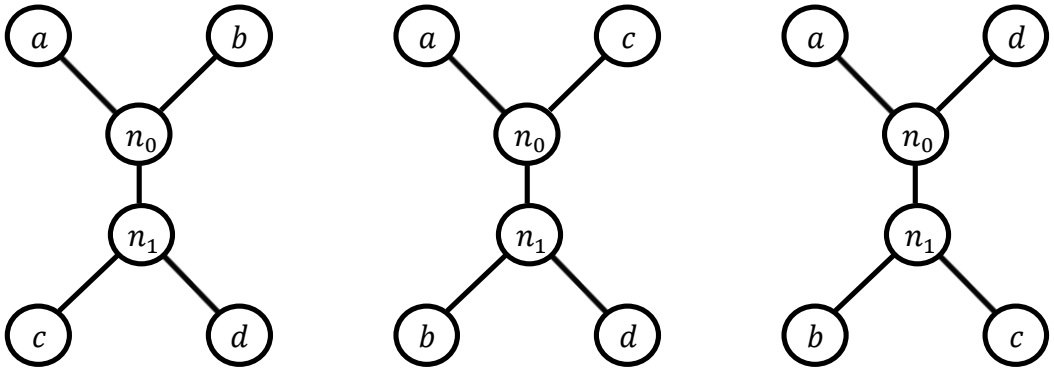

Figura 23 - Topologias possíveis para um grupo de 4 elementos {a,b,c,d}

[46] Tipo de diagrama em árvore binária (ver Figura 23).

Pode-se definir o custo de cada quarteto como sendo a soma das distâncias entre pares vizinhos (Cilibrasi et al., 2004, p. 54; Souto, 2014, p. 5), ou seja:

$$C_{ab|cd} = d(a,b) + d(c,d)\,, \tag{26}$$

$$C_{ac|bd} = d(a,c) + d(b,d)\,,$$

$$C_{ad|bc} = d(a,d) + d(b,c)\,.$$

De todas as árvores possíveis de contruir, a melhor é sempre aquela que tem o menor custo. Uma vez que é extremamente difícil encontrar a árvore perfeita, o *CompLearn*[47] recorre a métodos de *hill-climbing* e de escolha aleatória para a procurar (Souto, 2014, p. 5).

[47] Algoritmo de compressão utilizado no âmbito desta investigação.

Capítulo 4 - Dados utilizados

4.1. Obtenção das bases de dados

Para que fosse possível realizar esta dissertação, foram recolhidas, com sucesso, em períodos diferentes, um total de cinco Bases de Dados (BD). Destas cinco, três são respeitantes a capturas na *firewall* da rede de marinha (BD_{CAP}) e as outras duas correspondem a *logs*[48] com as classificações atribuídas pela *firewall* de perímetro *Checkpoint NG 77.20* (BD_{LOG}). A Tabela 9 enumera algumas caraterísticas das BD.

BD	Data	Hora início	Hora fim	Nº ligações	Tamanho
$BD1_{CAP}$	22/04/2014	05:38:46.416175	07:08:43.159648	44414971	4.2 GB
$BD2_{CAP}$	20/03/2015	10:12:27.045129	11:03:42.278141	45209318	31.1 GB
$BD2_{LOG}$	20/03/2015	00:00:00	11:59:59	198763	48.7 MB
$BD3_{CAP}$	30/06/2015	08:13:02.681945	08:28:48.398029	13403565	10.1GB
$BD3_{LOG}$	30/06/2015	08:00:00	10:00:00	37835	8.8 MB

Tabela 9 - Caraterísticas das BD

As BD foram obtidas através duma cooperação entre a Escola Naval e a Direção de Tecnologias de Informação e Comunicações (DITIC), por intermédio do 1TEN STP Baptista das Neves e do TEC. INF. Rodrigues Gomes.

4.2. Conteúdo das bases de dados

Quando se efetuam capturas, por norma os registos são guardados em formato PCAP. Assim, as BD_{CAP} encontravam-se todas neste formato, que é um formato binário. Portanto ao ser aberto sem um *software* apropriado é impercetível. Numa situação corrente, poder-se-ia utilizar o *software wireshark ou tshark,* para abrir este formato. Contudo dada a dimensão e tamanho das BD_{CAP}, tal não foi possível, pois o tamanho máximo de captura que estes *softwares* suportam é 2GB ("tshark - The Wireshark Network Analyzer 1.12.2," 2015).

[48] Registos de eventos.

4.2.1. Conteúdo da BD_{CAP}

Por forma a se preservar a totalidade dos dados agregados optou-se por convertê-los para um formato ASCII, nomeadamente para TXT. Para o fazer utilizou-se a ferramenta tcpdump (ver apêndice APN 1), disponível na maioria dos sistemas operativos Linux. A Figura 24 mostra uma linha desse ficheiro após ser convertida.

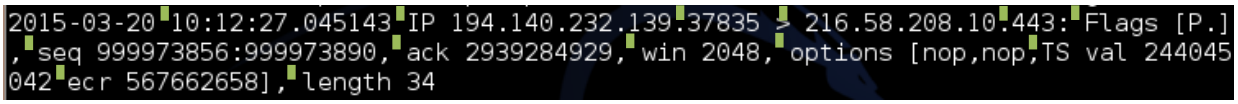

Figura 24 - Quinta linha da $BD2_{CAP}$ em formato TXT

Cada linha é composta por uma série de campos, descritos a seguir:

Campo 1 – Data (neste caso 2015-03-20), diz respeito, à data da ligação;

Campo 2 – Tempo a que a ligação foi registada na *firewall* (para esta ligação 10:12:27.045143). Neste campo o primeiro valor corresponde à hora, o segundo aos minutos, o terceiro aos segundos e o quarto aos nano segundos;

Campo 3 – IP de origem. O IP funciona como uma assinatura numérica para cada dispositivo numa rede de computadores e neste caso identifica o remetente da mensagem (194.140.232.139). Note-se que este valor não representa diretamente o computador que fez o pedido, pois conforme já foi referido no capítulo de enquadramento, a mensagem vai passando por vários intermediários até chegar ao destino;

Campo 4 – Porto de origem. Este número regista o porto que o remetente utilizou para se conectar ao destinatário (nesta ligação "37835");

Campo 5 – IP de destino. Este campo regista o IP do destinatário (neste caso 216.58.208.10);

Campo 6 – Porto de destino. Porto que o destinatário está a utilizar para esta conexão (nesta situação 443);

Campo 7 – *Flags*. A informação neste campo é representada de forma abreviada (nesta conexão "."). A Tabela 10 contém o significado de cada abreviatura;

.	S	F	R	P	E	W
ACK	SYN	FIN	RESET	PUSH	ECE	CWR

Tabela 10 - Flags

Os *ACK* servem para informar/verificar que o recetor recebeu a informação, os SYN para efetuar uma sincronização (informar que se vai iniciar uma conexão) e o FIN para informar que terminou a conexão. O RESET informa que houve um erro e que se deve terminar a sessão sem aguardar resposta. Tal como o nome indica PUSH significa envio de dados. A flag ECE (*Explicit Congestion Notification* (ECN) - *Echo*), definida pelo RFC 3168, indica que o par TCP tem capacidade ECN durante o procedimento *three-way handshake*. Por fim o *Congestion Window Reduced* (CWR) indica que foi recebido um pacote com a *flag* ECE (Sophos, 2015). Existem situações em que aparecem duas *flags* para a mesma instância, como no caso desta linha, em que surge "P." que significa que foi enviado um pacote e um *acknowledge*.

Campo 8 – Número de sequência. No caso de serem enviados dados, é também enviado o número de sequência. Quando se inicia uma sessão TCP, o primeiro número de sequência pode ser gerado de forma aleatória entre 0 e 4294967295. Contudo com o decorrer do procedimento do *three-way handshake* este passa a ser relevante para o estabelecimento da sessão, não podendo alterar-se de forma aleatória. Assim, nesta ligação pode-se observar que se está a efetuar uma transferência de pacotes, estando registada a sequência "999973856:999973890". Neste caso o primeiro valor representa o número final da sequência, transmitida no último pacote enviado, e o segundo valor é o resultado da soma desse valor com o tamanho do pacote a ser transmitido nesta instância. A diferença entre estes fica registada no último campo, que tem o tamanho dos pacotes transmitidos;

Campo 9 – Número do pacote ao qual se está a fazer *acknowledge* (neste caso 2939284929). Este valor serve para o emissor ter a certeza de que o recetor recebeu toda a informação e de forma correta;

Campo 10 – Tamanho da janela (para esta ligação 2048). Este campo é utilizado para registar o espaço de *buffer* que o remetente tem para receber dados. Segundo o RFC1072 o limite máximo para este campo é 65535 *bytes* (Jacobson & Braden, 1988),

contudo o RFC1323 e mais recentemente o RFC7323 vieram permitir que esse valor pudesse ser aumentado até cerca de 1GB, caso seja registada a opção "*window scale*" durante o procedimento de *three-way handshake*, por forma a garantir que tanto o emissor como o recetor se encontram preparados para tal (Borman, Scheffenegger, & Jacobson, 2014; Jacobson, Braden, & Borman, 1992);

Campo 11 – Campo das opções. Nesta ligação encontram-se registados dois "nop[49]". Estes servem simplesmente para completar os *bytes* necessários, para que este campo constitua um múltiplo de 4 (cada *nop* corresponde a um *byte*);

Após os *nops* surge o *Timestamp Value* e o *Timestamp Echo Reply*, definidos pela primeira vez no RFC1323. O primeiro corresponde ao tempo a que o pacote foi enviado e o segundo corresponde ao tempo a que foi enviado o último pacote recebido. Assim sempre que é iniciada uma sessão o remetente envia o *Timestamp Value* e coloca o *Timestamp Echo Reply* a 0. De seguida, quando o destinatário responder, este regista o *Timestamp Echo Reply* igual ao *Timestamp Value* recebido e envia o novo *Timestamp Value*. Estes valores servem para identificar possíveis atrasos, que requeiram ajustes de tempo na sincronização de *acknowledges* (Jacobson et al., 1992).

Campo 12 – Tamanho dos pacotes enviados nessa transmissão.

4.2.2. Conteúdo da BD_{LOG}

Para efeitos desta investigação as principais diferenças entre as BD_{LOG} e as BD_{CAP} a considerar são o número de ligações e a classificação de cada ligação. Tendo em conta que as BD_{LOG} são apenas um resumo dos eventos ocorridos, estas apresentam muito menos registos do que as BD_{CAP} para um igual período. Por outro lado, as BD_{LOG} têm um campo extra face às BD_{CAP}, que é ação realizada em cada evento. Uma vez que as BD_{LOG} foram disponibilizadas já em formato TXT, não houve necessidade de serem convertidas para outro formato. A Figura 25 mostra o cabeçalho e a primeira ligação da $BD2_{LOG}$.

[49] No Operation.

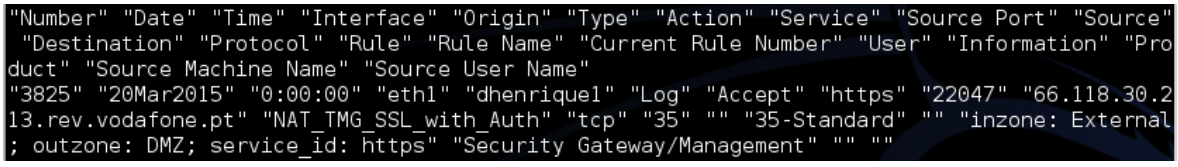

Figura 25 – Cabecalho e primeira linha da $BD2_{LOG}$ em formato TXT

Uma vez que a maioria dos campos são semelhantes aos da BD_{CAP} e que para efeitos deste estudo apenas iremos necessitar do campo "Action" destas BD, ir-se-á apenas explicitar o mesmo.

Conforme já referido, o campo "Action" regista as ações realizadas pela *firewall* de perímetro em uso na marinha portuguesa (ver Tabela 7) para cada evento. Assim sendo, neste campo podem-se observar as seguintes opções {Accept, Drop, Reject, Decrypt}. Tal como o nome indica, "Accept" significa que a ligação foi aceite. "Drop" significa que a ligação foi ignorada, isto é não foi permitida e não foi dada nenhuma informação ao remetente acerca disso. Por outro lado "Reject" indica que a ligação não foi permitida, mas foi enviada essa informação ao remetente. Por fim "Decrypt" indica que se trata de uma ligação através de VPN.

4.3. Programas utilizados para processamento

Tendo em conta o tamanho e número de ligações das bases de dados, não foi possível pré-processar os dados com os habituais processadores de texto utilizados no sistema operativo *windows*, como o *wordpad*, *notepad* ou *excel*, que requerem que os dados sejam carregados para a *ram* para serem processados (Duffy & Cram, 2007, p. 30, Cap. Windows). Assim surgiu a necessidade de se optar por programas que fossem capazes de processar documentos linha por linha, isto é, sem serem primeiro carregados para a *ram*. Desta forma optou-se por se recorrer ao sistema operativo Kali Linux, versão 3.14.5. Este sistema inclui várias ferramentas (*e.g.* awk, sed, *etc*) que, através de linhas de comandos, possibilitam que se criem códigos capazes de processar dados linha por linha.

4.4. Pré-processamento dos dados

Por forma a se poderem utilizar os dados, foi necessário eliminar-se informação desnecessária, ajustar-se os campos (devido a alguns erros, nem sempre o número de campos nas BD é o mesmo), e associar as ações das BD_{LOG} às BD_{CAP}. Foi assim produzida

uma nova BD, a BD_{CLASS} , onde se converteram todos os dados para numéricos (com exceção das *flags*) e dispuseram os dados em formato CSV (separado por vírgulas). A Figura 26 mostra as primeiras 5 linhas após estas terem sido processadas.

```
36747.045129,3264014475,44610,3627732994,80,.,0,0,3447915500,169,244045042,567660764,0,NOACTION
36747.045133,3264014475,44610,3627732994,80,.,0,0,529,170,244045042,567660764,0,NOACTION
36747.045137,3264014475,44610,3627732994,80,.,0,0,1310,179,244045042,567660764,0,NOACTION
36747.045140,1042029179,80,170786933,3260,.,0,0,1097479383,5530,916560072,860543292,0,Drop
36747.045143,3264014475,37835,3627733002,443,P.,999973856,999973890,2939284929,2048,244045042,567662658,34,NOACTION
```

Figura 26 - Primeiras 5 linhas da $BD2_{CLASS}$ em formato TXT

A sequência de ações efetuadas foi programada em *shell script*, por forma a poder ser efetuada automaticamente sempre que se introduzam novos dados e poderá ser visualizada em apêndice, assim como todos os respetivos *scripts*.

É ainda importante salientar que para se efetuarem as associações das BD_{LOG} às BD_{CAP}, se optou por fazer a associação em função do porto de origem, tendo-se definido o intervalo de procura como sendo 1 segundo. Esta escolha, no entanto tem um problema semelhante à escolha do k para o métodos dos k-vizinhos, se se escolher um tempo de intervalo muito grande, corre-se o risco de o *software* encontrar ligações que não se pretendem e se se escolher um intervalo muito pequeno, corre-se o risco de se obterem poucas instâncias classificadas. Assim realizaram-se vários testes, para diferentes intervalos. A Tabela 11 mostra o número de instâncias classificadas para cada ação, utilizando agrupamentos com diferentes durações, na $BD3_{CLASS}$.

	1 seg	2 seg	10 seg	86400 seg (24H)
Accept	3552394	4348056	5621641	5836384
Drop	176861	257920	361643	216154
NOACTION	9674313	8797592	7420284	7351030

Tabela 11 - Resultados obtidos para diferentes intervalos de procura

Conforme se pode observar, quanto maior o intervalo, menor o número de NOACTION's. Contudo, também menor é o rigor da classificação de cada ação e por esse motivo, optou-se por se escolher apenas um segundo de intervalo.

Após serem processados, decidiu-se elaborar caras de Chernoff, correspondentes a cada ação existente, por forma a mais facilmente se visualizar se de fato existem diferenças entre elas. Para evitar que fosse escolhida uma ligação atípica ou mal

classificada, optou-se por se elaborar uma matriz com a média para cada campo de todas as ligações existentes na $BD2_{CLASS}$. Estas ilustrações foram elaboradas recorrendo ao *software matlab* (ver apêndice APN 4), onde se deve considerar a média dos segundos como sendo o tamanho da cara, dos IPs de origem o comprimento do arco testa/queixo, do porto de origem o formato da testa, do IP de destino a forma da boca, do porto de destino a largura entre os olhos, as *flags* não foram consideradas por não terem formato numérico e porque em todas as ações o valor dominante era o ".", a média da sequência inicial a posição vertical dos olhos, da sequência final a altura dos olhos, do *acknowledge* a largura dos olhos, do tamanho da janela o ângulo dos olhos, do *Timestamp Value* a posição vertical das sobrancelhas, do *Timestamp Echo Reply* a largura das sobrancelhas e do tamanho do pacote o ângulo das sobrancelhas. A Figura 27 revela os resultados obtidos.

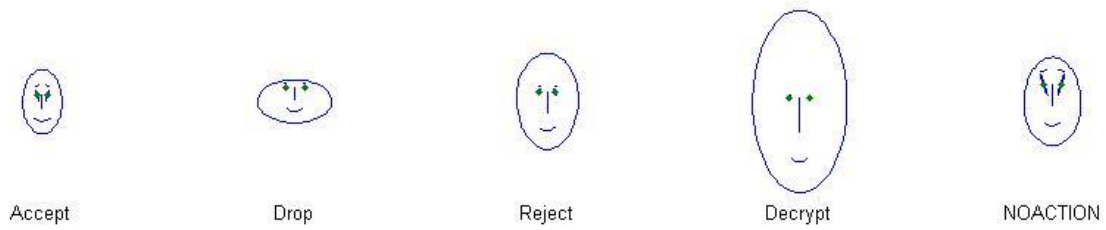

Figura 27 - Caras de Chernoff para cada ação existente na $BD2_{CLASS}$

Conforme se observa, não existe apenas um único critério que distinga ambas as ligações e mesmo com as caras de *Chernoff* não é fácil distinguir todas. No caso da ação "Reject" e "Decrypt" por exemplo torna-se difícil a comparação se se ignorar o tamanho da cara, que corresponde à média dos segundos em que surgiram estas ligações, fator esse que se sabe *a priori* não ser decisivo. Contudo denotam-se bem as diferenças entre as ações "*Accept*" e as "*Drop*", o que é bastante positivo. A ação "NOACTION", resultante da associação, parece ser um intermédio entre as ações "*Accept*" e as "*Drop*".

Capítulo 5 - Testes efetuados

Conforme anteriormente mencionado, é bastante importante que se consiga obter uma forma automática de detetar ameaças informáticas e o *data mining* pode ainda ter a vantagem de detetar ameaças não conhecidas. Tal pode ser conseguido através de dois métodos distintos: a aprendizagem supervisionada e a não supervisionada.

Ao longo desta investigação foram realizados vários testes, recorrendo a abordagens supervisionadas e não supervisionadas, tendo em conta os seguintes objetivos:

1. Obter um classificador capaz de classificar eventos com uma boa taxa de acerto;
2. Agrupar dados com ligações semelhantes, por forma a se poder saber se as fronteiras naturais correspondem às fronteiras pré-definidas.

5.1. Vantagens de se recorrer a ambos os métodos de aprendizagem

Por forma a se compreender melhor esta questão, considere-se o seguinte exemplo, imagine-se que se pretendia distinguir a população da América do Norte em função dos seus rendimentos mensais em dólares. E que após um inquérito se obtinham as seguintes informações:

Ind.	1	2	3	4	5	6	7	8	9
Rend.	1650	400	300	4000	1850	1900	1700	1800	1700
País	MEX	MEX	MEX	EUA	EUA	EUA	CAN	CAN	CAN

Se utilizássemos um método de classificação supervisionada, como os *k-vizinhos*, provavelmente o classificador conseguiria classificar bem quem era mexicano (a vermelho), americano (a azul) ou canadiano (a amarelo), bastava para tal observar quem eram os seus *k* vizinhos mais próximos.

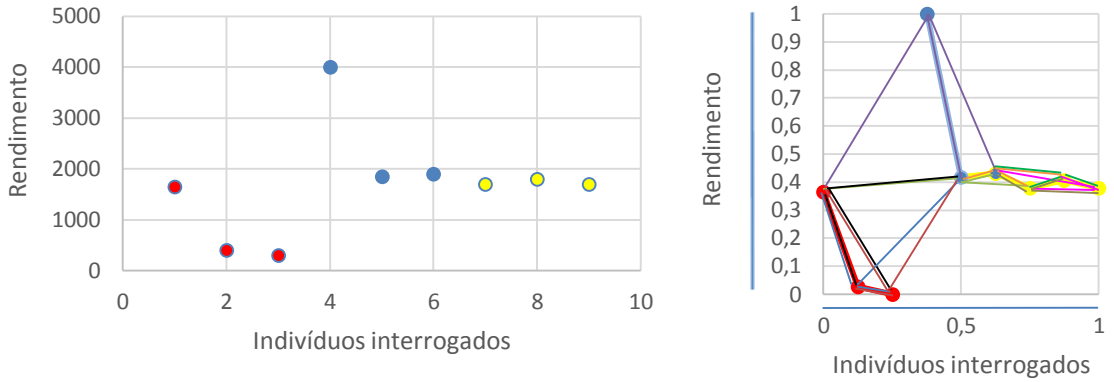

Figura 28 - Aprendizagem supervisionada vs não supervisionada – dados recolhidos e dados normalizados

Opte-se então por $k = 3$, basta observar-se o gráfico mais à direita (em que os eixos têm o mesmo comprimento e em que os dados foram normalizados pelo método do *min-max*[50]) para se perceber facilmente que se se fizesse um teste de validação cruzada com 9 partes, o classificador conseguiria classificar bem praticamente todos os elementos (apenas o 3º americano seria classificado como canadiano).

Por outro lado se pedíssemos a um algoritmo de *clustering* para agrupar os elementos, este provavelmente também criava três *clusters*, um com os indivíduos {1,5,6,7,8,9}, pois as suas caraterísticas socioeconómicas não diferem muito, outro com os indivíduos {2,3} e ainda outro apenas com o indivíduo {4}. Assim, podemos dizer que relativamente a esta caraterística, existem dois tipos de fronteiras diferentes, a pré-definida, que é imposta e diz qual o país de cada elemento e a natural que é observada em função dos rendimentos.

Em que é que estas informações nos podem ser úteis? Imagine-se agora que se comparam ambos os estudos. Desta forma é possível observar que à primeira vista se sobressaem dois *outliers*, o indivíduo 1 e o 4. Nesse caso, a articulação dos dois métodos tinha sido útil, pois o método de classificação apesar de útil e rapidamente capaz de identificar acertadamente quem era de que país, não ia ser capaz de observar "sozinho" estes dois casos estranhos. Assim, pode-se afirmar que o uso articulado destes dois métodos nos permite criar sinergias que de outra forma nunca poderiam ser obtidas, isto é, podemos utilizar um método para classificar os dados como pretendemos e outro para verificar se faz sentido os dados serem agrupados com classificações diferentes ou até se

[50] Para $y' \in [0,1]$ $y' = \left(\frac{y-min}{max-min}\right)$ (Kantardzic, 2011, p. 34).

esses dados correspondem mesmo à classificação obtida. Na realidade desta investigação esta articulação é bastante importante, pois só assim é possível identificar novas ameaças.

5.2. Aprendizagem supervisionada

Para se tentar obter um classificador capaz de distinguir os dados com uma boa taxa de acerto optou-se por utilizar métodos de aprendizagem supervisionada e portanto tiveram de se introduzir dados já classificados.

5.2.1. *Softwares* utilizados

No âmbito da aprendizagem supervisionada recorreu-se sobretudo ao *software weka*, que é uma ferramenta de prospeção de dados bastante utilizada para estudos académicos e aplicações empresariais. O *weka* é um *software open source* criado pela Universidade de Waikato, na Nova Zelândia e utiliza linguagem *Java* (Damasceno, 2005, p. 1; Hall et al., 2009, p. 10).

5.2.2. Classificação dos dados

Para se classificarem os dados efetuou-se uma associação à classificação atribuída pela *firewall* de perímetro da marinha portuguesa, isto é, associaram-se os dados da $\boldsymbol{BD_{CAP}}$ aos da $\boldsymbol{BD}_{LOG}$, dando-se origem à $\boldsymbol{BD}_{CLASS}$. Qual o problema de se fazer isto? Corre-se o risco de aquilo que é classificado como "Accept" poder ser na verdade uma ameaça que deveria ter sido rejeitada e vice-versa. Isto é, ao se introduzirem as classificações já atribuídas, está-se a assumir que estas estão corretas, podendo na verdade não estar. Em boa verdade, o classificador aqui obtido ficará sempre limitado aos métodos de classificação que a marinha já usa. Assim sendo, poderia ser questionado se valeria a pena estar-se a ter tanto trabalho para na realidade se "copiar" algo que já existe.

Este trabalho é útil e necessário para que se consiga evoluir, por duas razões. Em primeiro lugar porque se se pretendesse construir outro classificador, poder-se-ia utilizar exatamente o mesmo método, com a diferença de que em vez de se recorrem a classificações atribuídas por *softwares* de segurança, teria de se recorrer a uma *deep analisys*, isto é, teriam de ser pessoas especialistas nesta área a fazer a classificação de cada evento ou a verificar se estes eventos se encontram na realidade bem classificados. Tal trabalho é moroso e exige bastantes recursos a nível pessoal e financeiro, motivo pelo qual é geralmente feito por grandes organizações e está obviamente fora do âmbito desta

dissertação. Em segundo lugar porque mesmo que não se esteja a pensar criar um novo classificador, é sempre importante ter uma ideia de como este funciona e das alternativas que podem existir, quer seja para ser modificado ou melhorado.

Uma outra hipótese ainda seria fazer um *trace back*, isto é, poder-se-ia observar os dados recolhidos, selecionar um determinado evento que se soubesse ser malicioso e seguir-se-lhe o rasto, identificando-se essas ações como maliciosas. Qual o problema deste método? Desta forma o classificador apenas ia classificar como maliciosos o pequeno número de eventos semelhantes aos classificados como tal, apanhando-se assim apenas uma pequena parte dos ataques.

Para evitar tudo isto, conforme já referido, optou-se então por utilizar as análises já efetuadas como *ground truth*, devendo-se portanto ter em conta que uma vez que o "professor" é um sistema de classificação já existente, nunca se poderá pedir ao classificador que seja melhor que esse mesmo sistema.

5.2.3. Dados utilizados

Para se efetuarem os primeiros 4 testes retirou-se uma amostra das primeiras 30000 instâncias da $BD2_{CLASS}$, com a maioria dos dados convertidos para formato numérico, para ser utilizada como ficheiro de treino do classificador (ver apêndice APN 5.2).

A Figura 29 ilustra a distribuição dos dados por campo, onde se deve considerar o eixo das abcissas como sendo o valor de cada campo e o das ordenadas o número de vezes que esse valor ocorre. No último campo "Action" pode-se observar que 7540 estavam classificados como Accept (a azul), 15677 como Drop (a vermelho), 22 como Reject, 0 como Decrypt e 6761 como NOACTION (a beje).

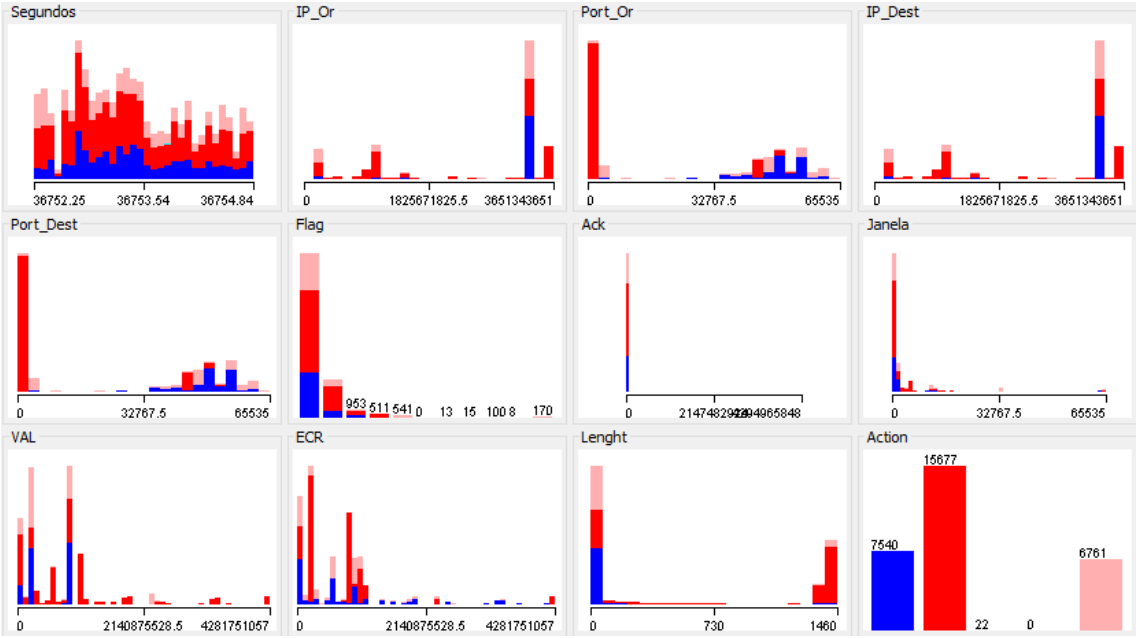

Figura 29 - Distribuição dos campos nas primeiras 30000 instâncias da $BD2_{CLASS}$

5.2.4. Métodos de classificação supervisionada utilizados

Conforme já havia sido mencionado no capítulo de *data mining*, para a realização deste estudo foram selecionados três métodos distintos: o naive de Bayes, o k-vizinhos (IBK) e as árvores de decisão (C4.5/J48).

5.2.5. Teste 1 – 20% para teste

No primeiro teste utilizou-se 80% (24000 instâncias) dos dados para treinar o classificador e 20% (6000 instâncias) para teste.

- ***Naive de Bayes***

Começou-se por se testar com o classificador *naive de Bayes*, conhecido por ser um classificador rápido, obtendo-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 4349	✓ 72.4833%
× Instâncias mal classificadas	× 1651	× 27.5167%

Tabela 12 - Teste 1, instâncias bem e mal classificadas para o método de naive de Bayes

Accept	Drop	Reject	Decrypt	NOACTION	Previsto / Real	Taxa de acerto por classe $(\frac{verdadeiros\ positivos}{total\ classe\ real})$
1349	14	0	0	172	Accept	87.9%
117	2583	0	0	428	Drop	82.6%
0	0	5	0	0	Reject	100%
0	0	0	0	0	Decrypt	-
786	134	0	0	412	NOACTION	30.9%

Tabela 13 – Teste 1, matriz de dispersão e taxa de acerto por classe para o método de naive de Bayes

Figura 30 – Teste 1, gráfico com dispersão dos erros para o método de naive de Bayes

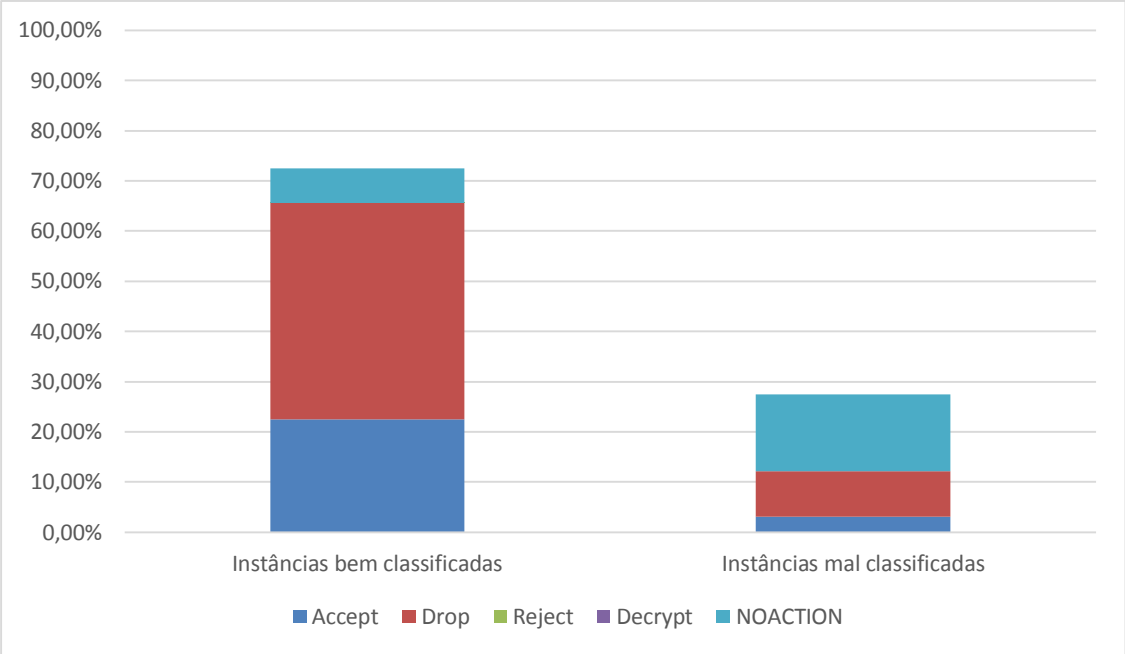

Figura 31 – Teste 1, gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real, para o método de naive de Bayes

- **K-vizinhos (IBK)**

De seguida utilizou-se o classificador IBK, começando-se por testar qual o melhor k, obtendo-se as seguintes taxas de acerto em função de k:

$k =$	1	2	3	4	5	6	7	11	51	111	1111
✓(%)	95.42	94.75	94.85	94.47	94.62	94.28	94	93.02	90.65	89.37	78.6

Tabela 14 - Teste 1, taxa de acerto em função de k para o método dos k-vizinhos

Conforme se pode observar, à medida que o k vai aumentando, a taxa de acerto vai diminuindo. Assim, selecionando-se $k = 1$, obtiveram-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 5725	✓ 95.4167%
× Instâncias mal classificadas	× 275	× 4.5833%

Tabela 15 – Teste 1, instâncias bem e mal classificadas para o método dos k-vizinhos

Accept	Drop	Reject	Decrypt	NOACTION	Previsto / Real	Taxa de acerto por classe $(\frac{verdadeiros\ positivos}{total\ classe\ real})$
1416	11	0	0	108	Accept	92.2%
12	3098	0	0	18	Drop	99%
0	0	5	0	0	Reject	100%
0	0	0	0	0	Decrypt	-
113	12	1	0	1206	NOACTION	90.5%

Tabela 16 – Teste 1, matriz de dispersão e taxa de acerto por classe para o método dos k-vizinhos

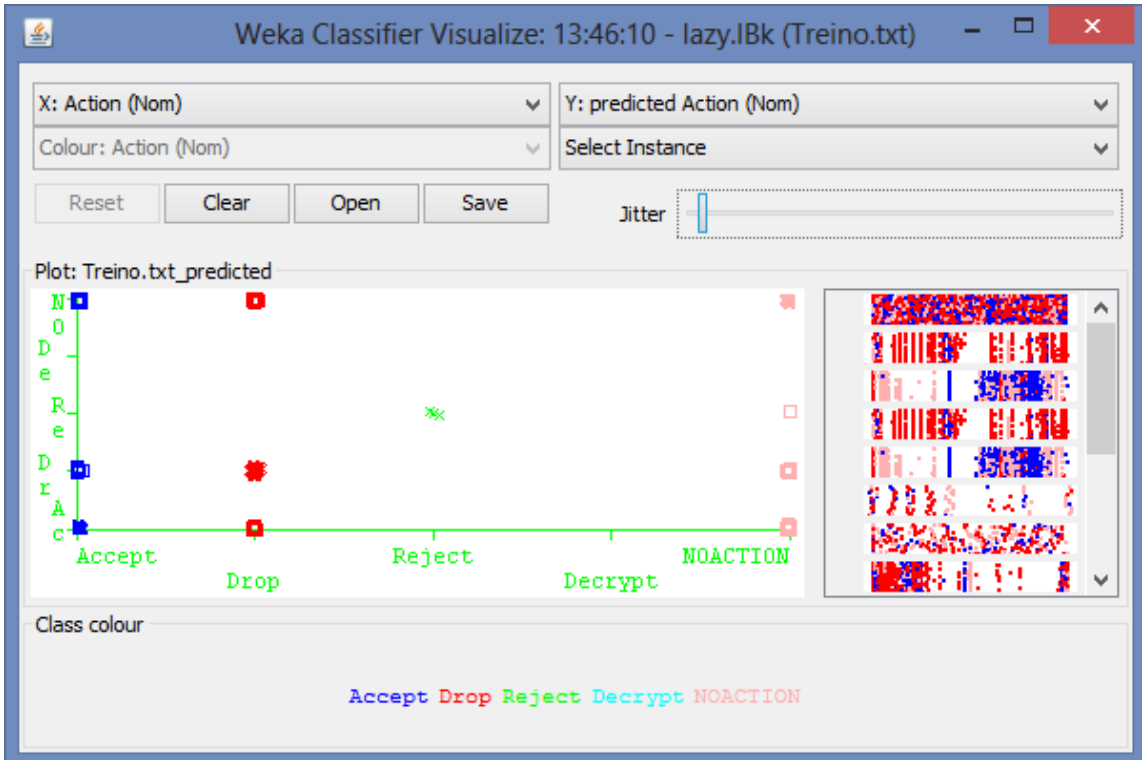

Figura 32 – Teste 1, gráfico com dispersão dos erros para o método dos k-vizinhos

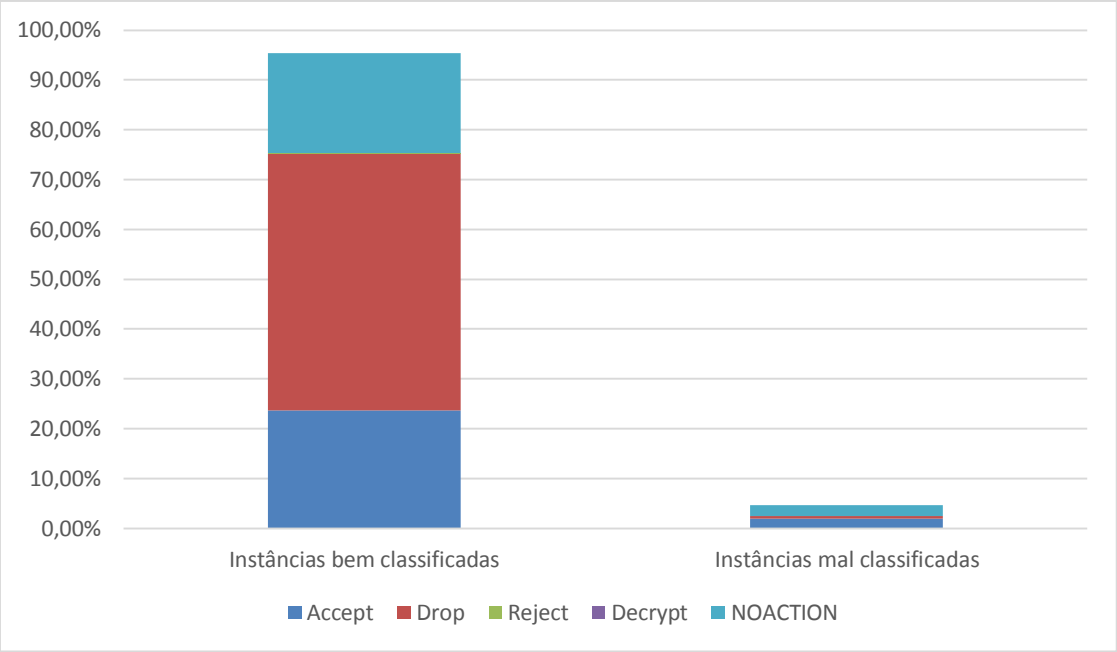

Figura 33 – Teste 1, gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real, para o método dos k-vizinhos

