UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS

FACULDADE DE COMPUTAÇÃO

ELIEL DOS SANTOS BENTES

COMPARAÇÃO DE MODELOS PREDITIVOS PARA DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES

BELÉM
2018

ELIEL DOS SANTOS BENTES

COMPARAÇÃO DE MODELOS PREDITIVOS PARA DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES

Trabalho de Conclusão de Curso apresentado para obtenção do grau de Bacharel em Sistemas de Informação. Faculdade de Computação. Instituto de Ciências Exatas e Naturais. Universidade Federal do Pará.

Orientador: Prof. Dr. Lídio Mauro de Lima

BELÉM
2018

ELIEL DOS SANTOS BENTES

COMPARAÇÃO DE MODELOS PREDITIVOS PARA DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES

Trabalho de Conclusão de Curso orientado pelo Profº. Dr. Lídio Mauro de Lima, apresentado ao Curso de Bacharelado em Sistemas de Informação do Instituto de Ciências Exatas e Naturais, como requisito para obtenção do grau de bacharel em Sistemas de Informação.

APROVADO EM: ____/____/_____/

BANCA EXAMINADORA:

_____

Prof. Dr. Lídio Mauro Lima de Campos
Orientador – UFPA

_____

Prof. Dr. Jefferson Magalhães de Moraes
Examinador Interno – UFPA

_____

Prof. Dr. Filipe de Oliveira Saraiva
Examinador Interno – UFPA

AGRADECIMENTOS

Agradeço a Deus, pelo dom da vida e dar as forças necessárias para enfrentar os desafios desta vida.

Agradeço ao meu orientador Prof. Dr. Lídio Mauro de Campos, pela paciência, pela atenção, pelo apoio, pela disponibilidade.

Agradeço imensamente sua preocupação e incentivo mesmo diante das dificuldades encontradas por mim neste período de elaboração deste trabalho.

Agradeço também a professora e ex coordenadora da FACOMP Marcelle Mota, ao atual coordenador Josivaldo Araújo e aos professores do curso de sistemas de informação, pela compreensão para com as necessidades dos alunos do período noturno.

Agradeço a meus pais que me incentivavam a sempre estudar quando pequeno.

RESUMO

Dentre as tarefas de mineração de dados a classificação é a que mais se destaca quando se deseja descobrir um modelo de conhecimento em banco de dados. Porém, para se construir um modelo eficiente é imprescindível que a classificação de dados seja realizada com dados mais próximos possíveis da realidade e com algoritmos que possuam ótimo desempenho. Nesse contexto, muitos estudos são feitos utilizando as técnicas de mineração de dados como ferramenta fundamental em um Sistema de Detecção de Intrusão. Esses sistemas são essenciais para complementar a segurança de um ambiente de redes de computadores, onde a análise do tráfego da rede precisa ser feita com rapidez e precisão a fim de impedir um acesso indesejado. Nesse trabalho de conclusão de curso realizaram-se simulações em um ambiente de detecção de intrusão onde foram analisados os desempenhos de dois algoritmos de aprendizado de máquina, o Árvore de Decisão e o *Naive Bayes* na tarefa de classificação de conexões normais ou anormais utilizando o *dataset KDDCUP'99*. A classificação do conjunto de dados foi realizada em duas etapas, na primeira com apenas duas classes de conexão (normal e anormal) utilizando a técnica de validação cruzada com valor de k menor ou igual a 10. Na segunda com cinco classes de detecção (quatro para ataques e uma normal) usando o valor de k maior ou igual a 10. As medidas de desempenho utilizadas para os algoritmos foram: taxa de acertos, taxa de erros e o tempo para construção do modelo. Os experimentos foram realizados utilizando o ambiente WEKA.

