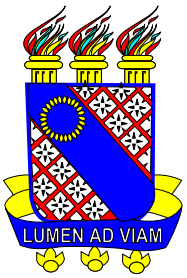

UNIVERSIDADE ESTADUAL DO CEARÁ

JOÃO DAVID DE LIRA

UMA COMPARAÇÃO DE ALGORITMOS DE APRENDIZADO DE MÁQUINA APLICADOS À CLASSIFICAÇÃO DE TRÁFEGO DE REDE

FORTALEZA - CEARÁ

2010

JOÃO DAVID DE LIRA

UMA COMPARAÇÃO DE ALGORITMOS DE APRENDIZADO DE MÁQUINA APLICADOS À CLASSIFICAÇÃO DE TRÁFEGO DE REDE

Monografia apresentada no Curso de Graduação em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: José Everardo Bessa Maia

FORTALEZA - CEARÁ

2010

L768c Lira, João David de.

Uma comparação de algoritmos de aprendizado de máquina aplicados à classificação de tráfego de rede / João David de Lira. – Fortaleza, 2010.

99 p.;il.

Orientador: Prof. Dr. José Everardo Bessa Maia

Monografia (Graduação em Ciência da Computação) - Universidade Estadual do Ceará, Centro de Ciências e Tecnologia.

1. Tráfego de rede 2. Aprendizado de máquina 3. C4.5 4. Naive Bayes 5. Curvas ROC I. Universidade Estadual do Ceará, Centro de Ciências e Tecnologia.

CDD:004.6

JOÃO DAVID DE LIRA

UMA COMPARAÇÃO DE ALGORITMOS DE APRENDIZADO DE MÁQUINA APLICADOS À CLASSIFICAÇÃO DE TRÁFEGO DE REDE

Monografia apresentada no Curso de Graduação em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial para obtenção do grau de Bacharel.

Aprovada em: 27/10/2010

BANCA EXAMINADORA

Prof. Dr. José Everardo Bessa Maia
Universidade Estadual do Ceará – UECE
Orientador

Prof. Dr. Gustavo Augusto Lima de Campos
Universidade Estadual do Ceará – UECE

Prof. Dr. Joaquim Celestino Júnior
Universidade Estadual do Ceará – UECE

AGRADECIMENTOS

À minha família por todo apoio e suporte durante todos os meus anos de estudos e vida. Principalmente ao meu irmão, João Daniel, o qual me apresentou a Ciência da Computação como algo aplicado e tangível, e por sua grande contribuição ao meu ingresso, percuso e conclusão do curso de Ciência da Computação.

Aos grandes amigos que me incentivaram durante todo o trajeto desta jornada. E, em especial, aos colegas que contribuíram diretamente no desenvolvimento deste trabalho, entre eles Silas Santiago, Cecília Rebouças, Rafael Carmo, Fabrício Gomes, Kayrena Melo, Rudy Matela e Sérgio Luis.

Agradecimento à Anna Sperotto, da Universidade de Twente, Holanda, por ter gentilmente feito apontamentos e esclarecimentos sobre a base de dados descrita em (SPEROTTO et al., 2009), e fornecido material sobre a mesma, parte do qual reproduzido aqui.

Aos professores que me instruíram e, em especial ao professor Edson Pessoa, que ministrou as duas melhores disciplinas que tive durante todo o curso, com as quais apresentou um horizonte de conhecimento muito além do que supunha referente ao curso, e a computação como uma ciência em termos e forma, além de reavivar meu interesse pela área.

E ao professor José Everardo Bessa Maia pela orientação, paciência e compreensão durante o desenvolvimento deste trabalho.

“Não tenhamos pressa, mas não percamos tempo.”
José Saramargo

RESUMO

Classificação do tráfego de Internet é um tema de importância crescente na medida da expansão do papel da Internet na vida da sociedade. Classificação de tráfego tem pelo menos duas aplicações relevantes na gerência de rede: classificação do tráfego para gerência de Qualidade de Serviço (QoS) e Detecção de Intrusos baseada em classificação, na área de segurança de redes. Este trabalho analisa a performance de dois classificadores, o algoritmo C4.5 e o classificador naive Bayes, sobre três conjuntos de dados de tráfego disponíveis na Internet. O primeiro e o segundo conjunto de dados são uma mistura de tráfego normal com tráfego contendo ameaças e ataques de rede, e o terceiro conjunto contém oito classes de aplicações. A unidade utilizada na classificação é o fluxo.

Palavras-Chave: Tráfego de rede. Aprendizado de máquina. C4.5. Naive Bayes. Curvas ROC.

ABSTRACT

Internet traffic Classification is an issue of growing importance as the growth of the Internet's role in society. Traffic classification has at least two relevant applications in network management: traffic classification for management of Quality of Service (QoS) and Intrusion Detection based on classification. This paper analyzes the performance of two classifiers, the algorithm C4.5 and naive Bayes classifier, on three sets of traffic data. The first and second sets of data are a mixture of normal traffic with traffic containing threats and network attacks, and the third set contains eight classes of applications. The unit used to classify is the flow.

Keywords: Network traffic. Machine learning. C4.5. Naive Bayes. ROC Curves.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

FP	False Positive
FTP	File Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
RFC	Request for Comments
ROC	Receiver Operating Characteristics
SSH	Secure Shell
TCP	Transmission Cronrol Protocol
TP	True Positive
UDP	User Datagram Protocol
WEKA	Waikato Environment for Knowledge Analysis

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

A administração de redes é uma tarefa altamente dependente das informações que se pode extrair destas sobre seu tráfego. Planejamento de expansão de hardware, *redesign* da sua topologia física, análise de incidentes e diversas outras atividades são executadas pelo administrador com base nas informações a que tem acesso sobre a rede e o trânsito de dados que ocorre nela, fazendo, com estas, projeções de crescimento, tomando medidas que visam adequar a rede ao tráfego atual e futuro, além de coibir sua utilização por aplicações desnecessárias, ilegais ou maliciosas. O reconhecimento de incidentes na rede, dentre eles ataques e tentativas de invasões, em tempo hábil para resposta ou, em momento posterior, visando auditoria, é uma habilidade indispensável a qualquer administrador. Se um ataque, ou mesmo constantes tentativas, é um cenário ruim, não estar sequer ciente disto é inaceitável.

Há diversos pontos na rede de onde se pode extrair informação. Seja dos hospedeiro ou mesmo dos dispositivos de rede, como roteadores ou comutadores da camada de enlace, cabe ao administrador coletar informações para tornar seu trabalho possível. Após um ataque, uma auditoria em um servidor invadido pode apresentar diversas indicações da intenção e das ações tomadas pelo invasor. Atividades simples, como a auditoria das conexões remotas em servidor, mesmo quando não se tem indícios de ataque, pode ser muito reveladora.

Porém, mesmo com todas as possibilidades existentes para coleta de informações, em diversos momentos existirão restrições que dificultarão ou impossibilitarão essa coleta. Em alguns momentos, até mesmo dados passíveis de serem coletados terão que ser ignorados para atender determinadas restrições. Estas restrições podem variar, contudo podemos resumi-las de forma simplificada nos seguintes tipos (KARAGIANNIS; PAPAGIANNAKI; FALOUTSOS, 2005):

1. Os usuários podem tornar a informação existente inconsistente a fim de utilizar recursos de rede bloqueados em condições normais;

2. Existem diversas restrições legais ao acesso e utilização da informação do usuário da rede sem sua permissão, visando respeitar sua privacidade, e seu desrespeito torna o responsável passível de ação judicial de acordo com a legislação local vigente;

3. Os usuários e aplicações podem, intencionalmente, fazer com que os dados transmitidos tornem-se inacessíveis a qualquer outra pessoa, exceto o receptor, através da codificação da transmissão (criptografia), fazendo com isso que informações relevantes à administração da rede, presentes junto a esses dados, também tornem-se inacessíveis.

Quanto às informações que se pode extrair da rede, o nível de profundidade destas varia conforme o ambiente e as restrições impostas por este, que impedem o acesso a determinadas informações. Com base nas informações disponíveis, uma atividade importante à tomada de decisões do administrador é classificar o tráfego existente na rede. Classificar o tráfego consiste em, através de técnicas diversas, categorizar o tráfego existente na rede como provenientes de categorias de aplicações específicas. De acordo com o nível de informação disponível, uma classificação mais precisa será possível. Ter conhecimento do tipo de tráfego que está em trânsito na rede propicia ao administrador informação necessárias para bloquear ou redirecionar tráfego ao qual a rede não se destina, tarifar adequadamente serviços sendo utilizados, responder à ataques e outras ações. Além disso, classificação de pacotes é uma técnica atrativa para sistemas de detecção de intrusão (IDS)(HALOI, 2004).

Em ambientes com restrições severas, as técnicas tradicionais de classificação de tráfego de rede se tornam pouco efetivas. Assim, técnicas alternativas, baseadas em análise estatísticas, como apresentadas em (KARAGIANNIS; PAPAGIANNAKI; FALOUTSOS, 2005) e em (ZUEV; MOORE, 2005), e em heurísticas, como apresentadas em (HALOI, 2004), vem sendo propostas para superar essas limitações.

Uma técnica de classificação de tráfego presente na literatura baseia-se na utilização de algoritmos de aprendizado de máquina (AULD; MOORE; GULL, 2006). Abordamos neste trabalho a utilização de algoritmos de aprendizado de máquina para a classificação de tráfego de rede sob restrições severas utilizando, como informação coletada, fluxos de rede, que podem ser coletados mesmo mesmo sob as restrições apresentadas.

Assim, este trabalho apresenta uma comparação de dois algoritmos de aprendizado de máquina aplicados na classificação de tráfego de rede sob o ambiente das restrições apresentadas, baseando-se na coleta de fluxos de rede. Os algoritmos abordados são: C4.5 e naive Bayes.

1.2 Objetivos

Objetivo Geral é analisar e comparar o desempenho dos classificadores C4.5 e naive Bayes na classificação de tráfego Internet baseada em fluxos de rede.

Objetivos específicos são:

1. Revisar o princípio de funcionamento do classificador C4.5;
2. Revisar o princípio de funcionamento do classificador naive Bayes;
3. Revisar as técnicas e medidas de avaliação de desempenho de classificadores;

4. Avaliar o desempenho dos classificadores C4.5 e naive Bayes sobre três conjuntos de dados nas áreas de classificação de aplicações e detecção de intrusos em redes.

1.3 Metodologia

A metodologia empregada neste trabalho consistiu em utilizar ferramentas de classificadores, software livre, para avaliar a performance dos algoritmos em foco sobre dados publicamente disponíveis na internet. A ferramenta escolhida para tal foi o software WEKA. Os passos da metodologia são os seguintes:

1. Estudo e exercícios da utilização do software WEKA;
2. Estudo, compreensão e descrição do classificador C4.5;
3. Estudo, compreensão e descrição do classificador naive Bayes;
4. Estudo, compreensão e descrição de medidas de performance e técnicas de comparação da performance de classificadores;
5. Obtenção e compreensão dos conjuntos de dados de tráfego com classes de aplicações;
6. Obtenção e compreensão dos conjuntos de dados de tráfego com ameaças e ataques de redes;
7. Execução de testes, registro e comparação dos resultados de desempenho dos classificadores;
8. Elaboração da monografia.

1.4 Organização do Trabalho

Após esta seção de introdução, é feita uma apresentação de conceitos e fundamentos necessários à compreensão do trabalho no capítulo 2. No capítulo 3 é apresentada uma revisão dos princípios conceituais dos algoritmos C4.5 e naive Bayes. No capítulo 4 são descritas as bases de dados utilizadas nos experimentos. No capítulo 5 os resultados de desempenho são apresentados e discutidos. As conclusões são apresentadas no capítulo 6.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Classificação de tráfego de rede

Este trabalho tem como foco principal a classificação de tráfego rede. Inciamos, assim, esclarecendo ao leitor o que é esta atividade e sua importância.

Segundo o dicionário Houaiss (HOUAISS, 2001), temos o verbete classificar como "distribuir em classes e nos respectivos grupos, de acordo com um sistema ou método de classificação".

Essa definição se aplica ao nosso contexto, porém devemos especificar o que queremos classificar, em quantos e quais grupos e com que método de classificação.

Primeiramente, desejamos classificar o tráfego de informações contido em uma rede qualquer, ou seja, desejamos categorizar a informação que está sendo transmitida pela rede em qualquer momento. Porém a informação não transita na rede em sua forma natural. Todo dado transita na rede na forma de **bits**, ou dígitos binários. Os *bits* são agrupados em **quadros**, e estes quadros em **pacotes** que, por sua vez, são agrupados em **segmentos**. Consideramos aqui estes segmentos. Assim, temos que todo dado, seja informação útil ao usuário ou não, que transita na rede é constituída de *bits*, que são agrupados em unidades maiores para seu transporte na rede, e acrescidos de meta informação sobre esse agrupamento.

Devemos definir ainda o que consideramos informação. No nosso contexto, informação são dados transmitidos entre aplicações da camada de aplicação da rede. Isso quer dizer que não consideramos informação simplesmente um arquivo ou dados do usuário, pois estes não são simplesmente empurrados para dentro da rede. Consideramos informação a troca de dados entre duas aplicações da camada de aplicação da rede, considerando não apenas os dados do usuário, mas também as informações adicionais acrescentadas pela camada de aplicação para a manipulação dessa informação pela aplicação, ou seja, todo o *overhead* gerado, como, por exemplo, os dados do protocolo da camada de aplicação. Contudo, não consideramos as informações de outras camadas, ou seja, não nos preocupamos em classificar o que está abaixo da camada de aplicação, o que não nos impede de usar esta informação, caso esteja disponível, para nos ajudar em nossa tarefa.

Em outras palavras, consideramos como informação a ser classificada todo o conteúdo do *payload* dos segmentos da camada de transporte, ou seja, toda a informação da camada de aplicação.

Aqui é válido fazer uma pequena observação sobre o que visamos capturar na rede para proceder com nossa classificação. Como acabamos de mencionar, o trabalho de classificação deve ocorrer sobre o conteúdo dos segmentos, porém capturar essa informação na rede requer

um dispositivo que implemente até a quarta camada da pilha de protocolo TCP/IP, o que não é comum para equipamentos de redes, e sim para hospedeiros. Por isso, é comum que haja a captura de pacotes da camada de redes, o que pode ser feitos em roteadores, e posteriormente um tratamento é feito nesses dados capturados para que ocorra a análise. Essa observação se faz necessária pois pode causar confusão ao leitor estarmos tratando durante todo o texto sobre a classificação do *payload* de segmentos e estarmos, contudo, manipulando pacotes da rede.

Definido o que desejamos classificar, devemos explicitar em quais grupos desejamos fazê-lo.

A classificação deve ser feita com base em alguma característica específica do tráfego analisado, característica esta que não necessariamente deve ser evidente, porém da qual desejamos estar cientes para promover uma mesma ação para todas as instâncias que a apresentarem. Como exemplo, podemos citar a classificação do tráfego em relação ao protocolo da camada de aplicação utilizado em cada instância, na qual poderíamos ter um grupo para cada tipo de protocolo existente que tivéssemos ciência e mais um grupo para as instâncias que não fossem classificadas por quaisquer motivos.

É importante ressaltar que a existência de meta-grupos para aquelas instâncias que não foram classificadas ou que não se encaixam em nenhum outro grupo definido deve ser levantada durante a definição da classificação a ser utilizada, podendo a quantidade desses grupos variar de acordo com a característica analisada e a ação a ser tomada em relação a esses casos, podendo mesmo se optar pela sua inexistência. Em um sistema IDS, por exemplo, a incapacidade de classificar uma instância pode ser motivo suficiente para considerar aquela instância como parte de um ataque em potencial.

Neste trabalho visamos classificar o tráfego quanto a existência ou não de atividade maliciosa. Essa classificação não é trivial sequer ao ponto de definição, pois há diversas questões a ponderar sobre o que é tráfego malicioso ou não. Atividade maliciosa em uma rede pode ter diversos significados, os quais podem levantar dúvidas e gerar dificuldade quanto à classificação fora de um contexto mais específico. *Scan de backdoors* é uma atividade maliciosa óbvia, porém o tráfego gerado de um servidor invadido não pode ser taxado de atividade maliciosa tão facilmente.

A intenção deste trabalho é classificar o tráfego existente na rede segundo os seguintes quesitos:

1. Identificar o tipo de serviço existente no tráfego;
2. Em um segundo momento, se o tráfego é malicioso ou não;
3. Em caso de tráfego malicioso, identificar a natureza deste.

Chegamos finalmente à definição do método de classificação utilizado.

Diversos métodos de classificação de tráfego já foram propostos na literatura. Os métodos tradicionais de classificação de tráfego se baseiam massivamente na análise das portas da camada de transporte utilizadas na comunicação entre os processos da camada de aplicação. Como dissemos anteriormente, o foco da classificação de tráfego de rede consiste em classificar o *payload* do segmento da camada de transporte. Algumas informações que podem ser coletadas dessa camada são de grande valia, como o número da porta utilizada na comunicação.

Faremos uma breve revisão sobre as técnicas tradicionais de classificação de tráfego de rede, porém antes revisemos alguns conceitos básicos relevantes.

2.2 Identificação de um processo na rede

Relembramos aqui, de forma superficial, alguns conceitos básicos de redes aos quais recorreremos durante todo o trabalho. Mais informações podem ser obtidas em (KUROSE; ROSS, 2006).

Iniciamos pelo **endereço IP**. O endereço IP consiste em um número de 32 *bits*, dividido em 4 grupos de 8 *bits* que identifica unicamente um hospedeiro em uma rede, ou seja, não pode haver um endereço IP repetido na mesma rede.

Já o protocolo da camada de transporte consiste no protocolo da terceira camada da pilha de protocolos e é utilizado na conversação entre processos e responsável pela divisão dos dados em pacotes. Existem algumas opções de protocolos nesta camada, cada uma oferecendo serviços específicos. Exemplos destes são o TCP e o UDP.

Por fim, outro conceito importante é o de porta da camada de transporte. **Porta** é um número de 16 *bits*, com valor que varia entre 0 e 65535. O número da porta visa identificar unicamente um processo que utiliza um determinado protocolo da camada de transporte, em comunicação com a rede, entre os processos de um determinado hospedeiro.

A alocação de portas para os processos segue algumas regras estabelecidas pela IANA que especifica três intervalos de portas e seus usos, como apresentado abaixo (POSTEL et al., 2005):

1. *Well Known Ports*: são as do intervalo de 0 até 1023. São utilizadas pelo próprio sistema ou executadas com privilégio de super usuário. É nesta faixa que estão os serviços básicos, como SSH, FTP e etc;

2. *Registered Ports*: são as do intervalo de 1024 até 49151. São utilizadas por processos do usuário ou programas executados com permissão de usuário ordinário;

3. *The Dynamic and/or Private Ports*: são as do intervalo de 49152 até 65535.

A listagem completa do relacionamento de portas e seus serviços associados pode ser encontrada em (POSTEL et al., 2005). Uma observação que deve ser ser feita referente à utilização de portas é em relação a sua alocação durante a comunicação. Como exemplo podemos citar um servidor web, ao qual é associada a porta *TCP* 80. Qualquer chamada a esta porta será respondida pelo servidor web com a página requisitada, como definido pela IANA. A observação a ser feita se refere a porta utilizada pelo hospedeiro que requisita o serviço; este utilizará uma porta qualquer da terceira faixa definida pela IANA, com o mesmo protocolo da camada de transporte.

É importante lembrar ainda que um mesmo número de porta pode ser utilizado por dois processos simultaneamente em um mesmo hospedeiro, desde que estes usem protocolos da camada de transporte distintos, ou seja, a porta *TCP* 30 é diferente e pode ser utilizada simultaneamente com a porta *UDP* 30, como exemplo. Por isso se faz necessário a especificação da porta e do protocolo da camada de transporte para identificar unicamente a comunicação de um processos em um hospedeiro.

Como vimos, o protocolo e a porta, ambos da camada de transporte, identificam unicamente um processo que se comunica através da rede em um determinado momento, entre os processos de um hospedeiro. Somado a isso temos o endereço IP da máquina, constituindo uma identificação única na rede de um processo que se comunica através desta em um determinado momento. Essa identificação única na rede é denominada *socket*.

2.3 Técnicas clássicas de classificação de tráfego de rede

As técnicas tradicionais de classificação de tráfego de rede se baseiam na análise da utilização das portas da camada de trasporte, como definidas pela IANA, para a classificação do *payload* do segmento (CHOI; CHOI, 2006). Ou seja, caso a comunicação seja endereçada à uma determinada porta, uma simples consulta à lista especificada pela IANA, que pode ser consultada em (POSTEL et al., 2005), nos diria que tipo de dado está sendo transportado.

Enquanto essa prática podia ser suficiente alguns anos atrás, apresenta-se pouco efetiva nos dias atuais pois pode fornecer informação insuficiente ou inconsistente sobre o tráfego que está ocorrendo devido a utilização de subterfúgios que permitem ao usuário tornar esta informação com sentido enganoso (KARAGIANNIS; PAPAGIANNAKI; FALOUTSOS, 2005). Como exemplo de subterfúgio comumente utilizado, o processo pode usar portas inconsistentes com o padrão definido pela IANA para ter acesso a serviços não disponíveis na rede, ou seja, utilizar um serviço em uma outra porta qualquer que não a definida pela IANA, ou usar uma porta associada com um serviço específico para outro processo. Outro exemplo de subterfúgio é

a utilização de protocolos tunelados sobre HTTP. Tunelamento sobre HTTP consiste em colocar os dados da aplicação que se deseja tunelar dentro do *payload* do protocolo HTTP. Assim, bastaria ao receptor (ou a um proxy no caminho deste) retirar os dados do *payload* antes de repassá-los à aplicação adequada.