- **Árvore de decisão C4.5 (J48)**

Por fim testou-se ainda com as árvores de decisão C4.5, método este que nos permitirá entender de forma mais intuitiva quais as regras utilizadas para se efetuar a classificação. Obtiveram-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 5881	✓ 98.0167%
× Instâncias mal classificadas	× 119	× 1.9833%

Tabela 17 – Teste 1, instâncias bem e mal classificadas para o método das árvores de decisão (C4.5)

Accept	Drop	Reject	Decrypt	NOACTION	Previsto / Real	Taxa de acerto por classe $(\frac{verdadeiros\ positivos}{total\ classe\ real})$
1471	2	0	0	62	Accept	95.8%
11	3114	0	0	3	Drop	99.6%
0	0	5	0	0	Reject	100%
0	0	0	0	0	Decrypt	-
39	2	0	0	1291	NOACTION	98%

Tabela 18 – Teste 1, matriz de dispersão e taxa de acerto por classe para o método das árvores de decisão (C4.5)

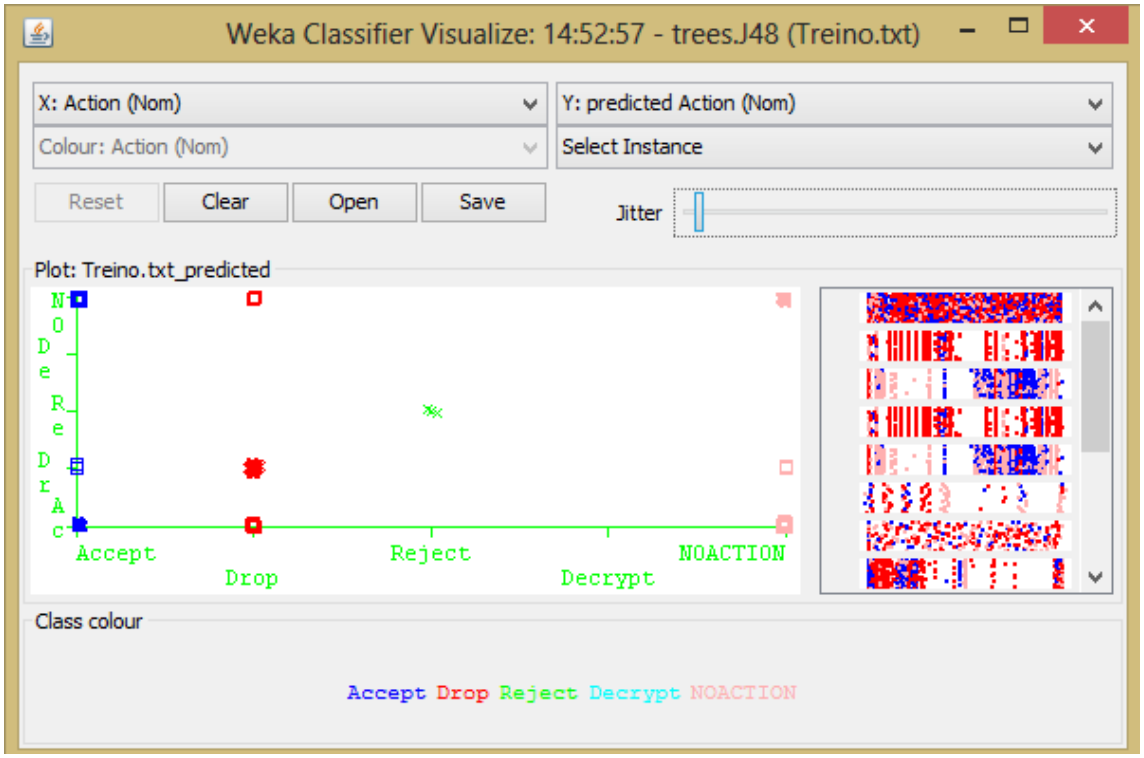

Figura 34 – Teste 1, gráfico com dispersão dos erros para o método das árvores de decisão (C4.5)

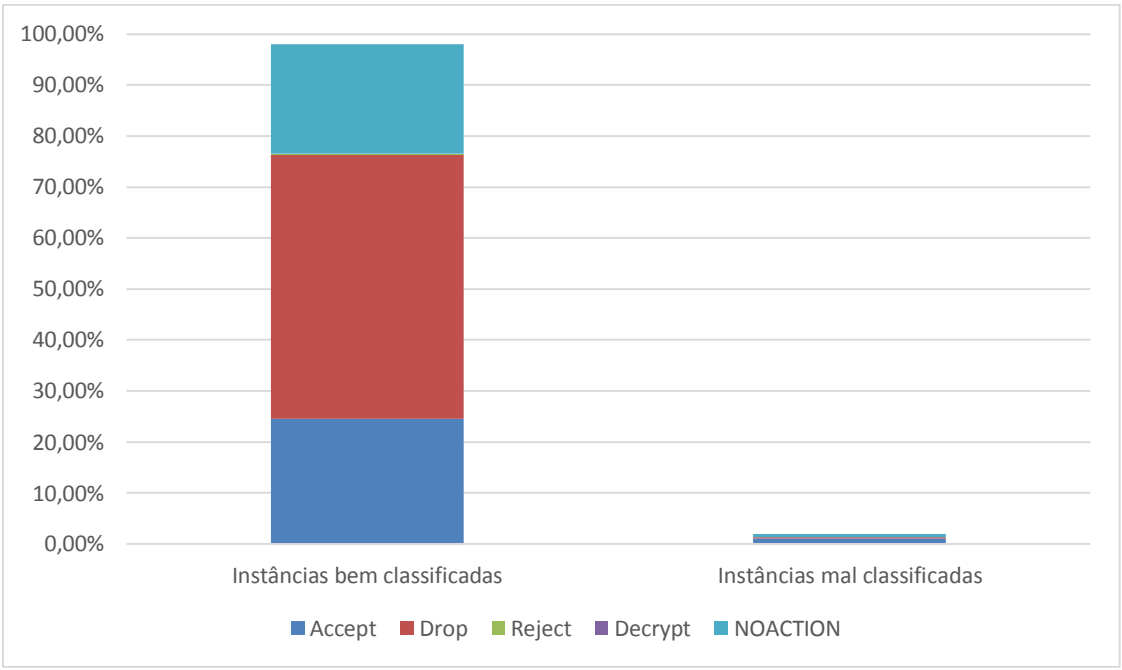

Figura 35 – Teste 1, gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real,para o método das árvores de decisão (C4.5)

Se se desenhar a árvore de decisão obtida, tem-se:

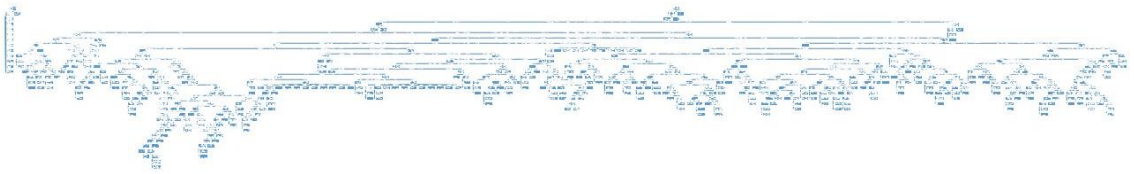

Figura 36 – Teste 1, árvore de decisão obtida

Conforme se pode observar, com a árvore ideal não é possível ter uma perceção de quais são os critérios que estão a ser utilizados, pois a árvore tem 625 nós. Assim, por forma a ser possível visualizar os critérios, obrigou-se a que cada nó tenha pelo menos 1000 objetos, o que diminuiu o número de nós para apenas 19 e obviamente também a taxa de acerto, para 88%. Obteve-se a árvore seguinte:

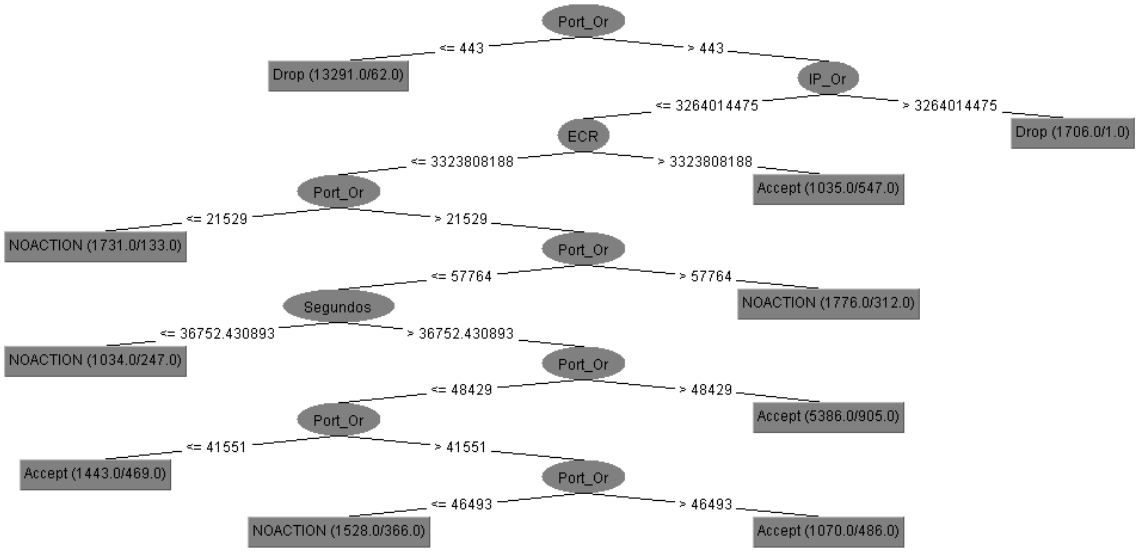

Figura 37 – Teste 1, árvore de decisão com apenas 19 nós

Como se pode observar, forçar que todos os nós tenham pelo menos 1000 instâncias, implica que nalguns casos não se consigam destinguir as diferentes classes (se os sub-grupos tiverem menos que 1000 instâncias). Observe-se por exemplo o nó que dita que todas as portas abaixo de 443 devem ser *drop*. 13291 instâncias desse nó ficam bem classificadas e as restantes 62 ficam mal. Na primeira árvore tal situação levaria a que se colocassem mais questões a estas 62 instâncias, daí a mesma ser tão extensa. Observe-se ainda que estas árvores foram construídas utilizando o conjunto de treino todo (apesar do software *weka* criar uma árvore com apenas 80% para fazer testes, no final a árvore apresentada utiliza sempre o conjunto de treino todo para ser construída, por forma a ser a "melhor árvore possível" a ser aplicada em classificações futuras).

- **Conclusões**

➢ Com a utilização de 20% dos dados para teste, foi possível prever com grande exatidão qual a classificação de cada ligação;
➢ O classificador que obteve melhores resultados foi a árvore de decisão C4.5 (J48), com 98.02% de instâncias bem classificadas.

5.2.6. Teste 2 – Validação cruzada

Tendo em conta os bons resultados obtidos no primeiro teste, poderia questionar-se se não foi apenas por mera sorte que aquele "pequeno" conjunto de 6000 instâncias aleatórias foi tão bem classificado. Por forma a se testar essa situação e a se obterem valores de taxa de acerto mais próximos dos reais, optou-se por se testar todo o conjunto de dados, recorrendo-se ao método da validação cruzada, para 10 conjuntos.

- ***Naive de Bayes***

Começou-se novamente por se testar com o classificador *naive de Bayes*, obtendo-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 21436	✓ 71.4533%
× Instâncias mal classificadas	× 8564	× 28.5467%

Tabela 19 – Teste 2, instâncias bem e mal classificadas para o método de naive de Bayes

Accept	Drop	Reject	Decrypt	NOACTION	Previsto / Real	Taxa de acerto por classe ($\frac{verdadeiros\ positivos}{total\ classe\ real}$)
6549	75	0	0	916	Accept	86.9%
727	12824	0	0	2126	Drop	81.8%
0	0	21	0	1	Reject	95.5%
0	0	0	0	0	Decrypt	-
3997	722	0	0	2042	NOACTION	30.2%

Tabela 20 – Teste 2, matriz de dispersão e taxa de acerto por classe para o método de naive de Bayes

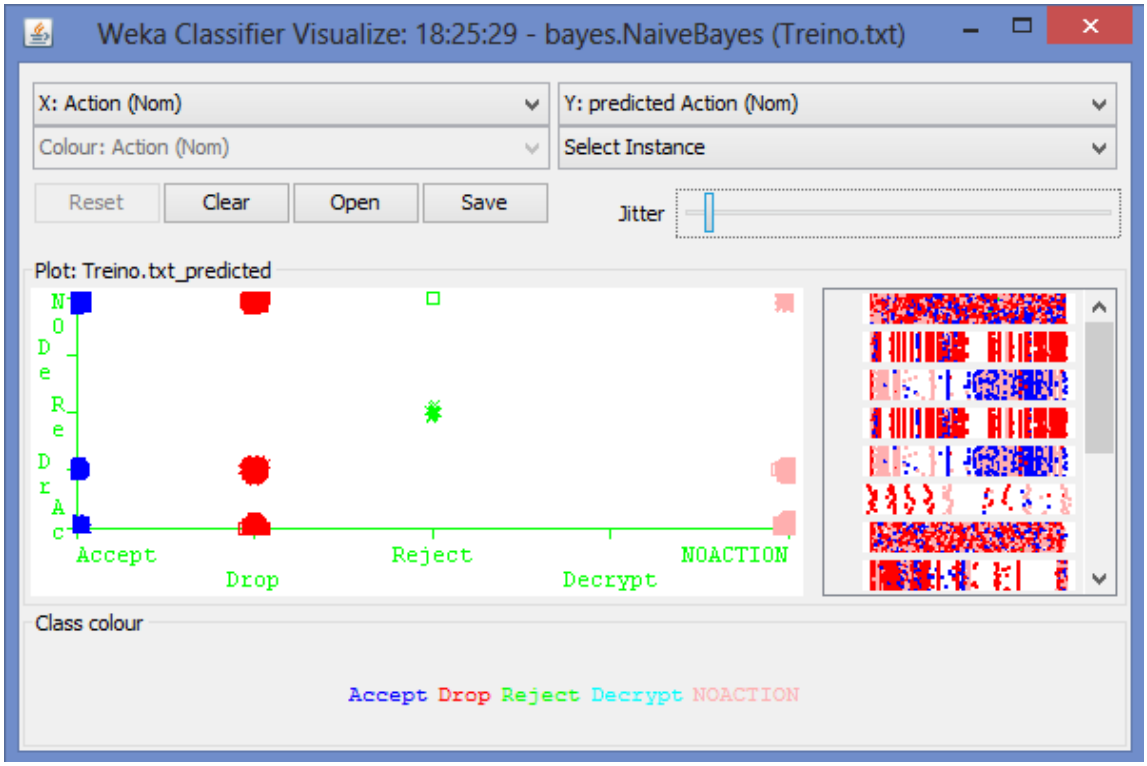

Figura 38 – Teste 2, gráfico com dispersão dos erros para o método de naive de Bayes

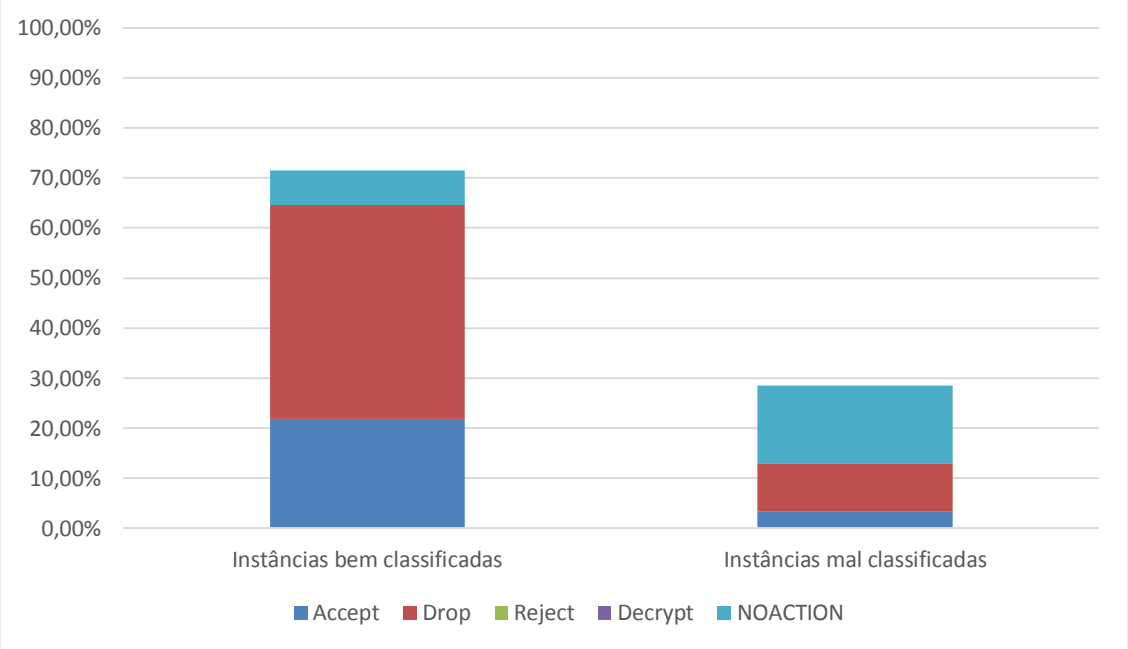

Figura 39 – Teste 2, gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real, para o método de naive de Bayes

- **K-vizinhos (IBK)**

De seguida utilizou-se o classificador IBK, começando-se por testar qual o melhor k, obtendo-se as seguintes taxas de acerto em função de k:

$k =$	1	11	111	1111
✓ (%)	95.48	93.13	89.11	78.96

Tabela 21 - Teste 2, taxa de acerto em função de k para o método dos k-vizinhos

Conforme se pode observar uma vez mais, à medida que o k vai aumentando, a taxa de acerto vai diminuindo. Assim, selecionando-se $k = 1$, obtiveram-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 28643	✓ 95.4767%
× Instâncias mal classificadas	× 1357	× 4.5233%

Tabela 22 – Teste 2, instâncias bem e mal classificadas para o método dos k-vizinhos

Accept	Drop	Reject	Decrypt	NOACTION	Previsto / Real	Taxa de acerto por classe $(\frac{verdadeiros\ positivos}{total\ classe\ real})$
6916	83	3	0	538	Accept	91.7%
75	15538	0	0	64	Drop	99.1%
1	0	21	0	0	Reject	95.5%
0	0	0	0	0	Decrypt	-
522	68	3	0	6168	NOACTION	91.2%

Tabela 23 – Teste 2, matriz de dispersão e taxa de acerto por classe para o método dos k-vizinhos

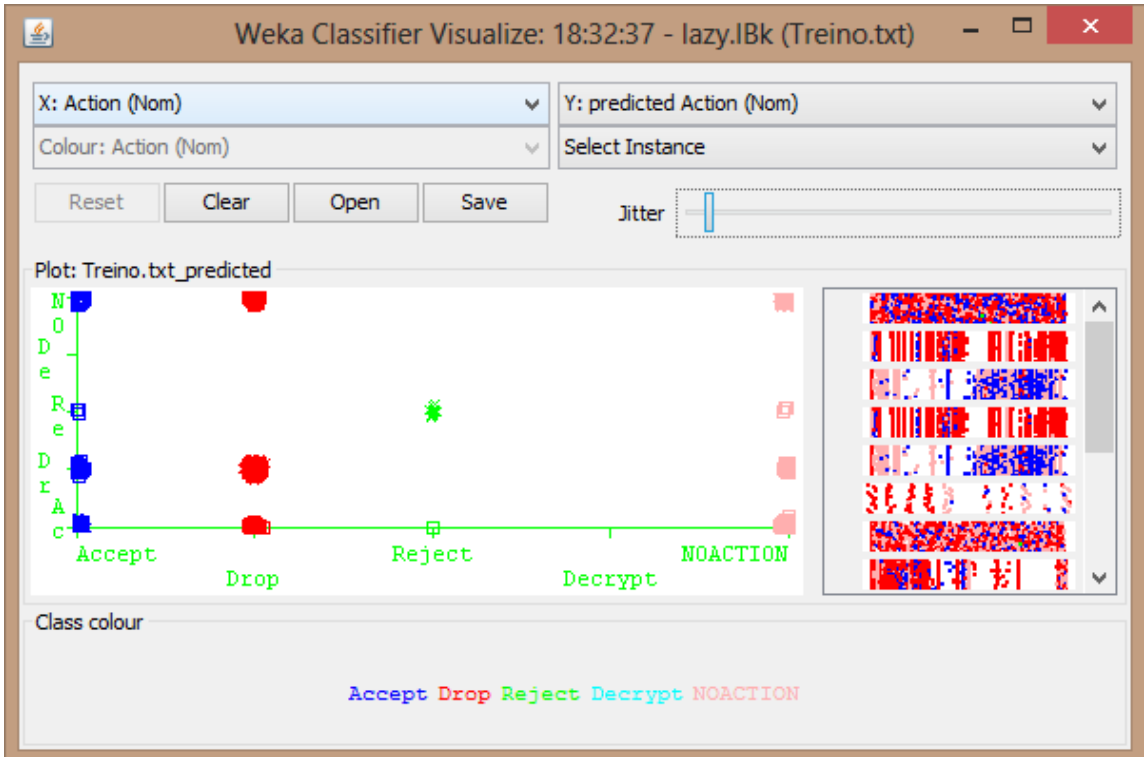

Figura 40 – Teste 2, gráfico com dispersão dos erros para o método dos k-vizinhos

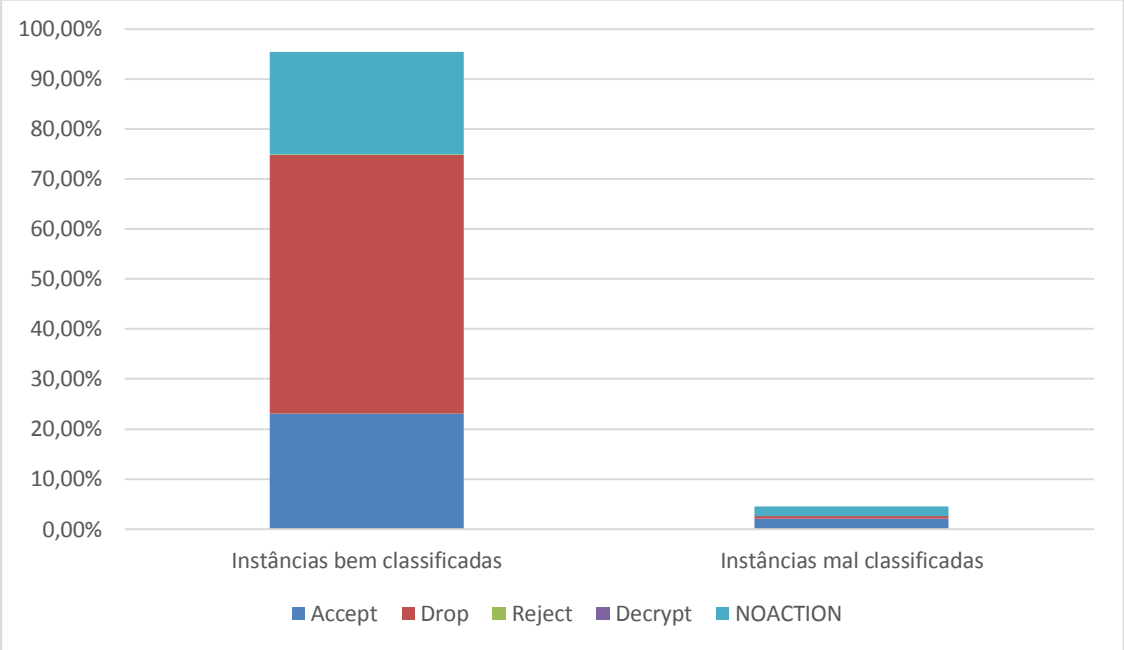

Figura 41 – Teste 2, gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real, para o método dos k-vizinhos

- **Árvore de decisão C4.5 (J48)**

Por fim testou-se ainda com as árvores de decisão C4.5, método este que nos permitirá entender de forma mais intuitiva quais as regras utilizadas para se efetuar a classificação. Obtiveram-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 29456	✓ 98.1867%
× Instâncias mal classificadas	× 544	× 1.8133%

Tabela 24 – Teste 2, instâncias bem e mal classificadas para o método das árvores de decisão (C4.5)

Accept	Drop	Reject	Decrypt	NOACTION	Previsto / Real	Taxa de acerto por classe $(\frac{verdadeiros\ positivos}{total\ classe\ real})$
7253	15	0	0	272	Accept	96.2%
41	15622	0	0	14	Drop	99.6%
0	0	22	0	0	Reject	100%
0	0	0	0	0	Decrypt	-
188	14	0	0	6559	NOACTION	97%

Tabela 25 – Teste 2, matriz de dispersão e taxa de acerto por classe para o método das árvores de decisão (C4.5)

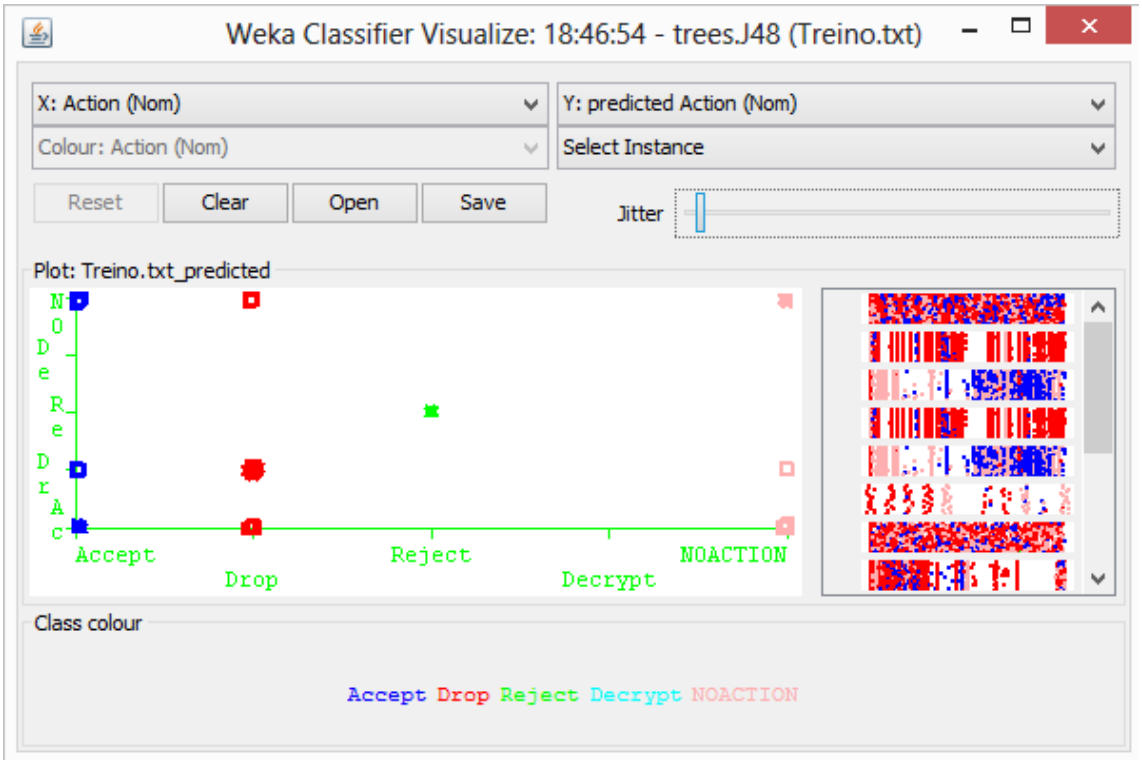

Figura 42 – Teste 2, gráfico com dispersão dos erros para o método das árvores de decisão (C4.5)

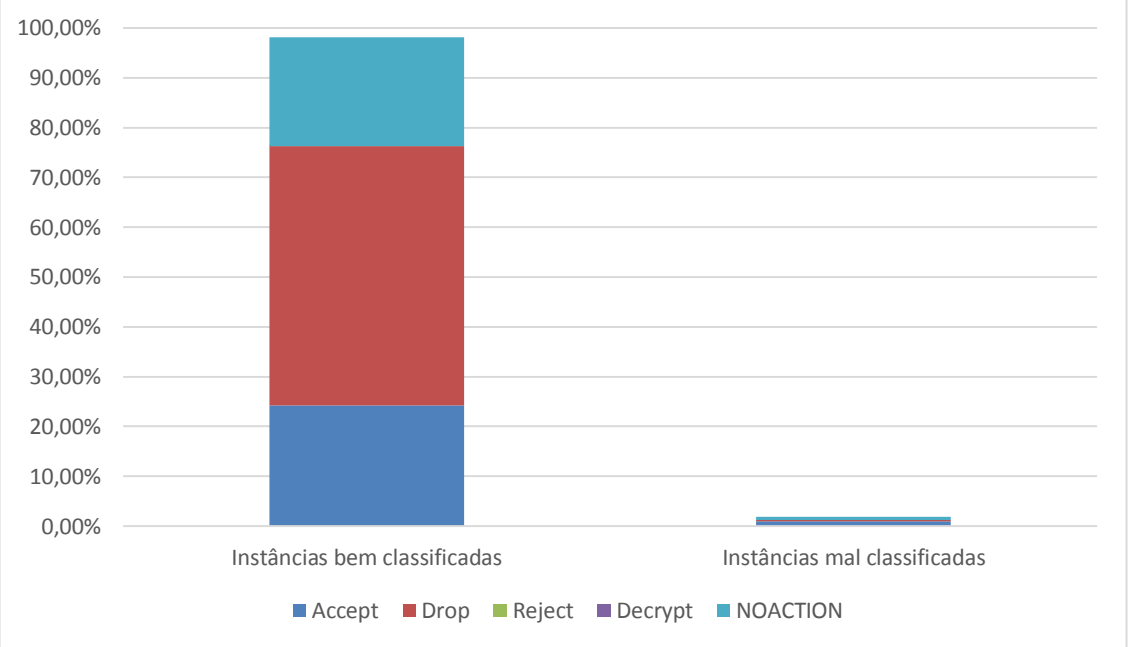

Figura 43 – Teste 2 gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real, para o método das árvores de decisão (C4.5)

Uma vez mais a árvore criou 625 nós. Assim, por forma a ser possível visualizar os critérios, obrigou-se a que cada nó tenha pelo menos 1000 objetos, o que diminuiu o número de nós para apenas 19 e a taxa de acerto, para 88%. Obteve-se a árvore seguinte:

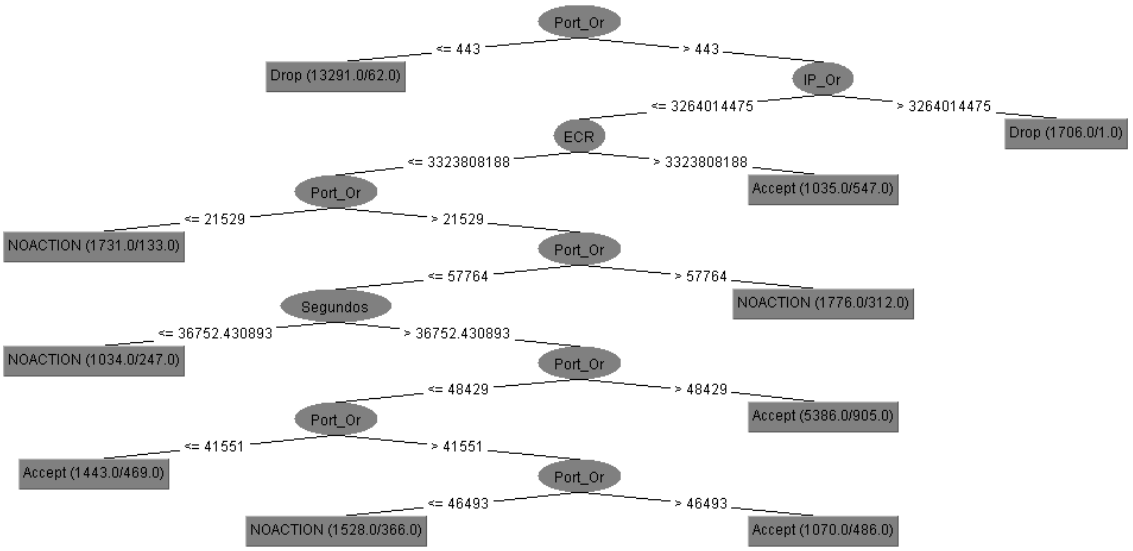

Figura 44 – Teste 2, árvore de decisão com apenas 19 nós

Como se pode observar, a árvore de decisão obtida é exatamente a mesma, pois também aqui foram utilizados todos os dados para construir o modelo de árvore.

- **Conclusões**

- Exeptuando para o classificador de Naive de Bayes, com a utilização do método de validação cruzada, os resultados foram ligeiramente melhores que com o método anterior;
- O classificador que obteve melhores resultados foi a árvore de decisão C4.5 (J48), com 98.19% de instâncias bem classificadas.

5.2.7. Teste 3 – Ficheiro de teste com as últimas 10 mil linhas da $BD2_{CLASS}$

Tendo em conta a continuação dos bons resultados obtidos no primeiro teste, poderia questionar-se ainda se não foi apenas por se estar a efetuar testes num intervalo tão curto de dados ($36754.84 - 36752.25 = 2.59$ segundos) que se obtiveram estes resultados. Por forma a se testar essa situação retirou-se uma nova amostra das últimas 10000 linhas para ser utilizada como ficheiro de teste. Note-se que entre a última instância do ficheiro de treino e a primeira instância do novo ficheiro de teste passaram mais de 51 minutos ($39827.57 - 36754.84 = 3072.73$ segundos).

A Figura 45 ilustra a distribuição dos dados por campo, onde se deve considerar o eixo das abcissas como sendo o valor de cada campo e o das ordenadas o número de vezes que esse valor ocorre. No último campo "Action" pode-se observar que 3925

estavam classificados como Accept (a azul), 5048 como Drop (a vermelho), 2 como Reject, 0 como Decrypt e 1025 como NOACTION (a beje).