Palavras-chave: Mineração de Dados. Sistemas de detecção de intrusão. Classificação

ABSTRACT

Among the tasks of data mining, classification is the one that stands out most when one wishes to discover a knowledge model in a database. However, in order to construct an efficient model, it is essential that the data classification be performed with data that is as close as possible to reality and with algorithms that perform optimally. In this context, many studies are done using data mining techniques as a fundamental tool in an Intrusion Detection System. These systems are essential to complement the security of a computer network environment, where network traffic analysis needs to be done quickly and accurately to prevent unwanted access. In this work, simulations were performed in an intrusion detection environment where the performance of two machine learning algorithms, the Decision Tree and the Naive Bayes, were analyzed in the task of classifying normal or abnormal connections using the dataset KDDCUP'99. The classification of the data set was performed in two stages, in the first one with only two connection classes (normal and abnormal) using the cross-validation technique with a value of k less than or equal to 10. In the second one with five detection classes (four for attacks and a normal) using the value of k greater than or equal to 10. The performance measures used for the algorithms were: hit rate, error rate and time for model construction. The experiments were performed using the WEKA environment.

Keywords: Data Mining. Intrusion Detection Systems. Classification.

LISTA DE GRÁFICOS

LISTA DE FIGURAS

LISTA DE ILUSTRAÇÕES - QUADRO

LISTA TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Atualmente observa-se que diversas organizações enfrentam o desafio de saber lidar com o grande volume de dados gerados por suas transações. Nesse contexto, vem ganhando destaque o termo *Big Data*, em tradução literal significa "Grandes Dados", uma forma direta de se referir ao grande volume de dados gerados. Amaral (2016) afirma que *Big Data* é o fenômeno em que dados são produzidos em vários formatos e armazenados por grande quantidade de dispositivos e equipamentos, explica ainda que o barateamento, miniaturização e aumento da capacidade de processamento levam a disseminação de equipamentos, dispositivos e processos capazes de produzir e armazenar dados. Ou seja, o *Big Data* é consequência quase natural da atual revolução tecnológica em que vivemos.

Porém os dados, brutos em si, possuem pouco ou quase nenhum valor para uma organização. O valor dos dados armazenados está tipicamente relacionado à capacidade de se extrair conhecimento de mais alto nível a partir deles, ou seja, informação útil que sirva para apoio à tomada de decisão, para exploração e melhor entendimento do fenômeno gerador dos dados (GOLDSCHMIDT, PASSOS e BEZERRA, 2015). Assim, os dados precisam ser analisados e trabalhados a fim de obter alguma informação deles. Esse processo já não é tão atual quanto o termo *Big Data*, na verdade o processo de extrair informações úteis a partir de dados estruturados já era bem difundido nos anos 90 sob a nomenclatura de KDD (em inglês: *Knowledge Discovery in Database*), que tem a mineração de dados como sua etapa mais significativa, e por isso muitas vezes sendo usada para se referir ao processo todo.

Diante disso, nota-se tanto para fins comerciais, como para fins de pesquisas acadêmicas, um interesse crescente em torno da análise de grandes conjuntos de dados para obtenção de conhecimento. Porém, para realizar esta extração de conhecimento é necessária uma base de dados com registros reais sobre a área de interesse que se pretende analisar. Apesar de existirem diversas base de dados disponíveis para fazer simulações de mineração de dados, escolheu-se a base do *dataset KDDCUP'99*, por ser uma base bastante conhecida e também possuir em seus registros algumas instâncias que se referem a termos utilizados na área de computação, o que torna mais acessível o entendimento do conteúdo dos registros e também algumas e fazer algumas alterações, caso seja necessário. Algo que não seria tão simples de fazer se considerar uma base de dados que possuísse registros de doenças, ou instâncias que se

referem ao tratamento do câncer, isto por que normalmente na tarefa de mineração de dados, o analista necessita ter conhecimento sobre o conteúdo dos dados que está manipulando.