Outros fatores ainda comprometem esta abordagem de classificação, como a constante emergência de novos protocolos e aplicações, que nem sempre são formalizadas e registradas junto ao padrão da IANA, e o domínio de p2p, que utiliza grandes faixas de portas alocadas dinamicamente para suas atividades (KARAGIANNIS; PAPAGIANNAKI; FALOUTSOS, 2005).

Uma outra técnica é a utilização do conceito de fluxo em sobreposição à utilização simples de portas para classificar tráfego na rede. Fluxo consiste em um conjunto de características de um pacote sendo transmitido que permitem o agrupar com os demais pacotes participantes da interação através da rede. Neste trabalho consideramos cinco propriedades definindo este conceito, que são (MOORE; ZUEV; CROGAN, 2005):

1. IP de origem;
2. IP de destino;
3. Porta de origem;
4. Porta de destino;
5. Protocolo.

Um intervalo de tempo de ociosidade na troca de pacotes é utilizado para delimitar fluxos com essas mesmas características. Também consideramos um fluxo como sendo bidirecional na rede, ou seja, a inversão dos atributos IP de origem e destino juntamente com a inversão das portas de origem e destino dentro do quadro de tempo considerado classifica o pacote como participante do fluxo.

Este agrupamento traz benefícios à classificação, como a diminuição do número de instâncias a serem classificadas. Isto é facilmente perceptível pois caso um pacote do fluxo seja classificado corretamente como pertencente a um ataque, todo o fluxo estará também classificado, dispensando o esforço computacional que seria utilizado nos demais pacotes daquele mesmo fluxo (NWANZE; SUMMERVILLE; SKORMIN, 2005).

Porém é visível que este simples agrupamento não permite um avanço muito grande na classificação de tráfego. Uma técnica complementar que dê informações seguras quanto ao conteúdo do tráfego se faz necessária.

A análise dos dados coletados para a classificação de tráfego pode ser realizada em dois momentos distintos. A análise feita em tempo real, que busca levantar informação para tomada

de ações imediatas, é denominada **análise online**. Já a análise feita em momento posterior à coleta dos dados na rede, para auditoria, é denominada **off-line**.

À análise *off-line* se aplicam métodos mais específicos, incompatíveis com a análise *online* devido ao seu custo computacional, que não as tornam factíveis em tempo hábil para uma análise em tempo real. A análise simples de portas é um exemplo de técnica aplicável à análise *online*, devido a sua baixa complexidade, podendo ser implementada em hardware (CHOI; CHOI, 2006). Já a análise do *payload* dos segmentos da camada de transporte, comentado em seguida, é um método caro computacionalmente, e com elevado consumo de memória, mais adequado à análise *off-line*.

Uma das técnicas que pode ser utilizada, quando o ambiente o permite, é o reconhecimento do protocolo através da análise do *payload* do pacote da camada de transporte. Porém mesmo quando o acesso a esses dados é possível, esta ainda não é uma tarefa trivial devido ao conhecimento necessário do protocolo e ao esforço computacional necessário para analisá-los. A principal complicação reside no fato de que a classificação de tráfego requer conhecimento *a priori* do protocolo, interações do mesmo e disposição de suas informações. Enquanto algumas das aplicações e protocolos são bem conhecidas e documentadas em detalhes, outras operam fora dos padrões e são proprietárias. E mesmo tendo toda a informação necessária disponível, o custo computacional para analisar e processar essa informação pode ser muito oneroso devido a limitações de hardware e complexidade, tornando-a inviável (KARAGIANNIS; PAPAGIANNAKI; FALOUTSOS, 2005).

Uma alternativa é a utilização de **assinaturas** características a cada protocolo. A *assinatura* de um protocolo consiste em um padrão de *bits* característico que representa o *handshake* inicial do protocolo na maioria das aplicações, e são identificadas nas RFC (Request for Comments) e em documentos públicos, no caso de protocolos bem documentados, ou por engenharia reversa e empiricamente derivando um conjunto de padrões de *bits* por monitoramento de tráfego (KARAGIANNIS; PAPAGIANNAKI; FALOUTSOS, 2005). Esta abordagem simplificaria, até determinado ponto, a classificação do *payload*, pois a esta ocorreria basicamente pela busca de um padrão sobre uma sequência de *bits*, e não mais com base em uma análise da estrutura de um protocolo por pacote.

Além disso, outro tipo de utilização de padrões, também chamado de *assinatura*, é para o reconhecimento de ataques propriamente. Como apresentado em (NWANZE; SUMMERVILLE; SKORMIN, 2005), padrões de comportamento normal da rede são levantados e, destes, gerado um padrão que serve de modelo para a análise da rede. Qualquer comportamento que difira deste padrão, ou *assinatura*, de comportamento da rede significa um incidente ou atividade maliciosa em potencial.

Porém, o escrutínio do *payload* dos segmentos nem sempre é viável devido as restrições de privacidade impostas pelo ambiente ou pelo próprio usuário.

O tipo de técnica utilizado para a classificação de tráfego de rede é dependente do ambiente no qual deve ser aplicado, pois este pode apresentar diversas restrições. Estas restrições podem variar, contudo podemos resumi-las de forma simplificada nos seguintes tipos (KARAGIANNIS; PAPAGIANNAKI; FALOUTSOS, 2005):

1. Os usuários podem tornar a informação existente inconsistente a fim de utilizar recursos de rede bloqueados em condições normais;
2. Existem diversas restrições legais ao acesso e utilização da informação do usuário da rede sem sua permissão, visando respeitar sua privacidade, e seu desrespeito torna o responsável passível de ação judicial de acordo com a legislação local vigente;
3. Os usuários e aplicações podem, intencionalmente, fazer com que os dados transmitidos tornem-se inacessíveis a qualquer outra pessoa, exceto o receptor, através da codificação da transmissão (criptografia), fazendo com isso que informações relevantes à administração da rede, presentes junto a esses dados, também tornem-se inacessíveis.

Visamos uma metodologia apta a ser utilizada mesmo na presença de todas estas restrições. Os demais casos, em que haverá restrições menos rigorosas e, com isso, uma maior disponibilidade de informações, serão, teoricamente, casos em que a classificação será mais simples, porém de forma alguma trivial.

As técnicas clássicas apresentadas possuem limitações e vulnerabilidades específicas, porém ainda são amplamente utilizadas. A abordagem vigente é a utilização destas técnicas de forma complementar, ou seja, usando mais de uma técnica, quando possível, para uma classificação tão segura quanto possível e que se adeque às restrições impostas no ambiente, como proposto em (CHOI; CHOI, 2006).

Levando em conta este cenário apresentado, podemos afirmar que ainda não há técnica definitiva que trate da classificação de tráfego de forma segura e independente das restrições do ambiente (KARAGIANNIS; PAPAGIANNAKI; FALOUTSOS, 2005).

Como formalizado em (KARAGIANNIS; PAPAGIANNAKI; FALOUTSOS, 2005), buscamos uma técnica com as seguintes propriedades:

1. Não depender de acesso ao *payload* dos dados;
2. Não depender somente de conhecimento do número das portas da camada de transporte;
3. Não depender de informação adicional a não ser a coletada com um monitor de fluxo.

Para atender estes requisitos, podemos citar a existência de metodologias que utilizam recursos estatísticos para análise do tráfego. Estas se baseiam na análise de propriedades como

média do tamanho do pacote, média de duração do fluxo, intervalo de tempo entre o recebimento dos pacotes e outras medições, estabelecendo modelos para cada tipo de tráfego, como abordado em (KARAGIANNIS; PAPAGIANNAKI; FALOUTSOS, 2005) e em (AULD; MOORE; GULL, 2006). Estes métodos são recentes e ainda estão em desenvolvimento.

2.4 Técnicas baseadas em algoritmos de Aprendizado de Máquina

Existem algumas abordagens mais recentes, propostas para superar as vulnerabilidades das técnicas de classificação de tráfego de rede clássicas, baseadas em técnicas estatísticas. Entre estas, estão presentes técnicas baseadas em **aprendizado de máquina**.

Aprendizado de máquina e mineração de dados são termos correlatos e o primeiro pode ser tratado como uma etapa do segundo. Aprendizado de máquina está relacionado a melhorias na performance de uma tarefa, ou pelo menos no potencial de performance, em novas situações. Ou seja, fazer com que o computador possa aprimorar seu desempenho em uma tarefa específica de forma automática.

Já mineração de dados, "consiste em resolver problemas analisando dados já presentes em bancos de dados. [...] É o processo de descoberta de padrões úteis, automática ou semi-automática, em extensas quantidades de dados" (WITTEN; FRANK, 2005).

Aplicado ao nosso contexto, de forma similar ao exposto anteriormente, visamos melhorar a performance em uma previsão de classificação de novas instâncias de tráfego de rede, utilizando técnicas para encontrar e descrever padrões estruturais em dados coletados previamente da rede, como uma ferramenta para explicar estes dados e fazer predições com eles. Estas técnicas são os algoritmos de aprendizado de máquina denominados **classificadores** e a abordagem utilizada é o **aprendizado supervisionado**, ou seja, criar modelos de previsão com base em um conjunto de dados de treino pré-existente, dados estes que, em nosso trabalho, são coletados da rede em forma de fluxo.

Algumas dessas técnicas são fruto da convergência de trabalhos desenvolvidos em paralelo nas áreas de aprendizado de máquina e estatística. Um exemplo é a técnica de indução de árvores de decisão (WITTEN; FRANK, 2005).

Propostas desta abordagem podem ser encontradas em (PAN et al., 2004), que utiliza o algoritmo C4.5, de indução de árvore de decisão, e em (AULD; MOORE; GULL, 2006) que utiliza o algoritmo naive Bayes. Abordamos e comparamos a utilização desses dois algoritmos para a classificação de tráfego de rede.

2.5 WEKA

WEKA, ou *Waikato Environment for Knowledge Analysis*, é um pacote de software onde está incluso uma vasta gama de algoritmos de aprendizado de máquina e mineração de dados, e ferramentas de pré-processamento de dados. Weka engloba ferramentas para pré-processamento, classificação, regressão, clusterização, regras de associação e visualização, e é projetado para facilitar o desenvolvimento de novos algoritmos, além de permitir o acesso aos códigos existente para estudo ou modificação (WITTEN; FRANK, 2005).

Os algoritmos presentes podem ser usados diretamente nos dados, através de uma interface gráfica que acompanha o pacote, ou instanciados a partir de código Java. Todos os algoritmos recebem sua entrada no formato ARFF, descrito nos apêndices deste trabalho.

Desenvolvido originalmente pela Universidade de Waikato, Nova Zelândia, é escrito em Java e distribuído sob os termos da GNU *General Public License*. Pode ser obtido em `http://www.cs.waikato.ac.nz/ml/weka/`.

Utilizaremos as implementações do algoritmos de classificação desse pacote neste trabalho. Maiores informações sobre este pacote estão presentes em (WITTEN; FRANK, 2005).

3 DESCRIÇÃO DOS ALGORITMOS UTILIZADOS

Neste capítulo descrevemos o funcionamento dos algoritmos de aprendizado de máquina comparados neste trabalho: C4.5 e naive Bayes. Apresentamos o algoritmo e os detalhes atrelados à implementação presente na plataforma WEKA, a qual utilizamos para proceder com a comparação.

3.1 Algoritmo C4.5

C4.5 é um algoritmo de construção de árvore de decisão para problemas de classificação em aprendizado de máquina e mineração de dados baseados em aprendizado supervisionado (WU et al., 2008) e desenvolvido por J.Ross Quinlan, da Universidade de Sydney, Austrália. Trata-se de um **classificador discreto**, ou seja, a saída para cada previsão será apenas uma classe.

Durante a escrita deste trabalho já estava disponível comercialmente a versão C5.0 deste algoritmo, porém este não possui código fonte aberto e sua melhorias não são acessíveis ao público em geral. Utilizamos a implementação J48 do algoritmo C4.5 padrão, que está disponível na plataforma WEKA.

Uma árvore de decisão é uma série de questões sistematicamente arranjadas para que cada questão aborde um atributo e gere ramos baseado nos valores deste atributo. Nas folhas da árvore são posicionadas predições da variável de classe que se deseja predizer (WU et al., 2008). O procedimento para a construção de uma árvore de decisão baseada nos atributos de um conjunto de dados é apresentado informalmente, porém de forma inteligível e clara, em (WITTEN; FRANK, 2005), como se segue:

> "Primeiramente, selecione um atributo para o nó raiz e crie um ramo para cada valor possível. Isto divide os exemplos (conjunto de dados original) em sub conjuntos (conjunto de dados menores), um para cada valor possível do atributo. Agora o processo pode ser repetido recursivamente para cada ramo criado, usando apenas aquelas instâncias que atualmente chegam àquele ramo. Se em qualquer momento todas as instâncias em um nó tem a mesma classificação, pare de desenvolver aquela parte da árvore. A única coisa deixada para decidir é como determinar em que atributo dividir os ramos da árvore, dado um conjunto de exemplos com diferentes classes."

Esta descrição informal já no dá quase todas a informações necessárias para a constru-

ção da árvore. Abaixo a formalização do algoritmo como proposto em (WU et al., 2008):

Algorithm 1 C4.5(D)
ENTRADA: conjunto de instâncias compostas de atributos D

$Arvore = \{\}$
if D *e homogeneo OR outro criterio de parada* **then**
  $termina$
end if
for all $atributos\ a \in D$ **do**
  $Compute\ criterio\ de\ divisao\ em\ a$
end for
$a_{melhor} = Melhor\ atributo\ de\ acordo\ com\ o\ criterio\ computado$
$Arvore = Crie\ um\ no\ de\ decisao\ que\ testa\ o\ a_{melhor}\ na\ raiz$
$D_v = Subconjuntos\ de\ dados\ de\ D\ apos\ o\ teste\ com\ a_{melhor}$
for all D_v **do**
  $Arvore_v = C4.5(D_v)$
  $Mescle\ a\ Arvore_v\ ao\ ramo\ correspondente\ da\ Arvore$
end for
return $Arvore$

Porém, são necessários alguns outros apontamentos. Sobre o critério de parada apresentado, que consiste na homogeneidade das instâncias em cada folha da árvore, também haverão outros critérios para casos específicos. A utilização de todos os atributos também comporá um critério de parada, sendo as folhas deste ramo classificadas pela maioria dos exemplos que apresentar.

Também é interessante ressaltar que buscamos árvores pequenas. Isto pode ser justificado por dois aspectos. O primeiro, apresentado em (LUGER, 2004), trata-se da heurística proposta pelo lógico William de Occan, em 1324, denominada ”Navalha de Occam”, na qual ele estabelece que ”é supérfluo fazer com mais o que pode ser feito com menos. Não se deve acrescentar entidades além do necessário”.

Adequando ao nosso contexto, devemos preferir árvores menores e mais simples que se ajustam corretamente aos dados, evitando ramificações desnecessárias. Outra justificativa, mais específica ao nosso contexto, consiste que em uma árvore menor a classificação de instâncias novas ocorrerá mais rapidamente. Além disso, árvores pequenas tendem a ser mais genéricas, evitando o vício da mesma ao seu conjunto de dados de treinamento (WITTEN; FRANK, 2005). Então nos é interessante que o algoritmo alcance o quanto antes seu critério de parada. Assim, buscamos uma forma de que cada divisão da árvore gere ramos tão puros, isto é, homogêneos em relação à classe que se busca classificar, quanto possível.

A medida da pureza que utilizaremos se baseia na Teoria da Informação, desenvolvida por Claude Elwood Shannon em 1948, e é chamada *informação* e sua unidade de medida são os *bits*. O sentido a ela atrelado no nosso contexto é de que quanto mais heterogêneo os nós dos

ramos gerados, mais informação será necessária à classificação das instâncias neste nó. Assim, buscamos uma divisão que gere nós com a menor *informação* possível.

Para isso, iniciamos nosso procedimentos calculando a *informação* atrelada à toda árvore. A *informação* intrínseca a cada nó é medida pela fórrmula 3.1:

$$informacao(c) = \sum_{i=1}^{n} -p_i \log_2 p_i \tag{3.1}$$

na qual c é a classificação sendo analisada, n o número de valores possíveis para aquela classificação, gerando um termo para cada valor possível, e p é a probabilidade de uma instância pertencer àquela classificação do termo, calculada pela fórmula 3.2:

$$probabilidade = \frac{n}{N} \tag{3.2}$$

na qual n é o número de exemplos da classificação em questão, e N é o número de exemplos total.

Utilizamos a fórmula 3.1 sobre as instâncias da raiz da árvore, ou seja, todas as instâncias de exemplo disponíveis, e a classificação geral que se pretende obter com a árvore. Com isso teremos a informação atrelada à toda árvore. Agora abordaremos cada atributo, para os quais calcularemos a *informação* atrelada à cada valor possível deste, utilizando a fórmula 3.1 para cada valor possível e, após, fazendo uma média com base nas probabilidade de uma instância pertencer àquele valor, com a seguinte fórmula 3.3 :

$$informacaoMedia(a) = \sum_{i=1}^{c} -p_{v_i} informacao(c_{v_i}) \tag{3.3}$$

na qual a é o atributo sendo analisada, c o número de valores possíveis para aquele atributo, gerando um termo para cada valor possível, p_{v_i} é a probabilidade de uma instância possuir aquele valor do atributo, e $informacao(c_{v_i})$ é a aplicação da fórmula 3.3 sobre as instâncias que possuem o valor deste atributo sendo considerado no termo.

Como resultado da fórmula 3.3 teremos a *informação média* atrelada a cada atributo naquele ponto de construção da árvore. Definimos agora *ganho de informação* por meio da fórmula 3.4:

$$ganhoDeInformacao_a = informacao(c) - informacaoMedia(a) \tag{3.4}$$

na qual a é o atributo analisado, *informação* é dada pela equação 3.1 considerando a classificação c geral que se deseja obter na árvore, e *informação média* é dada pela equação 3.3.

Buscamos o maior *ganho de informação possível* em cada escolha de atributo, ou seja, buscamos escolher o atributo que, após a divisão do conjunto de dados em seus valores, a *informação* necessária para se chegar à classificação final seja mínima. Colocando de duas formas, buscamos o menor resultado da equação 3.3, ou o maior resultado da equação 3.4, que serão o mesmo atributo.

Porém, apenas a utilização do *ganho de informação* para escolher o atributo tem pontos negativos que necessitam ser tratados. O *ganho de informação* é altamente influenciado pelo número de valores possíveis do atributo. Sendo assim, este tende a preferir atributos com um largo número de valores possíveis, mesmo que haja apenas uma instância em cada grupo. Para compensar esse vício, introduz-se a *taxa de ganho*, que leva em conta o número de nós filhos gerados, apresentada abaixo:

$$taxaDeGanho(a) = \frac{ganhoDeInformacao(a)}{informacao(a)} \tag{3.5}$$

onde *a* é o atributo analisado, *informação* é dada pela equação 3.1 e é aplicada diretamente ao atributo, sem levar em consideração a classificação geral, e *ganho de informação* é dado pela equação 3.4.

Este é o critério que utilizaremos na escolha do nó, optando pelo atributo que apresentar *taxa de ganho* com maior valor.

Assim, temos o algoritmo completo. *A priori*, este algoritmo apresentado se aplica apenas a valores nominais. Para sua aplicação em valores numéricos (ou datas, que também são tratadas como valores numéricos) a *informação* será calculada com base em uma escolha binária, que divide o conjunto de dados pelo atributo numérico com base em um limiar. Basicamente, testa-se a aplicação dos limiares dentre os valores intermediários dos atributos das instâncias de exemplo, testando-se a *taxa de ganho de informação* para cada limiar possível. Porém, segundo (WU et al., 2008), é preferível a utilização do *ganho de informação* no lugar da *taxa de ganho de informação* durante o procedimento de definição do limiar, para que apenas o número de instâncias em cada subconjunto definido pelo limiar não determine a sua posição, voltando contudo à utilização da *taxa de ganho de informação* durante a escolha do atributo a ser utilizado. Várias verificações podem ser feitas em atributos numéricos durante a construção de um único ramo da árvore, diferente dos valores nominais, que são utilizados apenas uma vez na construção de um ramo da árvore, o que já exaure toda a informação que se pode extrair destes.

Este algoritmo não se aplica a atributos *strings*, logo, os campos deste tipo devem ser excluídos do conjunto de dados ou, quando possível, modelados como valores nominais.

Outro problema a ser tratado é a questão dos atributos sem valores, que podem apare-

cer no conjunto de dados. Estes problema é tratado dividindo-se as instâncias (ou a instância) através dos ramos da árvore, mesmo que isso gere frações, o que não afetará os cálculos apresentados anteriormente, se for o caso do treinamento do modelo. Juntamente a isto se utiliza o critério de pesos, no caso da classificação de novas instâncias, para definir qual a classificação dessa instância, já que as partes da instância atingirão diversas folhas na árvore. Mais detalhes sobre este mecanismo podem ser encontrados em (WITTEN; FRANK, 2005).