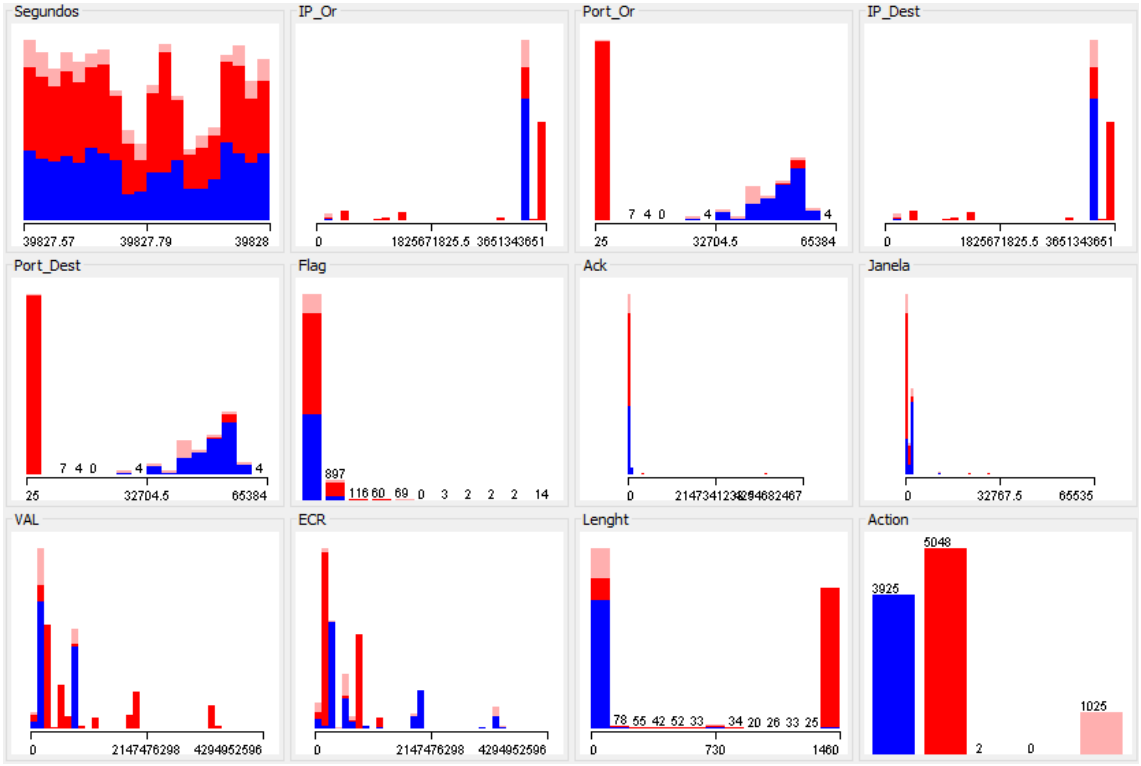

Figura 45 - Distribuição dos campos nas últimas 10000 instâncias da $BD2_{CLASS}$

- ***Naive de Bayes***

Começou-se novamente por se testarem os dados com o classificador *naive de Bayes*, obtendo-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 8691	✓ 86.91%
× Instâncias mal classificadas	× 1309	× 13.09%

Tabela 26 – Teste 3, instâncias bem e mal classificadas para o método de naive de Bayes

Accept	Drop	Reject	Decrypt	NOACTION	Previsto / Real	Taxa de acerto por classe $(\frac{verdadeiros\ positivos}{total\ classe\ real})$
3833	16	0	0	76	Accept	97.7%
244	4727	0	0	77	Drop	93.6%
0	1	0	0	1	Reject	0%
0	0	0	0	0	Decrypt	-
857	37	0	0	131	NOACTION	86.9%

Tabela 27 – Teste 3, matriz de dispersão e taxa de acerto por classe para o método de naive de Bayes

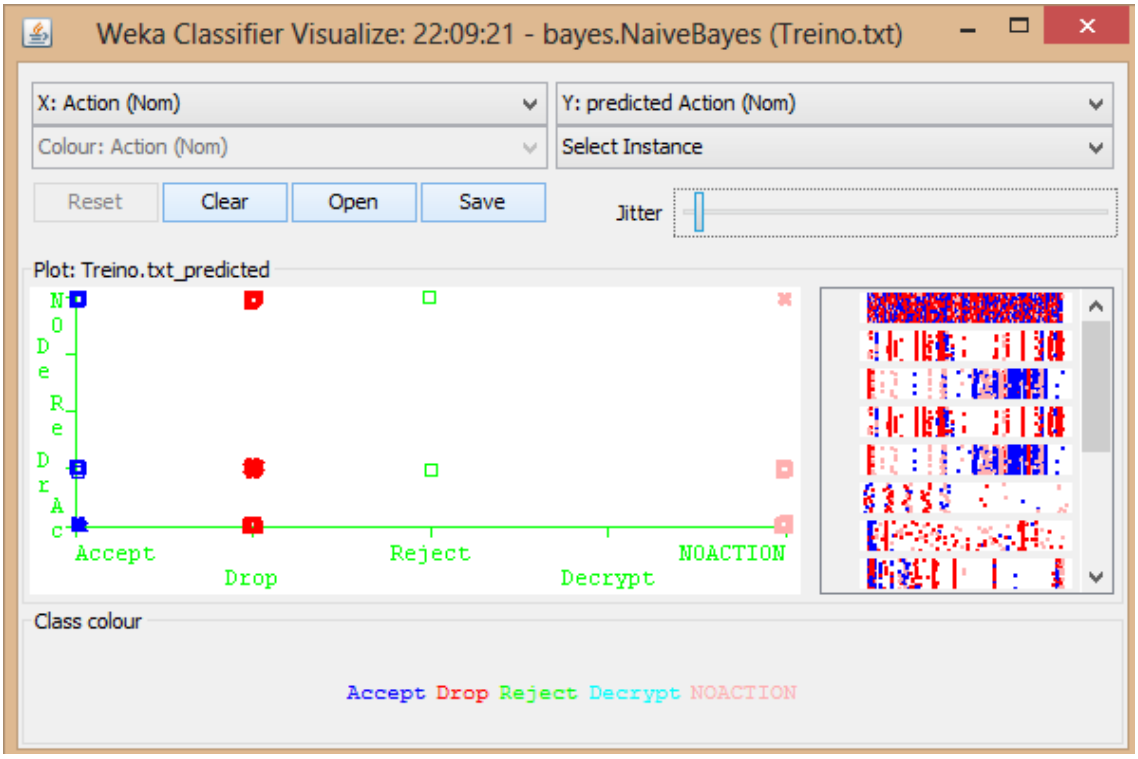

Figura 46 – Teste 3, gráfico com dispersão dos erros para o método de naive de Bayes

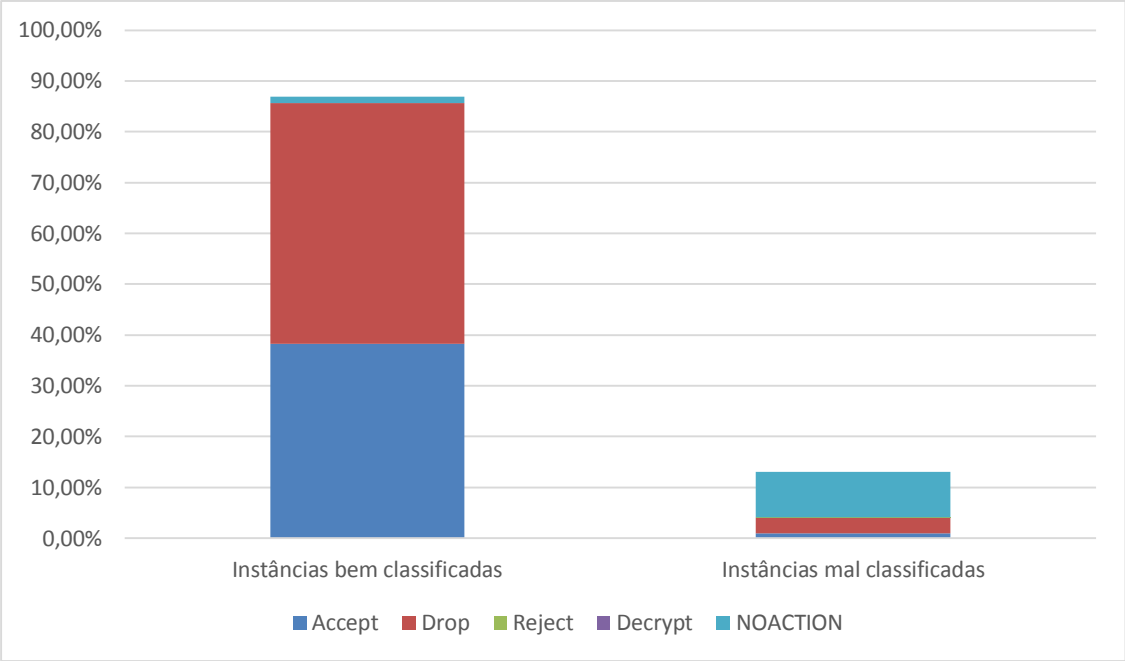

Figura 47 – Teste 3, gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real, para o método de naive de Bayes

- **K-vizinhos (IBK)**

De seguida utilizou-se o classificador IBK, começando-se por testar qual o melhor k, obtendo-se as seguintes taxas de acerto em função de k:

$k =$	1	11	111	1111	11111	29999
✓ (%)	91.78	78.21	70.61	73.4	86.76	50.48

Tabela 28 - Teste 3, taxa de acerto em função de k para o método dos k-vizinhos

Curiosamente, neste teste a taxa de acerto foi diminuindo até $k = 111$, começando a aumentar e caindo abruptamente assim que se colocou $k = 29999$. Denote-se que o melhor k encontrado continua a ser o $k = 1$ e mesmo que assim não fosse, ao se escolher um k entre 1111 e 29999 (intervalo onde se estima encontrar o melhor k) o tempo de demora da classificação já era demasiado elevado, motivo pelo qual provavelmente não se justificaria a opção desse k. Assim, selecionando-se $k = 1$ obtiveram-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 9178	✓ 91.78%
× Instâncias mal classificadas	× 822	× 8.22%

Tabela 29 – Teste 3, instâncias bem e mal classificadas para o método dos k-vizinhos

Accept	Drop	Reject	Decrypt	NOACTION	Previsto / Real	Taxa de acerto por classe $(\frac{verdadeiros\ positivos}{total\ classe\ real})$
3720	12	0	0	193	Accept	94.8%
233	4790	0	0	25	Drop	94.9%
0	0	0	0	2	Reject	0%
0	0	0	0	0	Decrypt	-
343	14	0	0	668	NOACTION	65.2%

Tabela 30 – Teste 3, matriz de dispersão e taxa de acerto por classe para o método dos k-vizinhos

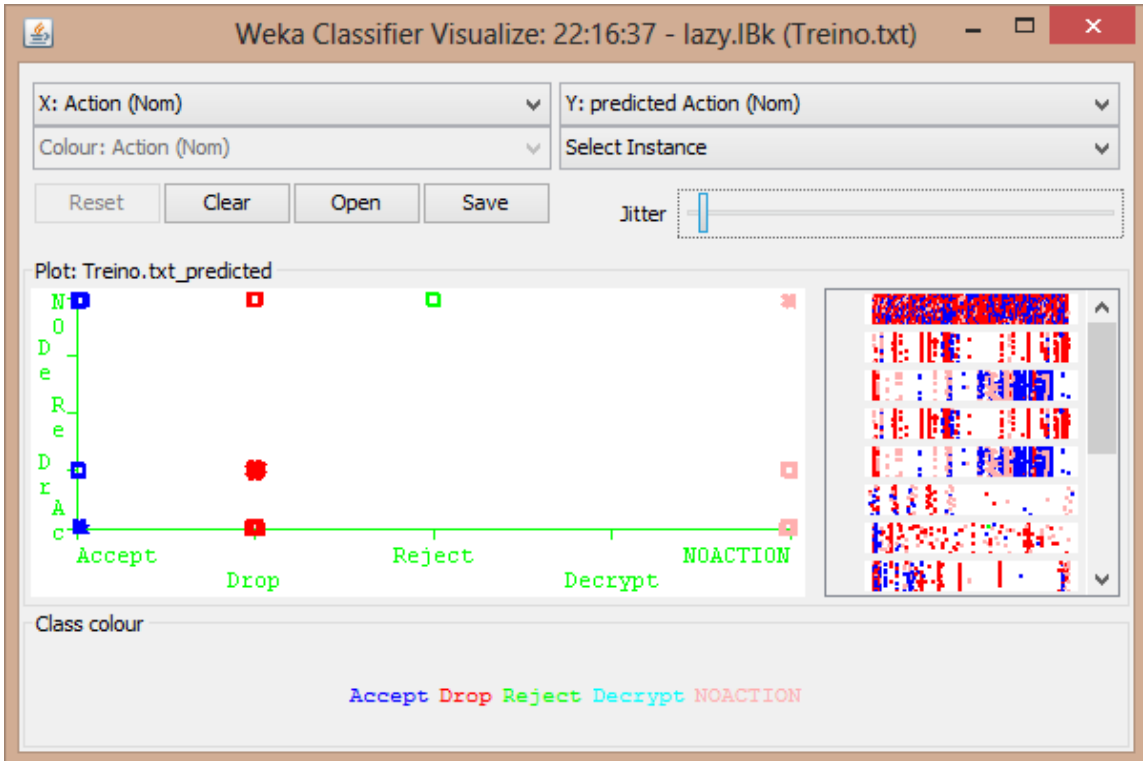

Figura 48 – Teste 3, gráfico com dispersão dos erros para o método dos k-vizinhos

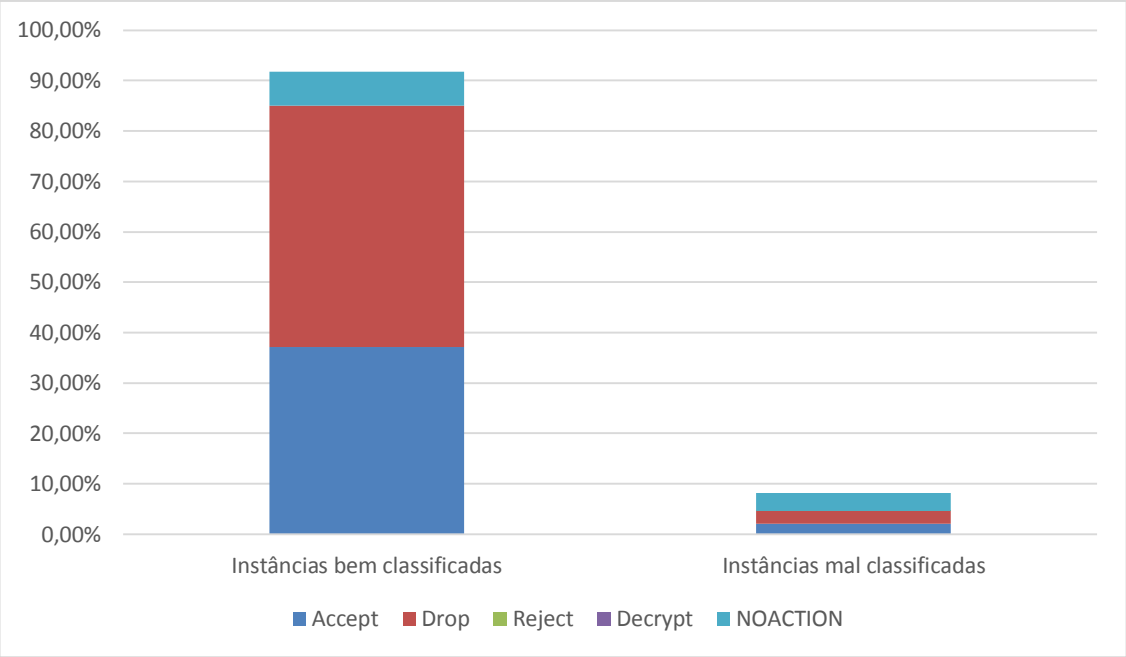

Figura 49 – Teste 3, gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real, para o método dos k-vizinhos

- **Árvore de decisão C4.5 (J48)**

Por fim testou-se com as árvores de decisão C4.5. Obtiveram-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 8802	✓ 88.02%
× Instâncias mal classificadas	× 1198	× 11.98%

Tabela 31 – Teste 3, instâncias bem e mal classificadas para o método das árvores de decisão (C4.5)

Accept	Drop	Reject	Decrypt	NOACTION	Previsto / Real	Taxa de acerto por classe $(\frac{verdadeiros\ positivos}{total\ classe\ real})$
3483	7	0	0	435	Accept	88.7%
234	4796	0	0	18	Drop	95%
0	0	0	0	2	Reject	0%
0	0	0	0	0	Decrypt	-
494	8	0	0	523	NOACTION	51%

Tabela 32 – Teste 3, matriz de dispersão e taxa de acerto por classe para o método das árvores de decisão (C4.5)

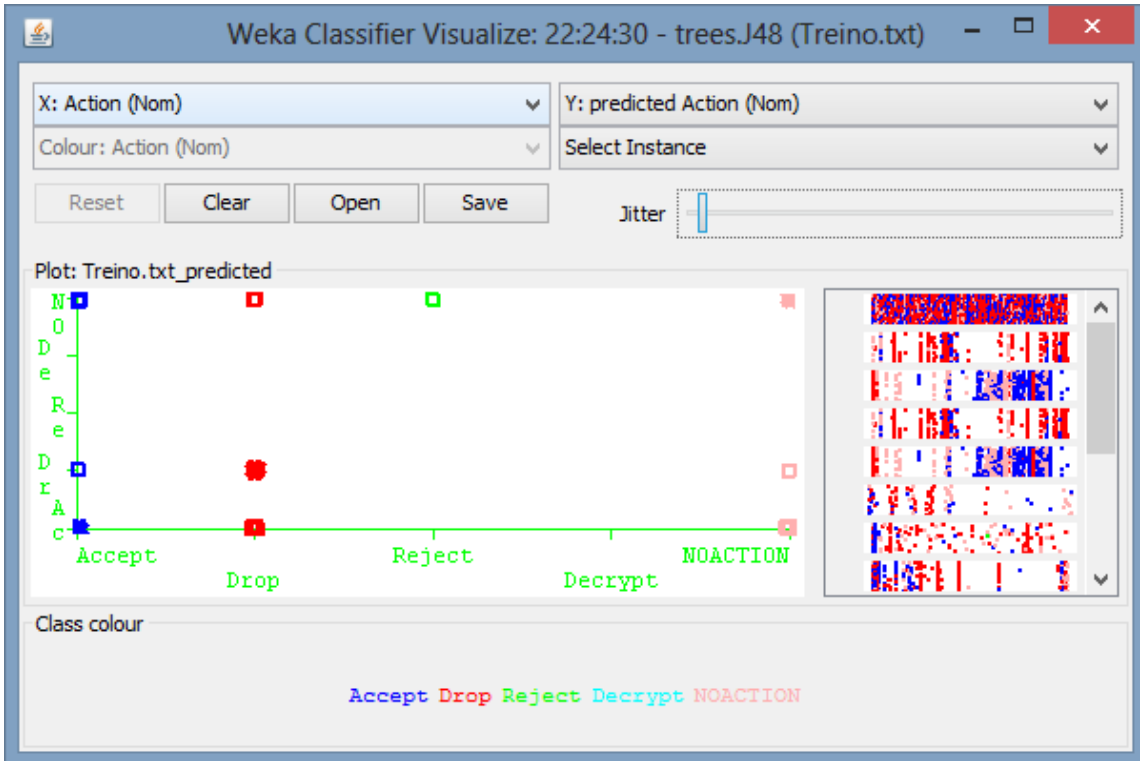

Figura 50 – Teste 3, gráfico com dispersão dos erros para o método das árvores de decisão (C4.5)

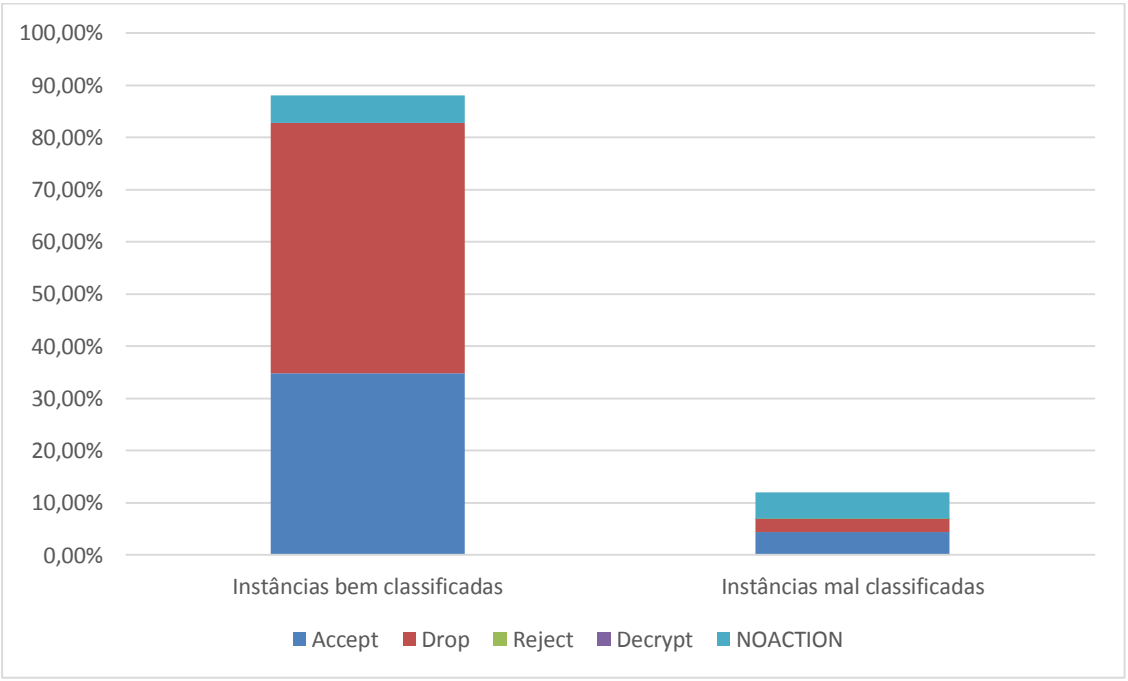

Figura 51 – Teste 3, gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real, para o método das árvores de decisão (C4.5)

Uma vez mais a árvore criou 625 nós, tal acontece porque sempre que se insere um ficheiro de teste, o *software weka* utiliza apenas o ficheiro de treino para efetuar a

árvore, assim sendo as árvores iam ser iguais às anteriores, motivo pelo qual se optou por não as apresentar.

- **Conclusões**

- Surpreendentemente, com a utilização das últimas 10 mil linhas da $BD2_{CAP}$, o classificador de *Naive de Bayes* obteve ainda melhores resultados que para os testes anteriores;
- Conforme seria de esperar, os outros classificadores obtiveram piores desempenhos que nos testes anteriores;
- O classificador que obteve melhores resultados foi o K-vizinhos (IBK), com 91.78% de instâncias bem classificadas.

5.2.8. Teste 4 – Ficheiro de teste com as últimas 10 mil linhas da $BD3_{CLASS}$

Após os resultados obtidos no teste anterior colocou-se em cima da mesa a possibilidade de os classificadores estarem a efetuar *overfitting*, isto é a decorar as ligações. Esta situação é descrita em *data mining* como sobreaprender e é comum acontecer quando os mesmos dados se repetem, e o classificador memoriza dados concretos sem generalizar.

Assim, por forma a garantir que tal não seria possível acontecer nesta investigação, efetuou-se uma nova captura ($BD3_{CAP}$) com mais de 100 dias de intervalo para a captura anterior ($BD2_{CAP}$).

Começou-se então por se recolher uma amostra de 10000 linhas, desta vez da $BD3_{CAP}$, para teste, e utilizar o mesmo ficheiro de treino que no teste anterior.

Inicialmente os dados obtidos não foram nada favoráveis, tendo-se obtido apenas 6.66% de instâncias corretamente classificadas para o método do *naive de Bayes*, 4.53% para o método dos k-vizinhos ($k = 1$) e 5.56% para a árvore de decisão C4.5.

Após uma detalhada observação das matrizes de correlação observou-se que o principal problema se encontrava na classe NOACTION, que era aquela que tinha maior número de instâncias "mal classificadas".

Assim sendo, ponderou-se inicialmente a possibilidade destas ligações estarem a ser rejeitadas pela *firewall* sem que essa informação estivesse a ser registada. Passaram-se então todas as ações NOACTION dos ficheiros de treino e teste anteriores para DROP, e testaram-se novamente os dados. Obteve-se 62.39% de instâncias corretamente

classificadas para o método do *naive de Bayes*, 62.82% para o método dos k-vizinhos e 93.35% para a árvore de decisão C4.5. A Tabela 33 mostra a matriz de dispersão e a taxa de acerto obtidas para o último método.

Accept	Drop	Reject	Decrypt	Previsto / Real	Taxa de acerto por classe $(\frac{verdadeiros\ positivos}{total\ classe\ real})$
27	286	0	0	Accept	8.6%
379	9308	0	0	Drop	96.1%
0	0	0	0	Reject	-
0	0	0	0	Decrypt	-

Tabela 33 – Teste com classe NOACTION substituída por Drop, matriz de dispersão e taxa de acerto por classe para o método das árvores de decisão (C4.5)

Conforme se pode observar, apesar dos aparentes bons resultados obtidos, a taxa de acerto para a classe Accept continuava muito baixa, ou seja existia algo que ainda não fazia sentido nestes dados.

Após alguma ponderação para se tentar compreender o porquê de existirem todas estas ligações NOACTION, e tendo-se em conta que o resultado da análise da *firewall* consiste apenas num resumo de ligações e que o *software* elaborado em *pyton* procura apenas num intervalo de 1 segundo, concluiu-se que estas ligações são ligações já anteriormente classificadas, motivo pelo qual não aparecem classificadas como DROP ou ACCEPT neste intervalo. Assim, para elaborar um classificador, não faria de todo sentido continuar a considerar estas ligações, motivo pelo qual estas foram eliminadas.

Após se eliminarem as linhas que continham a classe NOACTION nas BD, obtiveram-se então novos ficheiros, o de treino com as primeiras 30 mil linhas da $BD2_{CLASS}$ e o de teste com as últimas 10 mil da $BD3_{CLASS}$. A Figura 52 ilustra a distribuição dos dados por campo, no ficheiro de teste, onde se deve considerar o eixo das abcissas como sendo o valor de cada campo e o das ordenadas o número de vezes que esse valor ocorre. No último campo "Action" pode-se observar que 5114 estavam classificados como Accept (a azul), 4886 como Drop (a vermelho), 0 como Reject e 0 como Decrypt.

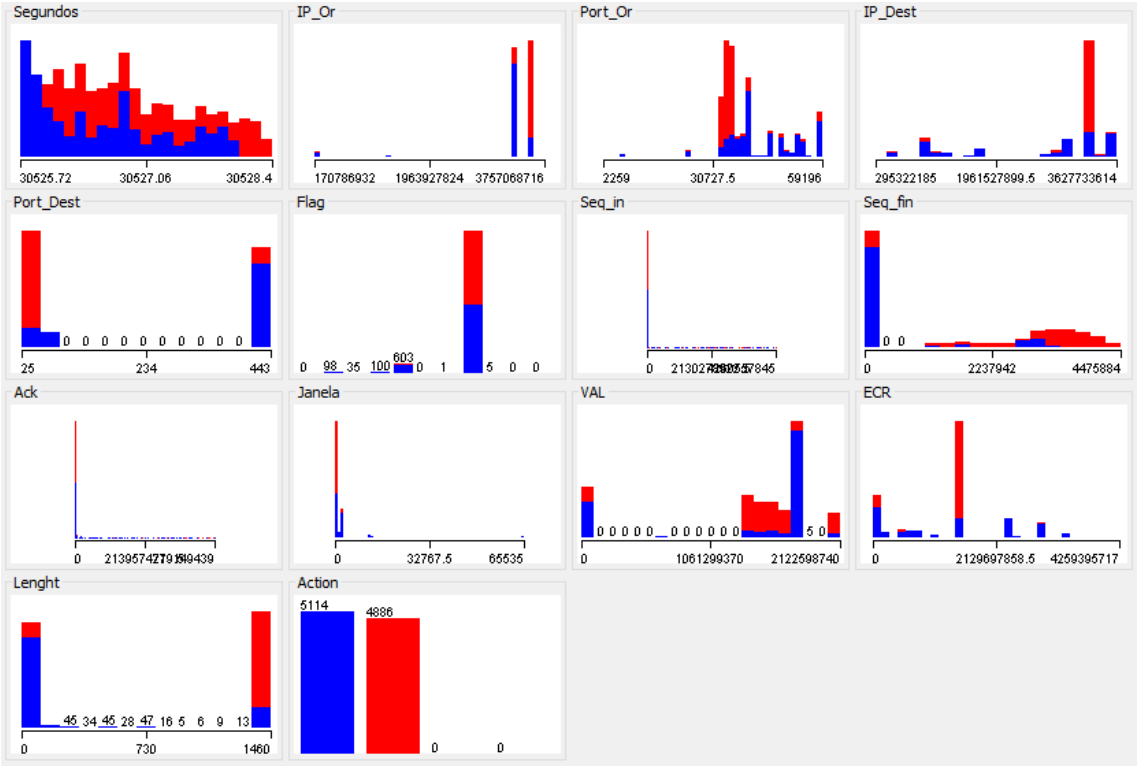

Figura 52 - Distribuição dos campos nas últimas 10000 instâncias da $BD3_{CLASS}$

- ***Naive de Bayes***

Começou-se novamente por testar os dados com o classificador *naive de Bayes*, obtendo-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 8165	✓ 81.65%
× Instâncias mal classificadas	× 1835	× 18.35%

Tabela 34 – Teste 4, instâncias bem e mal classificadas para o método de naive de Bayes

Accept	Drop	Reject	Decrypt	Previsto / Real	Taxa de acerto por classe $(\frac{verdadeiros\ positivos}{total\ classe\ real})$
3969	1145	0	0	Accept	77.6%
690	4196	0	0	Drop	85.9%
0	0	0	0	Reject	-
0	0	0	0	Decrypt	-

Tabela 35 – Teste 4, matriz de dispersão e taxa de acerto por classe para o método de naive de Bayes

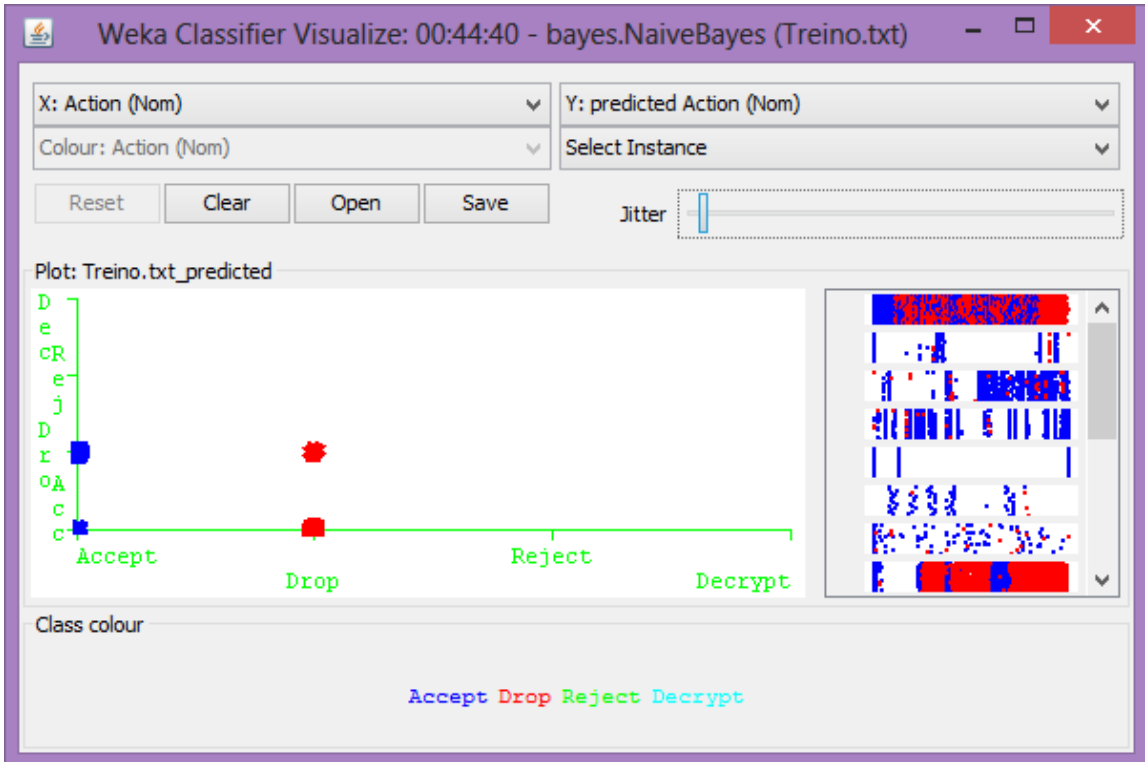

Figura 53 – Teste 3, gráfico com dispersão dos erros para o método de naive de Bayes

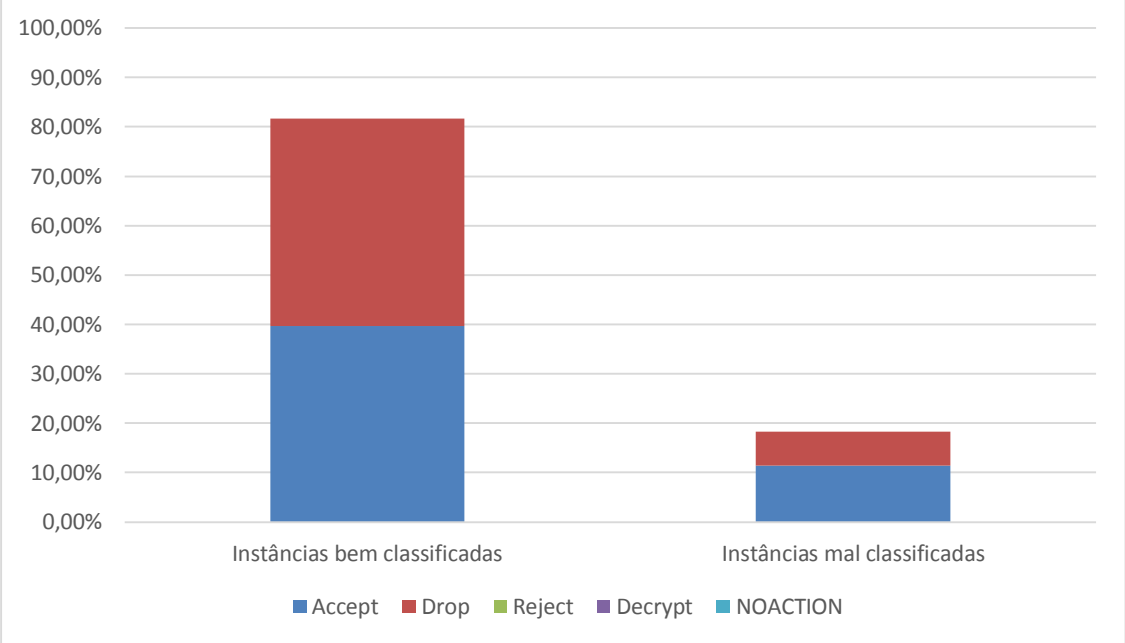

Figura 54 – Teste 4, gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real, para o método de naive de Bayes

- **K-vizinhos (IBK)**

De seguida utilizou-se o classificador IBK, começando-se por testar qual o melhor k, obtendo-se as seguintes taxas de acerto em função de k:

$k =$	1	11	111	1111	11111	29999
✓ (%)	83.19	83.1	83.97	80.67	55.65	48.86

Tabela 36 - Teste 4, taxa de acerto em função de k para o método dos k-vizinhos

Conforme se pode observar, contrariamente aos resultados anteriormente alcançados, e apesar de ser por uma pequena margem, o k que obteve melhores desempenhos foi o $k = 111$.

Assim, selecionando-se $k = 111$ obtiveram-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 8397	✓ 83.97%
× Instâncias mal classificadas	× 1603	× 16.03%

Tabela 37 – Teste 4, instâncias bem e mal classificadas para o método dos k-vizinhos

Accept	Drop	Reject	Decrypt	Previsto / Real	Taxa de acerto por classe ($\frac{verdadeiros\ positivos}{total\ classe\ real}$)
4228	886	0	0	Accept	82.7%
717	4169	0	0	Drop	85.3%
0	0	0	0	Reject	-
0	0	0	0	Decrypt	-

Tabela 38 – Teste 4, matriz de dispersão e taxa de acerto por classe para o método dos k-vizinhos

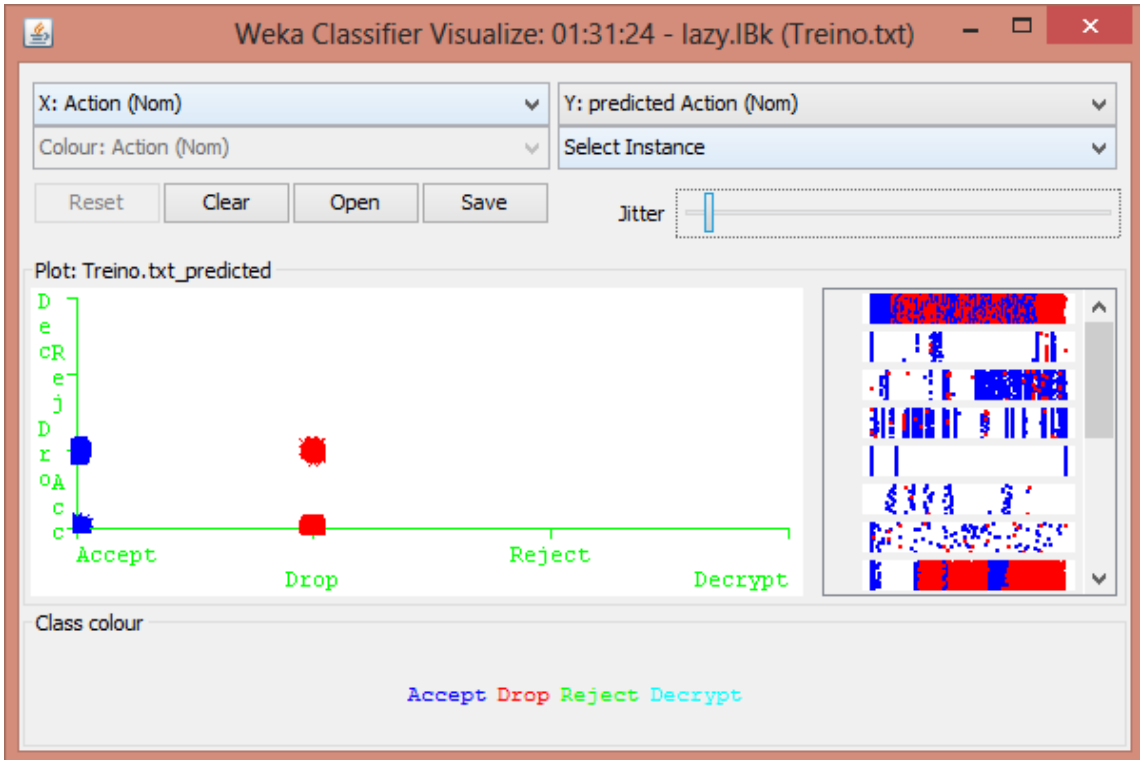

Figura 55 – Teste 4, gráfico com dispersão dos erros para o método dos k-vizinhos

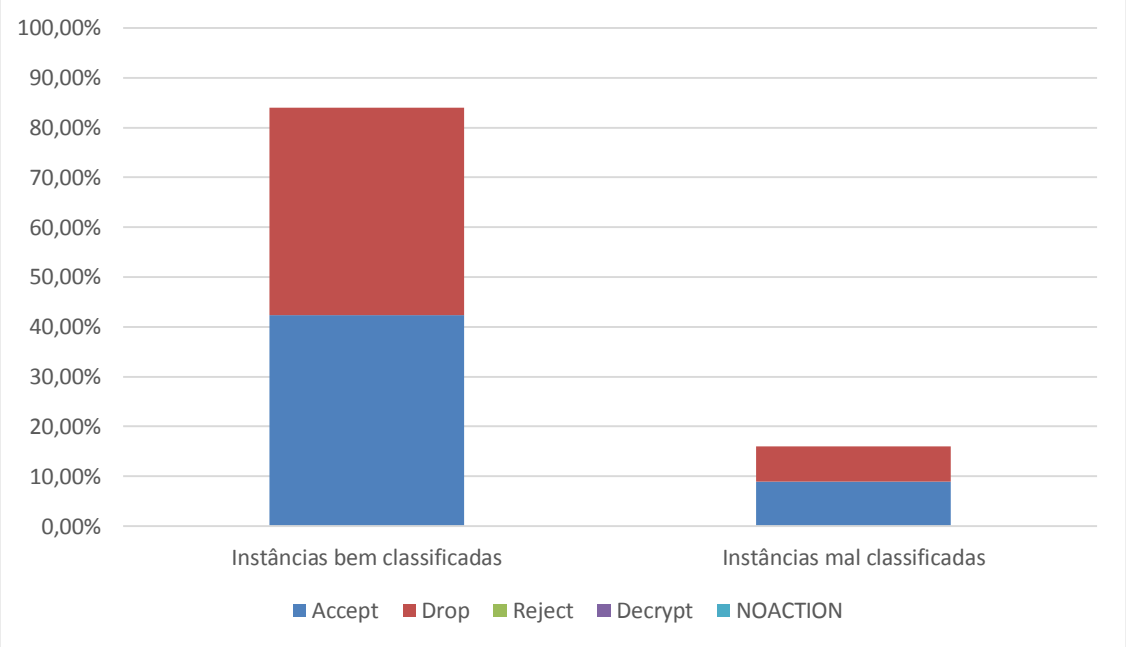

Figura 56 – Teste 4, gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real, para o método dos k-vizinhos

- **Árvore de decisão C4.5 (J48)**

Por fim testou-se com as árvores de decisão C4.5. Obtiveram-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 8112	✓ 81.12%
× Instâncias mal classificadas	× 1888	× 18.88%

Tabela 39 – Teste 4, instâncias bem e mal classificadas para o método das árvores de decisão (C4.5)

Accept	Drop	Reject	Decrypt	Previsto / Real	Taxa de acerto por classe $(\frac{verdadeiros\ positivos}{total\ classe\ real})$
3933	1142	39	0	Accept	76.9%
707	4179	0	0	Drop	85.5%
0	0	0	0	Reject	-
0	0	0	0	Decrypt	-

Tabela 40 – Teste 4, matriz de dispersão e taxa de acerto por classe para o método das árvores de decisão (C4.5)

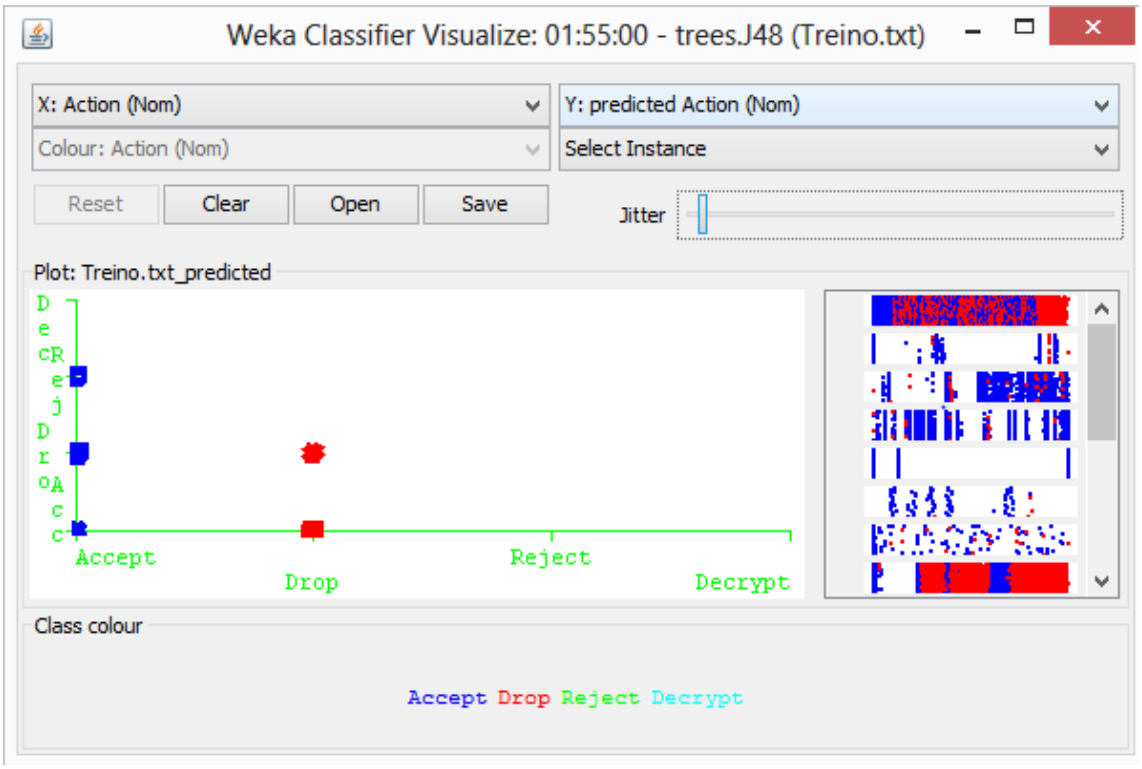

Figura 57 – Teste 4, gráfico com dispersão dos erros para o método das árvores de decisão (C4.5)

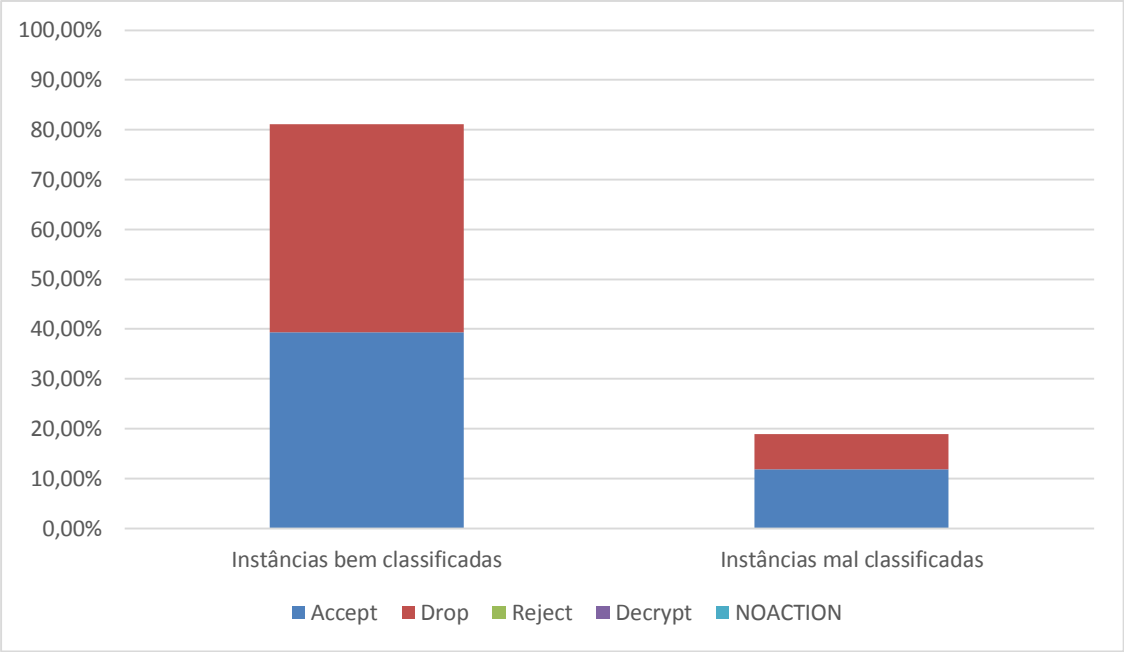

Figura 58 – Teste 4, gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real, para o método das árvores de decisão (C4.5)

Tendo em conta que o conjunto de treino já não apresenta a classe NOACTION, o número de nós reduziu para 155. A Figura 71 ilustra a árvore de decisão obtida.