Outro desafio enfrentado pelas organizações é como manter a segurança dos seus dados corporativos. Devido ao aumento de dispositivos e usuários conectados aos sistemas em que as organizações realizam suas atividades, muitas delas ficam expostas a ameaças, como ataques à rede ou aos sistemas operacionais, expostas também a acesso indevido às suas informações e também a abuso de privilégios cometidos por usuários internos.

Para garantir a segurança no ambiente de rede de computadores utilizam-se técnicas como criptografia de dados, *firewalls*, mecanismos de controle de acesso e sistemas de detecção de intrusão. Esta última técnica tem sido alvo de várias pesquisas que visam melhorar o seu desempenho na identificação de eventos suspeitos e aumentar sua eficiência em distinguir um comportamento anormal em um ambiente de rede.

Sistemas de detecção de intrusão ou simplesmente *IDS* (em inglês: Intrusion Detection System), são sistemas capazes de analisar um tráfego de uma rede ou uma atividade local de um computador e decidir se o evento constitui em um ataque ou se é uma operação rotineira (LIMA, 2012). Para analisar o tráfego e decidir sobre o comportamento do acesso à rede, o sistema de detecção deve ser capaz de além de analisar os dados, ter um tempo de resposta quase em tempo real. Dessa forma, muitos IDS utilizam as técnicas de aprendizado de máquina para ter um ótimo desempenho.

Dessa forma, diante do crescente interesse na área de mineração de dados procura-se aqui demonstrar um estudo de caso da utilização das técnicas de Extração de Conhecimento em Bancos de Dados, a partir da análise de dados reais de conexões de redes de computadores. Além disso, recomendar dentre os algoritmos apresentados, o que possui melhor desempenho em detectar intrusão em uma rede de computadores.

Muitas pesquisas têm surgido nesse contexto, sendo que uma das preocupações mais presentes nelas é a de como reduzir a taxa de falso positivo (FERREIRA, 2016). Ou qual técnica um IDS deve utilizar na análise de classificação de intrusão (MOLL, 2010). Portanto, a partir da utilização de recursos computacionais mais acessíveis, utilizando dados reais de conexão de rede, faz-se neste trabalho uma simulação de um ambiente de detecção de intrusão, a fim de verificar qual algoritmo de classificação seria mais recomendado para detecção de intrusão visando o melhor desempenho do modelo: Árvore de Decisão ou *Naive Bayes*.

1.2 JUSTIFICATIVA

O cenário atual indica que a vasta quantidade de aplicações financeiras disponíveis na *Internet* e as vulnerabilidades de sistemas e protocolos possibilitam um cenário a cada dia mais propenso as tentativas de intrusões (LIMA, 2012). Também, pode acrescentar um número crescente de dispositivos conectados, e consequentemente um número cada vez maior de usuários conectados à internet. Dessa forma, uma preocupação geral para analistas de sistemas ou empresas de segurança é a de como manter suas redes protegidas de ataques.

Logo, sistemas vulneráveis podem ser atacados a qualquer momento, tornando imperativa a necessidade de reconhecer e denunciar intrusões, a qual deve ocorrer o mais cedo possível, preferencialmente em tempo real (LIMA, 2012). Para detectar intrusões em um sistema computacional, usa-se uma ferramenta denominada IDS (em inglês: *Introduction Detection System*) ou Sistema de detecção de intrusão (SDI) o qual pode utilizar diversas técnicas de aprendizado de máquina para classificar uma conexão de rede como normal ou intrusão.

É desejável que um SDI tenha uma taxa de acertos alta para detecção e uma baixa taxa de falso positivo. Para alcançar estas metas, o sistema de detecção pode utilizar-se de um algoritmo de classificação que apresente um bom desempenho em distinguir as classes de intrusão (ataque) da classe normal. E para melhor resultados desta classificação é recomendado que se utilizasse uma base de dados mais próxima possível da realidade de um ambiente de rede tradicional.