Por fim tratamos de um processo de otimização e generalização da árvore denominado ”poda”, que é um processo recursivo que percorre toda a árvore no sentido das folhas para a raiz, efetuado após toda a construção da árvore. Este processo é crítico para aumentar a eficiência do classificador em novas instâncias (WU et al., 2008), mesmo que isso cause um aumento na taxa de erros no conjunto de dados de treino. A implementação J48 do algoritmo C4.5 implementa a ”poda pessimista”, que prevê o erro que pode ocorrer com base nas instâncias classificadas erroneamente durante o treino, recursivamente estimando o erro associado a cada nó a a partir de seus ramos.

No J48 são utilizadas duas operações de poda: *substituição de subárvore* e *elevação de subárvore*. Durante o processo de ”poda”, que ocorre das folhas para a raiz da árvore, deve-se avaliar se cada nós sofre a *substituição de subárvore*, a *elevação de subárvore* ou se permanece como está.

A *substituição de subárvore* consiste na substituição da subárvore completa por uma única folha. Já a *elevação de subárvore*, que é uma operação bem mais complexa e com alto consumo de tempo, consiste na substituição da árvore por uma de suas subárvores diretas, presente em um dos seus ramos, ou seja, a subárvore é elevada um nível na árvore, tomando o lugar do seu nó pai.

A avaliação que deve ser feita para decidir se deve ocorrer ”poda” e, em caso positivo, qual operação deve ser aplicada, se baseia na *taxa de erro*. Com base na *taxa de erro* do nó raiz, na *taxa de erro* de cada ramo e da média da *taxa de erro* dos ramos opta-se por uma das operações, da seguinte forma:

1. Não é feita poda: quando a média da taxa de erro dos ramos é a menor entre todas;
2. *Elevação de subárvore*: quando a taxa de erro de um dos ramos é a menor entre todas, este nó (subárvore) é elevado;
3. *Substituição de subárvore*: quando o nó pai é o que possui a menor taxa de erro, todos os ramos são podados e o nó pai se torna uma folha;

Para chegarmos à *taxa de erro*, como apresentado em (WITTEN; FRANK, 2005), iniciamos pelo cálculo da *taxa de erro observada em um nó* f, obtida pela aplicação da fórmula 3.6:

$$f = \frac{E}{N} \tag{3.6}$$

na qual N é o número instâncias que alcançam o nó, e E é o número de instâncias classificadas de forma errônea.

Como apontado anteriormente, consideramos para um nó qualquer que sua classificação corresponde à classificação predominante entre as instâncias que o alcançam. Consideremos então q como sendo a *probabilidade de erro real*.

Obtemos então os *limites de confiança z* considerando-se uma *confiança c* (por padrão 25% no J48, a qual manteremos), com a fórmula 3.7:

$$P\left[\frac{f-q}{\sqrt{\frac{q(1-q)}{N}}} > z\right] = c \tag{3.7}$$

onde N é o número de exemplos, f é a *taxa de erro observada* e q a *probabilidade ou taxa de erro real*. Isso nos leva a um limite superior z que usamos como estimativa para a *taxa de erro e* no nó para a fórmula 3.8:

$$e = \frac{f+\frac{z^2}{2N}+z\sqrt{\frac{f}{N}-\frac{f^2}{N}+\frac{z^2}{4N^2}}}{1+\frac{z^2}{N}} \tag{3.8}$$

onde z é o número de desvios padrões correspondentes à *confiança c*, onde para $c = 25\%$ é $z = 0.69$.

Assim, para a implementação J48, os parâmetros que devem ser passados são *confiança C*, que por padrão é 25%, e o mínimo de instâncias de exemplo requeridas em cada valor possível de um teste de atributo para este ser aceito como teste da árvore, denominado *mínimo M*, e que tem por valor padrão 2. Utilizaremos os valores padrões nos procedimento realizados com este algoritmo durante todo este trabalho.

Tendo apresentado os detalhes deste algoritmo nos falta apenas apresentar sua complexidade, segundo a fórmula 3.9 (WITTEN; FRANK, 2005).

$$\Theta(mn\log n) + \Theta(n(\log n)^2) \tag{3.9}$$

na qual n é o número de instâncias e m o número de atributos do conjunto de dados utilizado.

3.2 Naive Bayes

Abordamos nesta seção o segundo algoritmo de aprendizado de máquina que utilizaremos neste trabalho. Trata-se de um algoritmo de aprendizado supervisionado que se baseia em uma modelagem estatística, chamado naive Bayes. O termo *naive*, que em tradução direta para o português significa "ingênuo", vem do fato que este algoritmo se baseia na hipótese que os atributos são independentes, o que não condiz com a realidade. Porém, mesmo com base nessa falácia, o algoritmo apresenta bons resultados no mundo real (WITTEN; FRANK, 2005).

Diferente de um classificador discreto, este **classificador estatístico** retorna um valor escalar para cada classe possível, que indica a probabilidade da instância analisada pertencer àquela classe. Assim, a classe que apresentar a maior probabilidade atrelada será a selecionada para a instância em questão.

Apresentamos abaixo o processo de classificação com a utilização deste algoritmo, como descrito em (WITTEN; FRANK, 2005).

Esse algoritmo se baseia na regra de Bayes (Thomas Bayes, Inglaterra, 1702 - 1761), que é expressa pela fórmula 3.10 :

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \tag{3.10}$$

onde H é a hipótese, E o evento, $P(A)$ a probabilidade de um evento A e $P(A|B)$ é a probabilidade de um evento A condicional a outro evento B.

Aplicado ao nosso contexto, o algoritmo consistirá em duas etapas.

A primeira consiste em calcular a probabilidade atrelada a cada atributo e seus valores possíveis com base no conjunto de dados de treino. Isto é feito contabilizando no número N de exemplos existentes no conjunto de dados de treino, para cada atributo a_n e para cada valor possível deste, v_m, o número de exemplos associados a uma classe c_x. Assim teremos para cada atributo, e para cada valor possível deste, um valor associado ao par para cada classe de classificação existente. Dividimos este inteiro pelo número total de exemplos n_{c_x} classificados como a classe considerada c_x. Assim temos a fórmula 3.11 :

$$P(a_{n_{v_m \wedge c_x}}) = \frac{n_{a=v_m \wedge c=c_x}}{n_{c_x}} \tag{3.11}$$

E calculamos também a probabilidade atrelada a uma determinada classe, como apresentado na fórmula 3.12 :

$$P(c_x) = \frac{n_{c_x}}{N} \tag{3.12}$$

na qual P é a probabilidade, c_x é a classe sendo avaliada, n_{c_x} o número de instâncias da classe sendo avaliada, e N o número total de instâncias.

Após obter estes valores, utilizaremos o algoritmo na previsão da classe de novas instâncias. Isto é feito com base na probabilidade de cada classe, utilizando-se os valores obtidos no passo anterior para o valor de cada atributo que a nova instância apresentar. Aplicaremos este procedimento para cada classificação possível. Assim teremos a aplicação da fórmula 3.14 :

$$P(instancia \in classe) = (\prod_{n=1}^{z} P(a_{n_{v_m \wedge c_x}})) * P(c_x) \tag{3.13}$$

onde z é o número de atributos da instância, a_n o atributo do termo, v_m o valor do atributo na nova instância e c_x a classificação sendo considerada.

Este valor pode ser divido pela soma da probabilidade de todas as classificações para que seja expresso em porcentagem.

Assim, teremos uma probabilidade para cada classificação possível. Selecionaremos a classificação com maior probabilidade.

Utilizamos ainda a técnica chamada **estimador de Laplace** (Pierre Simon, Marquês de Laplace, França, 1749 — 1827) para tratar os casos em que o conjunto de dados de treino possua atributos que não possuem nenhum exemplo, o que acarretaria uma probabilidade 0 para aquele atributo no processo apresentado, e, consequentemente, 0 na probabilidade de classificação. O **estimador de Laplace** consiste na adição de uma constante na frequência de cada atributo, evitando assim valores nulos no cálculo da probabilidade para uma classe. Como compensador, adicionamos ao denominador, que possui o número total de instâncias daquela classe, a soma das constantes adicionadas a cada atributo.

A questão de valores de atributos ausentes é facilmente tratado neste algoritmo. O atributo que tem o valor ausente é simplesmente omitido. Na classificação de uma nova instância, o cálculo é feito apenas com base nos demais atributos, excluindo os atributos sem valores definidos. Já no treino do modelo, os valores ausentes apenas não são contabilizados para o cálculo da probabilidade utilizada para a previsão de novos valores.

O processo apresentado apenas se aplica à valores nominais. Já valores numéricos são trabalhados com a aplicação da função de densidade de probabilidade, apresentada na fórmula 3.14:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.14}$$

onde μ é a média e σ é o desvio padrão dos valores presentes no conjunto de dados de treino.

Com esta fórmula, aplicada ao valor presente na instância que se deseja classificar, conseguiremos o valor que deve ser inserido no lugar da probabilidade atrelada àquele atributo no cálculo da previsão da classe desta instância.

4 DESCRIÇÃO DOS DADOS

Neste capítulo apresentamos os conjuntos de dados utilizados no nosso procedimento, detalhando sua estrutura e apresentando o pré-processamento realizado. Utilizamos três conjuntos de dados, sendo que todos apresentam características desejadas para nosso procedimento. Estas características são:

1. Ser realista: consistir em tráfego real, ou seja, que reflita o comportamento real encontrado no dia-a-dia;
2. Anotado: ter os incidentes presentes devidamente reconhecidos e descritos, contendo os tipos dos ataque ;
3. Público: acesso público ao conjunto de dados, para que experimentos com este possam ser reproduzidos;
4. Dados em formato de fluxo: visando abordar o contexto de restrições apresentado anteriormente, este é o tipo de dado que foi buscado.

4.1 Conjunto de dados I - Tráfego Internet com ameaças (Sperotto et al.)

4.1.1 Descrição da base

Conjunto de dados elaborado por Sperotto et al. e definido em (SPEROTTO et al., 2009), construído a partir de um *honeypot* instalado na Universidade de Twente, Holanda. O conjunto de dados consiste em 14.2MB de fluxos, sendo 98% destes classificados. É um conjunto de dados público que pode ser obtido em `http://traces.simpleweb.org/netflow/netflow2/`, disponibilizado em um arquivo com extensão ".sql", para ser restaurado em um SGBD MySQL. A metodologia de desenvolvimento do conjunto de dados consistiu em dados capturados exclusivamente no *honeypot* exposto diretamente à internet, adicionado a isso vários dispositivos de *log*.

Os serviços inseridos neste *honeypot* são listados abaixo:

1. *ssh*: acesso remoto, utilizando a porta 22;
2. *web server*: servidor web, utilizando a porta 80;
3. *ftp*: transferência de arquivos remotos, utilizando a porta.

A estrutura dessa base é composta por 6 tabelas que apresentam os fluxos, seus atributos e suas classificações. Grande parte dos atributos fornecidos se referem à classificação

dos fluxos. As tabelas são explicadas de forma resumida a seguir com base no documento de descrição da base de dados disponível em `http://traces.simpleweb.org/netflow/netflow2/description.txt`.

As Figuras 1 e 2 apresentam o esquema da base de dados. A primeira foi gerada para este trabalho a partir da análise da base de dados. Já a segunda foi gentilmente cedida pela autora Anna Sperotto.

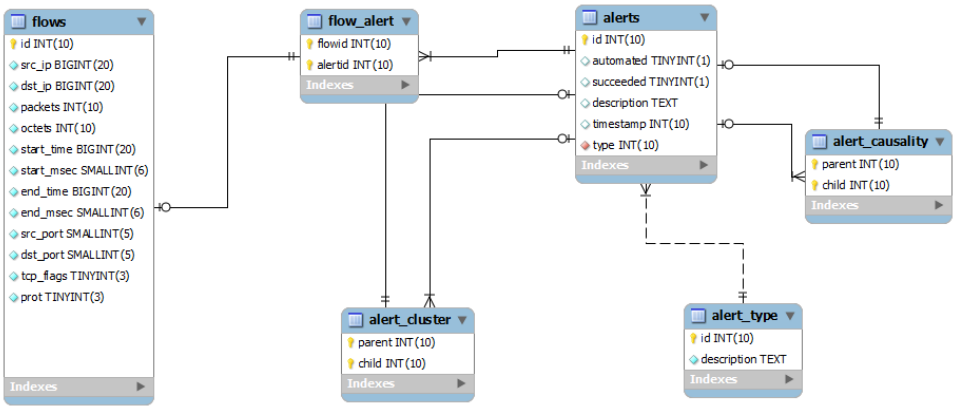

Figura 1: Esquema da base de dados do conjunto de dados de Sperotto et al. gerado a partir da base de dados.

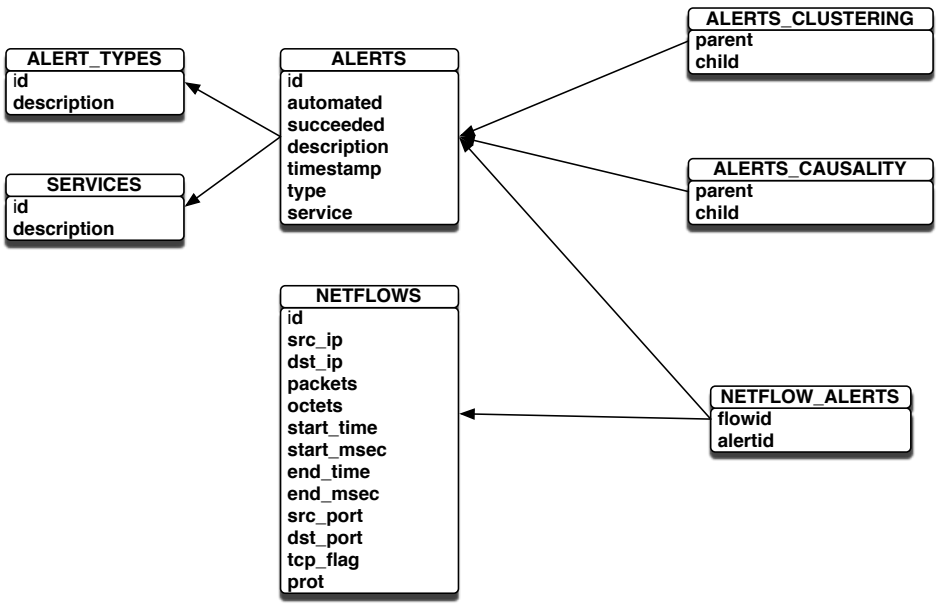

Figura 2: Esquema da base de dados do conjunto de dados de Sperotto et al. gentilmente fornecido pela autora.

A tabela 'flows' se refere aos fluxos, gerada a partir dos pacotes coletados. Os pacotes em questão não acompanham o conjunto de dados. Atributos da tabela 'flows':

- Coluna 'id': o identificador único do fluxo (chave primária);
- Coluna 'src_ip': IP de origem do fluxo. Os ips foram tornados anônimos. O ip do *honeypot*, que originalmente era 146.217.254.148, foi tornado 2463760020 com a função aplicada;
- Coluna 'dst_ip': IP de destino do fluxo. Os ips foram tornados anônimos;
- Coluna 'packets': número de pacotes do fluxo;
- Coluna 'octets': número de bytes do fluxo;
- Coluna 'start_time': tempo de início do fluxo no padrão UNIX (número de segundos);
- Coluna 'start_msec': tempo de início do fluxo (milissegundos não computados em 'start_time');
- Coluna 'end_time': tempo de término do fluxo no padrão UNIX (número de segundos);
- Coluna 'end_msec': tempo de término do fluxo (milissegundos não computados em 'end_time');
- Coluna 'src_port': número da porta de origem;
- Coluna 'dst_port': número da porta de destino;
- Coluna 'tcp_flags': Consiste na aplicação de uma operação 'OR' nos *flags* de todos os pacotes TCP no fluxo;
- Coluna 'prot': protocolo da camada de transporte. São utilizado os códigos do Cisco Netflow, que também são os da IANA. A lista completa pode ser encontrada em `http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml`, porém os valores se restringem, nesta base, a três códigos, que são: ICMP (1), TCP (6) and UDP (17).

A tabela 'alert_type' indica o tipo de incidentes e seus atributos são os seguintes:

- Coluna 'id': identificador único do tipo do alerta (chave primária);
- Coluna 'description': descrição textual do tipo do alerta.

A descrição do tipo de ataque é composta por dois termos. O primeiro se refere ao serviço ou aplicação ao qual pertence o fluxo em questão (ssh, ftp etc.). O segundo consiste no tipo de incidente. Estes tipos podem ser:

- CONN: são conexões maliciosas realizadas;
- SCAN: são agrupamentos, ou *clusters*, de conexões maliciosas (CONN), definidos com base em semelhança nestas. Um exemplo de um agrupamento deste seria um *cluster* formado pelos fluxos de um ataque de força bruta.
- SIDEEFFECT: consiste em tráfego oriundo de um ataque ou invasão anterior, como o tráfego gerado por um hospedeiro invadido.

Posto isto, os valores possíveis definidos no atributo 'description' da tabela 'alert_type' são listados abaixo :

- 'ssh_scan';
- 'ssh_conn';
- 'ftp_scan';
- 'ftp_conn';
- 'http_scan';
- 'http_conn';
- 'authident_sideeffect';
- 'irc_sideeffect';
- 'icmp_sideeffect';

É interessante notar que há alguns serviços listados no tipo de alerta que não fazem parte dos serviços oferecidos originalmente no *honeypot*. Estes serviços foram instalados por invasões ao *honeypot*.

Os incidentes ocorridos estão relacionados na tabela 'alerts', descrita abaixo:

- Coluna 'id': identificador único do alerta (chave primária);
- Coluna 'automated': indica se o ataque ocorreu de forma manual ou automática. Os valores são 1 se automático, 0 se não automático e *null* desconhecido;
- Coluna 'succeded': indica se o ataque obteve sucesso: 1 se teve sucesso, 0 se não e *null* se não se sabe;
- Coluna 'description': descrição textual do incidente;

- Coluna 'timestamp': *timestamp* do alerta. Caso o alerta consista em um *cluster* este campo é *null*;
- Coluna 'type': tipo de alerta (referência à tabela 'alert_type').

Em relação ao campo 'automated', podemos fazer alguns apontamentos. Os ataques *automated* consistem em ataques do tipo de força bruta, que basicamente fazem uma extensa tentativa de combinação entre várias possibilidades de nome de usuários e senhas de um grande dicionário que possuem.

O campo 'type' da tabela 'alerts' se relaciona com a tabela 'alert_type'. Este relacionamento não englobará nenhum tipo de alerta "SCAN", pois estes se referem à classificação de *clusters* apenas, e este relacionamento visa a classificação dos fluxos individualmente.

A tabela 'flow_alert' consiste em uma tabela que indica o relacionamento dos ataques e os respectivos fluxos que o constituem. Um fluxo pode se relacionar a apenas um alerta, porém um alerta pode ser constituído de vários fluxos, indicando um relacionamento de um para muitos. Esta tabela poderia ser substituída por um campo a mais na tabela 'flows', porém os autores do conjunto de dados fizeram esta opção, acreditamos que, por clareza na base. Os fluxos que não possuem este relacionamento são os que foram classificados como não maliciosos. Abaixo os atributos da tabela 'flow_alert':

- Coluna 'flowid': ID do fluxo, referenciando a tabela 'flows';
- Coluna 'alertid': ID do alerta, referenciando a tabela 'alerts'.

A tabela 'alert_cluster' consiste em um duplo relacionamento à tabela 'alerts'. Basicamente esta tabela constitui os *clusters* citados anteriormente, relacionando um *cluster* (tipos "SCAN" da tabela 'alert_type' registrados no campo 'parent') com os alertas simples (campo 'child'), que são os fluxos maliciosos especificamente. Abaixo os atributos da tabela 'alert_cluster':

- Coluna 'parent': identificador único do *cluster*, referenciando a tabela 'alerts';
- Coluna 'child': identificador único do alerta, referenciando a tabela 'alerts'.

A tabela 'alert_causality' indica o tráfego malicioso gerado por alerta malicioso anterior. Podemos exemplificar essa situação como o tráfego gerado a partir de um hospedeiro (campo 'child') após sua invasão (campo 'parent'). A seguir os atributos da tabela 'alert_causality':

- Coluna 'parent': identificador único do alerta inicial, referenciando a tabela 'alerts'
- Coluna 'child': identificador único do alerta posterior, gerado pelo alerta inicial, referenciando a tabela 'alerts'

Mais detalhes sobre esta base podem ser obtidos em `http://traces.simpleweb.org/netflow/netflow2/`.

4.1.2 Coleta das amostras

Foram coletadas 6 amostras de dados desta base. A primeira amostra foi composta por uma quantidade determinada de exemplos de cada classe coletados varrendo o conjunto de dados do início ao fim de forma sequencial utilizando-se a query de conversão apresentada no apêndice X . Estas quantidades são apresentadas na tabela 1. Esta constitui nossa amostra balanceada. O tipo "\N" se refere à tráfego não malicioso.

Tabela 1: Amostra 1 - Sperotto

Ataque	Quantidade de Exemplos
ssh_conn	1000
ftp_conn	12
http_conn	1000
authident_sideeffect	1000
irc_sideeffect	1000
icmp_sideeffect	1000
\N	1000

As demais amostras foram geradas através da variação do número de exemplos do tipo "\N" . As quantidades relativas do tipo "\N" para cada amostra são apresentadas na tabela 2. Nos demais tipos manteve-se a mesma quantidade de exemplos da amostra 1.