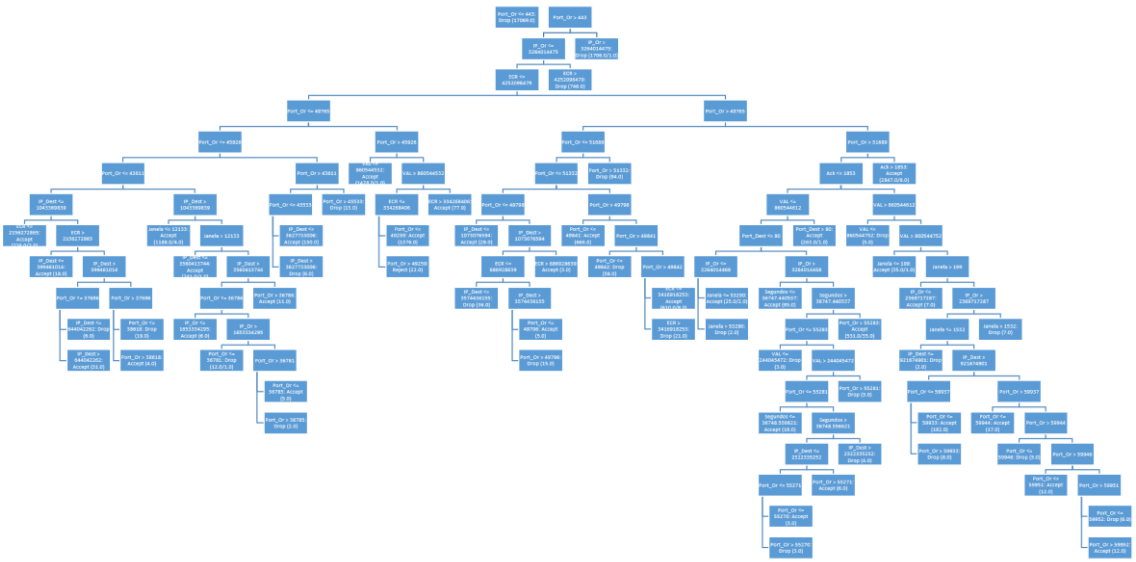

Figura 59 – Teste 4, árvore de decisão obtida

Conforme se pode observar, mais uma vez, com a árvore ideal não é fácil ter uma perceção de quais são os critérios que estão a ser utilizados, pois a árvore tem muitos nós. Assim, por forma a ser possível visualizar os critérios, obrigou-se novamente a que cada

nó tenha pelo menos 1000 objetos, o que diminuiu o número de nós para apenas 7 e ligeiramente a taxa de acerto, para 83.8%. Obteve-se a árvore seguinte:

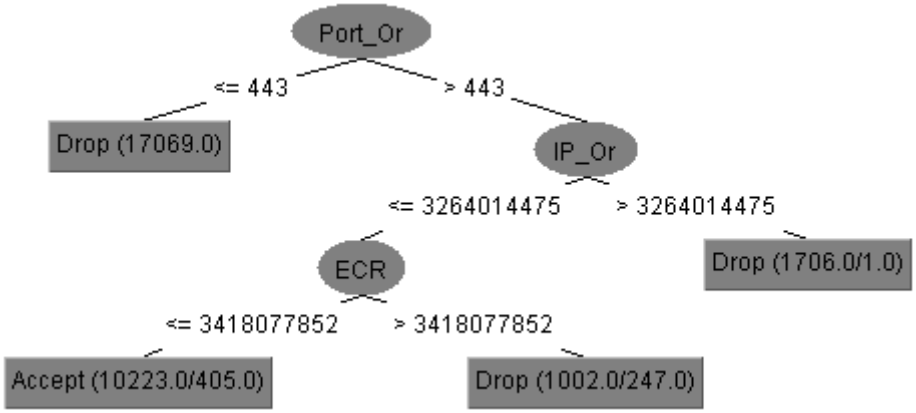

Figura 60 – Teste 4, árvore de decisão com apenas 7 nós

- **Conclusões**

- Inicialmente surgiu um problema com a classe NOACTION, que foi posteriormente resolvido;
- Os resultados obtidos foram bastante surpreendentes, face à separação temporal que existiu entre ambos os ficheiros;
- O classificador que obteve melhores resultados foi o K-vizinhos (IBK), com 83.97% de instâncias bem classificadas.

5.2.9. Teste 5 – *Big Data*

Por fim fez-se um último teste, com todos os dados, isto é, utilizou-se para treino o ficheiro com os dados correspondentes à $BD2_{CLASS}$ (sem as ligações NOACTION) e para teste o ficheiro com os dados correspondentes à $BD3_{CLASS}$ (sem NOACTION).

A Tabela 41 mostra a distribuição dos dados na $BD2_{CLASS}$ e na $BD3_{CLASS}$.

Conjunto / Classe	$Treino - BD2_{CLASS}$	$Teste - BD3_{CLASS}$
Accept	13691323	3552394
Drop	22867703	176861
Reject	17506	0
Decrypt	10722	0
Total	36587254	3729255

Tabela 41 - Teste 5, dados de treino e teste

- ***Naive de Bayes***

Começou-se por testar os dados com o classificador *naive de Bayes*, obtendo-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 34389484	✓ 93.99%
× Instâncias mal classificadas	× 2197770	× 6.01%

Tabela 42 – Teste 5, instâncias bem e mal classificadas para o método de naive de Bayes

Accept	Drop	Reject	Decrypt	Previsto / Real	Taxa de acerto por classe $(\frac{verdadeiros\ positivos}{total\ classe\ real})$
3412311	58020	82063	0	Accept	96.1%
170325	3424	3112	0	Drop	0.019%
0	0	0	0	Reject	-
0	0	0	0	Decrypt	-

Tabela 43 – Teste 5, matriz de dispersão e taxa de acerto por classe para o método de naive de Bayes

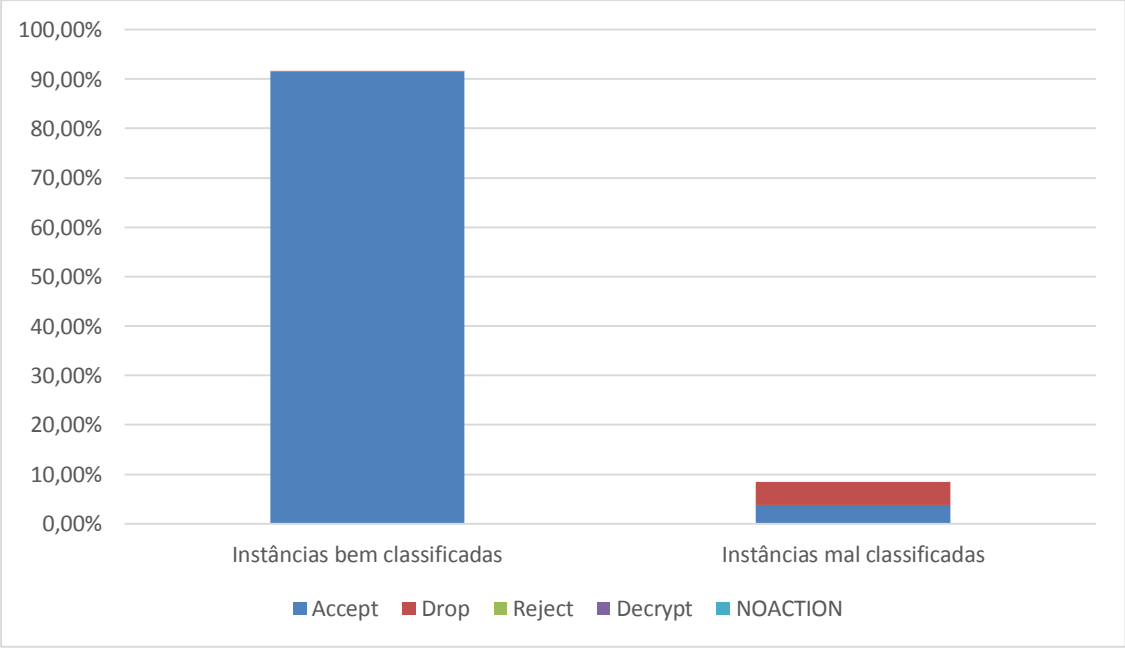

Figura 61 – Teste 5, gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real, para o método de naive de Bayes

- **Árvore de decisão C4.5 (J48)**

Por fim testou-se com as árvores de decisão C4.5. Obtiveram-se os seguintes resultados:

✓ Instâncias bem classificadas	✓ 3119689	✓ 83.65%
× Instâncias mal classificadas	× 609566	× 16.35%

Tabela 44 – Teste 5, instâncias bem e mal classificadas para o método das árvores de decisão (C4.5)

Accept	Drop	Reject	Decrypt	Previsto / Real	Taxa de acerto por classe $(\frac{verdadeiros\ positivos}{total\ classe\ real})$
3079114	462896	10290	94	Accept	86.7%
136227	40575	59	0	Drop	22.9%
0	0	0	0	Reject	-
0	0	0	0	Decrypt	-

Tabela 45 – Teste 5, matriz de dispersão e taxa de acerto por classe para o método das árvores de decisão (C4.5)

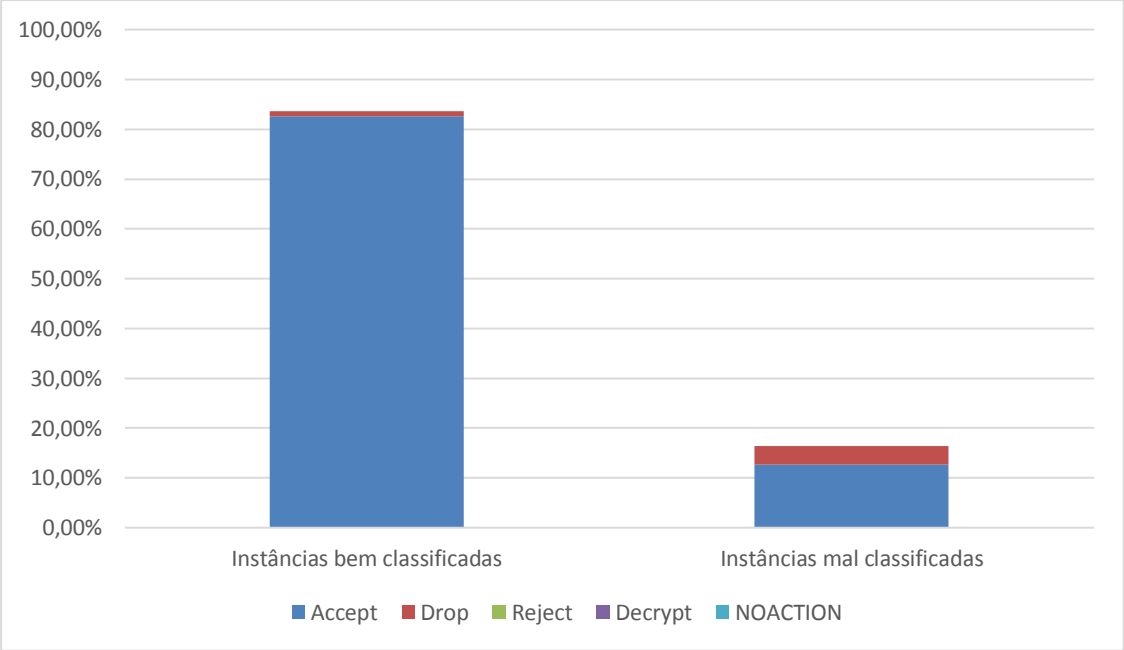

Figura 62 – Teste 5, gráfico com registo numérico das instâncias bem (verdadeiros positivos) e mal classificadas (falsos negativos) por classe real, para o método das árvores de decisão (C4.5)

Uma vez mais, com a árvore ideal é impossível ter uma perceção de quais são os critérios que estão a ser utilizados, pois a árvore tem 29764 nós. Assim, por forma a ser

possível visualizar os critérios, obrigou-se a que cada nó tenha pelo menos 100000 objetos, o que diminuiu o número de nós para apenas 15 e ligeiramente a taxa de acerto, para 96.9%. Obteve-se a árvore seguinte:

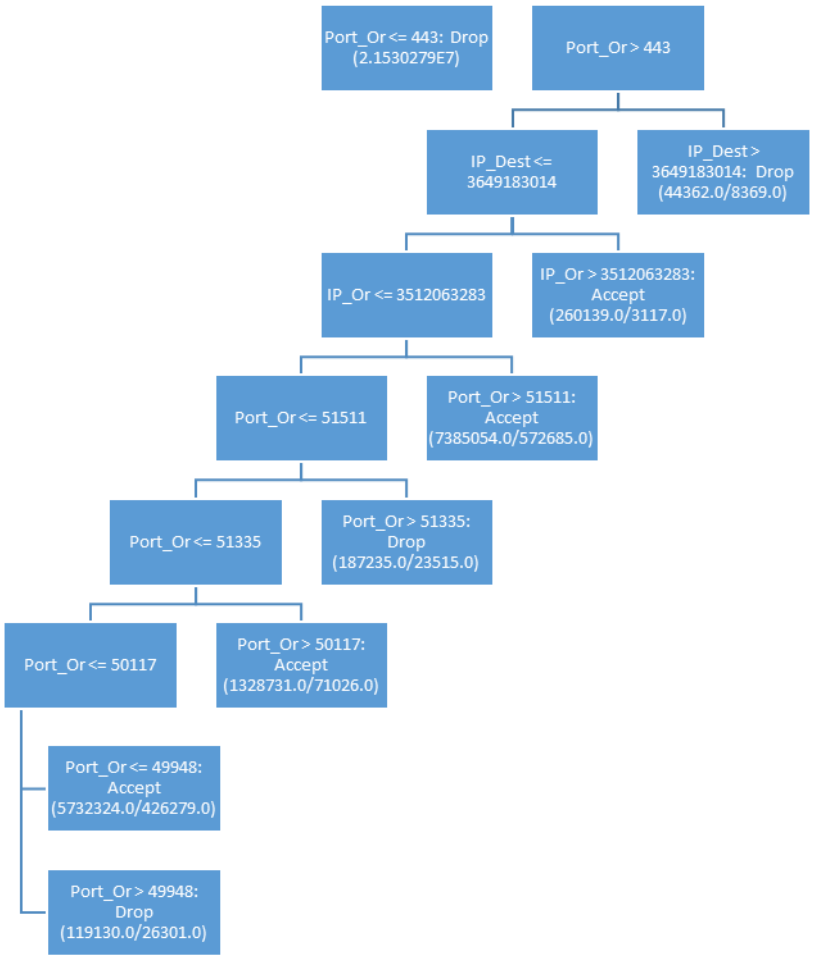

Figura 63 – Teste 5, árvore de decisão com apenas 7 nós

- **Conclusões**

- A grande quantidade de dados em análise não permitiu a utilização da interface gráfica do *software* weka (ver apêndice APN 3), exigiu que se recorressem a "supercomputadores" e ainda assim houve uma considerável demora no processamento dos mesmos, nomeadamente para o método dos K-vizinhos (IBK), motivo pelo qual não o foi possível concluir em tempo útil;
- Conforme seria de esperar, a utilização de um maior número de dados para treino fez com que os resultados melhorassem consideravelmente, face ao teste anterior;
- Apesar de se terem obtido boas taxas de acerto, denota-se que existe um número considerável de ligações Drop que foram classificadas como Accept, o que é normal, uma vez que os ficheiros são demasiado pequenos face à grande distância temporal que os separa;

- O classificador que obteve melhores resultados no geral foi o Naive de Bayes. Contudo, através de uma observação mais detalhada é possível verificar que este classificador não faz uma boa distinção entre classes, motivo pelo qual se considera que o classificador que obteve melhores resultados foi a Árvore de decisão C4.5 (J48).

5.3. Aprendizagem não supervisionada

Para se tentar agrupar dados com ligações semelhantes, optou-se por se recorrer ao algoritmo *Normalized Compression Distance* (NCD), baseado na complexidade de *Kolmogorov*.

5.3.1. Programas utilizados

No âmbito da aprendizagem não supervisionada recorreu-se ao *software complearn*, que é um programa de código aberto que reúne um conjunto de ferramentas utilizadas para processamento, descoberta e aprendizagem (Cilibrasi, Cruz, Rooij, & Keijzer, 2015).

5.3.2. Dados utilizados

Uma vez que aquilo se pretende aqui é que se encontrem relações entre dados semelhantes, as amostras de dados tiveram de ser recolhidas de forma diferente daquela que já havia sido testada durante a aprendizagem supervisionada. Assim sendo, optou-se por se irem recolhendo várias amostras de dados à medida que se iam efetuando testes e retirando conclusões.

5.3.3. Teste 1 – IP de origem 194.140.232.139

Começou-se por se recolherem as 100 primeiras instâncias de cada classe (com exceção da classe NOACTION, por não ser efetivamente uma classe) para o *IP* de origem 194.140.232.139 e retirou-se a informação sobre o mesmo. A Figura 64 ilustra as primeiras três linhas do ficheiro Accept1semip.txt.

```
36752.245149,37831,194.140.232.139,37831,P.,1485272053,2048,244045042,567662656,34
36752.245153,41670,194.140.232.139,41670,.,990583692,5,244045042,3323359366,0
36752.245224,55232,194.140.232.139,55232,.,1128975514,188,860543302,916560073,0
```

Figura 64 - Primeiras 3 linhas do ficheiro Accept1semip.txt

De seguida efetuou-se a compressão e criaram-se os *clusters*. A Figura 65 ilustra os resultados obtidos.

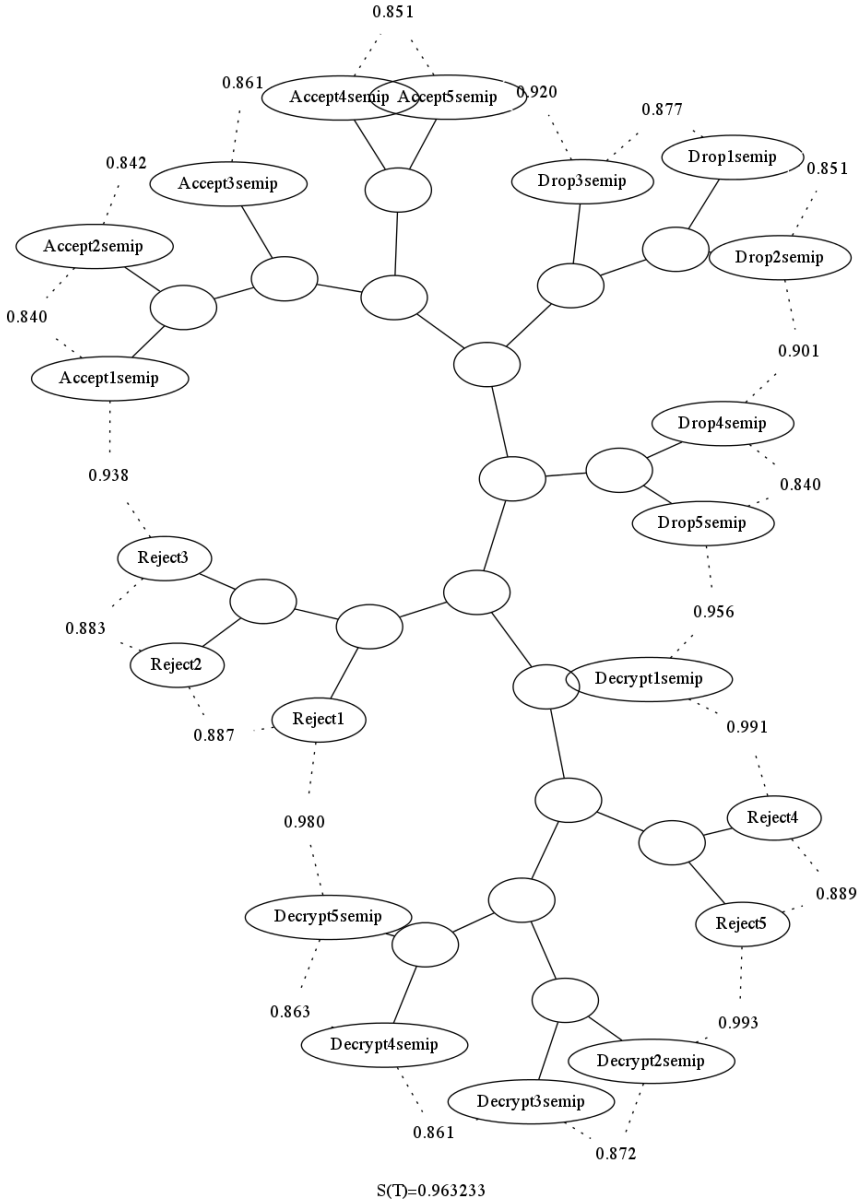

Figura 65 - Resultados obtidos para o Teste 1

Conforme se pode observar, os dados com classificação Accept ficaram todos agrupados num só *cluster*, o que é muito bom, pois significa que o algoritmo os está a distinguir bem dos restantes. Relativamente aos ficheiros Drop, ficaram em dois *clusters*, mas estão bastante próximos, portanto também se obtiveram bons resultados. Os Reject por algum motivo ficaram separados pelo ficheiro Decrypt1. Os restantes Decrypt encontram-se corretamente agrupados num só *cluster*.

Por forma a se tentar perceber se havia algum problema com os ficheiros Decrypt, retiraram-se os mesmos. A Figura 66 ilustra os resultados obtidos.

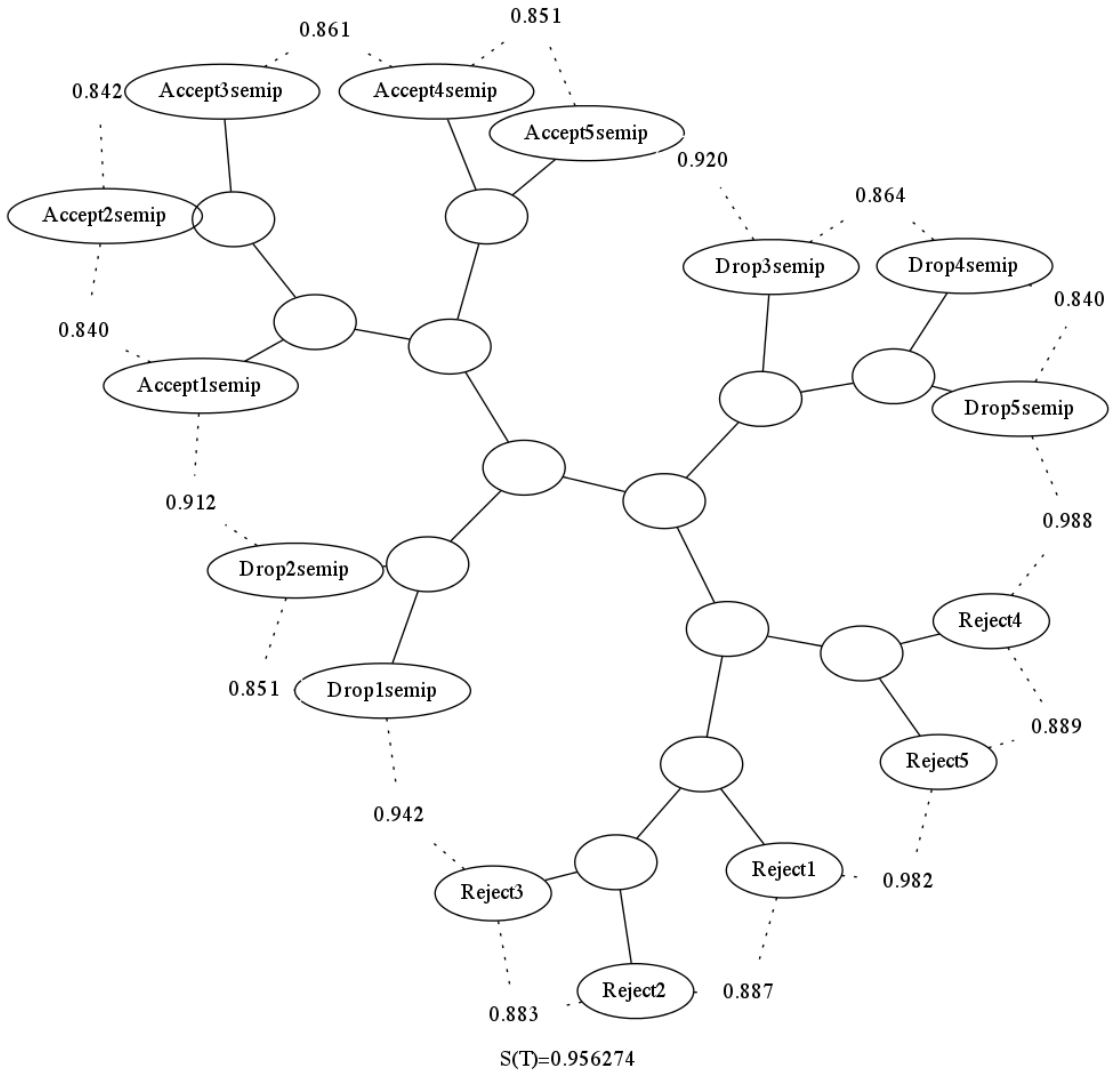

Figura 66 - Resultados obtidos para o Teste 1 (sem ficheiros Decrypt)

Conforme se pode observar, desta vez todos os ficheiros foram bem agrupados, observando-se que existe algum problema com os ficheiros Decrypt.

Para se confirmar se o problema está apenas no ficheiro Decrypt1, voltou-se a fazer o mesmo teste, mas desta vez apenas sem o Decrypt1. A Figura 67 ilustra os resultados obtidos.

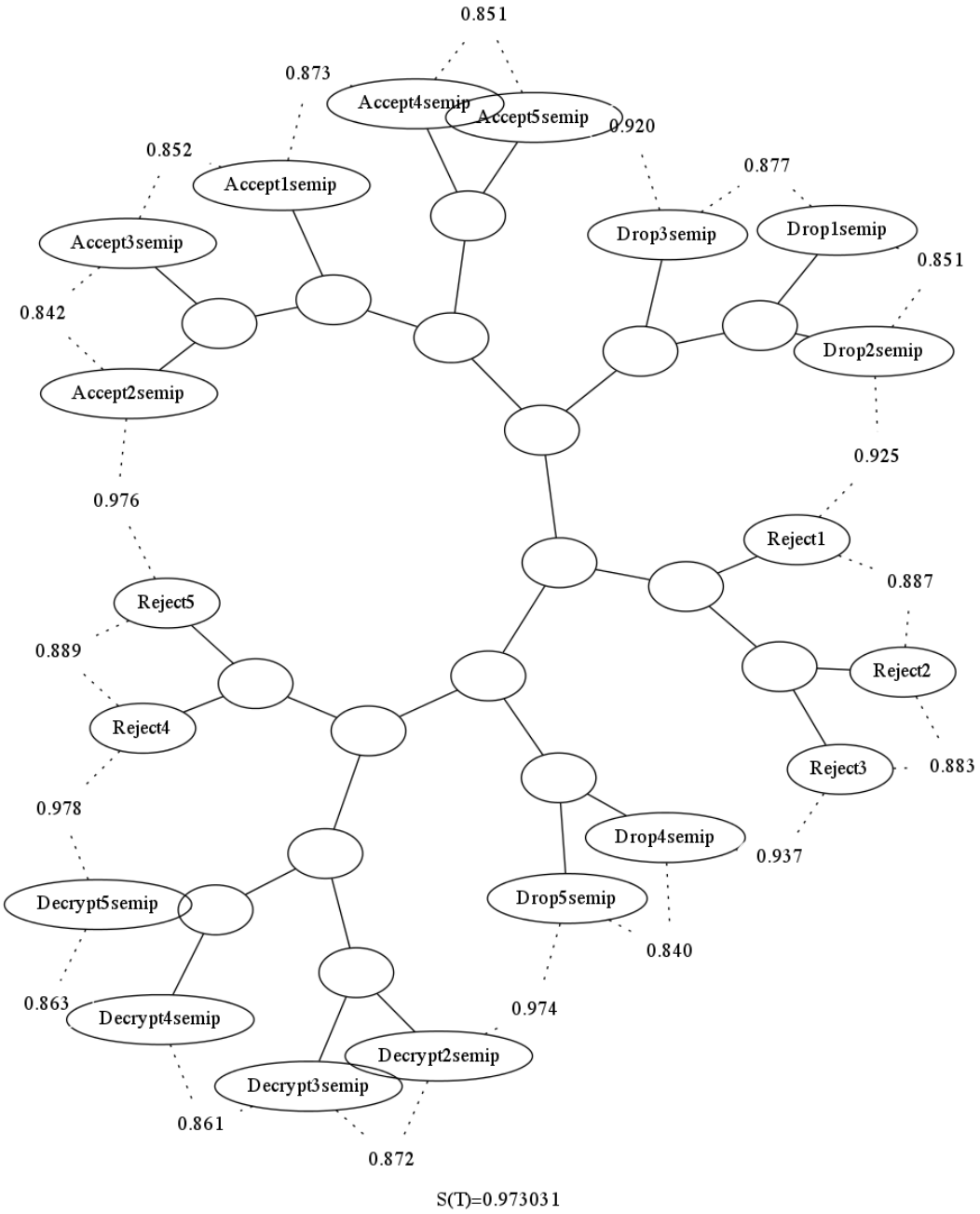

Figura 67 - Resultados obtidos para o Teste 1 (sem ficheiro Decrypt1semip)

Curiosamente, nesta tentativa ficaram dois ficheiros Drop a separar os mesmos ficheiros Reject que estavam anteriormente separados pelo Decrypt1.

Por forma a se tentar entender o que se passou com os resultados anteriores, introduziram-se os mesmos dados, desta vez sem os ficheiros Drop. A Figura 68 ilustra os resultados obtidos.

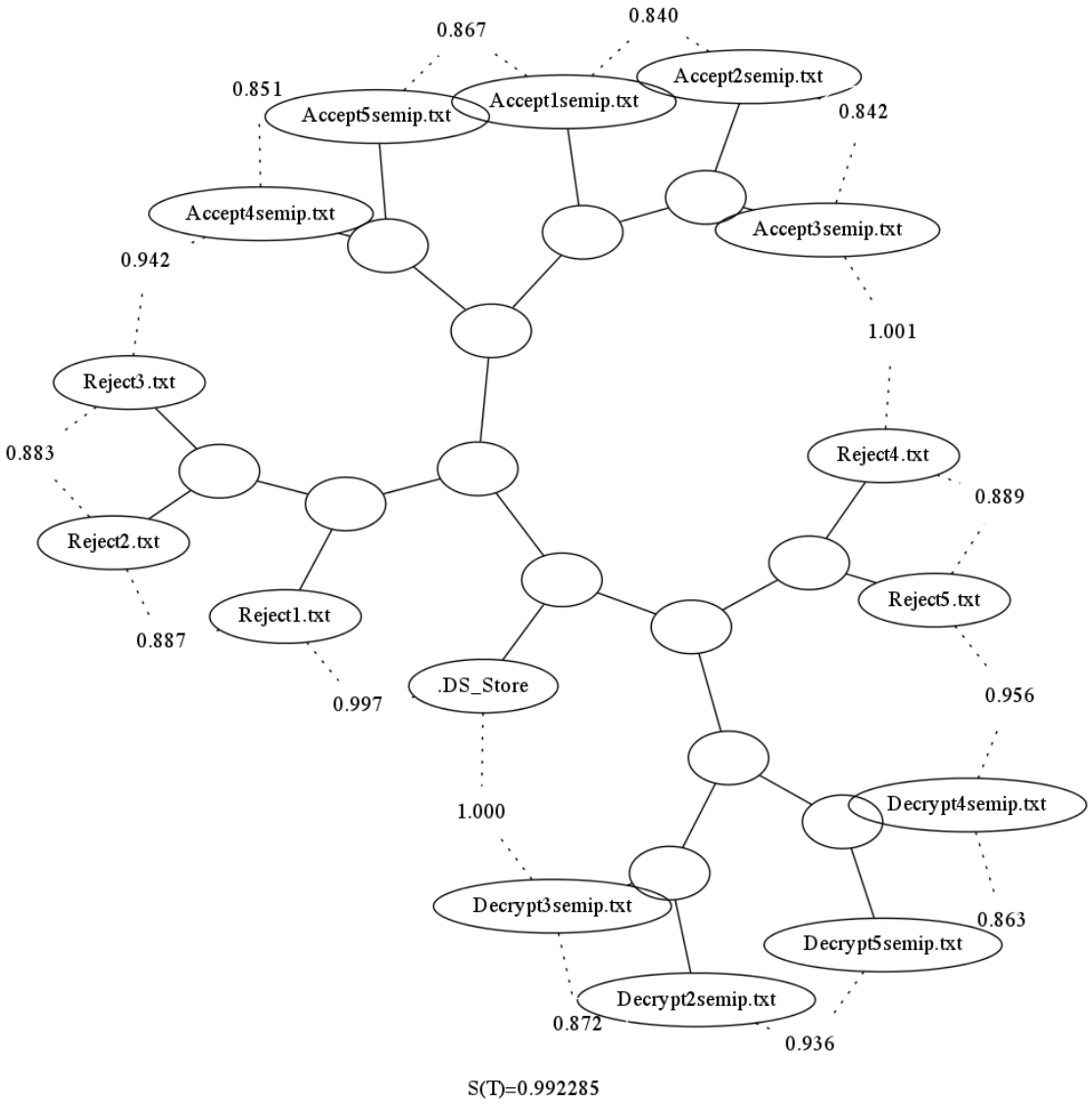

Figura 68 - Resultados obtidos para o Teste 1 (sem ficheiro Decrypt1semip e sem ficheiros Drop)

O ficheiro .DS_Store não deve ser considerado, uma vez que é um ficheiro gerado automaticamente pelo sistema operativo *Mackintosh*. No entanto não deixa de ser curioso que tenha ficado exatamente na separação entre os Rejects. Colocando essa situação de parte, todos os ficheiros ficaram bem agrupados.