Portanto, o IDS é uma técnica bastante recomendada para segurança de redes de computadores, e seu sucesso depende da qualidade de sua detecção. Além do mais, para que o mesmo seja eficiente é imprescindível que realize uma detecção rápida e bem precisa, com baixa taxa de falso positivo, independente dos recursos computacionais disponíveis. Dessa forma a escolha de um bom algoritmo de classificação pode determinar excelentes resultados na detecção de intrusão, assim como, o sucesso na proteção de uma rede de computadores.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Utilizar técnicas de mineração de dados para detectar intrusões em redes de computadores.

1.3.2 Objetivos Específicos

- Simular um ambiente de detecção de intrusão utilizando recursos computacionais mais acessíveis junto com o *dataset KDDCUP'99.*
- Descrever o *dataset KDDCUP'99* e gerar novo conjunto de dados dividido por categorias de ataques (*DOS, R2L, U2L* e *Probing*) para melhor realização dos experimentos.
- Realizar simulações com o *dataset KDDCUP'99* utilizando o software *WEKA*, visando analisar a desempenho, em termos de classificação, de dois algoritmos clássicos de mineração de dados (Árvore de Decisão – J.48 e *Naive Bayes*).
- Analisar dentre os dois algoritmos (Árvore de Decisão – J.48 e *Naive Bayes*) utilizados em mineração de dados, qual o mais eficiente para detectar intrusões ou conexões normais.
- Analisar o efeito da variação do parâmetro k, da técnica de validação cruzada, na acurácia dos dois modelos estudados.

1.4 METODOLOGIA

Este estudo teve por finalidade realizar uma simulação de um ambiente de detecção de intrusão em redes de computadores a partir de dados reais de conexão e com a utilização de dois algoritmos de classificação mais conhecidos na mineração de dados.

Devido à falta de conjunto de dados públicos disponíveis, para simular um IDS, escolheu-se para realização dos experimentos de detecção de intrusão, a base de dados do *dataset* KDDCUP'99. Apesar de não ser uma base recente trata-se de uma base de dados reais de conexões de rede bastante utilizada em outros estudos para simular o funcionamento de IDS, e acredita-se que seja um conjunto de dados de referência bastante eficaz para realizar tais experimentos.

Além de apresentar dados reais de conexão de redes de computadores, a base de dados utilizada, disponibiliza um subconjunto da base principal com 10% do total de seus registros, o que possibilita realizar experimentos com baixo custo computacional e, também, para testar um método de classificação que possa classificar adequadamente intrusões (ataques) e conexões normais.

Através do software *WEKA* foram testados dois algoritmos de classificação mais conhecidos nos estudos que envolvem mineração de dados, o de Árvore de Decisão e o *Naive Bayes* . Estes algoritmos são bastante utilizados quando se pretende detectar uma classe distinta dentre um conjunto grande de dados. Ambos os algoritmos podem ser implementados com a técnica de validação cruzada ou como é reconhecida na literatura: *k-fold cross-validation*. A partir da utilização desta técnica procura-se observar se a alteração no valor da constante *k* influenciará nos resultados obtidos para as taxas de acertos e taxa de erros.

A fim de garantir resultados bastante satisfatórios sobre a classificação feita com os dados, o conjunto de dados original foi alterado em dois momentos. No primeiro a configuração dos registros foi feita para classificar duas classes: normal e anormal (intrusão). No segundo, a configuração dos registros foi feita em quatro classes de ataque e uma classe normal.