Tabela 2: Quantidades de exemplos classificados como "\N" em cada amostra - Sperotto

Amostra	Quantidade de exemplos "\N"
1	1000
2	2000
3	3000
4	4000
5	5000
6	5968

4.1.3 Conversão dos dados

Visando adequar os dados ao formato exigido pela plataforma WEKA, realizou-se a a transformação da base de dados para arquivo ARFF. Esta transformação foi feita através da *query* apresentada no apêndice C. O conjunto de dados obtidos com essa transformação conta com 28 atributos, dentre eles a classificação do fluxo, que são apresentados com sua descrição nas tabelas 3 e 4. Alguns destes campos não estão presentes de forma direta na base de dados original, mas foram obtidos com base em campos existentes, entre eles os campos 'malicious' e 'duration'.

Nos fluxos que não são maliciosos, todos os atributos referentes a fluxo malicioso recebem o valor "\N", que significa que esse atributo não se aplica a este fluxo. Nos outros casos, em que o atributo não se aplicar ao fluxo, porém este for malicioso, o atributo receberá um valor específico, como é o caso do atributo 'cluster_succeeded', que neste contexto receberá o valor 'NO_CLUSTER'. Por último, ressaltamos que o campo que armazena a classificação do fluxo é o campo "alert_type".

Tabela 3: Atributos do ARFF gerado a partir da base de dados - parte 1 - Sperotto

#	Atributo	Descrição	Valores
1	src_ip	IP de origem do fluxo	numérico
2	dst_ip	IP de destino do fluxo	numérico
3	src_port	número da porta de origem	numérico
4	dst_port	número da porta de destino	numérico
5	packets	número de pacotes do fluxo	numérico
6	octets	número de bytes do fluxo	numérico
7	start_time	tempo de início do fluxo no padrão UNIX (número de segundos)	numérico
8	start_msec	tempo de início do fluxo (parte dos milissegundos)	numérico
9	end_time	tempo de término do fluxo no padrão UNIX (número de segundos)	numérico
10	end_msec	tempo de término do fluxo (parte dos milissegundos)	numérico
11	duration	duração do fluxo em milissegundos	numérico
12	tcp_flags	operação 'OR' nos flags de todos os pacotes TCP no fluxo	numérico
13	protocol	protocolo da camada de transporte	ICMP, TCP, UDP
14	malicious	indica se o fluxo é malicioso	FALSE, TRUE
15	automated	indica se o ataque ocorreu de forma automática	FALSE, TRUE, "\N"
16	succeeded	indica se o ataque obteve sucesso	FALSE, TRUE, "\N"
17	alert_description	descrição textual do incidente	sequência de caracteres
18	alert_type	classificação do fluxo	ssh_conn, ftp_conn, http_conn, authident_sideeffect, irc_sideeffect, icmp_sideeffect, "\N"
19	causality	indica se o fluxo é resultado de um incidente anterior	FALSE, TRUE, "\N"

Tabela 4: Atributos do arff gerado a partir da base de dados - parte 2 - Sperotto

#	Atributo	Descrição	Valores
20	causality_automated	indica se o ataque causador deste fluxo ocorreu de forma automática	FALSE, TRUE,"\N", NO_CAUSALITY
21	causality_succeeded	indica se o ataque causador deste fluxo obteve sucesso	FALSE, TRUE, "\N" , NO_CAUSALITY
22	causality_description	descrição textual do ataque causador deste fluxo	sequência de caracteres
23	causality_type	classificação do ataque causador do fluxo	ssh_scan, ftp_scan, http_scan, "\N", NO_CAUSALITY
24	clustered	indica se o fluxo está associado a um cluster	FALSE, TRUE, "\N"
25	cluster_automated	indica se o cluster deste fluxo ocorreu de forma automática	FALSE, TRUE, "\N", NO_CLUSTER
26	cluster_succeeded	indica se cluster obteve sucesso	FALSE, TRUE, "\N", NO_CLUSTER
27	cluster_description	descrição textual do cluster do fluxo	sequência de caracteres
28	cluster_type	classificação do cluster	ssh_scan, ftp_scan, http_scan, "\N", NO_CLUSTER

O cabeçalho do arquivo arff foi inserido posteriormente e é apresentado no apêndice D. Contudo, dos 28 atributos presentes neste conjunto de dados, apenas cinco atributos e a classificação foram utilizados para nossos procedimentos, sendo estes apresentados abaixo com sua descrição.

- ’packets’: número de pacotes do fluxo;
- ’octets’: número de bytes do fluxo;
- ’duration’: duração do fluxo em milissegundos;
- ’tcp_flags’: resultado de uma operação ’OR’ nos flags de todos os pacotes TCP no fluxo;
- ’protocol’: protocolo da camada de transporte;
- ’alert_type’: classificação do tipo de alerta do fluxo.

4.2 Conjunto de dados II - Tráfego Internet Aberto (Moore, 8 classes)

4.2.1 Descrição da base

Conjunto de dados desenvolvido por Moore et al. e descrito em (MOORE; ZUEV; CROGAN, 2005). Baseia-se em uma coleta de 24 horas de tráfego com sua respectivas anotações. Disponibilizado na forma de 10 arquivos ARFF, cada um consistindo em um período do dia. Conta com 248 atributos para cada flow e sua respectiva classificação, a qual define o tipo de tráfego. As categorias da classificação são apresentadas na tabela 5.

Tabela 5: Classificações - Moore

WWW	GAMES
MAIL	INTERACTIVE
FTP-CONTROL	SERVICES
FTP-PASV	MULTIMEDIA
ATTACK	FTP-DATA
P2P	DATABASE

Mais detalhes sobre esta base e seu download estão disponíveis em `http://www.cl.cam.ac.uk/research/srg/netos/nprobe/data/papers/sigmetrics/index.html`.

4.2.2 Coleta das amostras

As amostras dessa base foram geradas a partir de uma base maior, construída através da união de todos os 10 arquivos arff disponibilizados. A partir desta base gerada foram coletadas 6 amostras de dados. A primeira amostra foi composta por uma quantidade determinada de exemplos de cada classe coletados varrendo o conjunto de dados do início ao fim de forma sequencial. Estas quantidades são apresentadas na tabela 6. Esta constitui nossa amostra balanceada.

Tabela 6: Amostra 1 - Moore

Ataque	Quantidade de Exemplos
WWW	1000
MAIL	1000
FTP-CONTROL	1000
FTP-PASV	1000
ATTACK	1000
P2P	1000
DATABASE	1000
FTP-DATA	1000
MULTIMEDIA	576
SERVICES	1000
INTERACTIVE	110
GAMES	8

As demais amostras foram geradas através da variação do número de exemplos da categoria ”WWW”. As quantidades relativas da categoria ”WWW” para cada amostra são apresentadas na tabela 7. Nos demais tipos manteve-se a mesma quantidade de exemplos da amostra 1.

Tabela 7: Quantidades de exemplos classificados como ”WWW” em cada amostra - Moore

Amostra	Quantidade de exemplos ”WWW”
1	1000
2	2000
3	3000
4	4000
5	5000
6	6000

4.2.3 Conversão dos arquivos

O conjunto de dados não passou por nenhuma conversão, pois já é disponibilizado em formato ARFF. Porém, dos mais de 200 atributos de cada fluxo, apenas foram utilizados os 28 mais relevantes segundo o artigo (AULD; MOORE; GULL, 2006). Estes são apresentados na tabela 8 com sua respectiva descrição e numeração.

Uma observação a ser feita sobre a tabela 8 se refere à numeração apresentada. Esta numeração se refere à ordem de disposição dos atributos no arquivo ARFF, e não deve ser confundida com os nomes dos atributos existentes no arquivos, que também são números. A plataforma WEKA apresenta tanto a ordem quanto os nomes durante a manipulação do conjunto de dados, o que pode causar alguma dúvida.

Tabela 8: Atributos utilizados do conjunto de dados - Moore

#	Nome	Descrição
12	med_data_wire	Mediana dos bytes em pacotes(Ethernet). (ambas direções)
18	q1_data_ip	Primeiro quartil do tamanho do pacote em bytes. (todos os pacotes)
34	ack_pkts_sent_b a	Número total de pacotes com bit ACK setado. (servidor -> cliente)
37	sack_pkts_sent_a b	Número total de pacotes ACK com blocos TCP SACK. (cliente -> servidor)
45	actual_data_pkts_a b	Número total de pacotes com pelo menos um byte de payload TCP. (cliente -> servidor)
59	pushed_data_pkts_a b	Número total de pacotes com o bit PUSH setado no cabeçalho TCP. (cliente -> servidor)
60	pushed_data_pkts_b a	Número total de pacotes com o bit PUSH setado no cabeçalho TCP. (servidor -> cliente)
83	min_segm_size_a b	Menor tamanho de segmento. (cliente -> servidor)
86	avg_segm_size_b a	Média do tamanho de segmento da conexão. (servidor -> cliente)
87	max_win_adv_a b	Maior janela de aviso. (cliente->servidor)
95	initial_window-bytes_a b	Número total de bytes enviados na janela inicial. (client -> servidor)
96	initial_window-bytes_b a	Número total de bytes enviados na janela inicial. (servidor -> cliente)
108	data_xmit_time_b a	Total de dados transmitidos. (servidor -> cliente)
109	idletime_max_a b	Tempo de idle máximo. (cliente->servidor)
111	throughput_a b	Throughput médio. (cliente->servidor)
113	RTT_samples_a b	Número total de RTT (Round-Trip Time). (cliente -> servidor)
133	RTT_full_sz_stdev_a b	Desvio padrão do tamanho real de RTT. (cliente->servidor)
134	RTT_full_sz_stdev_b a	Desvio padrão do tamanho real de RTT. (servidor->cliente)
136	post-loss_acks_b a	Número de ACKs pós evento de perda. (servidor->cliente)
158	max_data_wire_a b	Máximo de bytes em um pacote (Ethernet).(cliente -> servidor)
160	min_data_ip_a b	Menor número total de bytes em um pacote IP. (cliente -> servidor)
162	med_data_ip_a b	Mediana do total de bytes em um pacote IP. (cliente -> servidor)
164	q3_data_ip_a b	Terceiro quartil do tamanho de pacotes em bytes. (cliente -> servidor)
180	var_data_wire_b a	Variância dos bytes em pacotes (Ethernet). (servidor -> cliente)
206	q3_IAT_b a	Terceiro quartil do tempo entre as chegadas de pacotes. (servidor -> cliente)
210	No._transitions_bulk/trans	Número de transições entre os modos de transferência 'transaction' e 'bulk'.
214	Time_spent_idle	Tempo gasto em idle.
241	FFT_b a	Terceiro maior componente FFT de pacote IAT. (servidor -> cliente)

4.3 Conjunto de Dados III - KDD Cup 1999 Data

4.3.1 Descrição da base

Conjunto de dados utilizado na Terceira Competição Internacional de Ferramentas de Descoberta de Conhecimento e Mineração de Dados (*The Third International Knowledge Discovery and Data Mining Tools Competition*) que tinha por objetivo a construção de uma ferramenta de detecção de intrusão de redes, ou seja, uma ferramenta capaz de distinguir entre conexões maliciosas e não maliciosas. Este conjunto de dados é baseado no conjunto de dados da DARPA, de 1998, construído e gerenciado pelo grupo de Tecnologia de Sistema de Informação (*Information System Technology* - IST) do MIT Lincoln Labs em contrato com a DARPA (*Defense Advanced Research Projects Agency*). Este conjunto de dados foi construído a partir da captura de 7 semanas de coleta de tráfego em uma rede local com vários ataques simulados. A informação gerada foi tratada e convertida para o formato de conexões, conceito apresentado na especificação da KDD CUP 99, em `http://kdd.ics.uci.edu/databases/kddcup99/task.html`, e reproduzido abaixo:

> "Uma conexão é uma sequência de pacotes TCP começando e terminando em um intervalo de tempo bem definido, no qual dados fluem de endereço IP fonte para um endereço IP alvo sob um protocolo bem definido".

Por apresentar semelhanças ao conceito de fluxo, apesar de diferir no aspecto das portas, o trataremos da mesma forma. Este conjunto de dados também foi classificado como malicioso ou normal, sendo especificada a classificação maliciosa do mesmo. Assim, temos um conjunto de dados, com cada instância contendo 42 atributos, sendo um destes sua classificação. Utilizamos todos estes 42 atributos no nosso trabalho.

Nas tabelas que seguem apresentamos a estrutura da base. A tabela 11 lista os atributos presentes na base, o seu tipo e o mapeamento adequado para a plataforma Weka. A tabela 10 apresenta os valores possíveis para o tipo de serviço de cada fluxo, relativos ao atributo ”service”. Por fim, a tabela 9 apresenta as classificações possíveis para cada conexão. Destas é importante ressaltar que a classificação ”NORMAL” corresponde a fluxos não maliciosos.

O download e mais detalhes sobre esta base estão disponíveis em `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`

Tabela 9: Classificação dos fluxos por tipo de ataque - KDDCup99

back	buffer_overflow	ftp_write	guess_passwd
imap	ipsweep	land	loadmodule
multihop	neptune	nmap	perl
phf	pod	portsweep	rootkit
satan	smurf	spy	teardrop
warezclient	warezmaster	normal	

Tabela 10: Serviços - KDDCup99

aol	auth	bgp	courier
csnet_ns	ctf	daytime	discard
domain	domain_u	echo	eco_i
ecr_i	efs	exec	finger
ftp	ftp_data	gopher	harvest
hostnames	http	http_2784	http_443
http_8001	imap4	IRC	iso_tsap
klogin	kshell	ldap	link
login	mtp	name	netbios_dgm
netbios_ns	netbios_ssn	netstat	nnsp
nntp	ntp_u	other	pm_dump
pop_2	pop_3	printer	private
red_i	remote_job	rje	shell
smtp	sql_net	ssh	sunrpc
supdup	systat	telnet	tftp_u
tim_i	time	urh_i	urp_i
uucp	uucp_path	vmnet	whois
X11	Z39_50a		

Tabela 11: Atributos I - KDDCup99

#	Atributo	Descrição	Valores	WEKA
1	duration	duração da conexão em segundos	inteiros	numeric
2	protocol_type	tipo do protocolo	udp,tcp,icmp	nominal
3	service	serviço de rede no destino	vide tabela 10	nominal
4	flag	status da conexão	OTH, REJ, RSTO, RSTOS0, RSTR, S0, S1, S2, S3, SF, SH	nominal
5	src_bytes	número de bytes de dados transferidos da fonte parao destino	inteiros	numeric
6	dst_bytes	número de bytes de dados transferidos do destino para a fonte	inteiros	numeric
7	land	indica se o endereço de origem e de destino e os números das portas são iguais	0(FALSO), 1(VERDADEIRO)	nominal
8	wrong_fragment	número de pacotes com checksum falho	inteiros	numeric
9	urgent	número de pacotes com o bit ”urgent” ativado	inteiros	numeric
10	hot	número de ações ”hot”	inteiros	numeric
11	num_failed_logins	número de tentativas falhas de login	inteiros	numeric
12	logged_in	indica se o login foi efetuado	0(FALSO), 1(VERDADEIRO)	nominal
13	num_compromised	número de erros ”NOT FOUND”	inteiros	numeric
14	root_shell	indica se um root shell foi obtido	0(FALSO), 1(VERDADEIRO)	numeric
15	su_attempted	indica se o comando ”su root” foi tentado	0(FALSO), 1(VERDADEIRO)	numeric
16	num_root	número de acessos ”root”	inteiros	numeric
17	num_file_creations	número de operações de criação de arquivo	inteiros	numeric
18	num_shells	número de logins de usuários normais	inteiros	numeric
19	num_access_files	número de operações de acesso de arquivos de controle	inteiros	numeric
20	num_outbound_cmds	número de comandos outbound em uma sessão FTP	inteiros	numeric
21	is_host_login	indica se o usuário está acessando como ”root” ou ”adm”	0(FALSO), 1(VERDADEIRO)	nominal

Tabela 12: Atributos II - KDDCup99

#	Atributo	Descrição	Valores	WEKA
22	is_guest_login	indica se o usuário está acessando como um ”guest”, ”anonymous” ou ”visitor”	0(FALSO), 1(VERDADEIRO)	nominal
23	count	número de conexões para o mesmo endereço IP dos últimos dois segundos	inteiros	numeric
24	srv_count	número de conexões para o mesmo número de porta de destino dos últimos dois segundos	inteiros	numeric
25	serror_rate	porcentagem de conexões que possuem erros ”SYN” entre as consideradas em ”count”	reais	numeric
26	srv_serror_rate	porcentagem de conexões que possuem erros ”SYN” entre as consideradas em ”srv_count”	reais	numeric
27	rerror_rate	porcentagem de conexões que possuem erros ”REJ” entre as consideradas em ”count”	reais	numeric
28	srv_rerror_rate	porcentagem de conexões que possuem erros ”REJ” entre as consideradas em ”srv_count”	reais	numeric
29	same_srv_rate	porcentagem de conexões para o mesmo serviço entre as consideradas em ”count”	reais	numeric
30	diff_srv_rate	porcentagem de conexões para serviços diferentes entre as consideradas em ”count”	reais	numeric
31	srv_diff_host_rate	porcentagem de conexões para hospedeiros diferentes entre as consideradas em ”srv_count”	reais	numeric
32	dst_host_count	número de conexões tendo como destino o mesmo endereço IP	inteiros	numeric
33	dst_host_srv_count	número de conexões tendo como destino o mesmo número de porta	inteiro	numeric
34	dst_host_same_srv_rate	porcentagem das conexões para o mesmo serviço entre as consideradas em ”dst_host_count”	reais	numeric
35	dst_host_diff_srv_rate	porcentagem das conexões para serviços diferentes entre as consideradas em ”dst_host_count”	reais	numeric
36	dst_host_same_src_port_rate	porcentagem das conexões para o mesmo número de porta entre as consideradas em ”dst_host_srv_count”	reais	numeric

Tabela 13: Atributos III - KDDCup99

#	Atributo	Descrição	Valores	WEKA
37	dst_host_srv_diff_host_rate	porcentagem das conexões para números de porta diferente entre as consideradas em ”dst_host_srv_count”	reais	numeric
38	dst_host_serror_rate	porcentagem de conexões que possuem erros ”SYN” entre as consideradas em ”dst_host_count”	reais	numeric
39	dst_host_srv_serror_rate	porcentagem de conexões que possuem erros ”SYN” entre as consideradas em ”dst_host_srv_count”	reais	numeric
40	dst_host_rerror_rate	porcentagem de conexões que possuem erros ”REJ” entre as consideradas em ”dst_host_count”	reais	numeric
41	dst_host_srv_rerror_rate	porcentagem de conexões que possuem erros ”REJ” entre as consideradas em ”dst_host_srv_count”	reais	numeric
42	classification	classificação do tráfego	vide tabela 9	nominal

4.3.2 Coleta das amostras

Foram coletadas 7 amostras de dados desta base. A primeira amostra foi composta por uma quantidade determinada de exemplos de cada classe coletados varrendo o conjunto de dados do início ao fim de forma sequencial. Estas quantidades são apresentadas na tabela 14. Esta constitui nossa amostra balanceada.

Tabela 14: Amostra 1 - KDDCup99

Ataque	Quantidade de Exemplos
normal	1000
back	1000
buffer_overflow	30
ftp_write	8
guess_passwd	53
imap	12
ipsweep	1000
land	21
loadmodule	9
multihop	7
neptune	1000
nmap	1000
perl	3
phf	4
pod	264
portsweep	1000
rootkit	10
satan	1000
smurf	1000
spy	2
teardrop	979
warezclient	1000
warezmaster	20

As demais amostras foram geradas através da variação do número de exemplos da categoria "NORMAL". As quantidades relativas à categoria "NORMAL"para cada amostra são apresentadas na tabela abaixo. Nos demais tipos manteve-se a mesma quantidade de exemplos da amostra 1.

Tabela 15: Quantidades de exemplos classificados como ”normal” em cada amostra - KDD-Cup99

Amostra	Quantidade de exemplos ”NORMAL”
1	1000
2	2000
3	3000
4	4000
5	5000
6	6000
7	7000

4.3.3 Conversão dos arquivos

O formato no qual esta base é disponibilizada já se assemelha em muito ao formato ARFF desejado. Esta é disponibilizada em um arquivo com um exemplo em cada linha, sendo cada atributo separado por vírgula e cada linha finalizada por ponto. Para adequá-lo bastou-nos duas modificações. A primeira foi a exclusão dos pontos no final de cada linha, feita com um editor de texto simples. A segunda foi a inserção de um cabeçalho de descrição para a plataforma WEKA. Este cabeçalho está presente nos anexos deste trabalho.

5 RESULTADOS

Nesta seção apresentamos os detalhes da execução do procedimento de comparação e os resultados obtidos. Detalhes sobre as métricas de desempenho utilizadas podem ser encontrados no apêndice B.

Apresentamos os resultados para cada conjunto de dados de forma separada.

O procedimento de execução de cada algoritmo para os conjuntos de dados foi realizado de forma padrão. Para o algoritmo *C4.5* utilizamos os parâmetros "-C 0.25" e "-M 2", apresentados no capítulo 3. Já para o algoritmo *naive Bayes* não foram inseridos parâmetros.

Para ambos algoritmos foi utilizada a *Cross validation*, que consiste na divisão dos dados utilizados para treino em partes, ou *folds*, das quais uma é utilizadas para teste e as demais para treino em cada rodada, até que todas sejam utilizadas para teste. Este procedimento é utilizado para validação do classificador, devendo-se ressaltar que o classificador e a matriz de confusão gerados ao fim do processo são referentes à utilização de todo o conjunto de dados tanto treino quanto para teste. O número de partes utilizadas foi 10 em cada execução dos algoritmos.