- **Conclusões**

- A grande maioria dos dados encontram-se bem agrupados, o que é extremamente positivo, pois demonstra que existem fronteiras naturais a separar os mesmos;
- Existe um problema com o ficheiro Decrypt1, há algo de diferente neste em relação aos restantes Decrypts;

- Existe um problema entre os ficheiros Reject 4 e Reject 5 e os restantes Rejects, há algo que os está a fazer estarem sempre separados.

5.3.4. Teste 2 – Qualquer IP

Após se terem efetuado testes para o mesmo IP, decidiu-se observar se os resultados favoráveis obtidos anteriormente se mantinham. Assim recolheram-se as 100 primeiras instâncias de cada classe (com exceção da classe NOACTION, por não ser efetivamente uma classe). A Figura 69 ilustra as primeiras sete linhas do ficheiro Accept1.txt.

```
36752.245149,194.140.232.139,37831,194.140.232.139,37831,P.,1485272053,2048,244045042,567662656,34
36752.245153,194.140.232.139,41670,194.140.232.139,41670,.,990583692,5,244045042,3323359366,0
36752.245224,194.140.232.139,55232,194.140.232.139,55232,.,1128975514,188,860543302,916560073,0
36752.245227,194.140.232.139,55232,194.140.232.139,55232,.,2,192,860543302,916560073,0
36752.245244,194.140.232.139,56266,194.140.232.139,56266,.,2073191264,1017,244045042,755697458,0
36752.245277,194.140.232.139,55232,194.140.232.139,55232,F.,2,192,860543302,916560073,0
36752.245293,85.139.148.163,49559,85.139.148.163,49559,.,2775500640,16425,NOVAL,NOECR,0
```

Figura 69 - Primeiras 3 linhas do ficheiro Accept1semip.txt

De seguida efetuou-se a compressão e criaram-se os *clusters*. A Figura 70 ilustra os resultados obtidos.

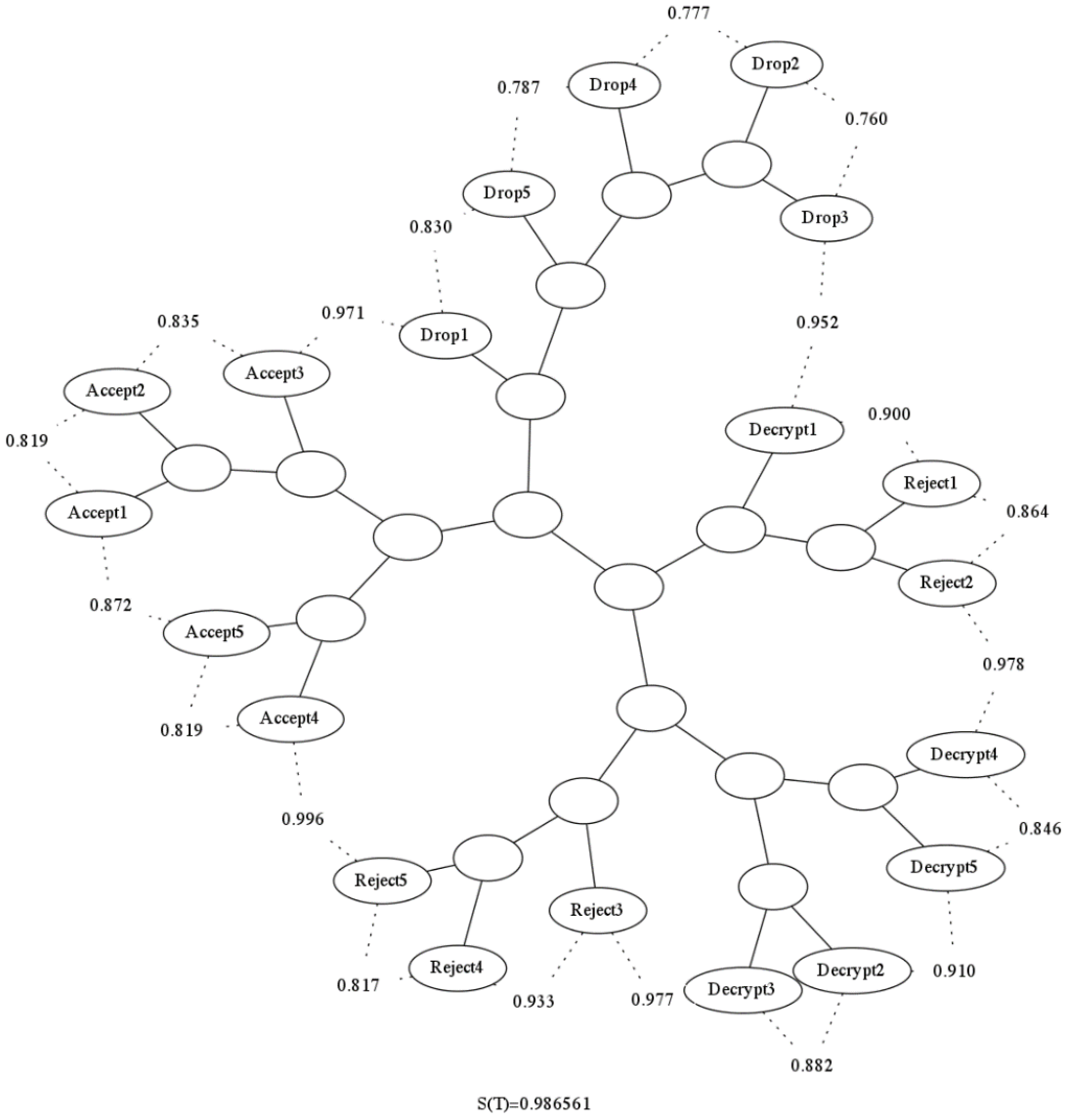

Figura 70 - Resultados obtidos para o Teste 2

Conforme se pode observar, os problemas continuam a ser exatamente os mesmos obtidos anteriormente. Com exceção desses, os resultados mantêm-se, o que é bastante positivo.

Voltou-se então novamente a retirar o Decrypt1. A Figura 71 ilustra os resultados obtidos.

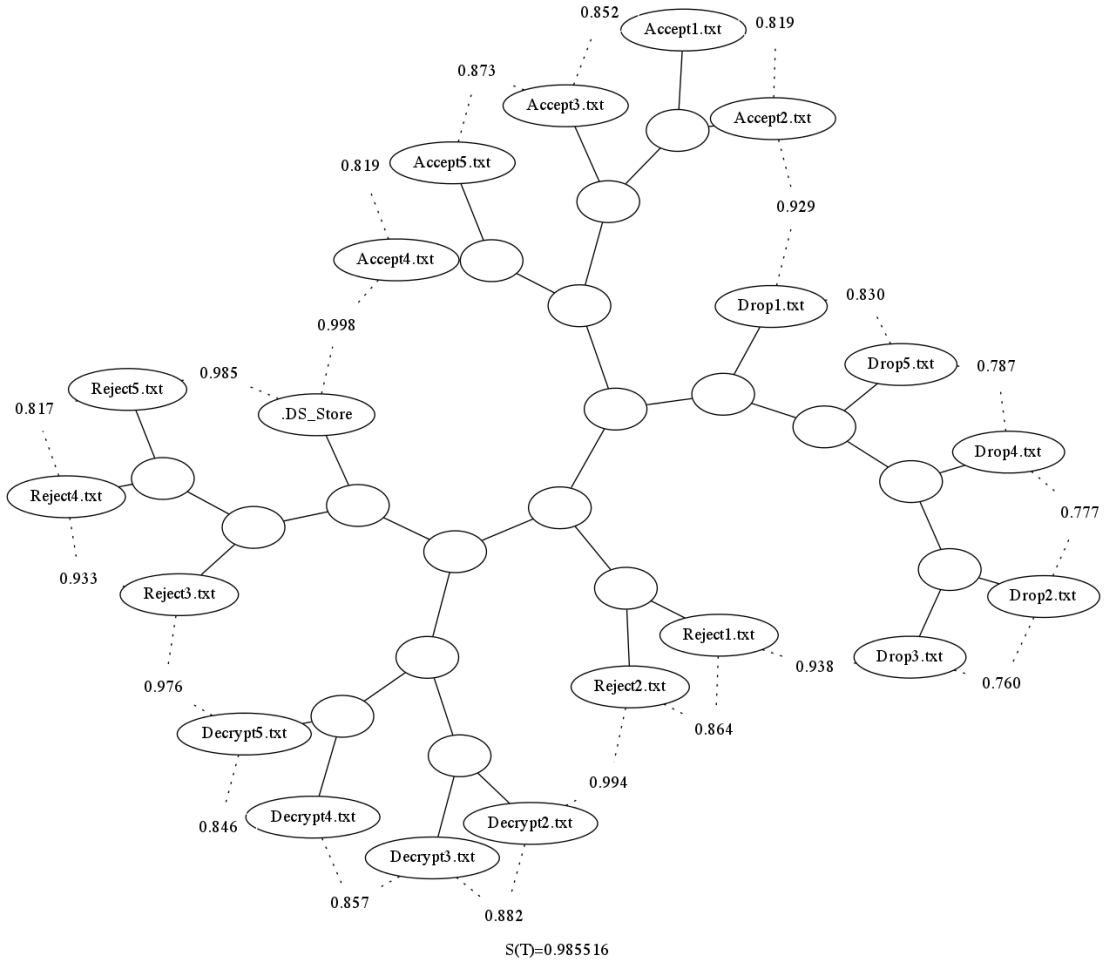

Figura 71 - Resultados obtidos para o Teste 2 (sem ficheiro Decrypt1)

Mais uma vez os problemas são exatamente os mesmos, os ficheiros Reject continuam a aparecer separados.

- **Conclusões**

➢ Capturarem-se ficheiros apenas com o mesmo *IP* de origem ou qualquer *IP* não tem grande influência.

5.3.5. Teste 3 – Com ficheiros de teste

Com os resultados obtidos anteriormente poder-se-ia novamente questionar se estes resultados se manteriam no futuro. Assim decidiu-se efetuar um teste com os mesmos dados que o Teste 2, contudo acrescentando-se um ficheiro com as últimas 100 instâncias de cada classe, ou seja com um intervalo de tempo superior a 51 minutos. A Figura 72 ilustra as primeiras três linhas do ficheiro ultimos100Accept.txt.

```
39827.988599,194.140.232.139,43232,194.140.232.139,43232,.,24797001,2025,247120302,1915703595,0
39827.988605,194.140.232.139,43232,194.140.232.139,43232,.,24798449,2025,247120302,1915703595,0
39827.988615,194.140.232.139,43232,194.140.232.139,43232,.,24799897,2025,247120302,1915703595,0
```

Figura 72 - Primeiras 3 linhas do ficheiro ultimos100Accept.txt

De seguida efetuou-se a compressão e criaram-se os *clusters*. A Figura 73 ilustra os resultados obtidos.

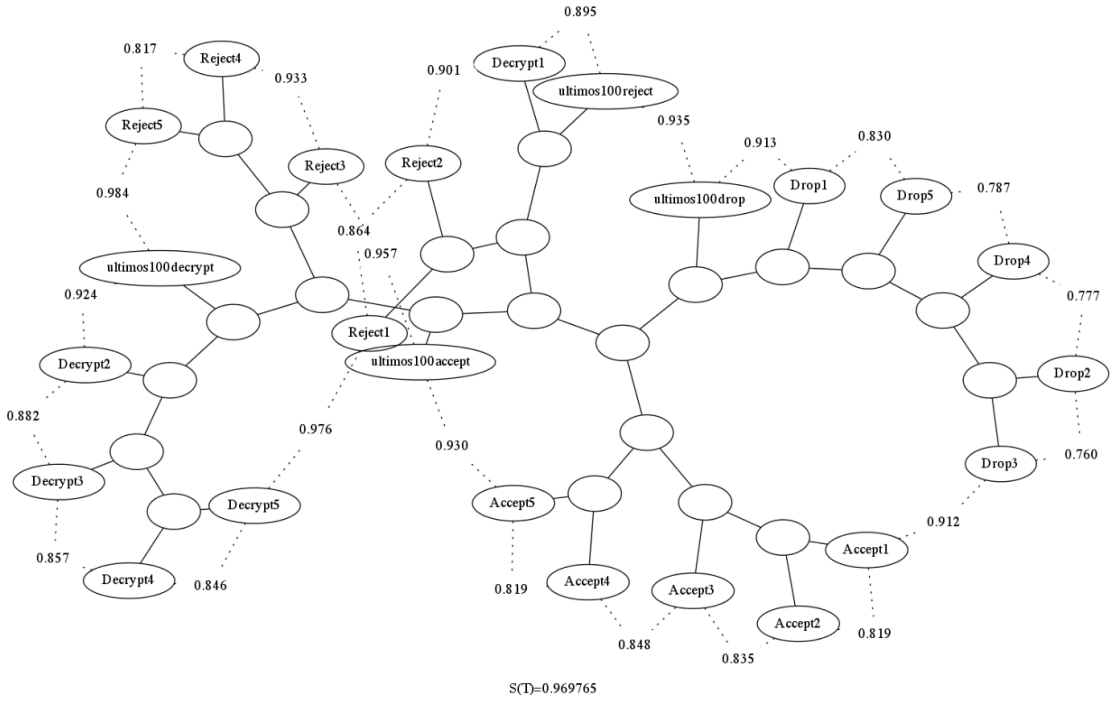

Figura 73 - Resultados obtidos para o Teste 3

Na ilustração obtida é possível observar que os ficheiros Drop se encontram extremamente bem agrupados num só *cluster*. No que toca aos Decrypt também se encontram bem agrupados, com exceção do Decrypt1, que continua a ser separado e agrupado próximo dos Reject. Os Reject, apesar de estarem em *clusters* diferentes, estão todos bastante próximos. Por fim, os Accept continuam bem agrupados, mas o ficheiro de teste, apesar de próximo, não se encontra no mesmo *cluster*.

Voltou-se então novamente a retirar o Decrypt1. A Figura 74 ilustra os resultados obtidos.

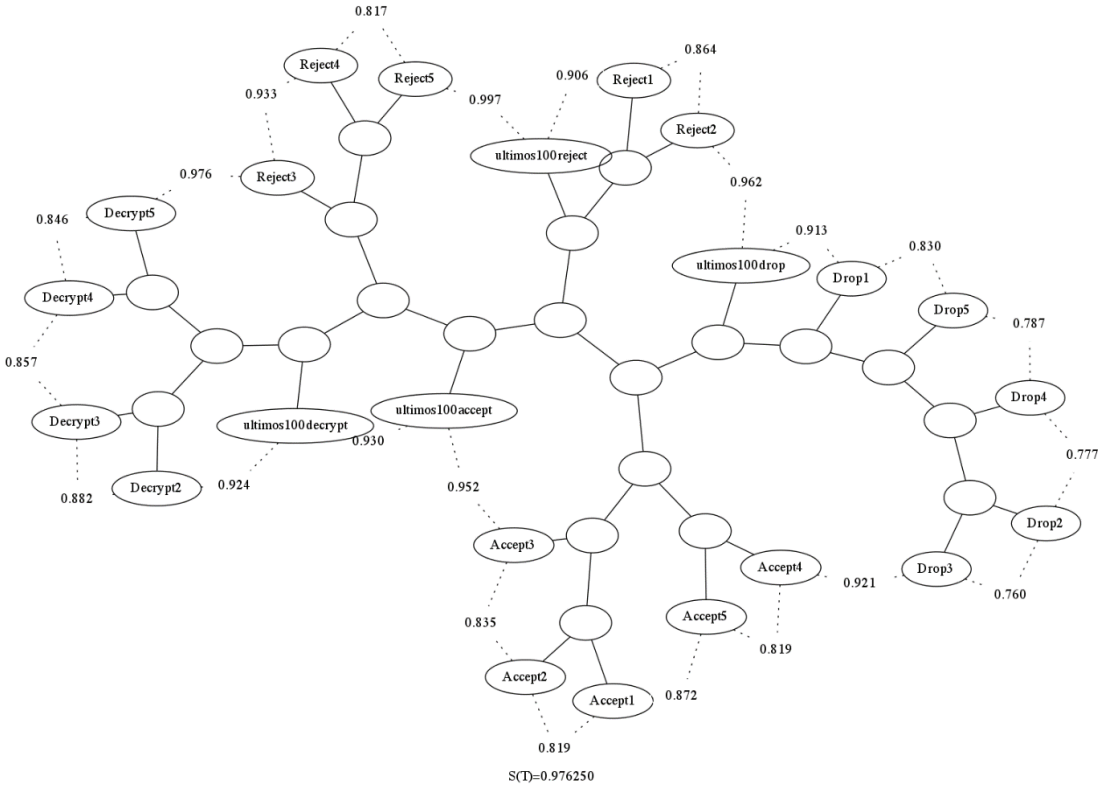

Figura 74 - Resultados obtidos para o Teste 3 (sem ficheiro Decrypt1)

Sem o ficheiro Decrypt1, os Decrypt já se encontram todos no mesmo *cluster*. Os Reject encontram-se separados pelo ficheiro de teste ultimos100Accept, que continua mal agrupado, apesar de próximo do seu *cluster*.

- **Conclusões**

- Apesar da pequeníssima quantidade de dados de treino face ao tempo que passou e ao número de dados testados (repare-se que se tem um intervalo de mais de 51 minutos, que corresponde a mais de 45 milhões de instâncias), os resultados obtidos foram bastante satisfatórios.
- Continua a confirmar-se que o ficheiro Decrypt1 contém ligações estranhas, face à classificação que lhe foi atribuída pela *firewall* de perímetro.
- Os ficheiros Reject também não estão totalmente em concordância.
- Infelizmente o ficheiro de teste da classificação Accept não ficou bem agrupado, motivo pelo qual será necessário testarem-se mais ficheiros.

5.3.6. Teste 4 – Treinando com mais ficheiros de cada classe

Após se concluírem os testes anteriores, questionou-se quais os resultados obtidos se se introduzisse uma maior quantidade de dados de "treino". Assim sendo, optou-se por

inicalmente se introduzirem 10 ficheiros de cada classe, e mais 1 ficheiro de cada classe correspondente às últimas 100 linhas. A Figura 75 ilustra os resultados obtidos.

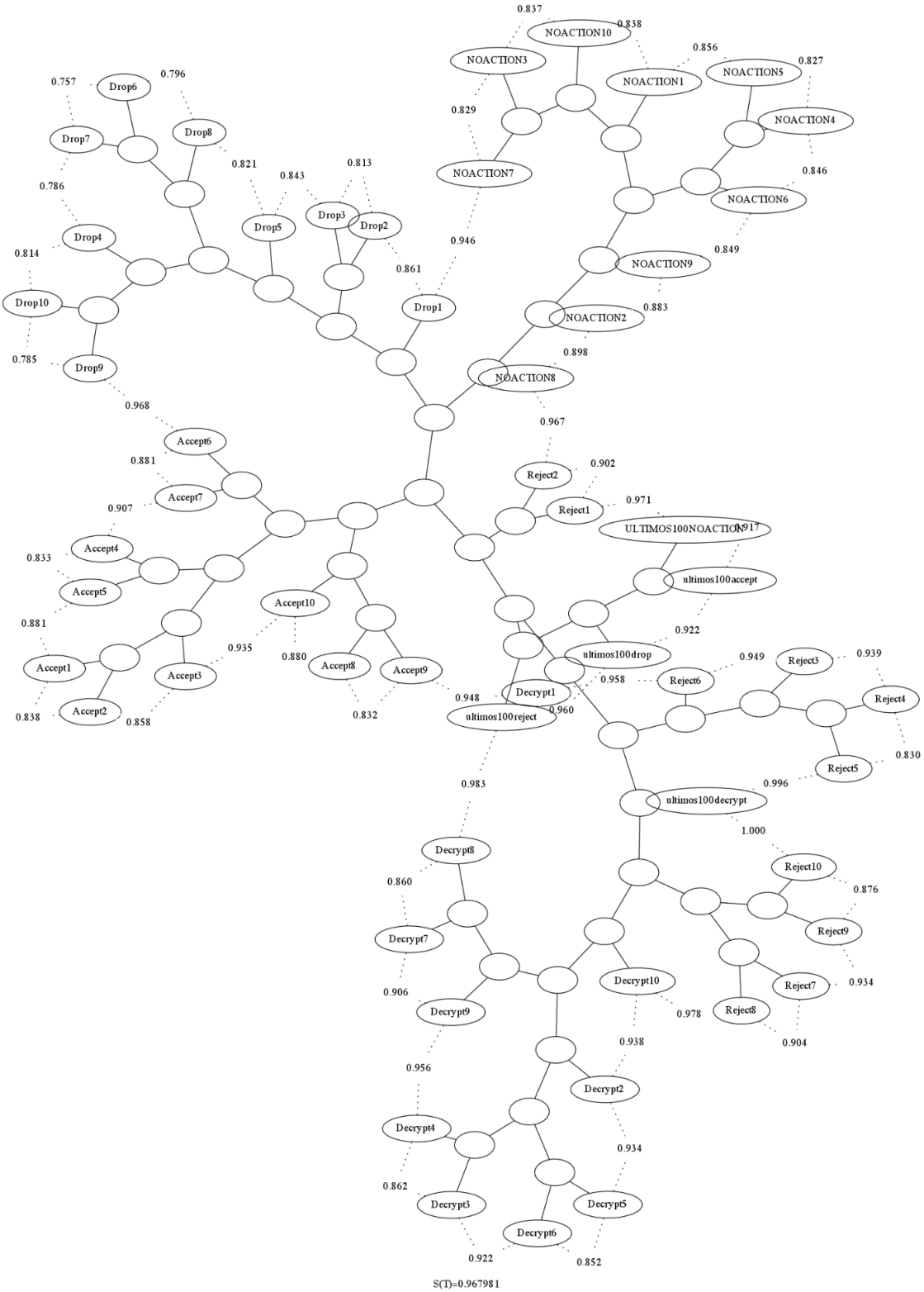

Figura 75 - Resultados obtidos para o Teste 4 (10 ficheiros + 1 por classe)

Conforme é possível observar, os Drops, NOACTION e Accept encontram-se corretamente agrupados, num só *cluster*. Contudo os últimos ficheiros de cada ficaram agrupados apenas num *cluster*. Com exceção do Decrypt1, que voltou a dar problemas, os Decrypts encontram-se bem agrupados, estando inclusive próximo do ficheiro ultimos100decrypt. Por outro lado os Reject ficaram partidos em 3 grupos.

Por fim, fez-se ainda um último teste utilizando 20 ficheiros para cada classe e mais um de cada classe com as últimas 100 instâncias de cada classe. A Figura 76 ilustra os resultados obtidos.

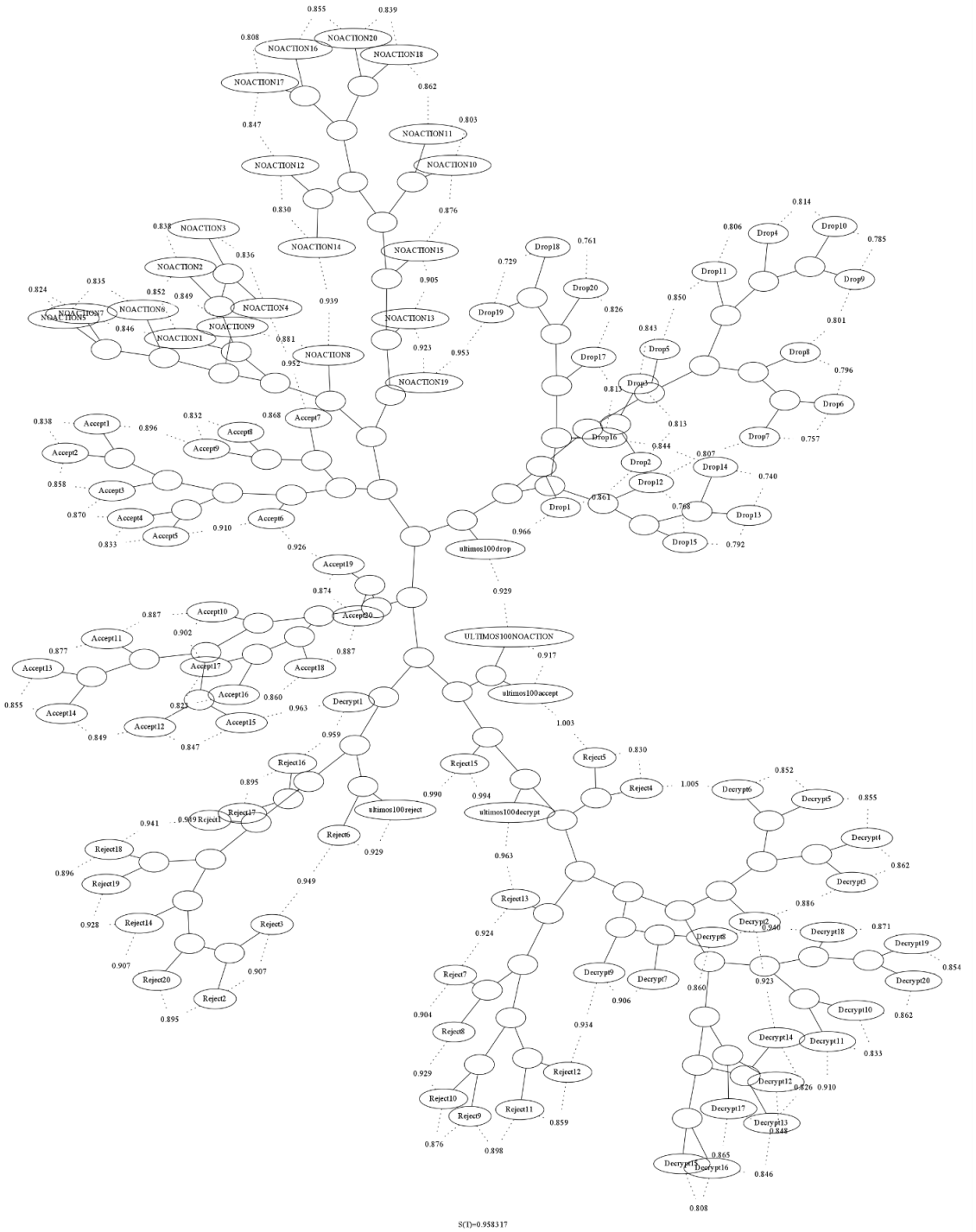

Figura 76 - Resultados obtidos para o Teste 4 (20 ficheiros + 1 por classe)

Observa-se que os ficheiros Drop se encontram todos corretamente classificados, inclusive o último já é agrupado no mesmo *cluster*. Os NOACTION encontram-se todos corretamente agrupados, contudo o ficheiro ULTIMOS100NOACTION encontra-se noutro *cluster*, com alguma distância entre os restantes NOACTION. Os Accept estão divididos em dois *clusters*, contudo ambos próximos. O último ficheiro da classe

encontra-se agrupado à parte. Os Decrypt, uma vez mais com excessão do Decrypt1 encontram-se agrupados. O ficheiro ultimos10decrypt encontra-se juntamente com alguns Reject. Os Reject continuam divididos em dois grupos, um mais próximo do Accept e outro mais próximo do Decrypt. O último desta classe encontra-se agrupado com o grupo que está perto do Accept.

- **Conclusões**

- Este teste revela que este ficheiro Decrypt1 continua a ser suspeito, pois tem registos que o diferem dos restantes (verifica-se a propriedade da complexidade de Kolmogorov de ser independente da máquina utilizada).
- Obtém-se relativamente bons resultados ao agrupar grandes quantidades de ficheiros, contudo a separação temporal entre ambos faz com que seja mais difícil o seu agrupamento, motivo pelo qual se observa que os últimos100 se encontram mal agrupados na maioria dos casos.

Capítulo 6 - Comentários finais

6.1. Contribuições do trabalho e conclusões

Nesta investigação foram abordados e testados vários métodos de *data mining* aplicados a registos de ligações na rede da marinha portuguesa.

Começou-se por dar uma breve introdução às redes de computadores e à cibersegurança, onde se pôde concluir que atualmente a sociedade se encontra bastante dependente das mesmas, não sendo a marinha uma exceção. Contudo apesar de úteis e necessárias, estas constituem uma vulnerabilidade que alguns indivíduos, a título pessoal ou organizacional, se encontram a tentar explorar. Neste capítulo pôde-se ainda concluir que a tecnologia de segurança atual tem duas grandes fraquezas: não é capaz de dar resposta a todo o tipo de ameaças e tem de estar constantemente a ser atualizada, tendo pouca capacidade de responder a novas ameaças.

De seguida explorou-se uma tecnologia relativamente recente e em expansão, a prospeção de dados, mais conhecida por *data mining*. Verificou-se que a prospeção de dados tem duas grandes famílias de problemas: problemas de aprendizagem supervisionada e não supervisionada. Na primeira existe um professor que ensina, dando exemplos do que é x e o que é y. Já na segunda tal não se aplica, o algoritmo tem de aprender a distinguir x de y por observação e comparação.

No capítulo 3 introduziu-se aquilo que seria a parte prática desta investigação, mostrando-se quais os dados que foram utilizados, como foram obtidos, qual o seu conteúdo e como foram pré-processados. Como se pôde observar, trabalhar com grandes quantidades de dados raramente é fácil, requer grande capacidade de processamento, bastante conhecimento de programação e muitíssimo tempo.

No último capítulo, foram apresentados os tão desejados testes. Foi aqui que foi possível colocar a teoria em prática e verificar se era possível recorrer-se à prospeção de dados para detetar ameaças informáticas na rede. Os resultados foram bastante surpreendentes e consideravelmente acima das espectativas. No que toca à abordagem supervisionada, verificou-se que inicialmente o classificador se encontrava a sobreaprender. Por este motivo tiveram de se recolher dados novos, o que diminuiu ligeiramente as taxas de acerto, continuando no entanto a ser extraordinárias.

Relativamente à abordagem não supervisionada, ficou demonstrado que é possível agrupar ligações com bastante exatidão, tendo em conta apenas as distâncias entre os mesmos.

Ao se fazer uma retrospetiva considera-se que todos os objetivos foram cumpridos e foi ainda possível ir mais além do que aquilo a que se tinha proposto no início desta investigação, pois para além da obtenção de um classificador capaz de identificar rapidamente quais os dados de tráfego que devem ser classificados como ameaças, exploraram-se ainda técnicas de análise não supervisionadas, o que possibilita que se detetem ligações estranhamente classificadas desse modo, possivelmente suspeitas.

Esta dissertação permitiu ao autor adquirir novos conhecimentos no âmbito das redes e da cibersegurança, aplicar e aprofundar muitos outros já adquiridos e ainda explorar alguns "mares nunca dantes navegados" (Camões, 1572). Por tudo isto e por se observar que esta é uma área ainda pouco explorada em Portugal e menos ainda na marinha portuguesa, se considera esta investigação de elevada importância e pertinência.

6.2. Trabalho futuro

Os bons resultados alcançados na presente investigação demonstraram uma vez mais o grande potencial da análise de grandes quantidades de dados. No entanto, considera-se de interesse futuro melhorar os resultados obtidos.

No âmbito da aprendizagem supervisionada, tal poderia ser feito através da repetição do método seguido, recorrendo-se no entanto à realização de uma *deep analisys*, o que permitiria ao novo classificador ser igual ou ainda melhor que o já atualmente existente.

Relativamente à aprendizagem não supervisionada, seria interessante tentar elaborar um sistema baseado em regras, que funcionasse como um classificador e que recorresse a métodos de *clustering* para classificar os dados.

Considera-se também que seria aliciante efetuar uma investigação semelhante, recorrendo-se a outros métodos de análise. Sugere-se, no final, uma comparação com os resultados aqui alcançados.

É ainda importante ressaltar que os métodos utilizados nesta investigação poderão ser aplicados nas mais variadas áreas, motivo pelo qual também se considera interessante investigar novas aplicações para os mesmos.

Referências

Aldrich, C., & Auret, L. (2013). *Unsupervised process monitoring and fault diagnosis with machine learning methods*: Springer.

Amor, N. B., Benferhat, S., & Elouedi, Z. (2004). *Naive bayes vs decision trees in intrusion detection systems.* Paper presented at the Proceedings of the 2004 ACM symposium on Applied computing.

Anubisnetworks. (2015). Appliance and VM solutions for Service Providers and Enterprises. Retrieved 04/06/2015, from https://www.anubisnetworks.com/products/email-security/mail-protection-systems

Arquilla, J., & Ronfeldt, D. (1993). Cyberwar is coming! *Comparative Strategy, 12*(2), 141-165.

Barbara, D., Couto, J., Jajodia, S., Popyack, L., & Wu, N. (2001). *ADAM: Detecting intrusions by data mining.* Paper presented at the In Proceedings of the IEEE Workshop on Information Assurance and Security.

Bastos, H., & Albini, L. (2012). Mecanismo de deteção de ataques Wormhole em redes tolerantes a atrasos e desconexões. *Simpósio brasileiro de telecomunicações*.

Borman, D., Scheffenegger, R., & Jacobson, V. (2014). TCP Extensions for High Performance. doi: urn:ietf:rfc:7323

Brandt, S. S. (2006). Maximum likelihood robust regression by mixture models. *Journal of Mathematical Imaging and Vision, 25*(1), 25-48.

Camões, L. d. (1572). Os lusíadas (1572).

Cegalla, D. P. (2008). *Dicionário escolar da língua portuguesa*: Companhia Editora Nacional.

Chowdhury, N. M. K., & Boutaba, R. (2010). A survey of network virtualization. *Computer Networks, 54*(5), 862-876.

Christodorescu, M., Jha, S., Seshia, S., Song, D., & Bryant, R. E. (2005). *Semantics-aware malware detection.* Paper presented at the Security and Privacy, 2005 IEEE Symposium on.

Cilibrasi, R., Cruz, A. L., Rooij, S., & Keijzer, M. (2015). CompLearn Home.

Cilibrasi, R., Vitányi, P., & De Wolf, R. (2004). Algorithmic clustering of music based on string compression. *Computer Music Journal, 28*(4), 49-67.

Collin, S. (2002). *Dictionary of Computing* (5 ed.): Evans Bros.

Comer, D. E. (2009). *Computer networks and internets* (L. o. C. Cataloging Ed. 5 ed.). New Jersey: Prentice Hall Press.

Damasceno, M. (2005). Introdução a Mineração de Dados Utilizando o Weka. *Instituto AAAFederal de Educação, Ciência e Tecnologia do Rio Grande do Norte*.

Domingos, H. (2012). Security Services for Routing Protocols in Wireless Sensor Networks.

Dua, S., & Du, X. (2011). *Data mining and machine learning in cybersecurity*: CRC press.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). Pattern classification. 2nd. *Edition. New York*.

Duffy, J., & Cram, C. (2007). *Microsoft Office Word 2007, Illustrated Complete*: Cengage Learning.

Duke, D. (2002). What is the difference between Denial-of-Service (DoS) and Distributed-Denial-of-Service (DDoS)? *Network Security, 5*(2002), 4.

Efron, B., & Tibshirani, R. J. (1994). *An introduction to the bootstrap*: CRC press.

Farlex. (2015). Aliases.

Fisher, D. H., Pazzani, M. J., & Langley, P. (2014). *Concept formation: Knowledge and experience in unsupervised learning*: Morgan Kaufmann.

Freitas, A. A. (2013). *Data mining and knowledge discovery with evolutionary algorithms*: Springer Science & Business Media.

Gama, J., & Brazdil, P. (1995). Characterization of classification algorithms *Progress in Artificial Intelligence* (pp. 189-200): Springer.

Glover, I., & Grant, P. M. (2010). *Digital communications*: Pearson Education.

Gupta, G. (2011). *Introduction to data mining with case studies*: PHI Learning Pvt. Ltd.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter, 11*(1), 10-18.

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: concepts and techniques: concepts and techniques*: Elsevier.

Hand, D. J., Mannila, H., & Smyth, P. (2001). *Principles of data mining*.

Hartigan, J. (1983). Bayes theory.

Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences, 44*(1), 1-12.

Hristea, F. T. (2012). *The Naïve Bayes Model for Unsupervised Word Sense Disambiguation: Aspects Concerning Feature Selection*: Springer Science & Business Media.

IDN12. (2013). *Estratégia da Informação e Segurança no Ciberespaço*. Lisboa: Instituto da Defesa Nacional.

Ifrah, G., Harding, E. F., Bellos, D., & Wood, S. (2000). *The universal history of computing: From the abacus to quantum computing*: John Wiley & Sons, Inc.

Jacobson, V., Braden, R., & Borman, D. (1992). TCP Extensions for High Performance. doi: urn:ietf:rfc:1323

Jacobson, V., & Braden, R. T. (1988). TCP extensions for long-delay paths. doi: urn:ietf:rfc:1072

Kantardzic, M. (2011). *Data mining: concepts, models, methods, and algorithms*: John Wiley & Sons.

Kim, W., Jeong, O.-R., Kim, C., & So, J. (2011). The dark side of the Internet: Attacks, costs and responses. *Information systems, 36*(3), 675-705.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection* (Vol. 1).

Larose, D. T. (2005). *Discovering knowledge in data: an introduction to data mining*: John Wiley & Sons.

Lee, J., Kim, W., Lee, S.-J., Jo, D., Ryu, J., Kwon, T., & Choi, Y. (2007). *An experimental study on the capture effect in 802.11 a networks.* Paper presented at the Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization.

Leite, R. M. S. R. (2007). Selecçao de Algoritmos de Classificaçao.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval *Machine learning: ECML-98* (pp. 4-15): Springer.

Li, L. (2014). 3 - Computer networks in academic learning environments. In L. Li (Ed.), *Scholarly Information Discovery in the Networked Academic Learning Environment* (pp. 61-91). Oxford: Chandos Publishing.

Li, M., Chen, X., Li, X., Ma, B., & Vitányi, P. (2004). The similarity metric. *Information Theory, IEEE Transactions on, 50*(12), 3250-3264.

Li, M., & Vitányi, P. (2008). An introduction to Kolmogorov complexity and its applications.

Liu, H., & Motoda, H. (2012). *Feature selection for knowledge discovery and data mining* (Vol. 454): Springer Science & Business Media.

Lobo, V. J. d. A. e. S. (2002). *Ship Noise Classification.* (Doctor), New University of Lisbon.

Lorena, S. (2010). A Aliança vai entrar no quinto domínio. *Público*.

MBA, D. E. M., & Hetico, H. R. (2007). *Dictionary of Health Information Technology and Security*: Springer Publishing Company.

Microsoft. (2013). Parts of computer. Retrieved from Windows website: http://windows.microsoft.com/en-us/windows/computer-parts#1TC=windows-7

Militão, O. P. (2014). Guerra da Informação: a cibersegurança, a ciberdefesa e os novos desafios colocados ao sistema internacional.

Nielsen, M. A., & Chuang, I. L. (2010). *Quantum computation and quantum information*: Cambridge university press.