Além disso, foram utilizadas algumas métricas para avaliar a precisão dos algoritmos classificadores: a taxa de acurácia (**Tacurácia),** a Taxa de erros por falsos positivos para a classe "normal" (**TEnormal**), a Taxa de erros por falsos positivos para a classe "anormal", (**TEanormal**) e a Taxa de erros total (**TEtotal1**). Para cinco classes foram utilizadas as seguintes métricas: Taxa de acurácia (**Tacurácia2**) que indica a proporção de acertos ocorridos na classificação, a Taxa de erro total (**TEtotal2**) que indica a proporção total de erros ocorridos na classificação, a taxa de erros por falsos positivos para a classe "*DOS*" (**TEdos**), que representa a proporção de erros ocorridos na classificação da classe "*DOS*", a taxa de erros por falsos positivos para a classe "*R2L*" (**TEr2l**), que representa a proporção de erros ocorridos na classificação da classe "*R2L*", a Taxa de erros por falsos positivos para a classe "*U2R*" (**TEu2r**), que representa a proporção de erros ocorridos na classificação da classe "*U2R*", a Taxa de erros por falsos positivos para a classe "*probing*" (**TEprobing**), que representa a proporção de erros ocorridos na classificação da classe "*probing*".

Também, compararam-se os valores de acurácia de cada classificador, para duas e cinco classes, de acordo com o valor de **k** utilizados em cada experimento. Ao final compararam-se os resultados obtidos pelos dos classificadores Árvore de Decisão e *Naive Bayes*, indicando-se o que obteve o melhor desempenho.

1.5 APRESENTAÇÃO DO TRABALHO

O Capítulo 2 apresenta conceitos, fundamentais, de aprendizagem de máquina e as suas aplicações, processo de descoberta de conhecimento em banco de dados, mineração de dados e sobre a ferramenta *WEKA*.

O Capítulo 3 detalha os materiais, métodos e as métricas utilizadas nessa pesquisa.

O Capítulo 4, os experimentos e resultados são detalhados e finalmente, no Capítulo 5, as principais conclusões são descritas.

2 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo são apresentados alguns conceitos básicos que serão utilizados no decorrer desse trabalho e que são essenciais para entender a técnica de mineração de dados, a qual é etapa principal do processo de descoberta de conhecimento em banco de dados. Além disso, os princípios que serviram de base para este estudo, também tem por objetivo descrever as etapas e técnicas computacionais que contribuem para a descoberta de conhecimento em banco de dados. Revisam-se, rapidamente, conceitos como Aprendizado de Máquina e a sua importância para mineração de dados. Assim como, também, apresenta-se uma breve introdução sobre os sistemas de detecção de intrusão ou IDS (em inglês: *Intrusion Detection System*). Por se tratar de simulação de um sistema genérico de detecção de intrusão em redes de computadores, alguns conceitos forma apresentados de forma sucinta, contextualizados ao escopo do trabalho.

2.1 INTRODUÇÃO

A mineração de dados é uma técnica que vem ganhando muito destaque nos últimos anos, especialmente devido ao fenômeno *big data*, e o surgimento da internet das coisas (IoT, em inglês: *Internet of Things*), marketing digital e em muitas outras áreas tradicionais da sociedade humana onde se faz presente o uso e armazenamento de dados. Outra questão, também, muito importante é a da segurança dos dados gerados, que envolve principalmente tomar conhecimento do tipo de acesso que os sistemas de gestão e armazenamento de dados estão tendo.

Mineração de dados é uma etapa do processo chamado de KDD (em inglês: *Knowledge Discovery in Database*), ou Descoberta de Conhecimento em Banco de Dados. Como o nome sugere trata-se do processo de obter informações a partir de dados brutos, e dessa forma construir conhecimento útil para tomada de decisões (ELMASRI & NAVATHE, 2011). Este trabalho pretende aplicar as técnicas de mineração de dados para extrair conhecimento a partir de um conjunto de dados de conexões de redes para que um sistema de detecção seja capaz de detectar um acesso anormal ao tráfego de rede.

Além disso, considerando os Sistemas de Detecção de Intrusão (SDI) como uma técnica de segurança que vem atraindo cada vez mais pesquisadores, e sendo, cada vez mais, difundida na área de segurança da informação, principalmente pela sua proximidade com a área

de aprendizado de máquina e de suas técnicas. Dessa forma, a mineração de dados constitui uma importante ferramenta para o desenvolvimento de analisadores de tráfego de um IDS, o qual permite, por meio da aplicação de técnicas e algoritmos do aprendizado de máquina, distinguir dentre os acessos ditos normais à rede, aqueles que apresentam algum tipo de anomalia.