Os atributos selecionados em cada conjunto de dados e a coleta das amostras foram apresentados no capítulo 4, e consistem em informações baseadas em fluxos e meta-informações destes, as quais se adequam às restrições que visamos atender, apresentados em (KARAGIANNIS; PAPAGIANNAKI; FALOUTSOS, 2005) e revisadas no capítulo 2 deste trabalho.

Assim, apresentamos para cada conjunto de dados os resultados obtidos na forma de matriz de confusão (apresentamos apenas a da amostra balanceada, definida no capítulo 4 para cada conjunto de dados), *TP Rate* e *FP rate* ponderados (a fórmula para o cálculo dessas métricas é apresentada no apêndice B) para cada amostra e a plotagem dos pontos correspondentes a estes valores no gráfico ROC, e a curva ROC construída com aplicação do fecho convexo (*convex hull*) sobre estes pontos, como descrito em (PRATI; BATISTA; MONARD, 2008) e reproduzido no apêndice B deste trabalho, e, por último, a plotagem das curvas ROC de C4.5 e *naive Bayes* em um mesmo gráfico para comparação.

Ressaltamos que todos os pontos e curvas ROC gerados utilizam os valores *TP Rate* e *FP rate* ponderados, pois estamos trabalhando com múltiplas classes, como definido no apêndice B deste trabalho.

Sobre o *layout* dos gráficos ROC apresentados, ressaltamos que a numeração existente sobre o ponto na curva ROC refere-se à numeração do ponto, e não a amostra que este representa. Esta numeração é apresentada na tabela referente a cada gráfico ROC.

5.1 Conjunto de dados I - Sperotto et al.

Apresentamos abaixo a *matriz de confusão* de nossa amostra balanceada para o algoritmo *C4.5(J48)*:

```
   a   b   c   d   e   f   g   <-- classified as
 707   1   0   1   0   0 291 |   a = ssh_conn
   3   9   0   0   0   0   0 |   b = ftp_conn
   3   0 997   0   0   0   0 |   c = http_conn
   2   2   0 986   0   0  10 |   d = authident_sideeffect
   0   0   0   0 993   0   7 |   e = irc_sideeffect
   0   0   0   0   0 987  13 |   f = icmp_sideeffect
   2   1   0  46  32  19 900 |   g = \N
```

E, a seguir, a *matriz de confusão* de nossa amostra balanceada para o algoritmo *naive Bayes*:

```
   a   b   c   d   e   f   g   <-- classified as
 647   7   1 297  32   0  16 |   a = ssh_conn
   2   3   7   0   0   0   0 |   b = ftp_conn
   1   0 999   0   0   0   0 |   c = http_conn
   0   0   0 996   2   0   2 |   d = authident_sideeffect
  45   1 657  50 211   0  36 |   e = irc_sideeffect
   0   0   0  85   0 915   0 |   f = icmp_sideeffect
  92  13  18 794  13  60  10 |   g = \N
```

A tabela 16 apresenta os valores de *TP Rate* e *FP rate* ponderados para cada amostra do conjunto de dados para os algoritmos *C4.5* e *naive Bayes*.

A tabela 17 apresenta todos os pontos e sua numeração gerados com o algoritmo *C4.5* para a plotagem destes no gráfico ROC, apresentado na figura 3, e os pontos do gráfico ROC após a aplicação do fecho convexo (*convex hull*), que por sua vez é expresso na figura 4 já como curva ROC.

A tabela 18 apresenta todos os pontos e sua numeração gerados com o algoritmo *naive Bayes* para a plotagem destes no gráfico ROC, apresentado na figura 5, e os pontos do gráfico ROC após a aplicação do fecho convexo (*convex hull*), que por sua vez é expresso na figura 6 já como curva ROC.

Por fim a figura 7 apresenta as duas curvas ROC plotadas no mesmo gráfico para comparação.

Tabela 16: Resultados - Sperotto

J48		
Amostra	TP Rate	FP Rate
1	0,928	0,014
2	0,925	0,024
3	0,907	0,034
4	0,891	0,044
5	0,892	0,092
6	0,894	0,1

Naive Bayes		
Amostra	TP Rate	FP Rate
1	0,629	0,073
2	0,551	0,075
3	0,487	0,074
4	0,432	0,072
5	0,398	0,069
6	0,356	0,067

Tabela 17: Pontos da Curva ROC para C4.5 - Sperotto

Pontos gerados dos classificadores J48			
Ponto	Amostra	TP Rate	FP Rate
1		0	0
2	1	0,928	0,014
3	2	0,925	0,024
4	3	0,907	0,034
5	4	0,891	0,044
6	5	0,892	0,092
7	6	0,894	0,1
8		1	1

Pontos da Curva ROC			
Ponto	Amostra	TP Rate	FP Rate
1		0	0
2	1	0,928	0,014
8		1	1

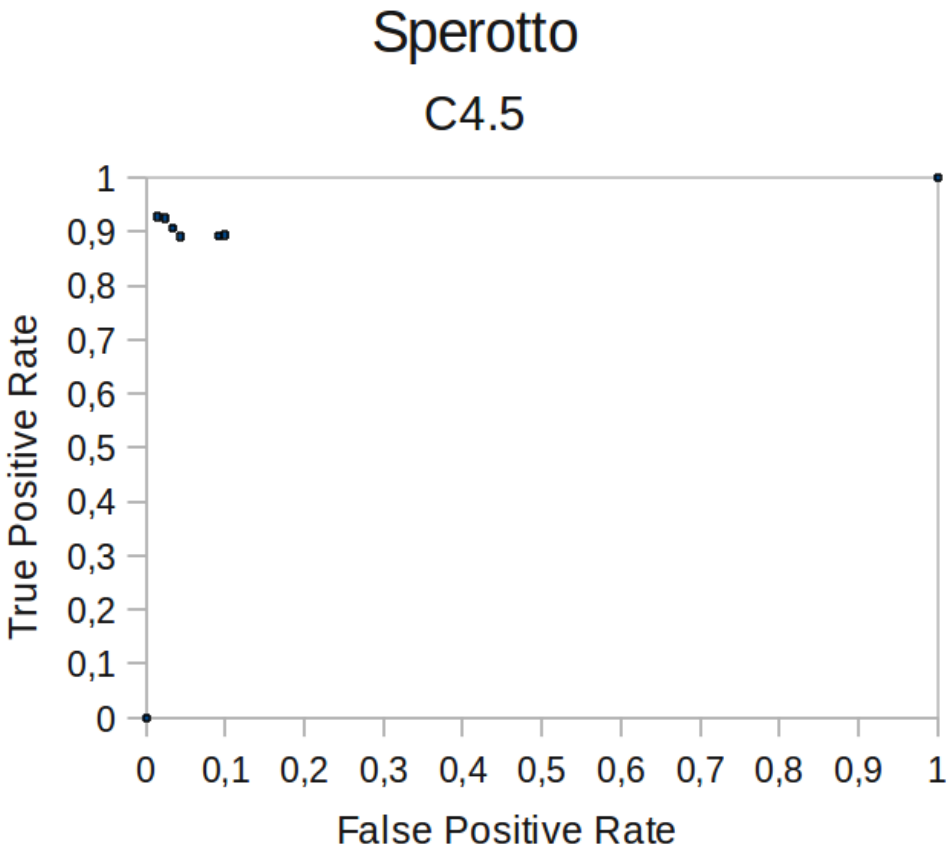

Figura 3: Plotagem dos pontos dos classificadores no gráfico ROC para C4.5 - Sperotto

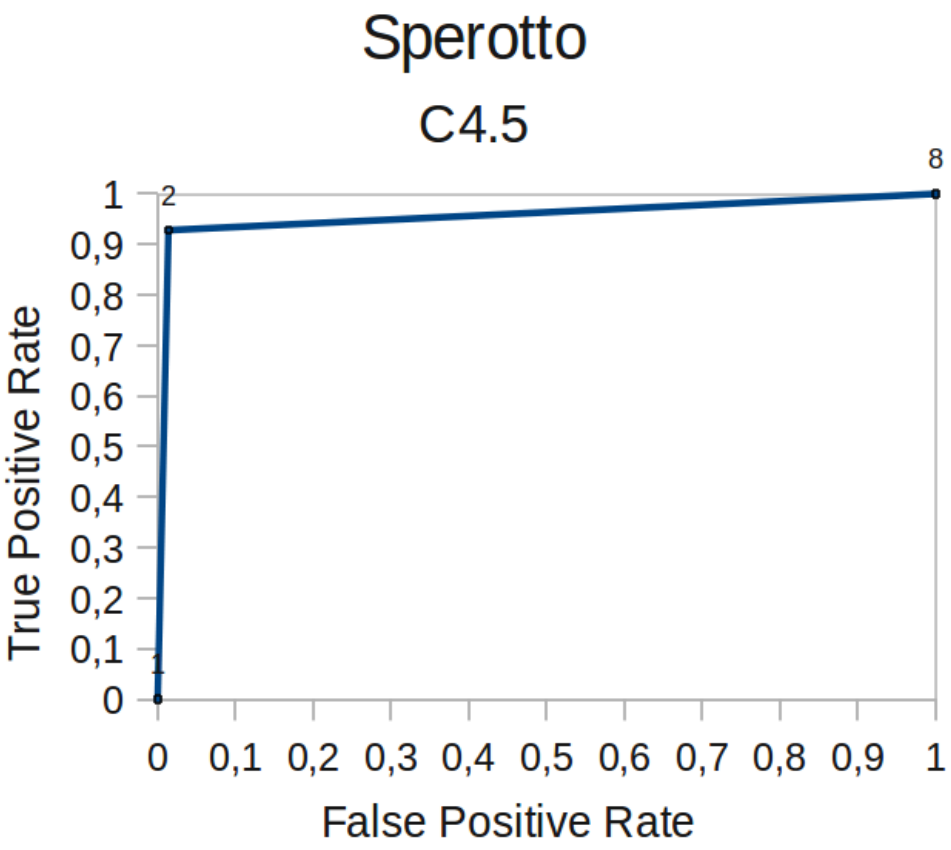

Figura 4: Plotagem da curva ROC para C4.5 - Sperotto

Tabela 18: Pontos da Curva ROC para Naive Bayes - Sperotto

Pontos gerados dos classificadores Naive Bayes			
Ponto	Amostra	TP Rate	FP Rate
1		0	0
2	1	0,629	0,073
3	2	0,551	0,075
4	3	0,487	0,074
5	4	0,432	0,072
6	5	0,398	0,069
7	6	0,356	0,067
8		1	1

Pontos da Curva ROC			
Ponto	Amostra	TP Rate	FP Rate
1		0	0
2	1	0,629	0,073
8		1	1

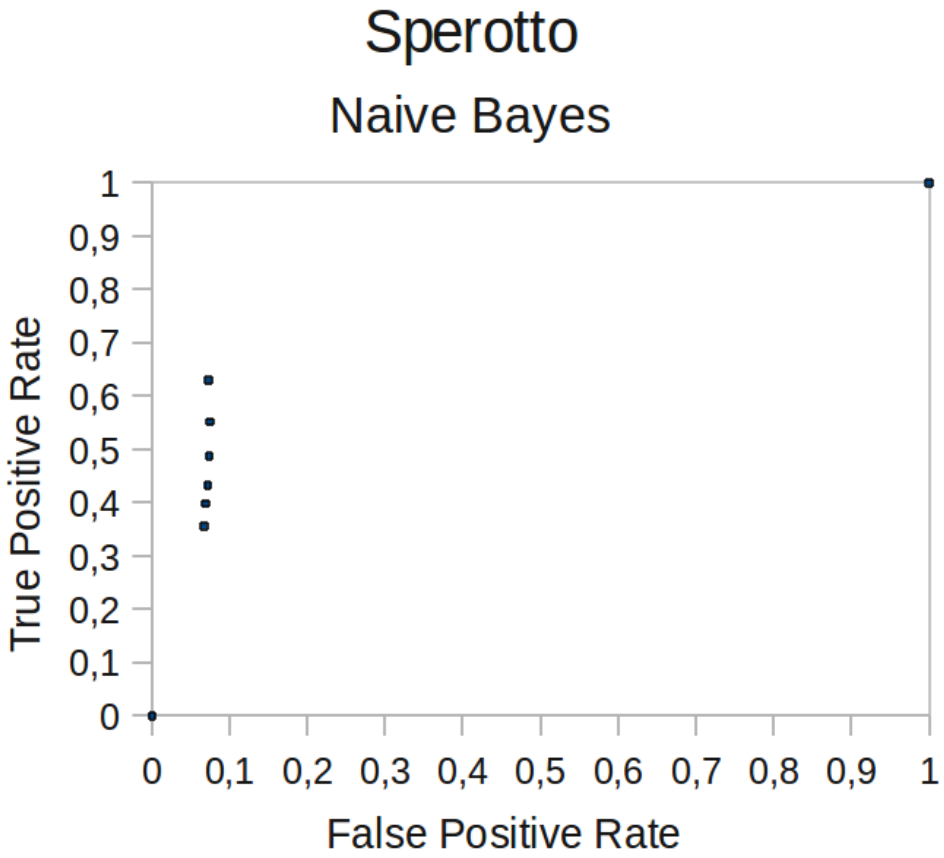

Figura 5: Plotagem dos pontos dos classificadores no gráfico ROC para *Naive Bayes* - Sperotto

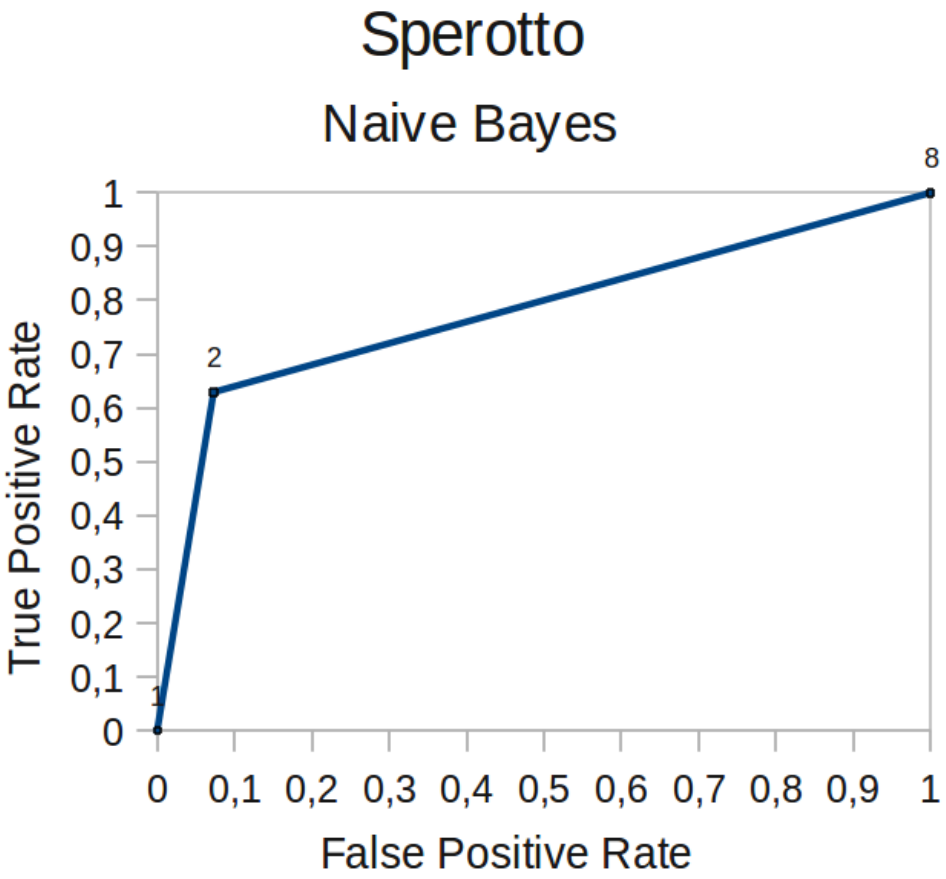

Figura 6: Plotagem da curva ROC para *Naive Bayes* - Sperotto

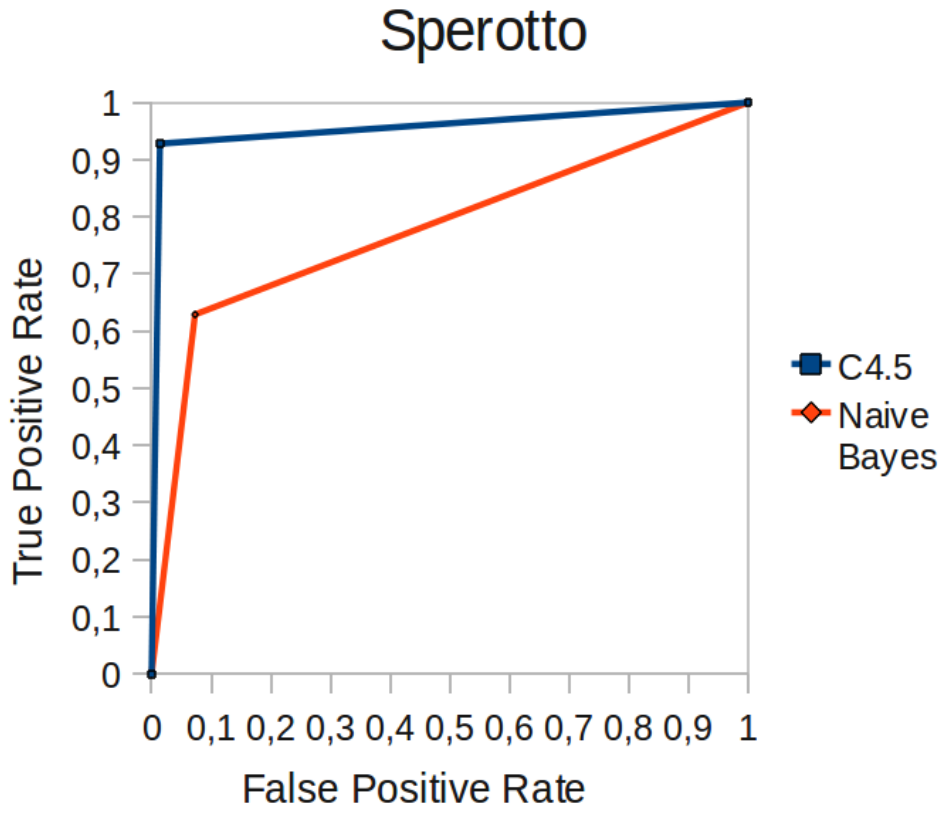

Figura 7: Plotagem para comparação das curvas ROC para C4.5 e *naive Bayes* - Sperotto

5.2 Conjunto de dados II - Moore

Apresentamos abaixo a *matriz de confusão* de nossa amostra balanceada para o algoritmo *C4.5(J48)*:

```
   a   b   c   d   e   f   g   h   i   j   k   l   <-- classified as
 998   0   0   0   0   2   0   0   0   0   0   0 |   a = WWW
   0 995   4   1   0   0   0   0   0   0   0   0 |   b = MAIL
   0   1 998   1   0   0   0   0   0   0   0   0 |   c = FTP-CONTROL
   0   3   0 992   1   2   0   1   1   0   0   0 |   d = FTP-PASV
   1   0   1   3 954  22   1   1  16   0   1   0 |   e = ATTACK
   5   0   1   1  21 946   6   1  14   2   3   0 |   f = P2P
   0   4   1   0   1   4 989   0   0   1   0   0 |   g = DATABASE
   0   0   0   0   0   0   0 1000   0   0   0   0 |   h = FTP-DATA
   0   0   1   0   0  87   0   0 488   0   0   0 |   i = MULTIMEDIA
   0   0   1   0   1   0   0   3   0 994   1   0 |   j = SERVICES
   0   1   0   0   0   4   0   0   2   0 103   0 |   k = INTERACTIVE
   0   0   0   0   0   1   0   0   1   0   0   6 |   l = GAMES
```

E, a seguir, a *matriz de confusão* de nossa amostra balanceada para o algoritmo *naive Bayes*:

```
   a   b   c   d   e   f   g   h   i   j   k   l   <-- classified as
 963   0   0   0  33   3   0   0   0   1   0   0 |   a = WWW
   0 960  15   0   8   0   1   0   5   2   8   1 |   b = MAIL
   0   0 947   0  20   8   3   0   6   3   7   6 |   c = FTP-CONTROL
   2   0  26 695  82   2   4   0   0 188   1   0 |   d = FTP-PASV
  19   0   1  12 893  15   3   0  40  12   1   4 |   e = ATTACK
  79   0  17   0 355 101   1   1 235 177  10  24 |   f = P2P
   2   0   0  21  42   5 275   0   8 605  18  24 |   g = DATABASE
   0   0   0   0   8   4   0 935   2  51   0   0 |   h = FTP-DATA
   0   0  27   1  15   2   1   0 469  52   9   0 |   i = MULTIMEDIA
   0   0   7   3   3   0   0   0   1 985   1   0 |   j = SERVICES
   0   0  17   0  29   2  18   0   1   0  31  12 |   k = INTERACTIVE
   0   0   0   0   3   1   0   0   1   0   1   2 |   l = GAMES
```

A tabela 19 apresenta os valores de *TP Rate* e *FP rate* ponderados para cada amostra do conjunto de dados para os algoritmos *C4.5* e *naive Bayes*.