ORTEGA Y GASSET, J. (2010). Eu sou eu e minha circunstância. *ESCÁMEZ SÁNCHEZ, Juan. Ortega y Gasset. Recife: Fundação Joaquim Nabuco*, 111-119.

Pellerin, C. (2010). Defense.gov News Article: Lynn: Cyberspace is the New Domain of Warfare. *U. S. Department of Defense, 2015*.

Peterson, L. L., & Davie, B. S. (2007). *Computer networks: a systems approach*: Elsevier.

Pióro, M., & Medhi, D. (2004). *Routing, flow, and capacity design in communication and computer networks*: Elsevier.

Pouwelse, J., Garbacki, P., Epema, D., & Sips, H. (2005). The bittorrent p2p file-sharing system: Measurements and analysis *Peer-to-Peer Systems IV* (pp. 205-216): Springer.

Pressman, R. S. (2005). *Software engineering: a practitioner's approach*: Palgrave Macmillan.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning, 1*(1), 81-106.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*: Elsevier.

Rouse, M. (2005). Network Protocols and Security. http://searchsecurity.techtarget.com/definition/probe

Russell, D., & Gangemi, G. (1991). *Computer security basics*: " O'Reilly Media, Inc.".

Saltzer, J. H., Reed, D. P., & Clark, D. D. (1984). End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS), 2*(4), 277-288.

Singhal, A. (2007). *Data warehousing and data mining techniques for cyber security* (Vol. 31): Springer Science & Business Media.

SIQUEIRA, D. d. (1990). História geral da aeronáutica brasileira: Rio de Janeiro: Instituto Histórico Cultural da Aeronáutica.

Sophos. (2015). UTM: Firewall log shows dropped packets with tcpflags="ACK RST" or "ACK FIN". *2015*.

Souto, A. (2014). Traffic analysis based on string compression. *SQIG - Instituto de Telecomunicações Departamento de Matemática - IST - Universidade de Lisboa*.

Tanenbaum, A. (2011). Computer Networks/Andrew S. Tanenbaum, David J. Wetherall.–: Prentice Hall, Cloth.

Te Ming Huang, V. K., & Kopriva, I. (2006). Kernel Based Algorithms for Mining Huge Data Sets: Supervised, Semisupervised, and Unsupervised Learning, volume 17 of Studies in Computational Intelligence: Springer.

Tseng, Y.-C., Ni, S.-Y., Chen, Y.-S., & Sheu, J.-P. (2002). The broadcast storm problem in a mobile ad hoc network. *Wireless networks, 8*(2-3), 153-167.

tshark - The Wireshark Network Analyzer 1.12.2. (2015). *2015*.

Tufféry, S. (2011). *Data mining and statistics for decision making*: John Wiley & Sons.

Tzu, S. (2012). *The art of war*: e-artnow.

Vazirgiannis, M., Halkidi, M., & Gunopulos, D. (2012). *Uncertainty handling and quality assessment in data mining*: Springer Science & Business Media.

Wehner, S. (2005). Analyzing worms and network traffic using compression. *arXiv preprint cs/0504045*.

Witten, I. H., & Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*: Morgan Kaufmann.

Witten, I. H., & Frank, E. (2011). *Data Mining: Practical machine learning tools and techniques*: Morgan Kaufmann.

Wu, B., Chen, J., Wu, J., & Cardei, M. (2007). A survey of attacks and countermeasures in mobile ad hoc networks *Wireless Network Security* (pp. 103-135): Springer.

Zezula, P., Amato, G., Dohnal, V., & Batko, M. (2006). *Similarity search: the metric space approach* (Vol. 32): Springer Science & Business Media.

Zhao, Z. A., & Liu, H. (2011). *Spectral feature selection for data mining*: CRC Press.

If I have seen further than others, it is by standing upon the shoulders of giants.

Sir Isaac Newton, 1643-1727

Glossário

Acknowledge

Sinal que é enviado de um recetor para indicar que a mensagem transmitida foi recebida e que está pronto para a próxima (Collin, 2002, p. 5).

Acknowledge Packet (ACK)

Pacote TCP de *acknowledge*.

Arpanet

Rede original que interconectava os computadores. Foi esta que formou o primeiro protopódio daquilo que viria a ser a atual *internet.* Foi desenvolvida pelo departamento de defesa dos Estados Unidos da América. (Collin, 2002, p. 21).

ASCII

Código que representa caracteres alfanuméricos em código binário (Collin, 2002, p. 22).

Bit

Unidade de código binário (2 dígitos), termo atribuído por John Tukey em 1949. É a unidade eletrónica mais pequena, pode ser considerada como sim/não, ligado/desligado ou ainda 0 ou 1 (MBA & Hetico, 2007, p. 33).

BitTorrent

Consiste num protocolo de *download* de ficheiros onde os ficheiros são divididos em várias partes (na proporção de 1 para 1000 por arquivo). Desta forma, enquanto o utilizador se encontra a efetuar *download* do ficheiro, este encontra-se também a efetuar *upload* das partes já descarregadas, otimizando assim a rede de partilha de ficheiros. Para além disso, este sistema possibilita ainda que cada *peer* maximize a sua própria taxa de download, entrando em contato com os seus *peers* mais adequados (Pouwelse, Garbacki, Epema, & Sips, 2005, p. 206).

Browser

Programa de *software* que é utilizado para navegar pelas *webpages* partilhadas na *internet.* Um *browser* pede ao servidor de *internet* para lhe enviar as informações da página, armazena-as, descodifica-as e mostra-as. (Collin, 2002, p. 47).

Buffer (retentor)

Zona da memória física utilizada para armazenar temporariamente os dados, enquanto o processador não está disponível para os processar (Collin, 2002, p. 47).

Byte

Conjunto de oito bit's (Collin, 2002, p. 50). Assim sendo, tendo em conta que um bit são dois valores, cada byte é capaz de representar $2^8 = 256$ caracteres diferentes.

Cabeçalho/*Header*

Pacote de dados que é enviado antes da informação/dados que fornece informação acerca do endereço de destino e de origem (Collin, 2002, p. 157).

Cache

Secção de memória utilizada com a finalidade de armazenar cópias de dados selecionados para acesso mais rápido (Collin, 2002, p. 51).

Connectionless

Transferência de dados que ocorre entre dois aparelhos que não têm um *link* fixo ou permanente e que portanto podem seguir rotas diferentes entre os dois aparelhos. (Collin, 2002, p. 79).

Cracker

Termo utilizado para designar um indivíduo que pratica a quebra de segurança (*craking*) de um sistema de segurança de forma ilegal ou sem ética (Cegalla, 2008, p. 375).

Declaração/*Statement*

Expressão usada para atribuir uma instrução ou definir um processo. [51] (Collin, 2002, p. 316).

Dispositivo periférico

Peça de *hardware* que se encontra ligada a um sistema de computadores (Collin, 2002, p. 251).

Download

Carregar um programa ou secção de dados a partir de um computador remoto, através de uma rede (Collin, 2002, p. 113).

Driver

Peça de *software* que se encontra situada entre o Windows e um dispositivo periférico, traduzindo as instruções do Windows para uma linguagem que o periférico entenda (Collin, 2002, p. 114).

Efeito de captura

Em teoria se existirem dois sinais emitidos na mesma frequência, apenas o sinal de maior potência conseguirá ser demodulado. Numa rede wireless, esta situação dependerá sempre da potência do sinal, e do timing de chegada das *frames* envolvidas (Lee et al., 2007, p. 19).

Email/Eletronic mail

Sistema de troca de mensagens entre utilizadores numa rede (Collin, 2002, p. 123).

End-to-end principle

Numa rede, as funções específicas da aplicação devem residir nos hospedeiros finais da rede e não nos intermédios, desde que possam ser implementadas completamente e corretamente nos hospedeiros finais (Saltzer, Reed, & Clark, 1984).

[51] Declaração (palavra ou programa).

Ethernet

Arquitetura de interconexão para LAN. Foi padronizada pelo IEEE como 802.3 (Collin, 2002, p. 128).

FDDI

Padrão ANSI[52] para redes de alta velocidade, que usa um cabo de fibra ótica com o formato de um duplo anel, transmitindo assim dados a 100 Mbps[53]. (Collin, 2002, p. 134).

File Transfer Protocol (FTP)

Padrão TCP/IP para transferência de ficheiros entre computadores. É um protocolo de partilha de ficheiros que opera nas layers 5, 6 e 7 do modelo OSI. (Collin, 2002, p. 146).

Frame

Unidade *standard* de informação, constituída por um pacote de 64 a 1518 *bytes* com cabeçalho/dados/*trailer* para marcar o início e final de uma comunicação (Collin, 2002, p. 145; MBA & Hetico, 2007, p. 129).

Gateway

Aparelho que liga duas redes diferentes (Collin, 2002, p. 149).

Hack

Invadir um sistema de computadores com propósitos maliciosos (Collin, 2002, p. 154).

Hacker

Pessoa que efetua *hacks* (Collin, 2002, p. 154).

Handshake

Conjunto de sinais padronizados entre dois dispositivos para garantir que o sistema está a trabalhar corretamente, que o equipamento é compatível e que os dados

[52] Organização americana responsável pela padronização de software e harware (Collin, 2002, p. 17).
[53] Megabytes (10^6 bytes) por segundo.

transferidos estão corretos. Estes sinais incluem os comandos "pronto a receber, "pronto para transmitir" e "dados corretos" (Collin, 2002, p. 155). No que toca ao controlo, este procedimento é bastante importante, pois permite que os computadores comuniquem com dispositivos mais lentos.

Hardware

Consiste no conjunto de elementos físicos que constituem um sistema de computadores. Como exemplo temos o monitor, rato, teclado e disco rígido (Ifrah, Harding, Bellos, & Wood, 2000, pp. 121-122; Microsoft, 2013).

Hash code

Sistema de código derivado dos códigos ASCII, em que são adicionados números de código para as primeiras três letras, resultando num novo número utilizado como código *hash* (Collin, 2002, p. 156).

Hill-climbing

Técnica de otimização matemática que começa com um valor num dado ponto aleatório, testa-o com dois ou mais pontos que estão a uma certa distância do ponto testado e continua a procura a partir do melhor ponto (Koza, 1992, p. 634).

Host

Adjetivo utilizado na gíria informática, que significa providenciar espaço para armazenamento num "computador servidor" para que um utilizador possa armazenar ficheiros ou dados no mesmo. Pode ainda servir para armazenar ficheiros necessários a um *website*. (Collin, 2002, p. 161).

Host computer

Computador usado para escrever e apagar *software* de outro computador; computador numa rede que providencia serviços especiais ou linguagens de programação a todos os utilizadores (Collin, 2002, p. 161).

Host server

Empresa que providencia ligações à *internet* e disponibiliza espaço nos seus computadores para se armazenarem ficheiros com a finalidade de serem acedidos pelos utilizadores dos *websites* (Collin, 2002, p. 161).

Hypertext Transfer Protocol (HTTP)

Conjunto de comandos utilizados por um *browser* para perguntar ao servidor de internet informação acerca de uma determinada página web. (Collin, 2002, p. 162).

Internet

Rede mundial de computadores e de sistemas de interligados, por um código (ex. JAVA, hipertexto), cuja finalidade consiste em trocar informações. Esta rede torna mais fácil a comunicação entre todo o mundo e pode ser feita através de linhas com ou sem fios. Consiste num recurso global compartilhado que não é propriedade ou regulamentado por ninguém (MBA & Hetico, 2007, p. 169).

Internet Protocol (IP)

Parte do protocolo TCP/IP que define como os dados são transferidos numa rede (Collin, 2002, p. 182).

Intranet

Rede privada de computadores dentro de uma organização, que fornece funções semelhantes à *internet*, como correio eletrónico, notícias e acesso à WWW, sem os riscos associados de se tornar informação pública ou ligar simplesmente a organização à rede pública (Collin, 2002, p. 181).

IP *address*/Endereço IP

Assinatura numérica de cada dispositivo que se encontra numa rede de computadores e utiliza o IP para efetuar comunicações (Collin, 2002, p. 182).

Jamming

Ataque de inativação a uma rede de computadores *wireless* de 2.4 GHz de banda, com interface de frequência de rádio (MBA & Hetico, 2007, p. 174).

JAVA

Poderosa linguagem de computadores criada pela Sun Microsystems, que permite a interatividade entre uma *página web* e a *internet* em toda uma série de diferentes tipologias, sistemas, linguagens e aplicações de *software* (MBA & Hetico, 2007, p. 174).

Linguagem de programação de nível 1/baixo nível

Linguagem de programação, própria de um sistema ou computador, onde cada máquina tem o seu próprio código de instruções. Assim sendo este tipo de programação é muito demorado e complexo. (Collin, 2002, p. 200).

Linguagem de programação de nível 2/alto nível

Linguagem de programação que é fácil de aprender, fácil de entender e que permite ao utilizador escrever programas usando palavras e comandos muito mais fáceis de entender[54]. (Collin, 2002, p. 159).

Link

Conexão entre duas ou mais redes de computadores. Ex. *Local Area Network (LAN), Metropolitan Area Network (MAN), Wide Area Network (WAN[55])* e *Internet* (MBA & Hetico, 2007, p. 182).

Machine learning

Conjunto de técnicas desenvolvidas para possibilitar que uma dada máquina aprenda. Como saber se uma máquina aprendeu? Esta questão levanta várias outras questões filosóficas, contudo podemos considerar que uma máquina aprendeu sempre que esta altera o seu comportamento para melhor do que no passado (Witten & Frank, 2005, pp. 7-9).

Mail Protection System (MPS)

É um *Security Email Gateway* para empresas e provedores de serviços. O MPS foi concebido a partir do zero para atender às necessidades da industria em termos de

[54] Estas palavras geralmente são expressões em inglês referentes à ordem que é dada ao computador.
[55] LAN (Rede de Área Local), MAN (Rede de Área Metropolitana) e WAN (Rede de Longa Distância).

segurança de *email* e tem funcionalidades como a gestão de cotas, auditoria, gestão de *aliases*[56] virtuais e resumos de quarentena (Anubisnetworks, 2015).

Malware

Programa que tem intenções maliciosas. Exemplos destes programas são vírus, trojans e worms (Christodorescu, Jha, Seshia, Song, & Bryant, 2005, p. 1).

Man-in-the-middle attacks

O invasor coloca-se entre o emissor e o recetor e vê a informação que está a ser enviada entre os dois extremos. Em alguns casos o invasor consegue persuadir o emissor a comunicar com o recetor e vice-versa. (Wu et al., 2007, p. 116).

Media Access Control (MAC)

Subcamada inserida na camada de ligação do modelo OSI, que providencia acesso aos meios de comunicação (Collin, 2002, p. 210).

Media Access Control Address (MAC *Address*) /Endereço MAC

Identificador utilizado em interfaces de rede para comunicações na camada física. Os *MAC addresses* são instalados e manufaturados de forma a garantidamente apenas existir um no mundo, o que faz destes identificadores convenientes e únicos (Tanenbaum, 2011, p. 339).

*Message Authentication Code (*MAC)

Código especial transmitido ao mesmo tempo que uma mensagem, como prova da sua autenticidade (Collin, 2002, p. 201).

Operating System (OS)

Conjunto básico de programas e utilitários que fazem o computador correr. No núcleo do OS encontra-se o *kernel*, que permite ao computador iniciar (*boot-up)* e correr aplicações ou programas (MBA & Hetico, 2007, p. 217).

Path

Rota entre dois pontos numa rede de comunicações (Collin, 2002, p. 249).

[56] Endereço de e-mail que reenvia os e-mails para um e-mail alternativo (Farlex, 2015).

Peer (par)

Cada um de dois dispositivos semelhantes a operar no mesmo nível de protocolo de rede (Collin, 2002, p. 251).

Router

Aparelho de comunicações que recebe pacotes de dados, num determinado protocolo, e os envia para a sua localização pela rota mais eficiente (Collin, 2002, p. 289).

Scan

Exame de uma imagem, objeto ou lista de itens para obter dados que os descrevam (Collin, 2002, p. 292).

Security Information and Events Management (SIEM)

É um *Security Email Gateway* para empresas e provedores de serviços. O MPS foi concebido a partir do zero para atender às necessidades da industria em termos de segurança de *email* e tem funcionalidades como a gestão de cotas, auditoria, gestão de *aliases*[57] virtuais e resumos de quarentena (Anubisnetworks, 2015).

Server/Servidor

Computador dedicado que executa uma função numa rede. Numa rede a máquina fixa é chamada o servidor e as unidades voláteis os satélites. Caso a rede seja em estrela, cada satélite encontra-se ligado individualmente a um servidor central. (Collin, 2002, p. 361).

Simple Mail Transfer Protocol (SMTP)

Protocolo padrão que permite que as mensagens de correio eletrónico sejam transferidas de um sistema para outro. Geralmente é utilizado como método de transferência de emails, de um servidor da internet para outro ou para enviar emails de um computador para um servidor. (Collin, 2002, p. 308).

[57] Endereço de e-mail que reenvia os e-mails para um e-mail alternativo (Farlex, 2015).

Sincronização

Garantir que dois ou mais dispositivos ou processos estão coordenados em ação e tempo (Collin, 2002, p. 322).

Software

Consiste num conjunto de instruções capazes de serem processadas para efetuarem ações específicas. Como exemplo temos os programas de computador (Microsoft, 2013; Pressman, 2005, p. 21).

Spam

Artigo publicado em mais de um grupo de notícias, portanto é provável que tenha conteúdos publicitários (Collin, 2002, p. 312).

Stream

Conjunto de dados armazenados na *Net Technology File System (NTFS)* em tempo real e contínuo, com os respetivos nomes (MBA & Hetico, 2007, p. 287). Também se pode referir a um longo fluxo de dados em série (Collin, 2002, p. 318).

Switch

Ponto de controlo num programa de computador; Dispositivo mecânico que consegue conectar ou isolar duas ou mais linhas (Collin, 2002, pp. 321-322).

***Synchronise Packet* (SYN)**

Pacote TCP de sincronização.

Telnet

É um protocolo TCP/IP que permite ao utilizador conectar-se e controlar o computador remotamente via *internet* (Collin, 2002, p. 329).

Token

Código interno que substitui uma declaração numa linguagem de nível 2 (alto nível). Numa LAN, consiste num pacote de controlo que é passado entre as estações de trabalho para controlar o acesso à rede (Collin, 2002, p. 334).

Trailer

Dados adicionais colocados no final de um bloco de dados a ser armazenado ou transmitido (Collin, 2002, p. 336). Estes dados geralmente contêm informações para o tratamento dos blocos ou marcam apenas a sua extremidade.

Transmission Control Protocol (TCP)

Protocolo de transferência utilizado em redes e sistemas de comunicação, geralmente utilizado pelas redes baseadas em *unix*. Este protocolo é utilizado para todas as comunicações através da *internet* (Collin, 2002, p. 328).

Transmission Control Protocol/Interface Program (TCP/IP)

Protocolo de transmissão de dados que possibilita uma comunicação completa, agrupa os dados em pacotes e verifica se existem erros (Collin, 2002, p. 328).

Trivial File Transfer Protocol (TFTP)

Simplificação do sistema padrão FTP, vulgarmente utilizado para carregar *software* do OS, a partir de um servidor, para uma estação de trabalho sem disco, quando esta inicia (Collin, 2002, p. 331).

Trojan Horse

Programa inserido num sistema por um *hacker* que copia constantemente informações classificadas a que depois um hacker pode aceder sem o conhecimento autorizado do utilizador (Collin, 2002, p. 338).

Turing machine

Dispositivo teórico concebido pelo matemático *Alan Turing*, muitos anos antes de existirem computadores. Uma máquina de *Turing* contém 4 elementos base: um programa, um estado de controlo finito (funciona como um processador), uma fita (funciona como memória) e um leitor/gravador de fita (que aponta para a posição a ler e a escrever). A Figura 77 representa uma máquina de *Turing* (Nielsen & Chuang, 2010, p. 122).

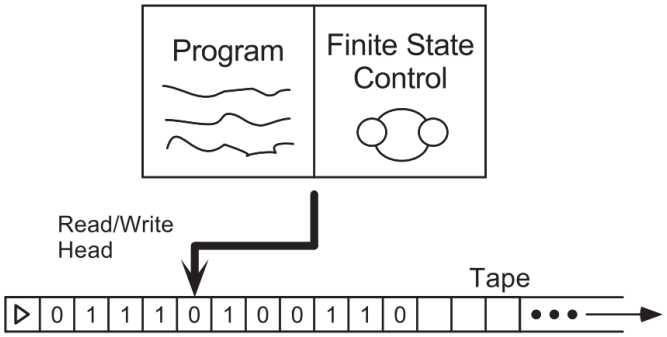

Figura 77 - Máquina de Turing (Nielsen & Chuang, 2010, p. 123)

Upload

Transferir um ficheiro de um computador para um computador hospedeiro (Collin, 2002, p. 334).

User Datagram Protocol (UDP)

Protocolo que faz parte do protocolo TCP/IP e que é normalmente utilizado na gestão da rede e aplicações *Simple Network Management Protocol (SNMP)* (Collin, 2002, p. 345).

Virtual Private Network (VPN)

Rede de comunicações dedicada, pertencente a uma ou mais empresas, que está distribuída por múltiplos locais e conectada através de tuneis nas redes de comunicações públicas (Ex. *Internet*) (Chowdhury & Boutaba, 2010, p. 863).

Virus

Programa que se insere, a si mesmo, num ficheiro executável e se copia a ele próprio para outros ficheiros executáveis, cada vez que um ficheiro infetado é corrido (Collin, 2002, p. 351).

Webpage/Página web

Conjunto de milhões de *website's* e *webpages* que juntos formam a parte da *internet* que é habitualmente mais visitada pelos utilizadores (Collin, 2002, p. 356).

Website

Conjunto de *webpages* que se encontram ligadas, relacionadas e que podem ser acedidas por um utilizador com um *web browser* (Collin, 2002, p. 356).

Weka software

Programa de análise computacional de dados, que agrega um conjunto de algoritmos provenientes de inteligência artificial, que permitem a um computador aprender (obter novo conhecimento) de forma indutiva ou dedutiva (Hall et al., 2009, p. 10).

Windows

Nome comercial de uma família de sistemas operativos, desenvolvida pela empresa Microsoft, que interage com os utilizadores através de uma interface gráfica. O *windows* é o sistema operativo utilizado pela maioria dos computadores (Collin, 2002, p. 358).

Wireless network/wireless LAN

Rede que não utiliza cabos para transmitir dados entre computadores. Em vez disso utiliza frequências rádio para transmitir sinais (Collin, 2002, p. 359).

Wireless technology

Frequência de condução rádio para sistemas WAN, MAN ou LAN, que permite uma maior mobilidade (MBA & Hetico, 2007, p. 321).

World Wide Web (WEB/WWW)

Conjunto de milhões de *website's* e *webpages* que juntos formam a parte da *internet* que é habitualmente mais visitada pelos utilizadores (Collin, 2002, p. 361).

Worm

Um *worm* é um programa independente. Este reproduz-se copiando-se a si mesmo de um computador para o outro, habitualmente através de uma rede. Tal como um vírus, o *worm* junta a capacidade de provocar danos com a velocidade de se espalhar rapidamente de um local para o outro. Contrariamente a um vírus que se anexa a um ficheiro, o *worm* mantém-se independente (Russell & Gangemi, 1991, p. 82).

Índice remissivo

R

S

T

U

V

W

Apêndices

APN 1. Tratamento dos dados (run.sh)

```
#!/bin/bash


###Para executar, colocar no terminal
     ###chmod +x run.sh
     ###./run.sh


###converter pcap para texto (com as portas já em numero)
     #tcpdump -n -tttt -r capture.pcap >
/root/capturas/texto/temp/captura.txt
     wait


###imprimir tempo e converter para segundos
     #awk -f /root/capturas/script/segundoscript
/root/capturas/texto/temp/captura.txt >
/root/capturas/texto/final/campo2.txt
     wait


###imprimir ip de origem e converter para numero
     ###imprimir campo 4 (ip e porta de origem)
          #awk -f /root/capturas/script/campo4script
/root/capturas/texto/temp/captura.txt >
/root/capturas/texto/temp/campo4.txt
          wait


     ###imprimir e converter ip de origem
          #awk -F. -f
/root/capturas/script/ipnumericoscript
/root/capturas/texto/temp/campo4.txt >
/root/capturas/texto/final/campo4ip_or.txt
          wait


###imprimir porta de origem
```

```
     #awk -F. -f /root/capturas/script/portasscript
/root/capturas/texto/temp/campo4.txt >
/root/capturas/texto/final/campo4porta_or.txt
     wait


###imprimir ip de destino e converter para numero
     ###imprimir campo 6 (ip e porta de destino)
          #awk -f /root/capturas/script/campo6script
/root/capturas/texto/temp/captura.txt >
/root/capturas/texto/temp/campo6.txt
          wait


     ###imprimir e converter ip de origem
          #awk -F. -f
/root/capturas/script/ipnumericoscript
/root/capturas/texto/temp/campo6.txt >
/root/capturas/texto/final/campo6ip_dest.txt
          wait


###imprimir porta de destino
     #awk -F. -f /root/capturas/script/portasscript
/root/capturas/texto/temp/campo6.txt >
/root/capturas/texto/final/campo6porta_dest.txt
     wait


###imprimir flags
     ###imprime campo 8
          #awk -f /root/capturas/script/flagsscript
/root/capturas/texto/temp/captura.txt >
/root/capturas/texto/final/campo8flags.txt
          wait


###imprimir sequencias
     ###imprime todas
          #awk -f /root/capturas/script/seqscript
/root/capturas/texto/temp/captura.txt >
/root/capturas/texto/temp/seq.txt
          wait
```

```
     ###imprime sequencia inicial
          #awk -F: -f /root/capturas/script/seqinscript
/root/capturas/texto/temp/seq.txt >
/root/capturas/texto/final/seqin.txt
          wait

     ###imprime sequencia final
          #awk -F: -f /root/capturas/script/seqfinscript
/root/capturas/texto/temp/seq.txt >
/root/capturas/texto/final/seqfin.txt
          wait

###imprimir acknowledge
     #awk -f /root/capturas/script/ackscript
/root/capturas/texto/temp/captura.txt >
/root/capturas/texto/final/ack.txt
     wait

###imprimir win size
     #awk -f /root/capturas/script/winscript
/root/capturas/texto/temp/captura.txt >
/root/capturas/texto/final/win.txt
     wait

###imprimir val
     #awk -f /root/capturas/script/valscript
/root/capturas/texto/temp/captura.txt >
/root/capturas/texto/final/val.txt
     wait

###imprimir ecr
     #awk -f /root/capturas/script/ecrscript
/root/capturas/texto/temp/captura.txt >
/root/capturas/texto/final/ecr.txt
     wait

###imprimir length
```

```
     #awk -f /root/capturas/script/lengthscript
/root/capturas/texto/temp/captura.txt >
/root/capturas/texto/final/length.txt

     wait


###juntar num ficheiro csv

     #paste /root/capturas/texto/final/campo2.txt
/root/capturas/texto/final/campo4ip_or.txt
/root/capturas/texto/final/campo4porta_or.txt
/root/capturas/texto/final/campo6ip_dest.txt
/root/capturas/texto/final/campo6porta_dest.txt
/root/capturas/texto/final/campo8flags.txt
/root/capturas/texto/final/seqin.txt
/root/capturas/texto/final/seqfin.txt
/root/capturas/texto/final/ack.txt
/root/capturas/texto/final/win.txt
/root/capturas/texto/final/val.txt
/root/capturas/texto/final/ecr.txt
/root/capturas/texto/final/length.txt -d , >
/root/capturas/texto/final/input1.txt

     wait


###associar ações tomadas pela firewall de perímetro às
ligações

     ###imprimir input2

          #awk -f /root/capturas/script/input2script
/root/capturas/logs.txt >
/root/capturas/texto/temp/input2.txt

          wait


     ###associar input1 a input2 e imprimir num ficheiro
chamado output

          #cd /root/capturas/script/

          #chmod +x comparar.py

          #./comparar.py
/root/capturas/texto/final/input1.txt
/root/capturas/texto/temp/input2.txt
/root/capturas/texto/final/output.txt

          wait


###imprimir ficheiro de verificação de erros
```

```
     #awk -f /root/capturas/script/vererrosscript
/root/capturas/texto/final/output.txt >
/root/capturas/texto/final/vererros.txt
     wait


###imprimir amostra para treino
     #sed -n 1,30000p /root/capturas/texto/final/output.txt
> /root/capturas/texto/temp/amostratreino.txt
     wait


###imprimir amostra para teste
     #tail -n 10000 /root/capturas/texto/final/output.txt >
/root/capturas/texto/temp/amostrateste.txt
     wait


###converter amostra de treino para arff
     #awk -f /root/capturas/script/treinoarffscript
/root/capturas/texto/temp/amostratreino.txt >
/root/capturas/texto/final/amostratreino.arff
     wait


###converter amostra de teste para arff
     #awk -f /root/capturas/script/testearffscript
/root/capturas/texto/temp/amostrateste.txt >
/root/capturas/texto/final/amostrateste.arff
     wait


###converter output para arff
     #awk -f /root/capturas/script/treinoarffscript
/root/capturas/texto/final/output.txt >
/root/capturas/texto/final/output.arff
     wait
```

APN 2. Scripts

APN 2.1. segundoscript

```
     Begin
{FS= " "}
{
#Lê o tempo e converte para segundos
{
     #verifica se é numero
     {
     if (substr ($2,1,2) ~ /^[0-9]+$/)
          {
          A = substr ($2,1,2) * 3600;
          B = substr ($2,4,2) * 60;
          C = substr ($2,7,9);
          total = A + B + C
          }
     else
          total = 0
     }
}
#Imprime com os valores arredondados a seis casas decimais
{printf "%.6f\n", total}
}
#correr
#awk -f segundoscript captura.txt
```

APN 2.2. campo4script

```
Begin
{FS= " "}
{print $4}
```

APN 2.3. ipnumericosscript

```
Begin { FS = "."}
{    {
     if ($1 ~ /^[0-9]+$/)
          {
          n=($1 * 256^3) + ($2 * 256^2) + ($3 * 256^1) +
($4 * 256^0)
          printf"%.0f\n", n
          }
     else
          print "0"
     }
}
#executar
#awk -F. -f campoipnumericosript campoA.txt > campoAip.txt
```

APN 2.4. portasscript

```
Begin { FS = "."}
{    {
     if ($1 ~ /^[0-9]+$/)
          {
          printf "%.0f\n", $5
          }
     else
          print "0"
     }
}
```

APN 2.5. campo6script

```
Begin
{FS= " "}
{print $6}
```

APN 2.6. flagsscript

```
{BEGIN {FS = " "}
{
   estado = 0
   for (i=1;i<=NF;i++)
#procura a palavra Flags
     if ($i == "Flags")
     {
#retira os dois carateres a seguir a [
          fi = substr($(i+1),2,2);
               {
#verifica se o segundo é igual a ]
               if (substr(fi,2,1) == "]")
                    print substr(fi,1,1);
               else
                    print fi
               }
          estado = 1
          break
     }
   if( estado == 0 )
     print "0";
}
   estado = 0
   for (i=1;i<=NF;i++)
#procura a palavra Flags
     if ($i == "Flags")
     {
#retira os dois carateres a seguir a [
          fi = substr($(i+1),2,2);
               {
#verifica se o segundo é igual a ]
               if (substr(fi,2,1) == "]")
                    print substr(fi,1,1);
```

```
                else
                     print fi
                }
           estado = 1
           break
      }
   if( estado == 0 )
     print "0";
}
```

APN 2.7. seqscript

```
BEGIN {FS = " "}
{
   estado = 0
   for (i=1;i<=NF;i++)
#procura palavra seq
     if ($i == "seq")
     {
#imprime tudo menos a virgula
          print substr($(i+1),1,length($(i+1))-1)
          estado = 1
          break
     }
   if( estado == 0 )
     print "0:0";
}
```

APN 2.8. seqinscript

```
Begin { FS = ":"}
{    {
     if ($1 ~ /^[0-9]+$/)
          {
          print $1
```

```
          }
     else
          print "0"
     }
}
```

APN 2.9. seqfinscript

```
Begin { FS = ":"}
{    {
     if ($2 ~ /^[0-9]+$/)
          {
          print $2
          }
     else
          print "0"
     }
}
```

APN 2.10. ackscript

```
BEGIN {FS = " "}
{
   estado = 0
   for (i=1;i<=NF;i++)
#procura palavra ack
     if ($i == "ack")
     {
#verifica se é numero
          if (substr($(i+1),1,length($(i+1))-1) ~ /^[0-
9]+$/)
               {
#imprime tudo menos a virgula
               print substr($(i+1),1,length($(i+1))-1)
               estado = 1
```

```
                break
                }
     }
   if( estado == 0 )
     print "0";
}
```

APN 2.11. winscript

```
BEGIN {FS = " "}
{
   estado = 0
   for (i=1;i<=NF;i++)
#procura palavra win
     if ($i == "win")
     {
#verifica se é numero
          if (substr($(i+1),1,length($(i+1))-1) ~ /^[0-
9]+$/)
                {
#imprime tudo menos a virgula
                print substr($(i+1),1,length($(i+1))-1)
                estado = 1
                break
                }
     }
   if( estado == 0 )
     print "0";
}
```

APN 2.12. valscript

```
BEGIN {FS = " "}
{
   estado = 0
```

```
   for (i=1;i<=NF;i++)
#procura palavra val
     if ($i == "val")
          {
#verifica se é numero
          if ($(i+1) ~ /^[0-9]+$/)
               {
#imprime o campo a seguir
               print $(i+1)
               estado = 1
               break
               }
          }
   if( estado == 0 )
     print "0";
}
```

APN 2.13. ecrscript

```
BEGIN {FS = " "}
{
   estado = 0
   for (i=1;i<=NF;i++)
#procura palavra ecr
     if ($i == "ecr")
     {
          if (substr($(i+1),1,length($(i+1))-2) ~ /^[0-
9]+$/)
               {
               print substr($(i+1),1,length($(i+1))-2)
               estado = 1
               break
               }
          else
```

```
#verificou-se que havia situações em que o ecr não estava
separado por espaço, mas sim por virgula
#este scrip corrige a situação, copiando apenas até à
virgula
               {d = 0;
               for (c=1;c=length($(i+1));c++)
                    {
                    if (substr($(i+1),d,1) == ",")
                         {
                         print substr($(i+1),1,d-1)
                         estado = 1
                         break
                         }
                    else
                         {d = d + 1}
                    }
               }
     }
   if( estado == 0 )
     print "0";
}
```

APN 2.14. lengthscript

```
BEGIN {FS = " "}
{
   estado = 0
   for (i=1;i<=NF;i++)
#procura palavra length
     if ($i == "length")
          {
          if ($(i+1) ~ /^[0-9]+$/)
               {
               print $(i+1)
               estado = 1
```

```
                break
                }
          else
#verificou-se que havia situações em que o length no final
tinha outros caracteres
#este scrip corrige a situação, copiando apenas até ao
ultimo numero
                {d = 0;
                for (c=1;c=length($(i+1));c++)
                     {
                     if (substr($(i+1),d,1) != /^[0-9]+$/)
#se nao for numero logo na primeira letra faz break
                          if (d == 0) {break}
                          else
                               {
                               print substr($(i+1),1,d-1)
                               estado = 1
                               break
                               }
                     else
                          {d = d + 1}
                     }
                }
          }
   if( estado == 0 )
     print "0";
}
```

APN 2.15. input2script

```
Begin
{FS= " "}
{
#verifica se tempo o formato de tempo
     {
```

```
     if (substr($3,3,1) == ":")
#acrecenta um 0 à esquerda da hora, se a hora é menor que
10
          {
          A = "0"substr($3,2,length($3)-2)
          }
     else
          if (substr($3,length($3)-3,1) == ":")
               {
               A = substr($3,2,length($3)-2)
               }
     else
          A = "0"
     }
#verifica se a porta tem formato numerico
     {
     if (substr($8,2,length($8)-2) ~ /^[0-9]+$/)
          {
          B = substr($8,2,length($8)-2)
          }
     else
          B = "0"
     }
#verifica se é uma das ações conhecidas
     {
     if (substr($7,2,length($7)-2) == "Drop"){C =
substr($7,2,length($7)-2)}
     else
     if (substr($7,2,length($7)-2) == "Accept"){C =
substr($7,2,length($7)-2)}
     else
     if (substr($7,2,length($7)-2) == "Reject"){C =
substr($7,2,length($7)-2)}
     else
     if (substr($7,2,length($7)-2) == "Decrypt"){C =
substr($7,2,length($7)-2)}
```

```
     else
          C = substr($7,2,length($7)-2) " erro"
     }
#imprime
print A ",0," B "," C
}
```

APN 2.16. comparar.py

```
#!/usr/bin/env python3

#correr:
#chmod +x comparar.py
#./comparar.py input1.txt input2.txt output.txt

import sys
import csv
import bisect
import re

timeRange = 1 # segundos

# O tempo tem de coincidir com o seguinte registo
timestampRegex = re.compile('[[0-9][0-9]\.*[0-9]*')

# Converte o tempo de uma string 'hh:mm:ss' para decimal
def timeToFloat(time):
    # Adiciona-se horas e minutos, caso estes não existam
    (s, m, h) = (['00', '00'] + time.split(':'))[::-1][0:3]
    result = float(h) * 3600 + float(m) * 60 + float(s)
    return result

if (len(sys.argv) != 4):
    print('Usage: {} <input file 1> <input file 2> <output
file>'.format(sys.argv[0]))
```

```
    exit(1)

inputFileName1 = sys.argv[1]
inputFileName2 = sys.argv[2]
outputFileName = sys.argv[3]

# Cada entrada será um vetor com o tempo como primeiro
elemento e a linha do ficheiro 2 como segundo elemento
file2Entries = []
with open(inputFileName2) as inputFile2:
    csvReader = csv.reader(inputFile2)
    for row in csvReader:
        if len(row) == 4:
            if not timestampRegex.match(row[0]):
                continue
            time = timeToFloat(row[0])
            file2Entries.append((time, row))

file1Entries = []
with open(inputFileName1) as inputFile1,
open(outputFileName, 'w') as outputFile:
    csvReader = csv.reader(inputFile1)
    # para cada linha do ficheiro 1 procura-se uma
correspondência com as linhas do ficheiro 2
    for row in csvReader:
        if len(row) == 13:
            if not timestampRegex.match(row[0]):
                # se não encontrar correspondência
                outputFile.write(','.join(row +
['NOACTION']))
                outputFile.write('\n')
                continue
            time = timeToFloat(row[0])
            # Encontra a primeira e ultima entrada do
ficheiro 2 com um intervalo de 1 segundo
```

```
            a = bisect.bisect_right(file2Entries, (time -
timeRange,))
            b = bisect.bisect_left(file2Entries, (time +
timeRange,))
            # ciclo para encontrar correspondencias (dentro
do intervalo de tempo)
            for entry in file2Entries[a:b]:
          #verifica a porta de destino
                if entry[1][2] == row[2]:
                    outputFile.write(','.join(row +
[entry[1][3]]))
                    outputFile.write('\n')
                    break;
          #verifica a porta de origem
                if entry[1][2] == row[4]:
                    outputFile.write(','.join(row +
[entry[1][3]]))
                    outputFile.write('\n')
                    break;
            else:
                # Caso não se encontre correspondência com
o ficheiro 2
                outputFile.write(','.join(row +
['NOACTION']))
                outputFile.write('\n')
```