2.2-APRENDIZADO DE MÁQUINA

Aprendizado de máquina é uma "área de pesquisa que visa desenvolver programas computacionais capazes de automaticamente melhorar seu desempenho por meio da experiência." (CASTRO e FERRARI, 2016, p. 14 apud TOM MITCHELL, 1997).

Ainda Castro e Ferrari (2016, apud ALPAYDIN, 2009), definem aprendizado de máquina como a "programação de computadores para aperfeiçoar um critério de desempenho usando experiências passadas, chamadas de exemplo ou simplesmente dados de entrada."

Aprendizado de máquina é "uma subárea da inteligência artificial que busca produzir algoritmos capazes de fazer com que computadores aprendam e não somente executem algoritmos." (AMARAL, 2016, pag. 214).

Como o nome sugere trata se da capacidade que o computador tem de aprender por si só a realizar determinada tarefa. Ou seja, uma forma de aprendizado automático. Logo percebe se que o objetivo primordial da área de Aprendizado de Máquina é desenvolver técnicas capazes de aprender a resolver problemas sem que haja uma resposta já programada.

Dentro do processo de KDD e para a etapa de mineração de dados, o Aprendizado de Máquina Computacional é a aplicação de técnicas computacionais na tentativa de encontrar padrões ocultos em dados. Entende se por padrões ocultos aqueles que não podem ser percebidos de forma eficaz pela maneira superficial que se limita a análise humana, necessitando dessa forma de técnicas computacionais para melhor análise destes dados, e dessa forma, melhores precisão na taxa de acertos e erros na criação de um modelo, ou seja, o aprendizado de máquina permite extrair informações ocultas em dados de maneira automática e com o melhor desempenho possível.

Para Amaral (2016) o aprendizado de máquina "pode ser dividido em três grandes grupos de tarefas: classificação, agrupamento e associação. A tarefa mais comum é a classificação. Na classificação, os dados devem possuir uma classe a qual queremos prever."

Convencionou se dividir as tarefas inerentes ao aprendizado de máquina em tarefas supervisionadas e não supervisionadas. De acordo com Amaral (2016) pode-se resumir:

> São tarefas supervisionadas em que existem uma classe, ou atributo ao qual se quer descrever ou prever. Classificação é então uma técnica supervisionada. Nas técnicas não supervisionadas não existe uma classe. Exemplos de técnicas não supervisionadas são as de agrupamento e regras de associação. (AMARAL, 2016, p. 87).

Também no aprendizado do tipo supervisionado para que se alcance um modelo de classe, o algoritmo escolhido trabalha com exemplos de entradas e saídas esperadas. Nesse trabalho utilizaremos a técnica supervisionada da classificação.

Dentre os algoritmos mais difundidos em mineração de dados destacamos o de Árvore de Decisão e *Naive Bayes*, pois são referenciados na literatura de classificação de dados como bons classificadores para grandes conjuntos de dados e também por necessitarem de baixo custo computacional para serem implementados. Portanto neste trabalho optou-se por fazer as simulações com estes dois algoritmos.

2.3- DESCOBERTA DE CONHECIMENTO EM BANCOS DE DADOS

> Mineração de dados, ou "*data mining*", é termo utilizado para nomear o processo de análise de conjuntos de dados com o objetivo de encontrar padrões que representem informações úteis e não triviais. Para tanto, utiliza-se de métodos matemáticos, heurísticas e algoritmos. A mineração de dados é parte de um processo maior e mais abrangente, o de descoberta de conhecimento em bancos de dados, que tem por objetivo extrair conhecimento de alto nível a partir de dados de baixo nível no contexto de grandes conjuntos de dados (FAYYAD, 1996).