A tabela 20 apresenta todos os pontos e sua numeração gerados com o algoritmo *C4.5* para a plotagem destes no gráfico ROC, apresentado na figura 8, e os pontos do gráfico ROC após a aplicação do fecho convexo (*convex hull*), que por sua vez é expresso na figura 9 já como curva ROC.

A tabela 21 apresenta todos os pontos e sua numeração gerados com o algoritmo *naive Bayes* para a plotagem destes no gráfico ROC, apresentado na figura 10, e os pontos do gráfico ROC após a aplicação do fecho convexo (*convex hull*), que por sua vez é expresso na figura 11 já como curva ROC.

Por fim, a figura 12 apresenta as duas curvas ROC plotadas no mesmo gráfico para comparação.

Tabela 19: Resultados - Moore

J48		
Amostra	TP Rate	FP Rate
1	0,976	0,003
2	0,979	0,002
3	0,98	0,002
4	0,981	0,002
5	0,982	0,001
6	0,982	0,002
Naive Bayes		
Amostra	TP Rate	FP Rate
1	0,749	0,026
2	0,775	0,022
3	0,792	0,019
4	0,806	0,017
5	0,82	0,015
6	0,828	0,014

Tabela 20: Pontos da Curva ROC para C4.5 - Moore

Pontos gerados dos classificadores J48			
Ponto	Amostra	TP Rate	FP Rate
1		0	0
2	1	0,976	0,003
3	2	0,979	0,002
4	3	0,98	0,002
5	4	0,981	0,002
6	5	0,982	0,001
7	6	0,982	0,002
8		1	1

Pontos da Curva ROC			
Ponto	Amostra	TP Rate	FP Rate
1		0	0
6	5	0,982	0,001
8		1	1

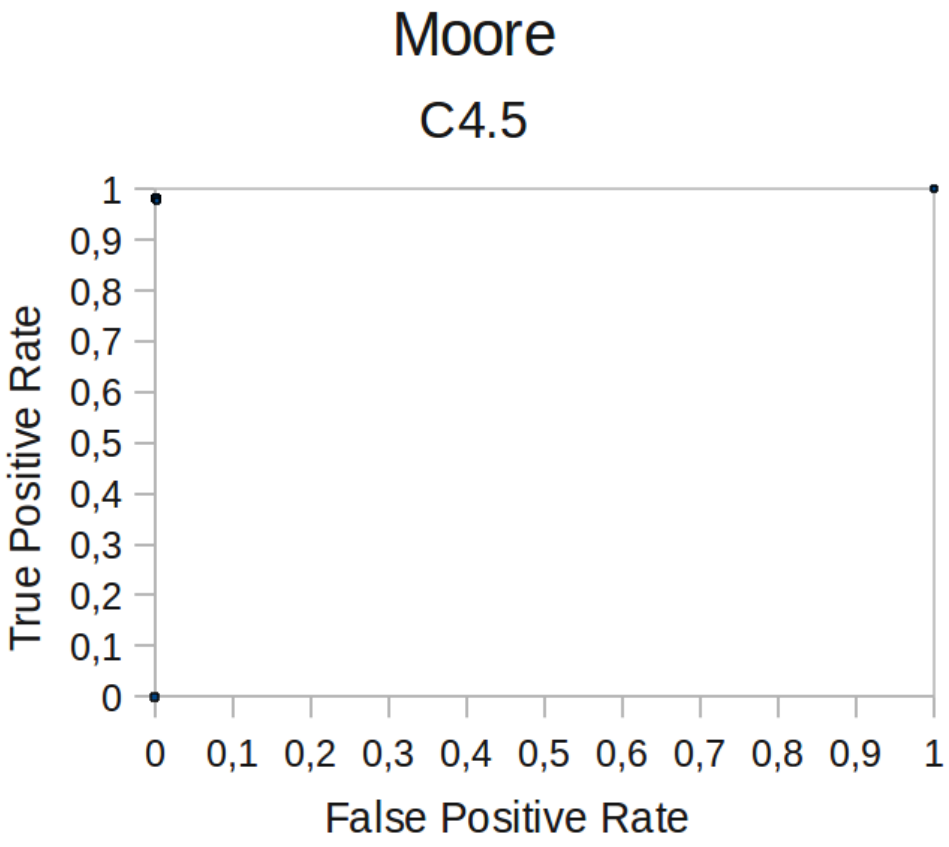

Figura 8: Plotagem dos pontos dos classificadores no gráfico ROC para C4.5 - Moore

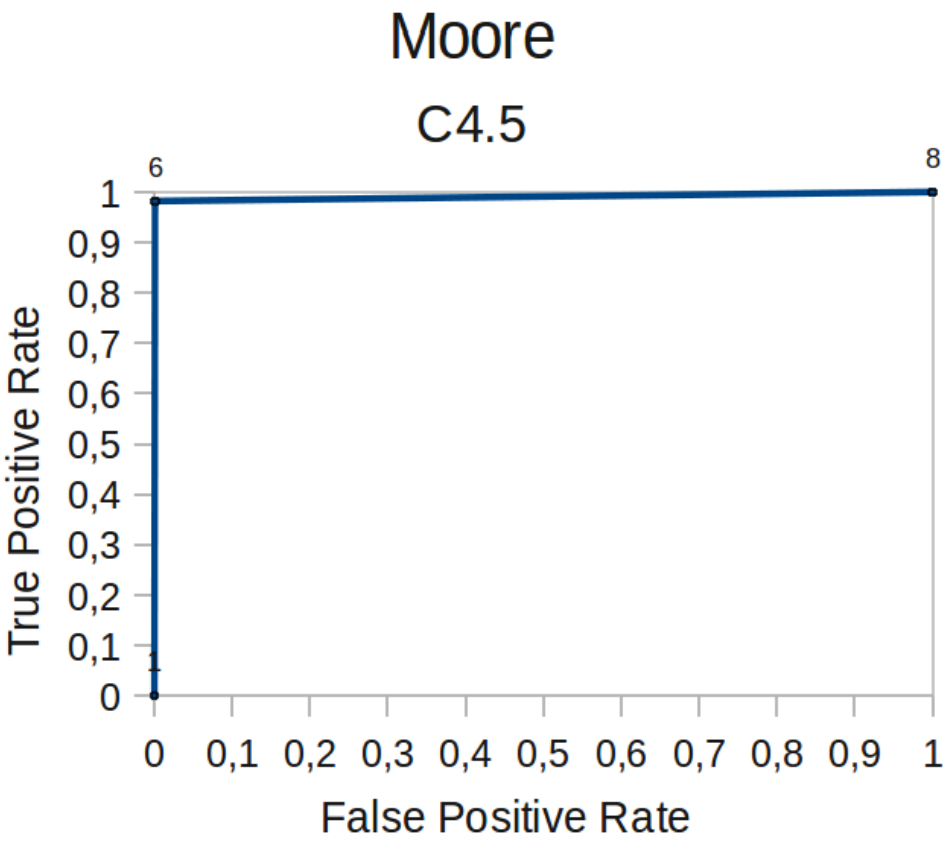

Figura 9: Plotagem da curva ROC para C4.5 - Moore

Tabela 21: Pontos da Curva ROC para Naive Bayes - Moore

Pontos gerados dos classificadores Naive Bayes			
Ponto	Amostra	TP Rate	FP Rate
1		0	0
2	1	0,749	0,026
3	2	0,775	0,022
4	3	0,792	0,019
5	4	0,806	0,017
6	5	0,82	0,015
7	6	0,828	0,014
8		1	1

Pontos da Curva ROC			
Ponto	Amostra	TP Rate	FP Rate
1		0	0
7	6	0,828	0,014
8		1	1

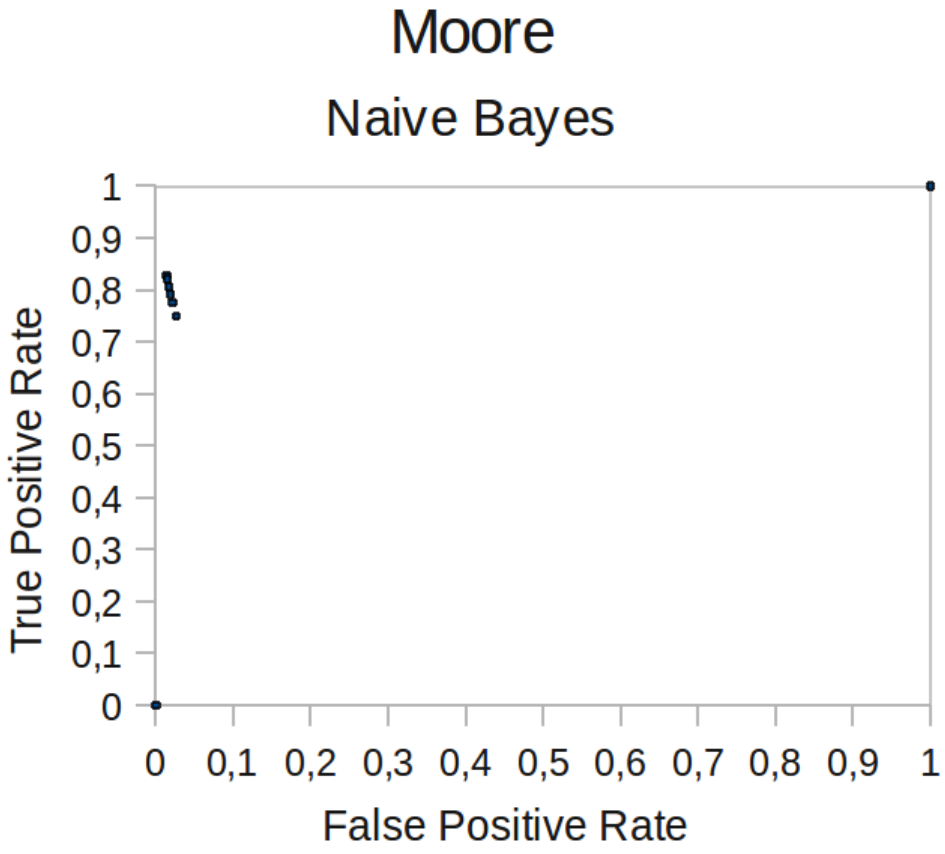

Figura 10: Plotagem dos pontos dos classificadores no gráfico ROC para *Naive Bayes* - Moore

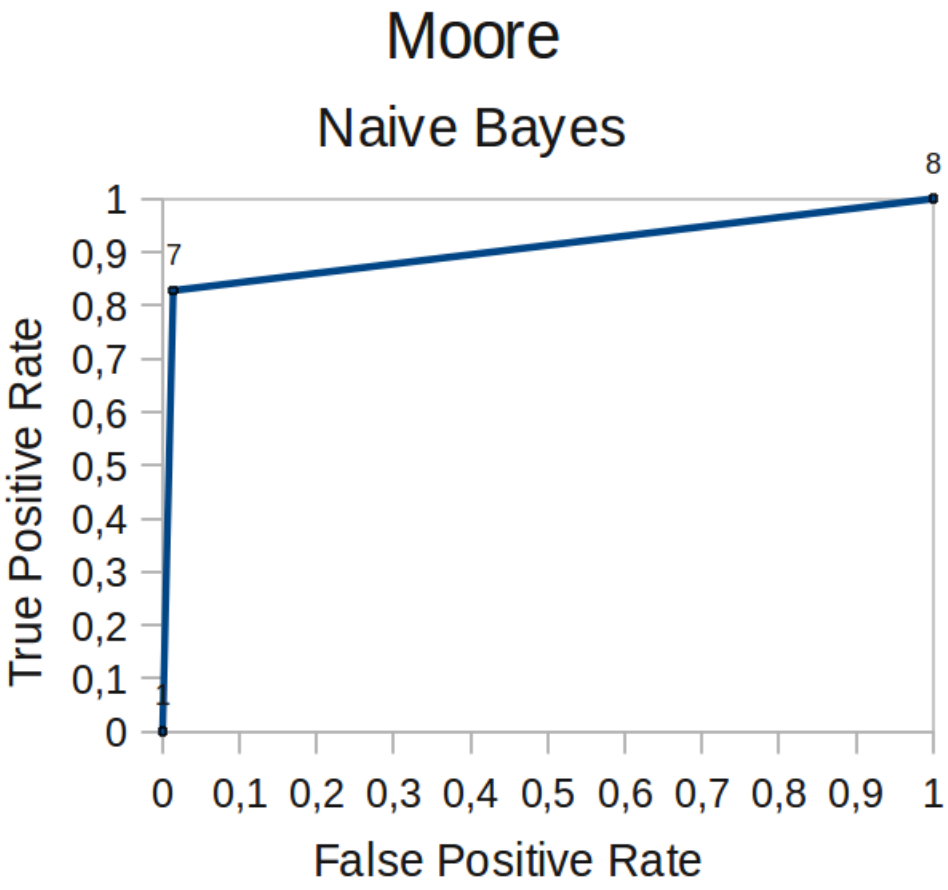

Figura 11: Plotagem da curva ROC para *Naive Bayes* - Moore

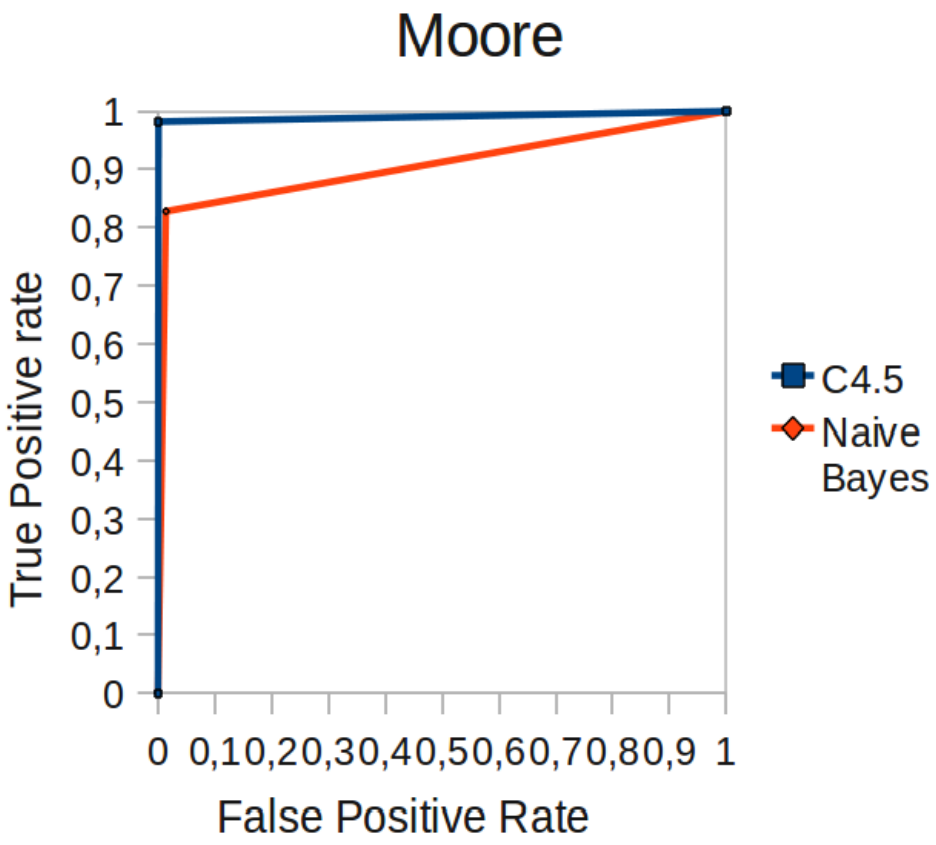

Figura 12: Plotagem para comparação das curvas ROC para C4.5 e *naive Bayes* - Moore

5.3 Conjunto de dados III - KDDCup99

Apresentamos abaixo a *matriz de confusão* de nossa amostra balanceada para o algoritmo *C4.5(J48)*:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	<-- classified as
1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	a = back
0	25	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	b = buffer_overflow
0	0	4	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	c = ftp_write
0	2	0	49	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	d = guess_passwd
0	1	1	0	6	0	0	0	0	1	3	0	0	0	0	0	0	0	0	0	0	0	0	e = imap
0	1	1	0	0	997	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	f = ipsweep
0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	g = land
0	2	0	1	0	0	0	5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	h = loadmodule
0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	2	i = multihop
0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	j = neptune
0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0	0	0	0	0	k = nmap
0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0	0	0	0	l = normal
0	0	0	0	0	0	0	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	m = perl
0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	n = phf
0	0	0	0	0	0	0	0	0	0	0	0	0	0	264	0	0	0	0	0	0	0	0	o = pod
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	995	0	4	0	0	0	0	0	p = portsweep
0	3	1	1	0	0	0	0	1	0	0	0	1	0	0	0	2	0	0	0	1	0	0	q = rootkit
0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	1	2	993	0	0	1	0	0	r = satan
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	0	0	0	0	s = smurf
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	t = spy
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	979	0	0	u = teardrop
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	999	0	v = warezclient
0	0	1	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	1	15	w = warezmaster

E, a seguir, a *matriz de confusão* de nossa amostra balanceada para o algoritmo *naive Bayes*:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	<-- classified as
980	2	0	0	0	2	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	14	0	a = back
0	5	3	0	0	0	0	10	3	0	0	2	0	0	0	0	3	1	0	0	0	0	3	b = buffer_overflow
0	0	7	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	c = ftp_write
0	0	1	51	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	d = guess_passwd
0	0	0	0	11	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	e = imap
0	0	8	5	0	926	0	0	0	0	0	0	0	0	61	0	0	0	0	0	0	0	0	f = ipsweep
0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	g = land
0	1	0	0	0	0	0	6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	h = loadmodule
0	0	1	0	1	0	0	0	2	0	0	0	0	0	0	0	2	0	0	0	0	0	1	i = multihop
0	0	0	0	0	0	17	0	0	981	1	0	0	0	0	1	0	0	0	0	0	0	0	j = neptune
0	0	0	0	0	0	2	5	0	0	990	0	0	0	0	3	0	0	0	0	0	0	0	k = nmap
0	0	2	0	0	0	0	0	1	0	0	997	0	0	0	0	0	0	0	0	0	0	0	l = normal
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	m = perl
1	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	n = phf
0	0	3	0	0	0	0	0	0	0	0	0	0	0	258	0	1	0	0	0	0	0	2	o = pod
0	0	0	0	0	5	2	0	0	0	0	0	0	0	0	952	0	41	0	0	0	0	0	p = portsweep
0	0	2	0	0	0	0	1	1	0	0	0	4	0	0	1	1	0	0	0	0	0	0	q = rootkit
0	0	0	0	0	0	0	0	0	0	0	0	6	0	7	21	9	956	0	0	1	0	0	r = satan
0	0	0	0	0	0	0	0	0	0	0	0	1	0	14	0	0	0	985	0	0	0	0	s = smurf
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	t = spy
0	0	0	0	0	0	0	0	0	0	0	0	3	0	1	0	2	0	0	0	973	0	0	u = teardrop
0	136	218	2	1	1	0	42	0	0	0	10	0	0	8	2	29	2	0	0	0	515	34	v = warezclient

```
0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2  17 |   w = warezmaster
```

A tabela 22 apresenta os valores de *TP Rate* e *FP rate* ponderados para cada amostra do conjunto de dados para os algoritmos *C4.5* e *naive Bayes*.

A tabela 23 apresenta todos os pontos e sua numeração gerados com o algoritmo *C4.5* para a plotagem destes no gráfico ROC, apresentado na figura 13, e os pontos do gráfico ROC após a aplicação do fecho convexo (*convex hull*), que por sua vez é expresso na figura 14 já como curva ROC.

A tabela 24 apresenta todos os pontos e sua numeração gerados com o algoritmo *naive Bayes* para a plotagem destes no gráfico ROC, apresentado na figura 15, e os pontos do gráfico ROC após a aplicação do fecho convexo (*convex hull*), que por sua vez é expresso na figura 16 já como curva ROC.

Por fim, a figura 17 apresenta as duas curvas ROC plotadas no mesmo gráfico para comparação.