APN 2.17. vererrosscript

```
BEGIN {FS = ","}
{
#campo 1 - tempo
if ($1 + 0 != $1) {print "erro $1 " NR}
#campo 2 - ip de origem
if ($2 + 0 != $2) {print "erro $2 " NR}
#campo 3 - porta de origem
if ($3 + 0 != $3) {print "erro $3 " NR}
#campo 4 - ip de destino
```

```
if ($4 + 0 != $4) {print "erro $4 " NR}
#campo 5 - porta de destino
if ($5 + 0 != $5) {print "erro $5 " NR}
#campo 6 - flags
flags[$6]++
#campo 7 - sequência inicial
if ($7 + 0 != $7) {print "erro $7 " NR}
#campo 8 - sequência final
if ($8 + 0 != $8) {print "erro $8 " NR}
#campo 9 - ack
if ($9 + 0 != $9) {print "erro $9 " NR}
#campo 10 - win
if ($10 + 0 != $10) {print "erro $10 " NR}
#campo 11 - val
if ($11 + 0 != $11) {print "erro $11 " NR}
#campo 12 - ecr
if ($12 + 0 != $12) {print "erro $12 " NR}
#campo 13 - length
if ($13 + 0 != $13) {print "erro $13 " NR}
#campo 14 - acao
acao[$14]++
#conta linhas
c=c+1;
}
END {
print ""
print ""
print ""
print "AÇÃO"
print ""
for (string in acao) { print string ":" acao[string]}
print ""
print "total de instâncias: " c
print ""
```

```
print ""
print "FLAGS"
print ""
for (bandeira in flags) { print bandeira ":"
flags[bandeira]}
}
#quando surgirem erros, para encontrar o tempo da instância
que está a gerar problemas, executar:
#awk -F,  '{if ($14 == "nome da acao a mais") print $1}'
/root/capturas/texto/final/output.txt
```

APN 2.18. treinoarffscript

```
BEGIN {FS = ","}
{
if (NR==1)
     {
     print "@relation Treino.txt"
     print ""
     print "@attribute Segundos numeric"
     print "@attribute IP_Or numeric"
     print "@attribute Port_Or numeric"
     print "@attribute IP_Dest numeric"
     print "@attribute Port_Dest numeric"
     print "@attribute Flag {SE,F.,R,S,P.,.W,FP,.,R.,S.,0}"
     print "@attribute Seq_in numeric"
     print "@attribute Seq_fin numeric"
     print "@attribute Ack numeric"
     print "@attribute Janela numeric"
     print "@attribute VAL numeric"
     print "@attribute ECR numeric"
     print "@attribute Lenght numeric"
     print "@attribute Action
{Accept,Drop,Reject,Decrypt,NOACTION}"
     print ""
     print "@data"
```

```
     print $0
     }
else
     print $0
}
```

APN 2.19. testearffscript

```
BEGIN {FS = ","}
{
if (NR==1)
     {
     print "@relation Teste.txt"
     print ""
     print "@attribute Segundos numeric"
     print "@attribute IP_Or numeric"
     print "@attribute Port_Or numeric"
     print "@attribute IP_Dest numeric"
     print "@attribute Port_Dest numeric"
     print "@attribute Flag {SE,F.,R,S,P.,.W,FP,.,R.,S.,0}"
     print "@attribute Seq_in numeric"
     print "@attribute Seq_fin numeric"
     print "@attribute Ack numeric"
     print "@attribute Janela numeric"
     print "@attribute VAL numeric"
     print "@attribute ECR numeric"
     print "@attribute Lenght numeric"
     print "@attribute Action
{Accept,Drop,Reject,Decrypt,NOACTION}"
     print ""
     print "@data"
     print $0
     }
else
     print $0
```

```
}
```

APN 2.20. mediasscript

```
BEGIN {FS = ","}

{

#conta Accept

if ($14 == "Accept") {a1 = a1 + $1; a2 = a2 + $2; a3 = a3 +
$3; a4 = a4 + $4; a5 = a5 + $5; fa[$6]++; a7 = a7 + $7; a8
= a8 + $8; a9 = a9 + $9; a10 = a10 + $10; a11 = a11 + $11;
a12 = a12 + $12; a13 = a13 + $13; ia = ia + 1}

#conta Drop

if ($14 == "Drop") {d1 = d1 + $1; d2 = d2 + $2; d3 = d3 +
$3; d4 = d4 + $4; d5 = d5 + $5; fd[$6]++; d7 = d7 + $7; d8
= d8 + $8; d9 = d9 + $9; d10 = d10 + $10; d11 = d11 + $11;
d12 = d12 + $12; d13 = d13 + $13; id = id + 1}

#conta Reject

if ($14 == "Reject") {r1 = r1 + $1; r2 = r2 + $2; r3 = r3 +
$3; r4 = r4 + $4; r5 = r5 + $5; fr[$6]++; r7 = r7 + $7; r8
= r8 + $8; r9 = r9 + $9; r10 = r10 + $10; r11 = r11 + $11;
r12 = r12 + $12; r13 = r13 + $13; ir = ir + 1}

#conta Decrypt

if ($14 == "Decrypt") {t1 = t1 + $1; t2 = t2 + $2; t3 = t3
+ $3; t4 = t4 + $4; t5 = t5 + $5; ft[$6]++; t7 = t7 + $7;
t8 = t8 + $8; t9 = t9 + $9; t10 = t10 + $10; t11 = t11 +
$11; t12 = t12 + $12; t13 = t13 + $13; it = it + 1}


#conta NOACTION

if ($14 == "NOACTION") {n1 = n1 + $1; n2 = n2 + $2; n3 = n3
+ $3; n4 = n4 + $4; n5 = n5 + $5; fn[$6]++; n7 = n7 + $7;
n8 = n8 + $8; n9 = n9 + $9; n10 = n10 + $10; n11 = n11 +
$11; n12 = n12 + $12; n13 = n13 + $13; ina = ina + 1}

#conta instâncias

i = i + 1

}

END {

ma1 = a1 / ia; ma2 = a2 / ia; ma3 = a3 / ia; ma4 = a4 / ia;
     ma5 = a5 / ia;      ma7 = a7 / ia; ma8 = a8 / ia; ma9
= a9 / ia;     ma10 = a10 / ia;    ma11 = a11 / ia;    ma12
= a12 / ia;    ma13 = a13 / ia;
```

```
md1 = d1 / id; md2 = d2 / id; md3 = d3 / id; md4 = d4 / id;
     md5 = d5 / id;      md7 = d7 / id; md8 = d8 / id; md9
= d9 / id;     md10 = d10 / id;    md11 = d11 / id;    md12
= d12 / id;    md13 = d13 / id;

mr1 = r1 / ir; mr2 = r2 / ir; mr3 = r3 / ir; mr4 = r4 / ir;
     mr5 = r5 / ir;      mr7 = r7 / ir; mr8 = r8 / ir; mr9
= r9 / ir;     mr10 = r10 / ir;    mr11 = r11 / ir;    mr12
= r12 / ir;    mr13 = r13 / ir;

mt1 = t1 / it; mt2 = t2 / it; mt3 = t3 / it; mt4 = t4 / it;
     mt5 = t5 / it;      mt7 = t7 / it; mt8 = t8 / it; mt9
= t9 / it;     mt10 = t10 / it;    mt11 = t11 / it;    mt12
= t12 / it;    mt13 = t13 / it;

mn1 = n1 / ina;     mn2 = n2 / ina;     mn3 = n3 / ina;
     mn4 = n4 / ina;     mn5 = n5 / ina;          mn7 = n7
/ ina;    mn8 = n8 / ina;     mn9 = n9 / ina;     mn10 =
n10 / ina;     mn11 = n11 / ina;   mn12 = n12 / ina;   mn13
= n13 / ina;

print ma1 "," ma2 "," ma3 "," ma4 "," ma5 "," ma7 "," ma8
"," ma9 "," ma10 "," ma11 "," ma12 "," ma13 ",Accept"

print md1 "," md2 "," md3 "," md4 "," md5 "," md7 "," md8
"," md9 "," md10 "," md11 "," md12 "," md13 ",Drop"

print mr1 "," mr2 "," mr3 "," mr4 "," mr5 "," mr7 "," mr8
"," mr9 "," mr10 "," mr11 "," mr12 "," mr13 ",Reject"

print mt1 "," mt2 "," mt3 "," mt4 "," mt5 "," mt7 "," mt8
"," mt9 "," mt10 "," mt11 "," mt12 "," mt13 ",Decrypt"

print mn1 "," mn2 "," mn3 "," mn4 "," mn5 "," mn7 "," mn8
"," mn9 "," mn10 "," mn11 "," mn12 "," mn13 ",NOACTION"

print ""

print "total de instâncias: " i

print "total de Accept: " ia

print "total de Drop: " id

print "total de Reject: " ir

print "total de Decrypt: " it

print "total de NOACTION: " ina

print ""

print ""

print "FLAGS"

print ""

print "Accept"

for (a in fa) { print a ":" fa[a]}
```

```
print ""
print "Drop"
for (d in fd) { print d ":" fd[d]}
print ""
print "Reject"
for (r in fr) { print r ":" fr[r]}
print ""
print "Decrypt"
for (t in ft) { print t ":" ft[t]}
print ""
print "NOACTION"
for (n in fn) { print n ":" fn[n]}
}
```

APN 3. Comandos weka (weka.sh)

```
#indica caminho das funções
     export CLASSPATH=/root/weka-3-7-12/weka.jar
#outra solução (o -Xmx2G diz que o tamanho permitido da
cache é 2GB, o -cp indica onde estão as funções, o -t o
conjunto de treino e o -T o conjunto de teste)
     #java -Xmx1G -cp /root/weka-3-7-12/weka.jar
weka.classifiers.bayes.NaiveBayes -t
"/root/capturas/texto/final/amostracap1.arff" -T
"/root/capturas/texto/final/amostracap2.arff"


#exportar dados
     #http://stackoverflow.com/questions/22139211/exporting
-weka-predictions-in-command-line


#naive de bayes
     #java -Xmx32G weka.classifiers.bayes.NaiveBayes -t
"/root/capturas/texto/final/cap1semnoaction.arff" -T
"/root/capturas/texto/final/cap2_2semnoaction.arff" >
"/root/weka/investigacao/semnoaction/tcap1snaTcap2_2sna.txt
"


#UPDATEABLE
     #http://weka.sourceforge.net/doc.dev/weka/classifiers/
UpdateableClassifier.html


     #naive de bayes updateable
          #java -Xmx16G
weka.classifiers.bayes.NaiveBayesUpdateable -t
"/root/capturas/texto/final/cap1semnoaction.arff" -T
"/root/capturas/texto/final/cap2_2semnoaction.arff" >
"/root/weka/investigacao/semnoaction/tcap1snaTcap2_2sna.txt
"


     #ibk
          #java -Xmx16G weka.classifiers.lazy.IBk -K 1 -W 0
-A "weka.core.neighboursearch.LinearNNSearch -A
\"weka.core.EuclideanDistance -R first-last\"" -t
"/root/capturas/texto/final/cap1semnoaction.arff" -T
"/root/capturas/texto/final/cap2_2semnoaction.arff" >
```

```
"/root/weka/investigacao/semnoaction/ibk/tcap1snaTcap2_2sna
.txt"


     #HoeffdingTree

          #java -Xmx128G
weka.classifiers.trees.HoeffdingTree -t
"/root/capturas/texto/final/cap1semnoaction.arff" -T
"/root/capturas/texto/final/cap2_2semnoaction.arff" >
"/root/weka/investigacao/semnoaction/trees/tcap1snaTcap2_2s
naHT.txt"


#J48

     #java -Xmx128G weka.classifiers.trees.J48 -C 0.25 -M 2
-t "/root/capturas/texto/final/cap1semnoaction.arff" -T
"/root/capturas/texto/final/cap2_2semnoaction.arff" >
"/root/weka/investigacao/semnoaction/trees/tcap1snaTcap2_2s
naJ48.txt"
```

APN 4. Caras de Chernoff

```
>> Acoes =
[38319,3.0221e+09,48943.1,2.74933e+09,372.923,2.67638e+07,1
.08519e+06,1.00943e+08,3948.97,5.29928e+08,9.07083e+08,93.1
974;

38354.1,2.58758e+09,3178.41,6.12842e+08,32298.5,1.00921e+08
,6.82057e+07,4.94437e+07,4404.77,9.3109e+08,5.40422e+08,113
2.01;

38442.1,2.72277e+09,37102.2,3.03297e+09,181.874,2.8462e+07,
1.4761e+06,2.61689e+07,19855,3.26081e+08,3.95057e+08,109.47
2;

38745.5,3.2492e+09,56510.7,3.56914e+09,438.622,3.14478e+06,
81.6794,1.11098e+07,1629.7,2.48709e+08,3.00904e+08,6.9197;

38397.9,2.83473e+09,43959.6,2.61675e+09,2296.73,5.70177e+07
,1.72827e+06,6.89654e+07,4189.96,4.8861e+08,9.55581e+08,689
.301]


Acoes =


  1.0e+009 *


  Columns 1 through 9


    0.0000    3.0221    0.0000    2.7493    0.0000
0.0268    0.0011    0.1009    0.0000

    0.0000    2.5876    0.0000    0.6128    0.0000
0.1009    0.0682    0.0494    0.0000

    0.0000    2.7228    0.0000    3.0330    0.0000
0.0285    0.0015    0.0262    0.0000

    0.0000    3.2492    0.0001    3.5691    0.0000
0.0031    0.0000    0.0111    0.0000

    0.0000    2.8347    0.0000    2.6168    0.0000
0.0570    0.0017    0.0690    0.0000


  Columns 10 through 12


    0.5299    0.9071    0.0000

    0.9311    0.5404    0.0000
```

```
    0.3261    0.3951    0.0000
    0.2487    0.3009    0.0000
    0.4886    0.9556    0.0000

>> c = char('Accept','Drop','Reject','Decrypt','NOACTION')

c =

Accept
Drop
Reject
Decrypt
NOACTION

>> glyphplot(Acoes,'glyph','face',...
            'obslabels',c,...
            'grid',[5 12],...
            'page',1);
```

APN 5. Ficheiros ARFF

APN 5.1. Exemplo de *cluster*

```
@relation exemplo

@attribute Nome String
@attribute Altura numeric
@attribute Peso numeric
@attribute Class {H,M}

@data
Alicia_Keys,167.64,54.49912326,M
Nicole_Kidman,179.07,55.49929443,M
Shakira,157.48,58.49980796,M
Rose_McGowan,160.02,46.50002181,M
Johnny_Depp,177.8,71.49976528,H
Gwen_Stefani,167.64,52.4987809,M
Jenny_McCarthy,170.18,55.49929443,M
Elisha_Cuthbert,158.75,50.49843855,M
Daniel_Radcliffe,166.37,54.49912326,H
Elle_MacPherson,182.88,58.49980796,M
Lil_Wayne,165.1,63.50066384,H
Heather_Locklear,165.1,50.49843855,M
Robert_Pattinson,185,78.99991512,H
Kate_Beckinsale,172.72,54.49912326,M
Orlando_Bloom,179.07,68.00030015,H
Carrie_Underwood,160.02,51.50087769,M
Rupert_Grint,177.8,62.50049266,H
Bruce_Willis,181.61,94.50143437,H
Matt_Damon,177.8,72.49993646,H
Vin_Diesel,182.88,98.99880271,H
Daniel_Craig,179.07,83.49955143,H
Brad_Pitt,180.34,90.50074966,H
Penelope_Cruz,172.72,49.50053534,M
```

```
Keira_Knightley,170.18,47.50019299,M
Zooey_Deschanel,167.64,54.49912326,M
Calista_Flockhart,167.64,44.49967946,M
Ryan_Seacrest,172.72,68.99820336,H
Patrick_Swayze,178,48.99931577,H
Jodie_Kidd,187.96,62.50049266,M
Nicole_Richie,154.94,44.49967946,M
Christian_Bale,182.88,86.50006496,H
Laetitia_Casta,170.18,56.49946561,M
Miley_Cyrus,164,48.00141256,M
Jason_Statham,175.26,77.50079234,H
Mark_Wahlberg,173.99,76.50062116,H
Ben_Stiller,168.91,77.50079234,H
Sylvester_Stallone,177.8,98.99880271,H
Reese_Witherspoon,157.48,52.4987809,M
Hulk_Hogan,193.04,136.4995519,H
Rodrigo_Santoro,187.96,81.00025747,H
Lisa_Kudrow,172.72,55.49929443,M
Shania_Twain,162.56,49.50053534,M
Naomi_Campbell,177.8,51.50087769,M
Lucy_Lawless,177.8,58.49980796,M
Michael_Jackson,175,50.80234544,H
Justin_Timberlake,185.42,73.50010763,H
George_Clooney,180.34,86.50006496,H
Leonardo_Dicaprio,181.61,76.50062116,H
Rihanna,172.72,59.49997913,M
Al_Pacino,170.18,77.50079234,H
```

APN 5.2. Treino – Primeiras 30000 instâncias da $BD2_{CLASS}$

```
@relation Treino.txt

@attribute Segundos numeric
@attribute IP_Or numeric
@attribute Port_Or numeric
@attribute IP_Dest numeric
@attribute Port_Dest numeric
@attribute Flag {.,P.,F.,S.,S,FP,FP.,R.,R,1au,vazio}
@attribute Ack numeric
@attribute Janela numeric
@attribute VAL numeric
@attribute ECR numeric
@attribute Lenght numeric
@attribute Action {Accept,Drop,Reject,Decrypt,NOACTION}

@data
36752.245129,3264014475,44610,3264014475,44610,.,3447915500,169,
244045042,567660764,0,NOACTION
36752.245133,3264014475,44610,3264014475,44610,.,529,170,2440450
42,567660764,0,NOACTION
36752.245137,3264014475,44610,3264014475,44610,.,1310,179,244045
042,567660764,0,NOACTION
36752.245140,1042029179,80,1042029179,80,.,1097479383,5530,91656
0072,860543292,0,Drop
36752.245143,3264014475,37835,3264014475,37835,P.,2939284929,204
8,244045042,567662658,34,NOACTION
36752.245146,3264014475,37835,3264014475,37835,P.,1,2048,2440450
42,567662658,34,NOACTION
36752.245149,3264014475,37831,3264014475,37831,P.,1485272053,204
8,244045042,567662656,34,Accept
36752.245153,3264014475,41670,3264014475,41670,.,990583692,5,244
045042,3323359366,0,Accept
36752.245156,1042029179,80,1042029179,80,.,1633186245,5529,91656
0074,860543292,0,Drop
36752.245159,1042029179,80,1042029179,80,P.,1921047764,5366,9165
60073,244045032,124,Drop
36752.245162,3264014475,55222,3264014475,55222,.,2518904468,190,
```

...

APN 5.3. Teste – Ultimas 10000 instâncias da $BD2_{CLASS}$

```
@relation Teste.txt

@attribute Segundos numeric
@attribute IP_Or numeric
@attribute Port_Or numeric
@attribute IP_Dest numeric
@attribute Port_Dest numeric
@attribute Flag {.,P.,F.,S.,S,FP,FP.,R.,R,1au,vazio}
@attribute Ack numeric
@attribute Janela numeric
@attribute VAL numeric
@attribute ECR numeric
@attribute Lenght numeric
@attribute Action {Accept,Drop,Reject,Decrypt,NOACTION}

@data
39827.572875,3587003342,443,3587003342,443,.,18722,524,283512257
,247119882,1448,Drop
39827.572878,1486733646,80,1486733646,80,P.,12895,284,756466166,
247119862,314,Drop
39827.572881,3272084278,443,3272084278,443,.,15444,473,723004083
,863618122,1448,Drop
39827.572885,3587003342,443,3587003342,443,.,18722,524,283512258
,247119882,1448,Drop
39827.573147,3264014475,57144,3264014475,57144,.,14847167,2025,2
47119882,283512252,0,Accept
39827.573153,3264014475,57144,3264014475,57144,.,14848615,2025,2
47119882,283512252,0,Accept
39827.573181,3264014475,35815,3264014475,35815,.,9498319,2039,86
3618142,570735467,0,Accept
39827.573414,3264014475,59663,3264014475,59663,.,5124737,169,863
618142,3326873767,0,Accept
39827.573424,3264014475,59663,3264014475,59663,.,5126185,169,863
618142,3326873767,0,Accept
39827.573430,3264014475,59663,3264014475,59663,.,5127633,146,863
618142,3326873767,0,Accept
39827.573436,3264014475,59663,3264014475,59663,.,5129081,124,863
```

…

Anexos

ANX 1. Código JAVA do *software* weka

ANX 1.1. *Naive de Bayes*

```
package weka.classifiers.bayes;

import java.util.Collections;
import java.util.Enumeration;
import java.util.Vector;
import weka.classifiers.AbstractClassifier;
import weka.core.Aggregateable;
import weka.core.Attribute;
import weka.core.Capabilities;
import weka.core.Capabilities.Capability;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.Option;
import weka.core.OptionHandler;
import weka.core.RevisionUtils;
import weka.core.TechnicalInformation;
import weka.core.TechnicalInformation.Field;
import weka.core.TechnicalInformation.Type;
import weka.core.TechnicalInformationHandler;
import weka.core.Utils;
import weka.core.WeightedInstancesHandler;
import weka.estimators.DiscreteEstimator;
import weka.estimators.Estimator;
import weka.estimators.KernelEstimator;
import weka.estimators.NormalEstimator;
import weka.filters.Filter;
import weka.filters.supervised.attribute.Discretize;

public class NaiveBayes
```

```
  extends AbstractClassifier
  implements OptionHandler, WeightedInstancesHandler,
TechnicalInformationHandler, Aggregateable<NaiveBayes>
{
  static final long serialVersionUID =
5995231201785697655L;
  protected Estimator[][] m_Distributions;
  protected Estimator m_ClassDistribution;
  protected boolean m_UseKernelEstimator = false;
  protected boolean m_UseDiscretization = false;
  protected int m_NumClasses;
  protected Instances m_Instances;
  protected static final double DEFAULT_NUM_PRECISION =
0.01D;
  protected Discretize m_Disc = null;
  protected boolean m_displayModelInOldFormat = false;

  public String globalInfo()
  {
    return "Class for a Naive Bayes classifier using
estimator classes. Numeric estimator precision values are
chosen based on analysis of the training data. For this
reason, the classifier is not an UpdateableClassifier
(which in typical usage are initialized with zero training
instances) -- if you need the UpdateableClassifier
functionality, use the NaiveBayesUpdateable classifier. The
NaiveBayesUpdateable classifier will  use a default
precision of 0.1 for numeric attributes when
buildClassifier is called with zero training
instances.\n\nFor more information on Naive Bayes
classifiers, see\n\n" +
getTechnicalInformation().toString();
  }

  public TechnicalInformation getTechnicalInformation()
  {
    TechnicalInformation result = new
TechnicalInformation(TechnicalInformation.Type.INPROCEEDING
S);
```

```
    result.setValue(TechnicalInformation.Field.AUTHOR,
"George H. John and Pat Langley");
    result.setValue(TechnicalInformation.Field.TITLE,
"Estimating Continuous Distributions in Bayesian
Classifiers");


    result.setValue(TechnicalInformation.Field.BOOKTITLE,
"Eleventh Conference on Uncertainty in Artificial
Intelligence");


    result.setValue(TechnicalInformation.Field.YEAR,
"1995");
    result.setValue(TechnicalInformation.Field.PAGES, "338-
345");
    result.setValue(TechnicalInformation.Field.PUBLISHER,
"Morgan Kaufmann");
    result.setValue(TechnicalInformation.Field.ADDRESS,
"San Mateo");


    return result;
  }


  public Capabilities getCapabilities()
  {
    Capabilities result = super.getCapabilities();
    result.disableAll();


result.enable(Capabilities.Capability.NOMINAL_ATTRIBUTES);

result.enable(Capabilities.Capability.NUMERIC_ATTRIBUTES);
    result.enable(Capabilities.Capability.MISSING_VALUES);


    result.enable(Capabilities.Capability.NOMINAL_CLASS);

result.enable(Capabilities.Capability.MISSING_CLASS_VALUES)
;
```

```
    result.setMinimumNumberInstances(0);

    return result;
  }

  public void buildClassifier(Instances instances)
    throws Exception
  {
    getCapabilities().testWithFail(instances);

    instances = new Instances(instances);
    instances.deleteWithMissingClass();

    this.m_NumClasses = instances.numClasses();

    this.m_Instances = new Instances(instances);
    if (this.m_UseDiscretization)
    {
      this.m_Disc = new Discretize();
      this.m_Disc.setInputFormat(this.m_Instances);
      this.m_Instances = Filter.useFilter(this.m_Instances,
this.m_Disc);
    }
    else
    {
      this.m_Disc = null;
    }
    this.m_Distributions = new
Estimator[this.m_Instances.numAttributes() -
1][this.m_Instances.numClasses()];

    this.m_ClassDistribution = new
DiscreteEstimator(this.m_Instances.numClasses(), true);
    int attIndex = 0;
    Enumeration<Attribute> enu =
this.m_Instances.enumerateAttributes();
```

```
    while (enu.hasMoreElements())
    {
      Attribute attribute = (Attribute)enu.nextElement();

      double numPrecision = 0.01D;
      if (attribute.type() == 0)
      {
        this.m_Instances.sort(attribute);
        if ((this.m_Instances.numInstances() > 0) &&
(!this.m_Instances.instance(0).isMissing(attribute)))
        {
          double lastVal =
this.m_Instances.instance(0).value(attribute);
          double deltaSum = 0.0D;
          int distinct = 0;
          for (int i = 1; i <
this.m_Instances.numInstances(); i++)
          {
            Instance currentInst =
this.m_Instances.instance(i);
            if (currentInst.isMissing(attribute)) {
              break;
            }
            double currentVal =
currentInst.value(attribute);
            if (currentVal != lastVal)
            {
              deltaSum += currentVal - lastVal;
              lastVal = currentVal;
              distinct++;
            }
          }
          if (distinct > 0) {
            numPrecision = deltaSum / distinct;
          }
        }
```

```
      }
      for (int j = 0; j < this.m_Instances.numClasses();
j++) {
        switch (attribute.type())
        {
        case 0:
          if (this.m_UseKernelEstimator) {
            this.m_Distributions[attIndex][j] = new
KernelEstimator(numPrecision);
          } else {
            this.m_Distributions[attIndex][j] = new
NormalEstimator(numPrecision);
          }
          break;
        case 1:
          this.m_Distributions[attIndex][j] = new
DiscreteEstimator(attribute.numValues(), true);

          break;
        default:
          throw new Exception("Attribute type unknown to
NaiveBayes");
        }
      }
      attIndex++;
    }
    Enumeration<Instance> enumInsts =
this.m_Instances.enumerateInstances();
    while (enumInsts.hasMoreElements())
    {
      Instance instance =
(Instance)enumInsts.nextElement();
      updateClassifier(instance);
    }
    this.m_Instances = new Instances(this.m_Instances, 0);
  }
```

```
  public void updateClassifier(Instance instance)
    throws Exception
  {
    if (!instance.classIsMissing())
    {
      Enumeration<Attribute> enumAtts =
this.m_Instances.enumerateAttributes();
      int attIndex = 0;
      while (enumAtts.hasMoreElements())
      {
        Attribute attribute =
(Attribute)enumAtts.nextElement();
        if (!instance.isMissing(attribute)) {

this.m_Distributions[attIndex][((int)instance.classValue())
].addValue(instance.value(attribute), instance.weight());
        }
        attIndex++;
      }

this.m_ClassDistribution.addValue(instance.classValue(),
instance.weight());
    }
  }


  public double[] distributionForInstance(Instance
instance)
    throws Exception
  {
    if (this.m_UseDiscretization)
    {
      this.m_Disc.input(instance);
      instance = this.m_Disc.output();
    }
    double[] probs = new double[this.m_NumClasses];
```

```
    for (int j = 0; j < this.m_NumClasses; j++) {
      probs[j] =
this.m_ClassDistribution.getProbability(j);
    }
    Enumeration<Attribute> enumAtts =
instance.enumerateAttributes();
    int attIndex = 0;
    while (enumAtts.hasMoreElements())
    {
      Attribute attribute =
(Attribute)enumAtts.nextElement();
      if (!instance.isMissing(attribute))
      {
        double max = 0.0D;
        for (int j = 0; j < this.m_NumClasses; j++)
        {
          double temp = Math.max(1.0E-75D,
Math.pow(this.m_Distributions[attIndex][j].getProbability(i
nstance.value(attribute)),
this.m_Instances.attribute(attIndex).weight()));

          probs[j] *= temp;
          if (probs[j] > max) {
            max = probs[j];
          }
          if (Double.isNaN(probs[j])) {
            throw new Exception("NaN returned from
estimator for attribute " + attribute.name() + ":\n" +
this.m_Distributions[attIndex][j].toString());
          }
        }
        if ((max > 0.0D) && (max < 1.0E-75D)) {
          for (int j = 0; j < this.m_NumClasses; j++) {
            probs[j] *= 1.0E75D;
          }
        }
      }
```

```
      attIndex++;
    }
    Utils.normalize(probs);
    return probs;
  }

  public Enumeration<Option> listOptions()
  {
    Vector<Option> newVector = new Vector(3);

    newVector.addElement(new Option("\tUse kernel density
estimator rather than normal\n\tdistribution for numeric
attributes", "K", 0, "-K"));

    newVector.addElement(new Option("\tUse supervised
discretization to process numeric attributes\n", "D", 0, "-
D"));

    newVector.addElement(new Option("\tDisplay model in old
format (good when there are many classes)\n", "O", 0, "-
O"));

newVector.addAll(Collections.list(super.listOptions()));

    return newVector.elements();
  }

  public void setOptions(String[] options)
    throws Exception
  {
    super.setOptions(options);
    boolean k = Utils.getFlag('K', options);
    boolean d = Utils.getFlag('D', options);
    if ((k) && (d)) {
      throw new IllegalArgumentException("Can't use both
kernel density estimation and discretization!");
```

```
    }
    setUseSupervisedDiscretization(d);
    setUseKernelEstimator(k);
    setDisplayModelInOldFormat(Utils.getFlag('O',
options));
    Utils.checkForRemainingOptions(options);
  }

  public String[] getOptions()
  {
    Vector<String> options = new Vector();

    Collections.addAll(options, super.getOptions());
    if (this.m_UseKernelEstimator) {
      options.add("-K");
    }
    if (this.m_UseDiscretization) {
      options.add("-D");
    }
    if (this.m_displayModelInOldFormat) {
      options.add("-O");
    }
    return (String[])options.toArray(new String[0]);
  }

  public String toString()
  {
    if (this.m_displayModelInOldFormat) {
      return toStringOriginal();
    }
    StringBuffer temp = new StringBuffer();
    temp.append("Naive Bayes Classifier");
    if (this.m_Instances == null)
    {
```

```
      temp.append(": No model built yet.");
    }
    else
    {
      int maxWidth = 0;
      int maxAttWidth = 0;
      boolean containsKernel = false;
      for (int i = 0; i < this.m_Instances.numClasses();
i++) {
        if
(this.m_Instances.classAttribute().value(i).length() >
maxWidth) {
          maxWidth =
this.m_Instances.classAttribute().value(i).length();
        }
      }
      for (int i = 0; i < this.m_Instances.numAttributes();
i++) {
        if (i != this.m_Instances.classIndex())
        {
          Attribute a = this.m_Instances.attribute(i);
          if (a.name().length() > maxAttWidth) {
            maxAttWidth =
this.m_Instances.attribute(i).name().length();
          }
          if (a.isNominal()) {
            for (int j = 0; j < a.numValues(); j++)
            {
              String val = a.value(j) + "  ";
              if (val.length() > maxAttWidth) {
                maxAttWidth = val.length();
              }
            }
          }
        }
      }
```

```
      for (Estimator[] m_Distribution :
this.m_Distributions) {
        for (int j = 0; j < this.m_Instances.numClasses();
j++) {
          if ((m_Distribution[0] instanceof
NormalEstimator))
          {
            NormalEstimator n =
(NormalEstimator)m_Distribution[j];
            double mean = Math.log(Math.abs(n.getMean())) /
Math.log(10.0D);
            double precision =
Math.log(Math.abs(n.getPrecision())) / Math.log(10.0D);

            double width = mean > precision ? mean :
precision;
            if (width < 0.0D) {
              width = 1.0D;
            }
            width += 6.0D;
            if ((int)width > maxWidth) {
              maxWidth = (int)width;
            }
          }
          else if ((m_Distribution[0] instanceof
KernelEstimator))
          {
            containsKernel = true;
            KernelEstimator ke =
(KernelEstimator)m_Distribution[j];
            int numK = ke.getNumKernels();
            String temps = "K" + numK + ": mean (weight)";
            if (maxAttWidth < temps.length()) {
              maxAttWidth = temps.length();
            }
            if (ke.getNumKernels() > 0)
            {
```

```
              double[] means = ke.getMeans();
              double[] weights = ke.getWeights();
              for (int k = 0; k < ke.getNumKernels(); k++)
              {
                String m = Utils.doubleToString(means[k],
maxWidth, 4).trim();
                m = m + " (" +
Utils.doubleToString(weights[k], maxWidth, 1).trim() + ")";
                if (maxWidth < m.length()) {
                  maxWidth = m.length();
                }
              }
            }
          else if ((m_Distribution[0] instanceof
DiscreteEstimator))
          {
            DiscreteEstimator d =
(DiscreteEstimator)m_Distribution[j];
            for (int k = 0; k < d.getNumSymbols(); k++)
            {
              String size = "" + d.getCount(k);
              if (size.length() > maxWidth) {
                maxWidth = size.length();
              }
            }
            int sum = ("" + d.getSumOfCounts()).length();
            if (sum > maxWidth) {
              maxWidth = sum;
            }
          }
        }
      }
      for (int i = 0; i < this.m_Instances.numClasses();
i++)
      {
```

```
        String cSize =
this.m_Instances.classAttribute().value(i);
        if (cSize.length() > maxWidth) {
          maxWidth = cSize.length();
        }
      }
      for (int i = 0; i < this.m_Instances.numClasses();
i++)
      {
        String priorP =
Utils.doubleToString(((DiscreteEstimator)this.m_ClassDistri
bution).getProbability(i), maxWidth, 2).trim();

        priorP = "(" + priorP + ")";
        if (priorP.length() > maxWidth) {
          maxWidth = priorP.length();
        }
      }
      if (maxAttWidth < "Attribute".length()) {
        maxAttWidth = "Attribute".length();
      }
      if (maxAttWidth < "  weight sum".length()) {
        maxAttWidth = "  weight sum".length();
      }
      if ((containsKernel) &&
        (maxAttWidth < "  [precision]".length())) {
        maxAttWidth = "  [precision]".length();
      }
      maxAttWidth += 2;

      temp.append("\n\n");
      temp.append(pad("Class", " ", maxAttWidth + maxWidth
+ 1 - "Class".length(), true));

      temp.append("\n");
```

```
      temp.append(pad("Attribute", " ", maxAttWidth -
"Attribute".length(), false));
      for (int i = 0; i < this.m_Instances.numClasses();
i++)
      {
        String classL =
this.m_Instances.classAttribute().value(i);
        temp.append(pad(classL, " ", maxWidth + 1 -
classL.length(), true));
      }
      temp.append("\n");

      temp.append(pad("", " ", maxAttWidth, true));
      for (int i = 0; i < this.m_Instances.numClasses();
i++)
      {
        String priorP =
Utils.doubleToString(((DiscreteEstimator)this.m_ClassDistri
bution).getProbability(i), maxWidth, 2).trim();

        priorP = "(" + priorP + ")";
        temp.append(pad(priorP, " ", maxWidth + 1 -
priorP.length(), true));
      }
      temp.append("\n");
      temp.append(pad("", "=", maxAttWidth + maxWidth *
this.m_Instances.numClasses() +
this.m_Instances.numClasses() + 1, true));

      temp.append("\n");

      int counter = 0;
      for (int i = 0; i < this.m_Instances.numAttributes();
i++) {
        if (i != this.m_Instances.classIndex())
        {
          String attName =
this.m_Instances.attribute(i).name();
```

```
          temp.append(attName + "\n");
          if ((this.m_Distributions[counter][0] instanceof
NormalEstimator))
          {
            String meanL = "  mean";
            temp.append(pad(meanL, " ", maxAttWidth + 1 -
meanL.length(), false));
            for (int j = 0; j <
this.m_Instances.numClasses(); j++)
            {
              NormalEstimator n =
(NormalEstimator)this.m_Distributions[counter][j];
              String mean =
Utils.doubleToString(n.getMean(), maxWidth, 4).trim();
              temp.append(pad(mean, " ", maxWidth + 1 -
mean.length(), true));
            }
            temp.append("\n");

            String stdDevL = "  std. dev.";
            temp.append(pad(stdDevL, " ", maxAttWidth + 1 -
stdDevL.length(), false));
            for (int j = 0; j <
this.m_Instances.numClasses(); j++)
            {
              NormalEstimator n =
(NormalEstimator)this.m_Distributions[counter][j];
              String stdDev =
Utils.doubleToString(n.getStdDev(), maxWidth, 4).trim();

              temp.append(pad(stdDev, " ", maxWidth + 1 -
stdDev.length(), true));
            }
            temp.append("\n");

            String weightL = "  weight sum";
            temp.append(pad(weightL, " ", maxAttWidth + 1 -
weightL.length(), false));
```

```
            for (int j = 0; j <
this.m_Instances.numClasses(); j++)
            {
              NormalEstimator n =
(NormalEstimator)this.m_Distributions[counter][j];
              String weight =
Utils.doubleToString(n.getSumOfWeights(), maxWidth,
4).trim();

              temp.append(pad(weight, " ", maxWidth + 1 -
weight.length(), true));
            }
            temp.append("\n");

            String precisionL = "  precision";
            temp.append(pad(precisionL, " ", maxAttWidth +
1 - precisionL.length(), false));
            for (int j = 0; j <
this.m_Instances.numClasses(); j++)
            {
              NormalEstimator n =
(NormalEstimator)this.m_Distributions[counter][j];
              String precision =
Utils.doubleToString(n.getPrecision(), maxWidth, 4).trim();

              temp.append(pad(precision, " ", maxWidth + 1
- precision.length(), true));
            }
            temp.append("\n\n");
          }
          else if ((this.m_Distributions[counter][0]
instanceof DiscreteEstimator))
          {
            Attribute a = this.m_Instances.attribute(i);
            for (int j = 0; j < a.numValues(); j++)
            {
              String val = "  " + a.value(j);
```

```
              temp.append(pad(val, " ", maxAttWidth + 1 -
val.length(), false));
              for (int k = 0; k <
this.m_Instances.numClasses(); k++)
              {
                DiscreteEstimator d =
(DiscreteEstimator)this.m_Distributions[counter][k];
                String count = "" + d.getCount(j);
                temp.append(pad(count, " ", maxWidth + 1 -
count.length(), true));
              }
              temp.append("\n");
            }
            String total = "  [total]";
            temp.append(pad(total, " ", maxAttWidth + 1 -
total.length(), false));
            for (int k = 0; k <
this.m_Instances.numClasses(); k++)
            {
              DiscreteEstimator d =
(DiscreteEstimator)this.m_Distributions[counter][k];
              String count = "" + d.getSumOfCounts();
              temp.append(pad(count, " ", maxWidth + 1 -
count.length(), true));
            }
            temp.append("\n\n");
          }
          else if ((this.m_Distributions[counter][0]
instanceof KernelEstimator))
          {
            String kL = "  [# kernels]";
            temp.append(pad(kL, " ", maxAttWidth + 1 -
kL.length(), false));
            for (int k = 0; k <
this.m_Instances.numClasses(); k++)
            {
              KernelEstimator ke =
(KernelEstimator)this.m_Distributions[counter][k];
```

```
              String nk = "" + ke.getNumKernels();
              temp.append(pad(nk, " ", maxWidth + 1 -
nk.length(), true));
            }
            temp.append("\n");

            String stdDevL = "  [std. dev]";
            temp.append(pad(stdDevL, " ", maxAttWidth + 1 -
stdDevL.length(), false));
            for (int k = 0; k <
this.m_Instances.numClasses(); k++)
            {
              KernelEstimator ke =
(KernelEstimator)this.m_Distributions[counter][k];
              String stdD =
Utils.doubleToString(ke.getStdDev(), maxWidth, 4).trim();

              temp.append(pad(stdD, " ", maxWidth + 1 -
stdD.length(), true));
            }
            temp.append("\n");
            String precL = "  [precision]";
            temp.append(pad(precL, " ", maxAttWidth + 1 -
precL.length(), false));
            for (int k = 0; k <
this.m_Instances.numClasses(); k++)
            {
              KernelEstimator ke =
(KernelEstimator)this.m_Distributions[counter][k];
              String prec =
Utils.doubleToString(ke.getPrecision(), maxWidth,
4).trim();

              temp.append(pad(prec, " ", maxWidth + 1 -
prec.length(), true));
            }
            temp.append("\n");
```

```
            int maxK = 0;
            for (int k = 0; k <
this.m_Instances.numClasses(); k++)
            {
              KernelEstimator ke =
(KernelEstimator)this.m_Distributions[counter][k];
              if (ke.getNumKernels() > maxK) {
                maxK = ke.getNumKernels();
              }
            }
            for (int j = 0; j < maxK; j++)
            {
              String meanL = "  K" + (j + 1) + ": mean
(weight)";
              temp.append(pad(meanL, " ", maxAttWidth + 1 -
meanL.length(), false));
              for (int k = 0; k <
this.m_Instances.numClasses(); k++)
              {
                KernelEstimator ke =
(KernelEstimator)this.m_Distributions[counter][k];
                double[] means = ke.getMeans();
                double[] weights = ke.getWeights();
                String m = "--";
                if (ke.getNumKernels() == 0)
                {
                  m = "0";
                }
                else if (j < ke.getNumKernels())
                {
                  m = Utils.doubleToString(means[j],
maxWidth, 4).trim();
                  m = m + " (" +
Utils.doubleToString(weights[j], maxWidth, 1).trim() + ")";
                }
                temp.append(pad(m, " ", maxWidth + 1 -
m.length(), true));
```

```
              }
              temp.append("\n");
            }
            temp.append("\n");
          }
          counter++;
        }
      }
    }
    return temp.toString();
  }

  protected String toStringOriginal()
  {
    StringBuffer text = new StringBuffer();

    text.append("Naive Bayes Classifier");
    if (this.m_Instances == null) {
      text.append(": No model built yet.");
    } else {
      try
      {
        for (int i = 0; i < this.m_Distributions[0].length;
i++)
        {
          text.append("\n\nClass " +
this.m_Instances.classAttribute().value(i) + ": Prior
probability = " +
Utils.doubleToString(this.m_ClassDistribution.getProbabilit
y(i), 4, 2) + "\n\n");

          Enumeration<Attribute> enumAtts =
this.m_Instances.enumerateAttributes();
          int attIndex = 0;
          while (enumAtts.hasMoreElements())
          {
```

```
            Attribute attribute =
(Attribute)enumAtts.nextElement();
            if (attribute.weight() > 0.0D) {
              text.append(attribute.name() + ":  " +
this.m_Distributions[attIndex][i]);
            }
            attIndex++;
          }
        }
      }
      catch (Exception ex)
      {
        text.append(ex.getMessage());
      }
    }
    return text.toString();
  }

  private String pad(String source, String padChar, int
length, boolean leftPad)
  {
    StringBuffer temp = new StringBuffer();
    if (leftPad)
    {
      for (int i = 0; i < length; i++) {
        temp.append(padChar);
      }
      temp.append(source);
    }
    else
    {
      temp.append(source);
      for (int i = 0; i < length; i++) {
        temp.append(padChar);
      }
```

```
    }
    return temp.toString();
  }

  public String useKernelEstimatorTipText()
  {
    return "Use a kernel estimator for numeric attributes
rather than a normal distribution.";
  }

  public boolean getUseKernelEstimator()
  {
    return this.m_UseKernelEstimator;
  }

  public void setUseKernelEstimator(boolean v)
  {
    this.m_UseKernelEstimator = v;
    if (v) {
      setUseSupervisedDiscretization(false);
    }
  }

  public String useSupervisedDiscretizationTipText()
  {
    return "Use supervised discretization to convert
numeric attributes to nominal ones.";
  }

  public boolean getUseSupervisedDiscretization()
  {
    return this.m_UseDiscretization;
  }
```

```
  public void setUseSupervisedDiscretization(boolean
newblah)
  {
    this.m_UseDiscretization = newblah;
    if (newblah) {
      setUseKernelEstimator(false);
    }
  }

  public String displayModelInOldFormatTipText()
  {
    return "Use old format for model output. The old format
is better when there are many class values. The new format
is better when there are fewer classes and many
attributes.";
  }

  public void setDisplayModelInOldFormat(boolean d)
  {
    this.m_displayModelInOldFormat = d;
  }

  public boolean getDisplayModelInOldFormat()
  {
    return this.m_displayModelInOldFormat;
  }

  public String getRevision()
  {
    return RevisionUtils.extract("$Revision: 10203 $");
  }

  public NaiveBayes aggregate(NaiveBayes toAggregate)
    throws Exception
  {
```

```
    if ((this.m_UseDiscretization) ||
(toAggregate.getUseSupervisedDiscretization())) {
      throw new Exception("Unable to aggregate when
supervised discretization has been turned on");
    }
    if
(!this.m_Instances.equalHeaders(toAggregate.m_Instances)) {
      throw new Exception("Can't aggregate - data headers
don't match: " +
this.m_Instances.equalHeadersMsg(toAggregate.m_Instances));
    }

((Aggregateable)this.m_ClassDistribution).aggregate(toAggre
gate.m_ClassDistribution);
    for (int i = 0; i < this.m_Distributions.length; i++) {
      for (int j = 0; j < this.m_Distributions[i].length;
j++) {

((Aggregateable)this.m_Distributions[i][j]).aggregate(toAgg
regate.m_Distributions[i][j]);
      }
    }
    return this;
  }

  public void finalizeAggregation()
    throws Exception
  {}

  public static void main(String[] argv)
  {
    runClassifier(new NaiveBayes(), argv);
  }
}
```