Descoberta de conhecimento em bancos de dados, ou "*Knowledge Discovery in Databases*", é o termo, criado em 1989, que se refere ao amplo processo de descobrir conhecimento em dados armazenados em bancos de dados. Tal processo objetiva extrair conhecimento implícito e previamente desconhecido, buscando informação potencialmente útil nos dados. O processo, descrito em (FAYYAD, 1996), consiste em uma sequência de cinco etapas, partindo dos dados existentes e chegando à descoberta do conhecimento extraído dos mesmos.

Seleção dos dados: a primeira etapa consiste em escolher qual o conjunto de dados que será submetido ao processo. Seleciona-se um conjunto de dados alvo, ou foca-se em um subconjunto de variáveis ou amostras de dados.

Pré-processamento: nesta etapa, os dados podem sofrer uma qualificação, a fim de corrigir erros e inconsistências que poderão existir. Incluem-se limpeza de dados, eliminação de dados ruidosos, falta de dados e normalização.

Transformação: aqui os dados são convertidos em um formato adequado para serem acessados pelos algoritmos de mineração. É nela que também se realiza uma possível redução no número de variáveis, sumarizando os dados que serão submetidos à mineração.

Mineração: é a etapa mais importante do processo. É nela que o algoritmo escolhido é aplicado sobre os dados a fim de se descobrir padrões interessantes. É fundamental para que esta etapa obtenha resultados de qualidade a correta aplicação dos passos anteriores.

Interpretação dos dados e Visualização: nesta última etapa do processo, os resultados obtidos na mineração são interpretados.

As técnicas de mineração de dados podem ser aplicadas a tarefas como classificação, estimativa, associação, segmentação e sumarização. Essas tarefas são descritas na Tabela 1.

Tabela 1 - Tarefas em Mineração de dados

Tarefa	Descrição	Exemplos
Classificação	Consiste em construir um modelo de algum tipo que possa ser aplicado a dados não classificados visando categorizá-los em classes. Um objeto é examinado e classificado de acordo com uma classe definida.	-classificar solicitações de pedidos de crédito. -esclarecer fraudes na declaração do imposto de renda.
Regressão	Regressão é aprender uma função que mapeia um item de dado para uma variável de predição real estimada.	-prever a demanda futura de um novo produto. -estimar expectativa de vida média dos brasileiros.
Associação	Identificação de grupos de dados que apresentem concorrência entre si.	-quais produtos são colocados juntos em carrinhos de supermercado.
Segmentação (ou Clustering)	Processo de partição de uma população heterogênea em	-agrupamento de clientes com comportamento de compras similar.

Tarefa	Descrição	Exemplos
	vários subgrupos ou grupos mais homogêneos.	-comportamento de clientes em compras realizadas na web para uso futuro.
Detecção de desvios (outliers)	Identificação de dados que deveriam seguir um padrão esperado, mas não o fazem.	-detecção de intrusão em redes de computadores.

Fonte: Adaptado de GOLDSCHMIDT, PASSOS e BEZERRA (2005)

2.4 - CLASSIFICAÇÃO DE DADOS

Como mencionou-se na subseção 2.3, um dos grupos de tarefas que compõe o processo de KDD é a classificação. Amaral (2016), descreve a tarefa de classificação como uma técnica supervisionada do campo de aprendizado de máquina, cujo objetivo é definir uma classe 'modelo' que será utilizada para classificar dados novos conforme esse modelo.

Elmasri e Navathe (2011) descrevem essa tarefa como a tarefa que irá analisar os dados já registrados e gerar um modelo para predizer qual a saída será gerada para os novos dados, e para tal será aplicado um dos algoritmos de aprendizado de máquina. Porém é importante ressaltar que a classificação é indicada para aplicar sobre dados nominais

Na classificação temos um atributo especial denominado classe. As técnicas e algoritmos de aprendizado de máquina são então utilizados sobre dados históricos, fatos ocorridos, ou seja, dados já classificados, para prever esta classe (AMARAL, 2016).