Tabela 22: Resultados - KDDCup

J48		
Amostra	TP Rate	FP Rate
1	0,994	0
2	0,996	0
3	0,995	0
4	0,995	0
5	0,996	0
6	0,996	0
7	0,996	0

Naive Bayes		
Amostra	TP Rate	FP Rate
1	0,925	0,001
2	0,931	0,001
3	0,926	0,001
4	0,93	0,001
5	0,936	0,001
6	0,939	0,001
7	0,94	0,001

Tabela 23: Pontos da Curva ROC para C4.5 - KDDCup

Pontos gerados dos classificadores J48			
Ponto	Amostra	TP Rate	FP Rate
1		0	0
2	1	0,994	0
3	2	0,996	0
4	3	0,995	0
5	4	0,995	0
6	5	0,996	0
7	6	0,996	0
8	7	0,996	0
9		1	1

Pontos da Curva ROC			
Ponto	Amostra	TP Rate	FP Rate
1		0	0
8	7	0,996	0
9		1	1

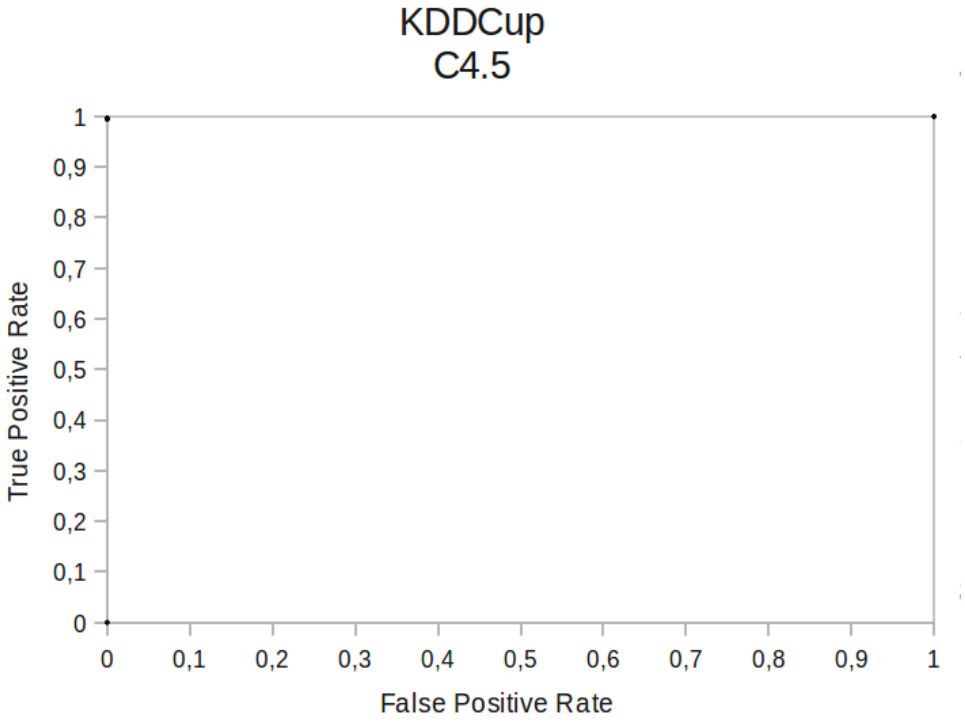

Figura 13: Plotagem dos pontos dos classificadores no gráfico ROC para C4.5 - KDDCup

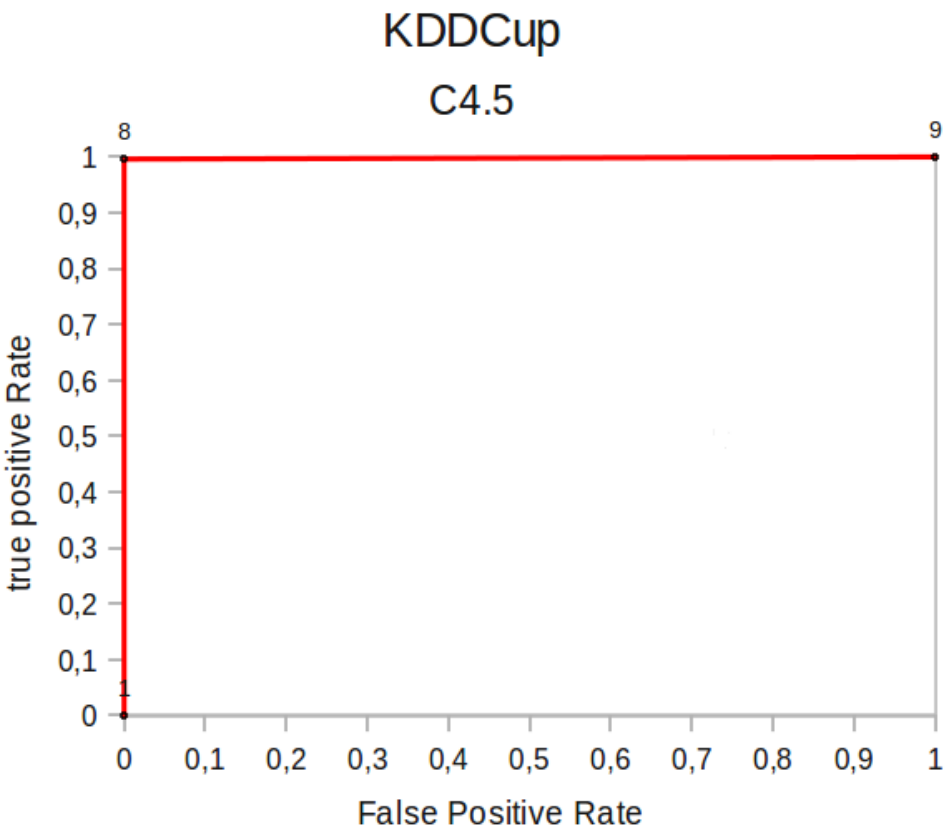

Figura 14: Plotagem da curva ROC para C4.5 - KDDCup

Tabela 24: Pontos da Curva ROC para Naive Bayes - KDDCup

Pontos gerados dos classificadores Naive Bayes			
Ponto	Amostra	TP Rate	FP Rate
1		0	0
2	1	0,925	0,001
3	2	0,931	0,001
4	3	0,926	0,001
5	4	0,93	0,001
6	5	0,936	0,001
7	6	0,939	0,001
8	7	0,94	0,001
9		1	1

Pontos da Curva ROC			
Ponto	Amostra	TP Rate	FP Rate
1		0	0
8	7	0,94	0,001
9		1	1

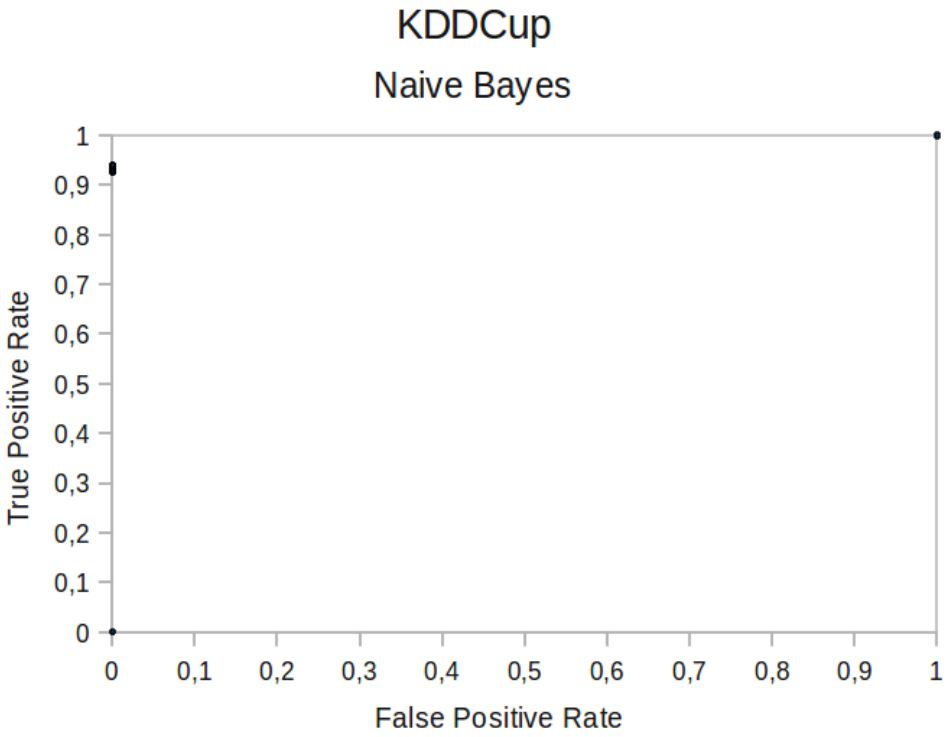

Figura 15: Plotagem dos pontos dos classificadores no gráfico ROC para *Naive Bayes* - KDD-Cup

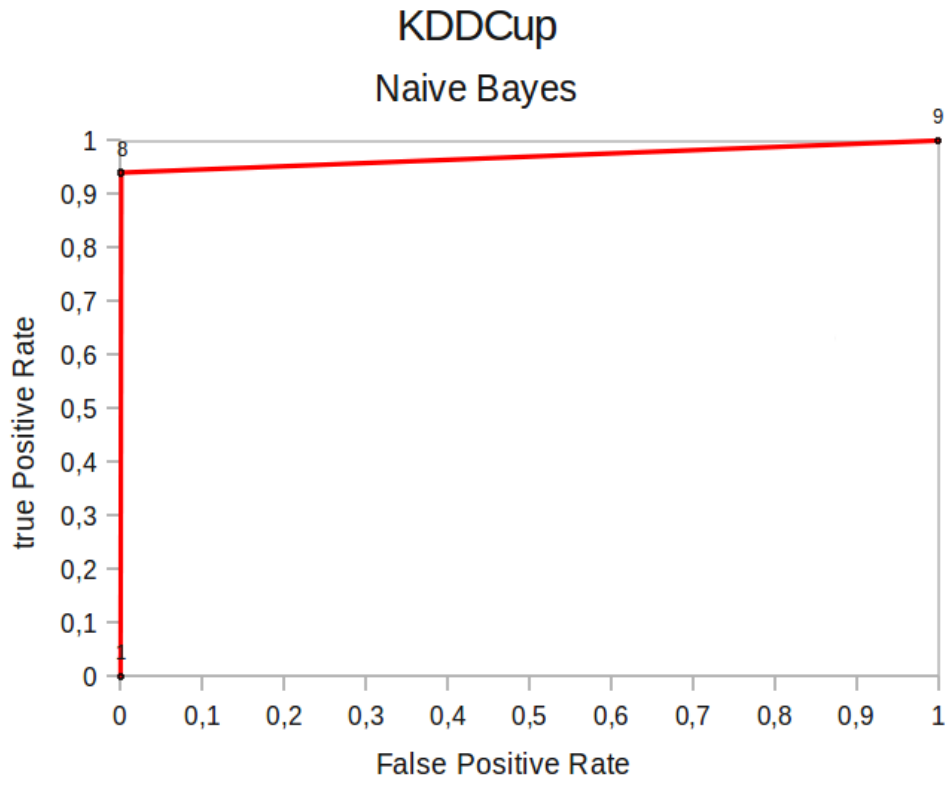

Figura 16: Plotagem da curva ROC para *Naive Bayes* - KDDCup

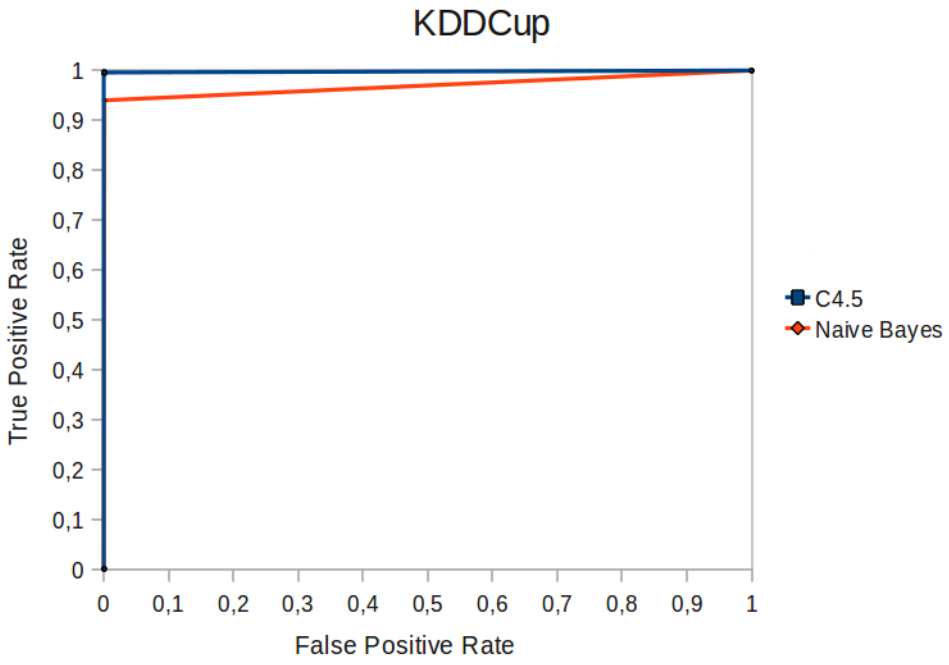

Figura 17: Plotagem para comparação das curvas ROC para C4.5 e *naive Bayes* - KDDCup

5.3.1 Discussão

Tendo estes resultados apresentados, podemos fazer algumas afirmações sobre o constatado.

Na aplicação dos dois algoritmos em todos os conjuntos de dados obtivemos resultados melhores que classificadores aleatórios. Consideramos um classificador aleatório aquele que estiver sobre a linha onde *TP Rate* = *FP rate*, ou seja, o segmento de reta que vai do ponto 0,0 a 1,1 (PRATI; BATISTA; MONARD, 2008). Todos os classificadores obtidos estão acima dessa linha, logo, todos apresentaram desempenho melhor que um classificador aleatório.

No geral obtivemos bons resultados, alguns se aproximando muito de classificadores ótimos (classificador que possui ponto no gráfico ROC plotado nas coordenadas 0,1), como foi verificado na utilização do algoritmo C4.5 nos conjuntos de dados Moore e KDDCup, apresentados nas figuras 9 e 14, respectivamente.

É perceptível que nos três conjuntos de dados obtivemos melhores resultados com o algoritmo C4.5.

Isto pode ser afirmado com base na comparação apresentada nos gráficos apresentados nas figuras 7, 12 e 17, pois todos apresentam a linha ROC relativa à aplicação do algoritmo C4.5 dominantes sobre a gerada pelo algoritmo *naive Bayes*, sem apresentar interseções (PRATI; BATISTA; MONARD, 2008). Assim podemos afirmar que o desempenho do algoritmo C4.5 é superior ao naiver Bayes em todos os caso apresentados .

Porém não é coerente extrapolar essa verificação para o mundo real. O cenário abordado neste trabalho é muito restrito se comparado ao mundo real. Um exemplo disso é a pequena variedade de tipos de ataque do conjunto de dados de Sperotto et al. .

Outra observação que pode ser feita é que obtivemos resultados mais próximos entre os algoritmos utilizados nas bases que apresentam uma maior variedade de informação, ou seja, uma maior quantidade de atributos. Isso é evidente quando comparamos os resultados obtidos no conjunto de dados Sperotto e al. e os outros dois conjuntos de dados. Nesse conjunto de dados utilizamos apenas 5 atributos, e o resultado obtido na curva ROC foi o mais discrepante entre os obtidos.

Assim, podemos afirmar que há uma tendência para um melhor desempenho do algoritmo C4.5, que se acentua em conjunto de dados com poucos atributos, e que os dois algoritmos apresentaram bons resultados na classificação.

6 CONCLUSÃO

Este trabalho teve como principal foco analisar e comparar o desempenho dos classificadores C4.5 e naive Bayes na classificação de tráfego Internet baseada em fluxos de rede. Como apresentado no capítulo 5, obtivemos um melhor desempenho com o algoritmo C4.5, o qual em todos os casos apresentados foi superior ao algoritmo naive Bayes, segundo a técnica de avaliação de desempenho utilizada.

Porém não é correto dizer que este resultado pode ser considerado conclusivo quanto à supremacia do algoritmo C5.4 sobre naive Bayes no contexto de classificação de tráfego. Mesmo utilizando três bases diferentes, que ofereciam tecnicamente cenários distintos, ainda é uma variedade de cenários muito restrita a encontrada no mundo real.

Além disso, foi mostrado que sob determinadas circunstâncias o algoritmo naive Bayes se aproxima muito do desempenho do algoritmo C4.5 como ocorreu nos nossos testes com a base KDDCup 99, apresentados no capítulo 5, na qual podemos apontar a existência de um maior número de informações sobre o tráfego (refletida na quantidade de atributos utilizados do conjunto de dados, maior em número que nas outras duas bases) como aspecto relevante.

Contudo, de forma geral podemos perceber neste trabalho uma tendência a um melhor desempenho do algoritmo C4.5.

Além do objetivo principal, vários outros benefícios foram conseguidos, direta ou indiretamente, com o desenvolvimento deste trabalho. Desde uma introdução ao problema de classificação tráfego de rede, os seus diversos obstáculos e desafios, até o estudo de algoritmos de aprendizado de máquina. Propiciou-nos um entendimento, mesmo que superficial, da área de aprendizado de máquina e de duas técnicas amplamente utilizadas, que são o algoritmo C4.5 e naive Bayes, ambas aplicadas a um problema real. O primeiro contato e utilização da técnica de avaliação curvas ROC, de aplicação em diversos contextos e áreas. O contato com bases reais de dados de tráfego de rede, as quais seu próprio desenvolvimento já consiste um grande desafio. E, não menos importante, a utilização do pacote WEKA, que é uma ferramenta extremamente útil na utilização e desenvolvimento de técnicas de aprendizado de máquina, e uma abordagem da implementação dos algoritmos utilizados aqui que este oferece.

Acreditamos que este trabalho possibilite diversas extensões. Apresentamos a seguir algumas ideias levantadas com este fim:

1. Análise e comparação de performance de outros algoritmos de aprendizado de máquina aplicados à classificação de tráfego de rede;

2. Comparação de performance das técnicas clássicas e de técnicas estatísticas recentes com as técnicas de aprendizado de máquina na classificação de tráfego de rede, sob diversos

níveis de restrições;

3. Utilização de algoritmos de aprendizado de máquina geradores de regras aplicados à otimização de técnicas clássicas de classificação de tráfego de rede;

4. Análise de desempenho da integração de técnicas na classificação de tráfego de rede;

5. Estudo e definição dos ambientes e das restrições impostas à classificação de tráfego de rede de forma concisa, visando orientar o desenvolvimento e aplicação de novas técnicas para cada ambiente;

6. Desenvolvimento de uma *suite* ou *framework*, nos moldes do pacote WEKA, que englobe todo o processo de classificação de tráfego de rede, desde a captura do tráfego sob diversos formatos, até a aplicação de diversas técnicas para sua análise e classificação, propiciando a integração de técnicas, análise de desempenho e uma fácil extensão.

Acreditamos que, apesar de apesar de sua simplicidade, e não ter chegado a uma resposta conclusiva sobre o problema abordado, tenha sido desenvolvido um trabalho útil, pois propiciou uma introdução consistente à classificação de tráfego de rede, apresentando a problemática enfrentada, técnicas clássicas e técnicas recentes, e abordando problemas correlatos, como as restrições existentes na utilização dos dados disponíveis na rede.

Estamos cientes das limitações do trabalho apresentado, porém acreditamos ter alcançado um resultado satisfatório e que responde, mesmo que de forma parcial, aos objetivos abordados.

BIBLIOGRAFIA

AULD, T.; MOORE, A.W.; GULL, S.F. Bayesian neural networks for internet traffic classification. *Neural Networks, IEEE Transactions on*, IEEE, v. 18, n. 1, p. 223–239, 2006.

CHOI, K.; CHOI, J.K. Pattern Matching of Packet Payload for Network Traffic Classification. In: *The Joint International Conference on Optical Internet and Next Generation Network, 2006. COIN-NGNCON 2006*. [S.l.: s.n.], 2006. p. 130–132.

HALOI, T.A.M.R.A. Network Traffic Classification for Intrusion Detection. 2004.

HOUAISS, A. Dicionário Houaiss da língua portuguesa. *São Paulo*, 2001.

KARAGIANNIS, T.; PAPAGIANNAKI, K.; FALOUTSOS, M. BLINC: multilevel traffic classification in the dark. *ACM SIGCOMM Computer Communication Review*, ACM, v. 35, n. 4, p. 240, 2005.

KUROSE, J.; ROSS, K. *Redes de Computadores e Internet*. [S.l.]: São Paulo: Person, 2006.

LUGER, G.F. *Inteligência artificial: estruturas e estratégias para a solução de problemas complexos*. [S.l.]: Porto Alegre: Bookmann, 2004.

MOORE, A.W.; ZUEV, D.; CROGAN, M. Discriminators for use in flow-based classification. *RR-05.13 Department of Computer Science, University of London*, Citeseer, 2005.

NWANZE, N.; SUMMERVILLE, D.H.; SKORMIN, VA. Real-time identification of anomalous packet payloads for network intrusion detection. In: *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*. [S.l.: s.n.], 2005. p. 448–449.

PAN, Z.S. et al. Hybrid neural network and C4. 5 for misuse detection. In: IEEE. *Machine Learning and Cybernetics, 2003 International Conference on*. [S.l.], 2004. v. 4, p. 2463–2467.

POSTEL, J. et al. WELL KNOWN PORT NUMBERS. Citeseer, 2005.

PRATI, RC; BATISTA, G.; MONARD, MC. Curvas ROC para avaliação de classificadores. *Revista IEEE América Latina*, v. 6, n. 2, p. 215–222, 2008.

SPEROTTO, A. et al. A Labeled Data Set For Flow-based Intrusion Detection. *IP Operations and Management*, Springer, p. 39–50, 2009.

WITTEN, I.H.; FRANK, E. *Data Mining: Practical machine learning tools and techniques*. [S.l.]: Morgan Kaufmann Pub, 2005.

WU, X. et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, Springer, v. 14, n. 1, p. 1–37, 2008.

ZUEV, D.; MOORE, A.W. Traffic classification using a statistical approach. *Passive and Active Network Measurement*, Springer, p. 321–324, 2005.

APÊNDICE

APÊNDICE A – O FORMATO ARFF

Definimos nesta seção o formato dos dados utilizados, ao qual convertemos os conjuntos utilizados. O formato consiste em um arquivo de texto plano, de extensão ".arff". É interessante ressaltar que a estrutura do arquivo em questão não indica o propósito dos dados que ele armazena. Ele apenas funciona como um repositório de dados, podendo estes dados serem utilizados para quaisquer tipos de análise. As definições do que se deseja fazer ocorrerão já na aplicação WEKA, não sendo definido na hora da construção do arquivo.