ANX 1.2. K-vizinhos (IBK)

```
package weka.classifiers.lazy;

import java.io.PrintStream;
import java.util.Collections;
import java.util.Enumeration;
import java.util.Vector;
import weka.classifiers.AbstractClassifier;
import weka.classifiers.UpdateableClassifier;
import weka.classifiers.rules.ZeroR;
import weka.core.AdditionalMeasureProducer;
import weka.core.Attribute;
import weka.core.Capabilities;
import weka.core.Capabilities.Capability;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.Option;
import weka.core.OptionHandler;
import weka.core.RevisionUtils;
import weka.core.SelectedTag;
import weka.core.Tag;
import weka.core.TechnicalInformation;
import weka.core.TechnicalInformation.Field;
import weka.core.TechnicalInformation.Type;
import weka.core.TechnicalInformationHandler;
import weka.core.Utils;
import weka.core.WeightedInstancesHandler;
import weka.core.neighboursearch.CoverTree;
import weka.core.neighboursearch.LinearNNSearch;
import weka.core.neighboursearch.NearestNeighbourSearch;

public class IBk
  extends AbstractClassifier
```

```
  implements OptionHandler, UpdateableClassifier,
WeightedInstancesHandler, TechnicalInformationHandler,
AdditionalMeasureProducer
{
  static final long serialVersionUID = -
3080186098777067172L;
  protected Instances m_Train;
  protected int m_NumClasses;
  protected int m_ClassType;
  protected int m_kNN;
  protected int m_kNNUpper;
  protected boolean m_kNNValid;
  protected int m_WindowSize;
  protected int m_DistanceWeighting;
  protected boolean m_CrossValidate;
  protected boolean m_MeanSquared;
  protected ZeroR m_defaultModel;
  public static final int WEIGHT_NONE = 1;
  public static final int WEIGHT_INVERSE = 2;
  public static final int WEIGHT_SIMILARITY = 4;
  public static final Tag[] TAGS_WEIGHTING = { new Tag(1,
"No distance weighting"), new Tag(2, "Weight by
1/distance"), new Tag(4, "Weight by 1-distance") };
  protected NearestNeighbourSearch m_NNSearch = new
LinearNNSearch();
  protected double m_NumAttributesUsed;

  public IBk(int k)
  {
    init();
    setKNN(k);
  }

  public IBk()
  {
    init();
```

```
  }

  public String globalInfo()
  {
    return "K-nearest neighbours classifier. Can select
appropriate value of K based on cross-validation. Can also
do distance weighting.\n\nFor more information, see\n\n" +
getTechnicalInformation().toString();
  }

  public TechnicalInformation getTechnicalInformation()
  {
    TechnicalInformation result = new
TechnicalInformation(TechnicalInformation.Type.ARTICLE);
    result.setValue(TechnicalInformation.Field.AUTHOR, "D.
Aha and D. Kibler");
    result.setValue(TechnicalInformation.Field.YEAR,
"1991");
    result.setValue(TechnicalInformation.Field.TITLE,
"Instance-based learning algorithms");
    result.setValue(TechnicalInformation.Field.JOURNAL,
"Machine Learning");
    result.setValue(TechnicalInformation.Field.VOLUME,
"6");
    result.setValue(TechnicalInformation.Field.PAGES, "37-
66");

    return result;
  }

  public String KNNTipText()
  {
    return "The number of neighbours to use.";
  }

  public void setKNN(int k)
  {
```

```
    this.m_kNN = k;
    this.m_kNNUpper = k;
    this.m_kNNValid = false;
  }

  public int getKNN()
  {
    return this.m_kNN;
  }

  public String windowSizeTipText()
  {
    return "Gets the maximum number of instances allowed in
the training pool. The addition of new instances above this
value will result in old instances being removed. A value
of 0 signifies no limit to the number of training
instances.";
  }

  public int getWindowSize()
  {
    return this.m_WindowSize;
  }

  public void setWindowSize(int newWindowSize)
  {
    this.m_WindowSize = newWindowSize;
  }

  public String distanceWeightingTipText()
  {
    return "Gets the distance weighting method used.";
  }

  public SelectedTag getDistanceWeighting()
```

```
  {
    return new SelectedTag(this.m_DistanceWeighting,
TAGS_WEIGHTING);
  }

  public void setDistanceWeighting(SelectedTag newMethod)
  {
    if (newMethod.getTags() == TAGS_WEIGHTING) {
      this.m_DistanceWeighting =
newMethod.getSelectedTag().getID();
    }
  }

  public String meanSquaredTipText()
  {
    return "Whether the mean squared error is used rather
than mean absolute error when doing cross-validation for
regression problems.";
  }

  public boolean getMeanSquared()
  {
    return this.m_MeanSquared;
  }

  public void setMeanSquared(boolean newMeanSquared)
  {
    this.m_MeanSquared = newMeanSquared;
  }

  public String crossValidateTipText()
  {
    return "Whether hold-one-out cross-validation will be
used to select the best k value between 1 and the value
specified as the KNN parameter.";
  }
```

```
  public boolean getCrossValidate()
  {
    return this.m_CrossValidate;
  }

  public void setCrossValidate(boolean newCrossValidate)
  {
    this.m_CrossValidate = newCrossValidate;
  }

  public String nearestNeighbourSearchAlgorithmTipText()
  {
    return "The nearest neighbour search algorithm to use
(Default: weka.core.neighboursearch.LinearNNSearch).";
  }

  public NearestNeighbourSearch
getNearestNeighbourSearchAlgorithm()
  {
    return this.m_NNSearch;
  }

  public void
setNearestNeighbourSearchAlgorithm(NearestNeighbourSearch
nearestNeighbourSearchAlgorithm)
  {
    this.m_NNSearch = nearestNeighbourSearchAlgorithm;
  }

  public int getNumTraining()
  {
    return this.m_Train.numInstances();
  }
```

```
  public Capabilities getCapabilities()
  {
    Capabilities result = super.getCapabilities();
    result.disableAll();


result.enable(Capabilities.Capability.NOMINAL_ATTRIBUTES);

result.enable(Capabilities.Capability.NUMERIC_ATTRIBUTES);
    result.enable(Capabilities.Capability.DATE_ATTRIBUTES);
    result.enable(Capabilities.Capability.MISSING_VALUES);

    result.enable(Capabilities.Capability.NOMINAL_CLASS);
    result.enable(Capabilities.Capability.NUMERIC_CLASS);
    result.enable(Capabilities.Capability.DATE_CLASS);

result.enable(Capabilities.Capability.MISSING_CLASS_VALUES)
;

    result.setMinimumNumberInstances(0);

    return result;
  }

  public void buildClassifier(Instances instances)
    throws Exception
  {
    getCapabilities().testWithFail(instances);

    instances = new Instances(instances);
    instances.deleteWithMissingClass();

    this.m_NumClasses = instances.numClasses();
    this.m_ClassType = instances.classAttribute().type();
    this.m_Train = new Instances(instances, 0,
instances.numInstances());
```

```
    if ((this.m_WindowSize > 0) &&
(instances.numInstances() > this.m_WindowSize)) {
      this.m_Train = new Instances(this.m_Train,
this.m_Train.numInstances() - this.m_WindowSize,
this.m_WindowSize);
    }
    this.m_NumAttributesUsed = 0.0D;
    for (int i = 0; i < this.m_Train.numAttributes(); i++)
{
      if ((i != this.m_Train.classIndex()) &&
((this.m_Train.attribute(i).isNominal()) ||
(this.m_Train.attribute(i).isNumeric()))) {
        this.m_NumAttributesUsed += 1.0D;
      }
    }
    this.m_NNSearch.setInstances(this.m_Train);

    this.m_kNNValid = false;

    this.m_defaultModel = new ZeroR();
    this.m_defaultModel.buildClassifier(instances);
  }

  public void updateClassifier(Instance instance)
    throws Exception
  {
    if (!this.m_Train.equalHeaders(instance.dataset())) {
      throw new Exception("Incompatible instance types\n" +
this.m_Train.equalHeadersMsg(instance.dataset()));
    }
    if (instance.classIsMissing()) {
      return;
    }
    this.m_Train.add(instance);
    this.m_NNSearch.update(instance);
    this.m_kNNValid = false;
```

```
    if ((this.m_WindowSize > 0) &&
(this.m_Train.numInstances() > this.m_WindowSize))
    {
      boolean deletedInstance = false;
      while (this.m_Train.numInstances() >
this.m_WindowSize)
      {
        this.m_Train.delete(0);
        deletedInstance = true;
      }
      if (deletedInstance == true) {
        this.m_NNSearch.setInstances(this.m_Train);
      }
    }
  }


  public double[] distributionForInstance(Instance
instance)
    throws Exception
  {
    if (this.m_Train.numInstances() == 0) {
      return
this.m_defaultModel.distributionForInstance(instance);
    }
    if ((this.m_WindowSize > 0) &&
(this.m_Train.numInstances() > this.m_WindowSize))
    {
      this.m_kNNValid = false;
      boolean deletedInstance = false;
      while (this.m_Train.numInstances() >
this.m_WindowSize) {
        this.m_Train.delete(0);
      }
      if (deletedInstance == true) {
        this.m_NNSearch.setInstances(this.m_Train);
      }
```

```
    }
    if ((!this.m_kNNValid) && (this.m_CrossValidate) &&
(this.m_kNNUpper >= 1)) {
      crossValidate();
    }
    this.m_NNSearch.addInstanceInfo(instance);

    Instances neighbours =
this.m_NNSearch.kNearestNeighbours(instance, this.m_kNN);
    double[] distances = this.m_NNSearch.getDistances();
    double[] distribution = makeDistribution(neighbours,
distances);

    return distribution;
  }

  public Enumeration<Option> listOptions()
  {
    Vector<Option> newVector = new Vector(7);

    newVector.addElement(new Option("\tWeight neighbours by
the inverse of their distance\n\t(use when k > 1)", "I", 0,
"-I"));

    newVector.addElement(new Option("\tWeight neighbours by
1 - their distance\n\t(use when k > 1)", "F", 0, "-F"));

    newVector.addElement(new Option("\tNumber of nearest
neighbours (k) used in classification.\n\t(Default = 1)",
"K", 1, "-K <number of neighbors>"));

    newVector.addElement(new Option("\tMinimise mean
squared error rather than mean absolute\n\terror when using
-X option with numeric prediction.", "E", 0, "-E"));

    newVector.addElement(new Option("\tMaximum number of
training instances maintained.\n\tTraining instances are
```

```
dropped FIFO. (Default = no window)", "W", 1, "-W <window
size>"));

    newVector.addElement(new Option("\tSelect the number of
nearest neighbours between 1\n\tand the k value specified
using hold-one-out evaluation\n\ton the training data (use
when k > 1)", "X", 0, "-X"));

    newVector.addElement(new Option("\tThe nearest
neighbour search algorithm to use (default:
weka.core.neighboursearch.LinearNNSearch).\n", "A", 0, "-
A"));

newVector.addAll(Collections.list(super.listOptions()));

    return newVector.elements();
  }

  public void setOptions(String[] options)
    throws Exception
  {
    String knnString = Utils.getOption('K', options);
    if (knnString.length() != 0) {
      setKNN(Integer.parseInt(knnString));
    } else {
      setKNN(1);
    }
    String windowString = Utils.getOption('W', options);
    if (windowString.length() != 0) {
      setWindowSize(Integer.parseInt(windowString));
    } else {
      setWindowSize(0);
    }
    if (Utils.getFlag('I', options)) {
      setDistanceWeighting(new SelectedTag(2,
TAGS_WEIGHTING));
```

```
    } else if (Utils.getFlag('F', options)) {
      setDistanceWeighting(new SelectedTag(4,
TAGS_WEIGHTING));
    } else {
      setDistanceWeighting(new SelectedTag(1,
TAGS_WEIGHTING));
    }
    setCrossValidate(Utils.getFlag('X', options));
    setMeanSquared(Utils.getFlag('E', options));

    String nnSearchClass = Utils.getOption('A', options);
    if (nnSearchClass.length() != 0)
    {
      String[] nnSearchClassSpec =
Utils.splitOptions(nnSearchClass);
      if (nnSearchClassSpec.length == 0) {
        throw new Exception("Invalid NearestNeighbourSearch
algorithm specification string.");
      }
      String className = nnSearchClassSpec[0];
      nnSearchClassSpec[0] = "";


setNearestNeighbourSearchAlgorithm((NearestNeighbourSearch)
Utils.forName(NearestNeighbourSearch.class, className,
nnSearchClassSpec));
    }
    else
    {
      setNearestNeighbourSearchAlgorithm(new
LinearNNSearch());
    }
    super.setOptions(options);

    Utils.checkForRemainingOptions(options);
  }
```

```
  public String[] getOptions()
  {
    Vector<String> options = new Vector();
    options.add("-K");options.add("" + getKNN());
    options.add("-W");options.add("" + this.m_WindowSize);
    if (getCrossValidate()) {
      options.add("-X");
    }
    if (getMeanSquared()) {
      options.add("-E");
    }
    if (this.m_DistanceWeighting == 2) {
      options.add("-I");
    } else if (this.m_DistanceWeighting == 4) {
      options.add("-F");
    }
    options.add("-A");
    options.add(this.m_NNSearch.getClass().getName() + " "
+ Utils.joinOptions(this.m_NNSearch.getOptions()));

    Collections.addAll(options, super.getOptions());

    return (String[])options.toArray(new String[0]);
  }

  public Enumeration<String> enumerateMeasures()
  {
    if (this.m_CrossValidate)
    {
      Enumeration<String> enm =
this.m_NNSearch.enumerateMeasures();
      Vector<String> measures = new Vector();
      while (enm.hasMoreElements()) {
        measures.add(enm.nextElement());
      }
```

```
      measures.add("measureKNN");
      return measures.elements();
    }
    return this.m_NNSearch.enumerateMeasures();
  }

  public double getMeasure(String additionalMeasureName)
  {
    if (additionalMeasureName.equals("measureKNN")) {
      return this.m_kNN;
    }
    return
this.m_NNSearch.getMeasure(additionalMeasureName);
  }

  public String toString()
  {
    if (this.m_Train == null) {
      return "IBk: No model built yet.";
    }
    if (this.m_Train.numInstances() == 0) {
      return "Warning: no training instances - ZeroR model
used.";
    }
    if ((!this.m_kNNValid) && (this.m_CrossValidate)) {
      crossValidate();
    }
    String result = "IB1 instance-based classifier\nusing "
+ this.m_kNN;
    switch (this.m_DistanceWeighting)
    {
    case 2:
      result = result + " inverse-distance-weighted";
      break;
    case 4:
```

```
      result = result + " similarity-weighted";
    }
    result = result + " nearest neighbour(s) for
classification\n";
    if (this.m_WindowSize != 0) {
      result = result + "using a maximum of " +
this.m_WindowSize + " (windowed) training instances\n";
    }
    return result;
  }

  protected void init()
  {
    setKNN(1);
    this.m_WindowSize = 0;
    this.m_DistanceWeighting = 1;
    this.m_CrossValidate = false;
    this.m_MeanSquared = false;
  }

  protected double[] makeDistribution(Instances neighbours,
double[] distances)
    throws Exception
  {
    double total = 0.0D;
    double[] distribution = new double[this.m_NumClasses];
    if (this.m_ClassType == 1)
    {
      for (int i = 0; i < this.m_NumClasses; i++) {
        distribution[i] = (1.0D / Math.max(1,
this.m_Train.numInstances()));
      }
      total = this.m_NumClasses / Math.max(1,
this.m_Train.numInstances());
    }
    for (int i = 0; i < neighbours.numInstances(); i++)
```

```
    {
      Instance current = neighbours.instance(i);
      distances[i] *= distances[i];
      distances[i] = Math.sqrt(distances[i] /
this.m_NumAttributesUsed);
      double weight;
      switch (this.m_DistanceWeighting)
      {
      case 2:
        weight = 1.0D / (distances[i] + 0.001D);
        break;
      case 4:
        weight = 1.0D - distances[i];
        break;
      default:
        weight = 1.0D;
      }
      weight *= current.weight();
      try
      {
        switch (this.m_ClassType)
        {
        case 1:
          distribution[((int)current.classValue())] +=
weight;
          break;
        case 0:
          distribution[0] += current.classValue() * weight;
        }
      }
      catch (Exception ex)
      {
        throw new Error("Data has no class attribute!");
      }
      total += weight;
```

```
    }
    if (total > 0.0D) {
      Utils.normalize(distribution, total);
    }
    return distribution;
  }

  protected void crossValidate()
  {
    try
    {
      if ((this.m_NNSearch instanceof CoverTree)) {
        throw new Exception("CoverTree doesn't support
hold-one-out cross-validation. Use some other NN method.");
      }
      double[] performanceStats = new
double[this.m_kNNUpper];
      double[] performanceStatsSq = new
double[this.m_kNNUpper];
      for (int i = 0; i < this.m_kNNUpper; i++)
      {
        performanceStats[i] = 0.0D;
        performanceStatsSq[i] = 0.0D;
      }
      this.m_kNN = this.m_kNNUpper;
      for (int i = 0; i < this.m_Train.numInstances(); i++)
      {
        if ((this.m_Debug) && (i % 50 == 0)) {
          System.err.print("Cross validating " + i + "/" +
this.m_Train.numInstances() + "\r");
        }
        Instance instance = this.m_Train.instance(i);
        Instances neighbours =
this.m_NNSearch.kNearestNeighbours(instance, this.m_kNN);
        double[] origDistances =
this.m_NNSearch.getDistances();
```

```
        for (int j = this.m_kNNUpper - 1; j >= 0; j--)
        {
          double[] convertedDistances = new
double[origDistances.length];
          System.arraycopy(origDistances, 0,
convertedDistances, 0, origDistances.length);

          double[] distribution =
makeDistribution(neighbours, convertedDistances);

          double thisPrediction =
Utils.maxIndex(distribution);
          if (this.m_Train.classAttribute().isNumeric())
          {
            thisPrediction = distribution[0];
            double err = thisPrediction -
instance.classValue();
            performanceStatsSq[j] += err * err;
            performanceStats[j] += Math.abs(err);
          }
          else if (thisPrediction != instance.classValue())
          {
            performanceStats[j] += 1.0D;
          }
          if (j >= 1) {
            neighbours = pruneToK(neighbours,
convertedDistances, j);
          }
        }
      }
      for (int i = 0; i < this.m_kNNUpper; i++)
      {
        if (this.m_Debug) {
          System.err.print("Hold-one-out performance of " +
(i + 1) + " neighbors ");
        }
```

```
        if (this.m_Train.classAttribute().isNumeric())
        {
          if (this.m_Debug) {
            if (this.m_MeanSquared) {
              System.err.println("(RMSE) = " +
Math.sqrt(performanceStatsSq[i] /
this.m_Train.numInstances()));
            } else {
              System.err.println("(MAE) = " +
performanceStats[i] / this.m_Train.numInstances());
            }
          }
        }
        else if (this.m_Debug) {
          System.err.println("(%ERR) = " + 100.0D *
performanceStats[i] / this.m_Train.numInstances());
        }
      }
      double[] searchStats = performanceStats;
      if ((this.m_Train.classAttribute().isNumeric()) &&
(this.m_MeanSquared)) {
        searchStats = performanceStatsSq;
      }
      double bestPerformance = NaN.0D;
      int bestK = 1;
      for (int i = 0; i < this.m_kNNUpper; i++) {
        if ((Double.isNaN(bestPerformance)) ||
(bestPerformance > searchStats[i]))
        {
          bestPerformance = searchStats[i];
          bestK = i + 1;
        }
      }
      this.m_kNN = bestK;
      if (this.m_Debug) {
        System.err.println("Selected k = " + bestK);
```

```
      }
      this.m_kNNValid = true;
    }
    catch (Exception ex)
    {
      throw new Error("Couldn't optimize by cross-
validation: " + ex.getMessage());
    }
  }


  public Instances pruneToK(Instances neighbours, double[]
distances, int k)
  {
    if ((neighbours == null) || (distances == null) ||
(neighbours.numInstances() == 0)) {
      return null;
    }
    if (k < 1) {
      k = 1;
    }
    int currentK = 0;
    for (int i = 0; i < neighbours.numInstances(); i++)
    {
      currentK++;
      double currentDist = distances[i];
      if ((currentK > k) && (currentDist != distances[(i -
1)]))
      {
        currentK--;
        neighbours = new Instances(neighbours, 0,
currentK);
        break;
      }
    }
    return neighbours;
  }
```

```
  public String getRevision()
  {
    return RevisionUtils.extract("$Revision: 10141 $");
  }

  public static void main(String[] argv)
  {
    runClassifier(new IBk(), argv);
  }
}
```

ANX 1.3. Árvore de decisão C4.5 (J48)

```
package weka.classifiers.trees.j48;

import java.util.Enumeration;
import weka.core.Attribute;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.RevisionUtils;
import weka.core.Utils;

public class C45Split
  extends ClassifierSplitModel
{
  private static final long serialVersionUID =
3064079330067903161L;
  protected int m_complexityIndex;
  protected final int m_attIndex;
  protected final int m_minNoObj;
  protected final boolean m_useMDLcorrection;
  protected double m_splitPoint;
  protected double m_infoGain;
  protected double m_gainRatio;
  protected final double m_sumOfWeights;
  protected int m_index;
  protected static InfoGainSplitCrit infoGainCrit = new
InfoGainSplitCrit();
  protected static GainRatioSplitCrit gainRatioCrit = new
GainRatioSplitCrit();

  public C45Split(int attIndex, int minNoObj, double
sumOfWeights, boolean useMDLcorrection)
  {
    this.m_attIndex = attIndex;

    this.m_minNoObj = minNoObj;
```

```
    this.m_sumOfWeights = sumOfWeights;

    this.m_useMDLcorrection = useMDLcorrection;
  }

  public void buildClassifier(Instances trainInstances)
    throws Exception
  {
    this.m_numSubsets = 0;
    this.m_splitPoint = Double.MAX_VALUE;
    this.m_infoGain = 0.0D;
    this.m_gainRatio = 0.0D;
    if
(trainInstances.attribute(this.m_attIndex).isNominal())
    {
      this.m_complexityIndex =
trainInstances.attribute(this.m_attIndex).numValues();
      this.m_index = this.m_complexityIndex;
      handleEnumeratedAttribute(trainInstances);
    }
    else
    {
      this.m_complexityIndex = 2;
      this.m_index = 0;

trainInstances.sort(trainInstances.attribute(this.m_attInde
x));
      handleNumericAttribute(trainInstances);
    }
  }

  public final int attIndex()
  {
    return this.m_attIndex;
```

```
  }

  public double splitPoint()
  {
    return this.m_splitPoint;
  }

  public final double classProb(int classIndex, Instance
instance, int theSubset)
    throws Exception
  {
    if (theSubset <= -1)
    {
      double[] weights = weights(instance);
      if (weights == null) {
        return this.m_distribution.prob(classIndex);
      }
      double prob = 0.0D;
      for (int i = 0; i < weights.length; i++) {
        prob += weights[i] *
this.m_distribution.prob(classIndex, i);
      }
      return prob;
    }
    if (Utils.gr(this.m_distribution.perBag(theSubset),
0.0D)) {
      return this.m_distribution.prob(classIndex,
theSubset);
    }
    return this.m_distribution.prob(classIndex);
  }

  public final double codingCost()
  {
    return Utils.log2(this.m_index);
```

```
  }

  public final double gainRatio()
  {
    return this.m_gainRatio;
  }

  private void handleEnumeratedAttribute(Instances
trainInstances)
    throws Exception
  {
    this.m_distribution = new
Distribution(this.m_complexityIndex,
trainInstances.numClasses());

    Enumeration<Instance> enu =
trainInstances.enumerateInstances();
    while (enu.hasMoreElements())
    {
      Instance instance = (Instance)enu.nextElement();
      if (!instance.isMissing(this.m_attIndex)) {

this.m_distribution.add((int)instance.value(this.m_attIndex
), instance);
      }
    }
    if (this.m_distribution.check(this.m_minNoObj))
    {
      this.m_numSubsets = this.m_complexityIndex;
      this.m_infoGain =
infoGainCrit.splitCritValue(this.m_distribution,
this.m_sumOfWeights);
      this.m_gainRatio =
gainRatioCrit.splitCritValue(this.m_distribution,
this.m_sumOfWeights, this.m_infoGain);
    }
  }
```

```
  private void handleNumericAttribute(Instances
trainInstances)
    throws Exception
  {
    int next = 1;
    int last = 0;
    int splitIndex = -1;


    this.m_distribution = new Distribution(2,
trainInstances.numClasses());


    Enumeration<Instance> enu =
trainInstances.enumerateInstances();
    int i = 0;
    while (enu.hasMoreElements())
    {
      Instance instance = (Instance)enu.nextElement();
      if (instance.isMissing(this.m_attIndex)) {
        break;
      }
      this.m_distribution.add(1, instance);
      i++;
    }
    int firstMiss = i;


    double minSplit = 0.1D * this.m_distribution.total() /
trainInstances.numClasses();
    if (Utils.smOrEq(minSplit, this.m_minNoObj)) {
      minSplit = this.m_minNoObj;
    } else if (Utils.gr(minSplit, 25.0D)) {
      minSplit = 25.0D;
    }
    if (Utils.sm(firstMiss, 2.0D * minSplit)) {
      return;
```

```
    }
    double defaultEnt =
infoGainCrit.oldEnt(this.m_distribution);
    while (next < firstMiss)
    {
      if (trainInstances.instance(next -
1).value(this.m_attIndex) + 1.0E-5D <
trainInstances.instance(next).value(this.m_attIndex))
      {
        this.m_distribution.shiftRange(1, 0,
trainInstances, last, next);
        if ((Utils.grOrEq(this.m_distribution.perBag(0),
minSplit)) && (Utils.grOrEq(this.m_distribution.perBag(1),
minSplit)))
        {
          double currentInfoGain =
infoGainCrit.splitCritValue(this.m_distribution,
this.m_sumOfWeights, defaultEnt);
          if (Utils.gr(currentInfoGain, this.m_infoGain))
          {
            this.m_infoGain = currentInfoGain;
            splitIndex = next - 1;
          }
          this.m_index += 1;
        }
        last = next;
      }
      next++;
    }
    if (this.m_index == 0) {
      return;
    }
    if (this.m_useMDLcorrection) {
      this.m_infoGain -= Utils.log2(this.m_index) /
this.m_sumOfWeights;
    }
    if (Utils.smOrEq(this.m_infoGain, 0.0D)) {
```

```
      return;
    }
    this.m_numSubsets = 2;
    this.m_splitPoint =
((trainInstances.instance(splitIndex +
1).value(this.m_attIndex) +
trainInstances.instance(splitIndex).value(this.m_attIndex))
/ 2.0D);
    if (this.m_splitPoint ==
trainInstances.instance(splitIndex +
1).value(this.m_attIndex)) {
      this.m_splitPoint =
trainInstances.instance(splitIndex).value(this.m_attIndex);
    }
    this.m_distribution = new Distribution(2,
trainInstances.numClasses());
    this.m_distribution.addRange(0, trainInstances, 0,
splitIndex + 1);
    this.m_distribution.addRange(1, trainInstances,
splitIndex + 1, firstMiss);


    this.m_gainRatio =
gainRatioCrit.splitCritValue(this.m_distribution,
this.m_sumOfWeights, this.m_infoGain);
  }


  public final double infoGain()
  {
    return this.m_infoGain;
  }


  public final String leftSide(Instances data)
  {
    return data.attribute(this.m_attIndex).name();
  }


  public final String rightSide(int index, Instances data)
  {
```

```
    StringBuffer text = new StringBuffer();
    if (data.attribute(this.m_attIndex).isNominal()) {
      text.append(" = " +
data.attribute(this.m_attIndex).value(index));
    } else if (index == 0) {
      text.append(" <= " +
Utils.doubleToString(this.m_splitPoint, 6));
    } else {
      text.append(" > " +
Utils.doubleToString(this.m_splitPoint, 6));
    }
    return text.toString();
  }


  public final String sourceExpression(int index, Instances
data)
  {
    StringBuffer expr = null;
    if (index < 0) {
      return "i[" + this.m_attIndex + "] == null";
    }
    if (data.attribute(this.m_attIndex).isNominal())
    {
      expr = new StringBuffer("i[");
      expr.append(this.m_attIndex).append("]");

expr.append(".equals(\"").append(data.attribute(this.m_attI
ndex).value(index)).append("\")");
    }
    else
    {
      expr = new StringBuffer("((Double) i[");
      expr.append(this.m_attIndex).append("])");
      if (index == 0) {
        expr.append(".doubleValue() <=
").append(this.m_splitPoint);
```

```
      } else {
        expr.append(".doubleValue() >
").append(this.m_splitPoint);
      }
    }
    return expr.toString();
  }

  public final void setSplitPoint(Instances allInstances)
  {
    double newSplitPoint = -1.7976931348623157E308D;
    if
((allInstances.attribute(this.m_attIndex).isNumeric()) &&
(this.m_numSubsets > 1))
    {
      Enumeration<Instance> enu =
allInstances.enumerateInstances();
      while (enu.hasMoreElements())
      {
        Instance instance = (Instance)enu.nextElement();
        if (!instance.isMissing(this.m_attIndex))
        {
          double tempValue =
instance.value(this.m_attIndex);
          if ((Utils.gr(tempValue, newSplitPoint)) &&
(Utils.smOrEq(tempValue, this.m_splitPoint))) {
            newSplitPoint = tempValue;
          }
        }
      }
      this.m_splitPoint = newSplitPoint;
    }
  }

  public final double[][] minsAndMaxs(Instances data,
double[][] minsAndMaxs, int index)
```

```
  {
    double[][] newMinsAndMaxs = new
double[data.numAttributes()][2];
    for (int i = 0; i < data.numAttributes(); i++)
    {
      newMinsAndMaxs[i][0] = minsAndMaxs[i][0];
      newMinsAndMaxs[i][1] = minsAndMaxs[i][1];
      if (i == this.m_attIndex) {
        if (data.attribute(this.m_attIndex).isNominal()) {
          newMinsAndMaxs[this.m_attIndex][1] = 1.0D;
        } else {
          newMinsAndMaxs[this.m_attIndex][(1 - index)] =
this.m_splitPoint;
        }
      }
    }
    return newMinsAndMaxs;
  }

  public void resetDistribution(Instances data)
    throws Exception
  {
    Instances insts = new Instances(data,
data.numInstances());
    for (int i = 0; i < data.numInstances(); i++) {
      if (whichSubset(data.instance(i)) > -1) {
        insts.add(data.instance(i));
      }
    }
    Distribution newD = new Distribution(insts, this);
    newD.addInstWithUnknown(data, this.m_attIndex);
    this.m_distribution = newD;
  }

  public final double[] weights(Instance instance)
```

```
  {
    if (instance.isMissing(this.m_attIndex))
    {
      double[] weights = new double[this.m_numSubsets];
      for (int i = 0; i < this.m_numSubsets; i++) {
        weights[i] = (this.m_distribution.perBag(i) /
this.m_distribution.total());
      }
      return weights;
    }
    return null;
  }

  public final int whichSubset(Instance instance)
    throws Exception
  {
    if (instance.isMissing(this.m_attIndex)) {
      return -1;
    }
    if (instance.attribute(this.m_attIndex).isNominal()) {
      return (int)instance.value(this.m_attIndex);
    }
    if (Utils.smOrEq(instance.value(this.m_attIndex),
this.m_splitPoint)) {
      return 0;
    }
    return 1;
  }

  public String getRevision()
  {
    return RevisionUtils.extract("$Revision: 10531 $");
  }
}
```