A.0.2 Estrutura

Podemos dividir a estrutura deste formato em duas seções: cabeçalho e corpo.

Estas seções podem ser permeadas de comentários, que consistem em linhas iniciadas pelo caractere "%".

A primeira seção corresponde ao cabeçalho do arquivo, que apresenta as definições de nome e formato de cada campo das instâncias.

Esta seção inicia com uma linha com a palavra chave *@relation* seguida por um nome, que nomeia a relação apresentada neste arquivo. Caso este nome apresente espaços, deve ser cercado por aspas simples.

Formato:

@relation <nome>

Exemplo:

@relation flows

As linhas seguintes definem os atributos de cada instância, nomeando e indicando seu formato. Estas linhas iniciam com a palavra chave *@attribute* seguida pelo nome do atributo e por seu formato.

Os tipos principais possíveis são:

1. *numeric*: representa inteiros e valores reais;
2. *nominal*: representa valores discretos, sendo necessário definir os valores possíveis como uma lista após o nome do atributo;

3.*string*: utilizada para textos. Os valores para este campo devem ser cercados por aspas;

4.*date*: utilizada para representar datas, porém é tratado como o formato *numeric* internamente. Deve seguir o formato "yyyy-MM-dd'T'HH:mm:ss".

Formato:

@attribute <nome> <tipo>

Exemplo:

@attribute port_number numeric@attribute tipo {tipo1,tipo2,tipo3}

A utilização dos formatos deve ser compatível ao tipo de dados que se quer analisar, e a sua utilização em cada algoritmo varia um pouco.

Segue-se à descrição dos atributos uma linha com a palavra chave *@data* que indica o início da listagem das instâncias. Cada linha corresponde a uma instância, sendo constituída por seus valores de atributos, separados por vírgulas, na exata ordem que os atributos foram definidos no cabeçalho. Os valores de cada campo devem respeitar o tipo definido no cabeçalho. Caso haja alguma campo do qual não se tenha o valor, deve-se colocar o caractere '?' para representar sua ausência. As instâncias são independentes, não apresentam ordenação e não apresentam relações entre si.

Exemplo:

2042,113,1,48,1222254970,95,?,95,0,2,TCP,TRUE,TRUE

Valores de *string* e nominais são *case sensitive*, e em qualquer campo que contenha espaços é necessário delimitá-los com aspas simples. As palavras chave *@RELATION*, *@ATTRIBUTE* e *@DATA* não são *case sensitive*.

Há outras possibilidades na construção deste arquivo, mas nos restringimos ao que foi utilizado neste trabalho. Mais informações sobre este formato podem se encontradas em (WITTEN; FRANK, 2005).

APÊNDICE B – MÉTRICAS DE DESEMPENHO

B.0.3 Métricas *TP rate* e *FP rate*

As métricas $TP\ rate$ e $FP\ rate$ consistem em uma maneira natural de apresentar as estatísticas de um modelo de classificação, e se baseiam na frequência absoluta das classificações reais das instâncias no conjunto de dados e nas previsões realizadas pelo classificador. Por si só fornecem uma métrica de avaliação de um modelo, porém as apresentamos como base para a métrica utilizada neste trabalho, a ser detalhada na seção seguinte.

Considerando-se apenas duas classe P e N, respectivamente $positiva$ e $negativa$, temos:

$$TP\ rate = \frac{TP}{P} \tag{B.1}$$

$$FP\ rate = \frac{FP}{N} \tag{B.2}$$

nas quais:

- P é o número de instâncias da classe P;
- N é o número de instâncias da classe N;
- TP (*True Positive*) é o número de exemplos da classe P classificados como P;
- FP (*False Positive*) é o número de exemplos da classe N classificados como P;

Quando forem tratadas múltiplas classes, haverá um $TP\ rate$ e um $FP\ rate$ para cada classe, que serão calculados com as fórmulas apresentadas em B.1 e B.2, porém considerando-se a classe abordada como classe P e todas as outras classes como uma classe única, N.

B.0.4 Curvas ROC

Apresentamos nesta seção a análise ROC, que constitui a técnica de avaliação de desempenho de classificadores utilizada neste trabalho.

Análise ROC (*Receiver Operating Characteristics*) é uma ferramenta útil para visualização e avaliação de modelos de classificação. Consiste em um modelo gráfico para avaliação, organização e seleção de sistemas de predição, provendo uma avaliação mais rica que a partir de uma única medida (PRATI; BATISTA; MONARD, 2008).

O procedimento de construção desse avaliador consiste na plotagem de um gráfico bidimensional, cujo eixo X reflete o $FP\ rate$ e o eixo Y reflete o $TP\ rate$. É inserido no gráfico um ponto para cada classificador de acordo com o valor de sua $FP\ rate$ e sua $TP\ rate$ verificados. No caso de um classificador trabalhar com múltiplas classes, como ocorre nos classificadores abordados durante este trabalho, uma técnica para definir os valores $FP\ rate$ e $TP\ rate$ utilizados no gráfico é utilizar a média ponderada destes valores de todas as classes.

As fórmulas B.3 e B.4 expressam a aplicação da média ponderada às métricas $FP\ rate$ e $TP\ rate$.

$$TPrate_{ponderada} = \frac{\sum_{i=1}^{x} TPrate_i * n_i}{N} \tag{B.3}$$

$$FPrate_{ponderada} = \frac{\sum_{i=1}^{x} FPrate_i * n_i}{N} \tag{B.4}$$

nas quais x é o número de classes, $TPrate_i$ e $FPrate_i$ são, respectivamente, o $TP\ rate$ e o $FP\ rate$ da classe i, n_i é o número de instâncias reais da classe i, N o número total de instâncias do conjunto de dados.

Estes valores são fornecidos na saída padrão do pacote WEKA, bastando utilizá-los de acordo com o nosso propósito.

A estes pontos adicionamos o ponto (0,0), que representa a estratégia de nunca classificar um exemplo como positivo, e o ponto (1,1) que indica a estratégia de sempre classificar um novo exemplo como positivo.

Classificadores derivados tendem a formar uma nuvem de pontos no gráfico ROC. Nos pontos plotados para um classificador e suas derivações, podemos considerar apenas o envelope externo convexo (*convex hull*) que mais se aproximam do ponto (0,1), ponto que representa os modelos que podem ser considerados ótimos, dada uma certa condição operacional. Os modelos que não fazem parte do envelope convexo podem ser descartados (PRATI; BATISTA; MONARD, 2008).

Assim, para um conjunto de classificadores derivados, teremos um subconjunto de pontos, que fazem parte do envelope convexo considerado. Com estes pontos nos é possível plotar no gráfico a curva ROC referente a este conjunto de classificadores derivados.

Da mesma forma podemos plotar mais de uma curva em um mesmo gráfico para conjuntos distintos de classificadores derivados. Assim, em um mesmo gráfico, teremos uma curva para um conjunto de classificadores derivados, o que nos permite fazer uma comparação entre estes. Nos casos em que não há intersecções entre as curvas, os modelos de classificação derivados da curva mais próxima ao ponto (0,1), que representa o classificador em que todos

os exemplos positivos e negativos são corretamente classificados, serão sempre melhores do que os modelos derivados a partir de outra curva. Nos casos em que há intersecção, os classificadores pertencentes ao trechos da curva mais próximos do ponto (0,1) serão os melhores, sendo necessário delimitar estes trechos para efetuar uma comparação adequada (PRATI; BATISTA; MONARD, 2008).

APÊNDICE C – TRANSFORMAÇÃO DA BASE DE DADOS SPEROTTO ET AL.

Apresentamos aqui o *script* SQL utilizado para a transformação da base de dados desenvolvida por Sperotto et al. para o formato ARFF, utilizado na plataforma WEKA. Este script é específico para o SGBD MySQL.

C.0.5 SQL

```
1  /*
2  SQL utilizado para a transformacao da base de dados (MYSQL)
       elaborada por Sperotto et al. em formato arff, compativel
       com a plataforma WEKA.
3  */
4   SELECT
5           /*
6           Campos referentes a tabela 'flows'.
7           */
8       f.src_ip,
9       f.dst_ip,
10      f.src_port,
11      f.dst_port,
12      f.packets,
13      f.octets,
14      f.start_time,
15      f.start_msec,
16      f.end_time,
17      f.end_msec,
18          /*
19          Campo acrescido que indica a duracao do fluxo em
               milisegundos baseado nos campos 'start_time',
               'start_msec', 'end_time', 'end_msec' da tabela
               'flows'.
20          */
21      (f.end_time - f.start_time) * 1000 + (f.end_msec -
           f.start_msec) as duration,
22      f.tcp_flags,
23
24          /*
```

```
25          Mapeamento do protocolo com base na especificacao do
                Cisco Netflow.
26          */
27      CASE f.prot
28          WHEN 1  THEN 'ICMP'
29          WHEN 6  THEN 'TCP'
30          WHEN 17 THEN 'UDP'
31      END as protocol,
32
33          /*
34          Campo acrescido que indica se o fluxo e malicioso ou
                nao.
35          */
36      CASE
37          WHEN flow_alert.alertid IS NULL THEN 'FALSE'
38          ELSE 'TRUE'
39      END as malicious ,
40
41          /*
42          A partir deste campo, caso o fluxo nao seja malicioso,
                todos os campos constarao como NULL, sendo
                representado pela string '\N'.
43          Isto e realizado atraves da verificacao do campo
                'alertid' da tabela 'flow_alert' pois caso este
                campo seja nulo significa que o fluxo em questao
                nao se relaciona com nenhum alerta.
44          */
45
46          /*
47          Neste campo foi acrescida a verficiacao de nulidade do
                campo mesmo o fluxo deste sendo malicioso, pois
                alguns fluxos nao tiveram este campo definido,
                recebendo entao o simbolo '?' para indicar que nao
                foi possivel identificar seu valor. Varios outros
                campos subsequentes recebem o mesmo tratamento.
48          */
49          /*
50          Os proximos campos sao relativos a tabela 'alerts',
                indicando o evento de alerta do fluxo.
```

```
51          */
52      CASE
53          WHEN flow_alert.alertid IS NOT NULL THEN
54              CASE
55                  WHEN a.automated IS NULL THEN '?'
56                  WHEN a.automated = 0     THEN 'FALSE'
57                  WHEN a.automated = 1     THEN 'TRUE'
58              END
59          ELSE NULL
60      END as automated ,
61
62      CASE
63          WHEN flow_alert.alertid IS NOT NULL THEN
64              CASE
65                  WHEN a.succeeded IS NULL THEN '?'
66                  WHEN a.succeeded = 0     THEN 'FALSE'
67                  WHEN a.succeeded = 1     THEN 'TRUE'
68              END
69          ELSE NULL
70      END as succeeded,
71
72      CASE
73          WHEN flow_alert.alertid IS NOT NULL THEN
              CONCAT('\'',a.description,'\'')
74          ELSE NULL
75      END as alert_description,
76
77          /*
78          Campo da tabela 'alert_type' que indica a
                classificacao do fluxo.
79          */
80
81      t.description,
82
83          /*
84          Campo acrescido que indica se o fluxo foi originado
                por um evento anterior.
85          */
86
```

```
87      CASE
88         WHEN flow_alert.alertid IS NOT NULL THEN
89               CASE
90                  WHEN c.parent IS NULL THEN 'FALSE'
91                  ELSE 'TRUE'
92               END
93         ELSE NULL
94      END as causality,
95
96         /*
97         Os proximos campos sao relativos a tabela 'alerts',
               indicando o evento 'causality' que gerou o fluxo
               atual.
98         */
99
100     CASE
101        WHEN flow_alert.alertid IS NOT NULL THEN
102           CASE
103              WHEN c.parent IS NOT NULL THEN
104                 CASE
105                    WHEN a2.automated IS NULL THEN '?'
106                    WHEN a2.automated = 0     THEN 'FALSE'
107                    WHEN a2.automated = 1     THEN 'TRUE'
108                 END
109              ELSE 'NO_CAUSALITY'
110           END
111        ELSE NULL
112     END as causality_automated,
113
114
115     CASE
116         WHEN flow_alert.alertid IS NOT NULL THEN
117           CASE
118              WHEN c.parent IS NOT NULL THEN
119                 CASE
120                    WHEN a2.succeeded IS NULL THEN '?'
121                    WHEN a2.succeeded = 0     THEN 'FALSE'
122                    WHEN a2.succeeded = 1     THEN 'TRUE'
123                 END
```

```
124             ELSE 'NO_CAUSALITY'
125         END
126     ELSE NULL
127 END as causality_succeeded,
128
129 CASE
130     WHEN flow_alert.alertid IS NOT NULL THEN
131         CASE
132             WHEN c.parent IS NOT NULL THEN
                    CONCAT('\'',a2.description,'\'')
133             ELSE 'NO_CAUSALITY'
134         END
135     ELSE NULL
136 END as causality_description,
137
138 CASE
139     WHEN flow_alert.alertid IS NOT NULL THEN
140         CASE
141             WHEN c.parent IS NOT NULL THEN  t2.description
142             ELSE 'NO_CAUSALITY'
143         END
144     ELSE NULL
145 END as causality_type,
146
147     /*
148     Campo acrescido que indica se o fluxo foi associado a
            um 'cluster'.
149     */
150
151  CASE
152     WHEN flow_alert.alertid IS NOT NULL THEN
153             CASE
154                 WHEN ac.parent IS NULL THEN 'FALSE'
155                 ELSE 'TRUE'
156             END
157     ELSE NULL
158 END as clustered,
159
160     /*
```

```
161         Os proximos campos sao relativos a tabela 'alerts',
              indicando o 'cluster' a que foi associado o fluxo
              atual.
162         */
163
164       CASE
165        WHEN flow_alert.alertid IS NOT NULL THEN
166           CASE
167              WHEN ac.parent IS NOT NULL THEN
168                 CASE
169                    WHEN a3.automated IS NULL THEN '?'
170                    WHEN a3.automated = 0     THEN 'FALSE'
171                    WHEN a3.automated = 1     THEN 'TRUE'
172                 END
173              ELSE 'NO_CLUSTER'
174           END
175        ELSE NULL
176       END as cluster_automated,
177
178
179      CASE
180         WHEN flow_alert.alertid IS NOT NULL THEN
181           CASE
182              WHEN ac.parent IS NOT NULL THEN
183                 CASE
184                    WHEN a3.succeeded IS NULL THEN '?'
185                    WHEN a3.succeeded = 0     THEN 'FALSE'
186                    WHEN a3.succeeded = 1     THEN 'TRUE'
187                 END
188              ELSE 'NO_CLUSTER'
189           END
190        ELSE NULL
191       END as cluster_succeeded,
192
193
194       CASE
195          WHEN flow_alert.alertid IS NOT NULL THEN
196             CASE
```

```
197             WHEN ac.parent IS NOT NULL THEN
                    CONCAT('\'',a3.description,'\'')
198             ELSE 'NO_CLUSTER'
199         END
200     ELSE NULL
201 END as cluster_description,
202
203 CASE
204     WHEN flow_alert.alertid IS NOT NULL THEN
205         CASE
206             WHEN ac.parent IS NOT NULL THEN  t3.description
207             ELSE 'NO_CLUSTER'
208         END
209     ELSE NULL
210 END as cluster_type
211
212     /*
213     Join das tabelas que compoem o banco de dados.
214     */
215
216 FROM flows as f                                 LEFT JOIN
217  flow_alert  ON f.id = flow_alert.flowid        LEFT JOIN
218  alerts as a ON flow_alert.alertid = a.id       LEFT JOIN
219  alert_type as t ON a.type = t.id               LEFT JOIN
220
221  alert_causality as c ON c.child = a.id         LEFT JOIN
222  alerts as a2 ON a2.id = c.parent               LEFT JOIN
223  alert_type as t2 ON t2.id = a2.type            LEFT JOIN
224
225  alert_cluster as ac ON ac.child = a.id         LEFT JOIN
226  alerts as a3 ON a3.id = ac.parent              LEFT JOIN
227  alert_type as t3 ON t3.id = a3.type
228
229     /*
230     Instrucoes para que o resultado da query seja salvo em
          um arquivo de extensao .cvs, onde os campos sao
          separados por virgulas.
231     Este formato adequa-se ao formato .arff para
          utilizacao com a plataforma WEKA, sendo necessario
```

```
            apenas o acrescimo do cabecalho com informacos
            referentes aos tipos dos campos.
232         */
233
234     INTO OUTFILE "C:/output.csv"
235     FIELDS TERMINATED BY ','
236     LINES TERMINATED BY "\n" ;
```

APÊNDICE D – CABEÇALHOS

Apresentamos aqui os cabeçalhos desenvolvidos neste trabalho para a utilização das bases de dados desenvolvida por Sperotto et al., descrita em (SPEROTTO et al., 2009), e KDDCup99 na plataforma WEKA, necessários ao formato ARFF.

D.0.6 Cabeçalho da base Sperotto et al.

```
1  @relation flows
2
3  @attribute src_ip        numeric
4  @attribute dst_ip        numeric
5  @attribute src_port      numeric
6  @attribute dst_port      numeric
7  @attribute packets       numeric
8  @attribute octets        numeric
9  @attribute start_time    numeric
10 @attribute start_msec    numeric
11 @attribute end_time      numeric
12 @attribute end_msec      numeric
13 @attribute duration      numeric
14 @attribute tcp_flags     numeric
15 @attribute protocol      {ICMP, TCP, UDP}
16 @attribute malicious     {FALSE, TRUE}
17 @attribute automated     {FALSE, TRUE, \N}
18 @attribute succeeded     {FALSE, TRUE, \N}
19 @attribute alert_description string
20 @attribute alert_type {ssh_conn, ftp_conn, http_conn,
      authident_sideeffect, irc_sideeffect, icmp_sideeffect, \N}
21 @attribute causality {FALSE, TRUE, \N}
22 @attribute causality_automated {FALSE, TRUE, \N , NO_CAUSALITY}
23 @attribute causality_succeeded {FALSE, TRUE, \N , NO_CAUSALITY}
24 @attribute causality_description string
25 @attribute causality_type {ssh_scan, ftp_scan, http_scan, \N,
     NO_CAUSALITY}
26 @attribute clustered     {FALSE, TRUE, \N}
27 @attribute cluster_automated {FALSE, TRUE, \N , NO_CLUSTER}
28 @attribute cluster_succeeded {FALSE, TRUE, \N , NO_CLUSTER}
```

```
29 @attribute cluster_description string
30 @attribute cluster_type {ssh_scan, ftp_scan, http_scan, \N,
     NO_CLUSTER}
```

D.0.7 Cabeçalho KDDCup99

```
1  @relation kddcup
2
3  @attribute duration      numeric
4  @attribute protocol_type {udp,tcp,icmp}
5  @attribute service {aol, auth, bgp, courier, csnet_ns, ctf,
     daytime, discard, domain, domain_u, echo, eco_i, ecr_i,
     efs, exec, finger, ftp, ftp_data, gopher, harvest,
     hostnames, http, http_2784, http_443, http_8001, imap4,
     IRC, iso_tsap, klogin, kshell, ldap, link, login, mtp,
     name, netbios_dgm, netbios_ns, netbios_ssn, netstat, nnsp,
     nntp, ntp_u, other, pm_dump, pop_2, pop_3, printer,
     private, red_i, remote_job, rje, shell, smtp, sql_net, ssh,
     sunrpc, supdup, systat, telnet, tftp_u, tim_i, time, urh_i,
     urp_i, uucp, uucp_path, vmnet, whois, X11, Z39_50}
6  @attribute flag { OTH, REJ, RSTO, RSTOS0, RSTR, S0, S1, S2,
     S3, SF, SH }
7  @attribute src_bytes     numeric
8  @attribute dst_bytes     numeric
9  @attribute land          {0,1}
10 @attribute wrong_fragment numeric
11 @attribute urgent        numeric
12 @attribute hot           numeric
13 @attribute num_failed_logins numeric
14 @attribute logged_in     {0,1}
15 @attribute num_compromised numeric
16 @attribute root_shell    numeric
17 @attribute su_attempted numeric
18 @attribute num_root      numeric
19 @attribute num_file_creations numeric
20 @attribute num_shells    numeric
21 @attribute num_access_files numeric
22 @attribute num_outbound_cmds numeric
23 @attribute is_host_login {0,1}
24 @attribute is_guest_login {0,1}
25 @attribute count         numeric
26 @attribute srv_count     numeric
27 @attribute serror_rate   numeric
```

```
28 @attribute srv_serror_rate numeric
29 @attribute rerror_rate  numeric
30 @attribute srv_rerror_rate numeric
31 @attribute same_srv_rate numeric
32 @attribute diff_srv_rate numeric
33 @attribute srv_diff_host_rate numeric
34 @attribute dst_host_count numeric
35 @attribute dst_host_srv_count numeric
36 @attribute dst_host_same_srv_rate numeric
37 @attribute dst_host_diff_srv_rate numeric
38 @attribute dst_host_same_src_port_rate numeric
39 @attribute dst_host_srv_diff_host_rate numeric
40 @attribute dst_host_serror_rate numeric
41 @attribute dst_host_srv_serror_rate numeric
42 @attribute dst_host_rerror_rate numeric
43 @attribute dst_host_srv_rerror_rate numeric
44 @attribute classification {back, buffer_overflow, ftp_write,
   guess_passwd, imap, ipsweep, land, loadmodule, multihop,
   neptune, nmap, normal, perl, phf, pod, portsweep, rootkit,
   satan, smurf, spy, teardrop, warezclient, warezmaster}
```