Os dados históricos são os registros que compõem a base de dados, estes registros são formados pelas instâncias, cada instância traz o conjunto de valores para cada atributo que compõe a base de dados, onde dentre esses valores haverá um que corresponde a um atributo especial que determinará a classe, geralmente, em um arquivo ARFF (em inglês: *Atribute-Relation File Format*), o último campo de atributo determina a que classe ele pertence.

Para determinar uma classe deve-se particionar o conjunto de dados em dois subconjuntos: os dados de treinamento e os dados de teste. Os dados de treinamento são para aprendizado do modelo, pois estes dados já estão rotulados como sendo de uma classe específica (CASTRO e FERRARI, 2016).

No treinamento o classificador vai fazer a distinção entre os valores de atributos que pertencem a cada classe, ou seja, ele vai aprender que aqueles atributos específicos são de uma determinada classe e não de outra, nessa etapa o classificador cria um modelo de classe.

Ainda dentro da tarefa de classificação os dados separados para teste serão usados para testar esse modelo.

2.4.1 – Partição do conjunto de dados e criação do modelo

Para a criação de um modelo o primeiro passo é a partição do conjunto de dados, dividindo o grupo de dados em um grupo de treinamento e um grupo de teste. O primeiro é utilizado para gerar um modelo e o segundo é utilizado para testar o desempenho desse modelo gerado. Essa partição dos dados é de suma importância para a criação do modelo gerado.

Conforme explicado na subseção anterior esse modelo é gerado a partir dos dados de treinamento e será testado com os dados de teste, porém estes dados de teste, apesar de fazer parte da mesma base de dados, serão dados novos ao modelo, com registros inéditos ao classificador, a fim de verificar seu desempenho.

A divisão entre dados de treino e dados de teste pode ser do tipo *hold out* onde os dados são divididos aleatoriamente sem distribuição em 70% para treino e 30% para teste, outro tipo de divisão também muito usada é validação cruzada (AMARAL, 2016). Porém a técnica de validação cruzada é considerada em muitos estudos como sendo mais eficiente devido ao grande número de interações que realiza com as partições dos dados.

Validação cruzada é a técnica em cada registro é usado o mesmo número de vezes para treino e para teste, são trocados. Repete se o processo um determinado número de vezes. O número de interações e conhecido como *folds*, ou partições. (AMARAL, 2016). A técnica de validação cruzada será descrita em mais detalhes no capítulo 3 (Material e Métodos) no tópico que trata da partição do *dataset KDDCUP'99.*

Depois de gerado um modelo de classe, os dados de teste são usados para testar este modelo gerado, onde toda vez que uma nova instancia for submetida a este modelo, o classificador determinará, conforme os valores de seus atributos, que ela pertence a uma determinada classe. Ou seja, uma instância que contenha um conjunto específico de valores dos atributos que correspondem aos dados do modelo será, então, classificada como sendo uma instância pertencente a classe predita.

Normalmente na tarefa de classificação, após treinamento e teste do modelo, os resultados são representados numa tabela de forma a analisar a quantidade de instâncias classificadas corretamente e quantidade de instâncias classificadas incorretamente, esta tabela

é chamada de Matriz de Confusão. Essa matriz é gerada automaticamente no software *WEKA*, após a tarefa de classificação, e a partir dela é possível calcular as métricas que medem o desempenho do algoritmo classificador.

2.5-ALGORITMOS DE CLASSIFICAÇÃO UTILIZADOS

2.5.1-Árvores de Decisão

A árvore de decisão é um dos tipos mais conhecidos dentre os classificadores, e é chamada dessa forma, pois o modelo criado é semelhante a uma árvore com nós.

Castro e Ferrari (2016) definem como uma estrutura em forma de arvore na qual cada nó interno corresponde a um teste de um atributo, cada ramo representa um resultado do teste e os nós folhas representam classes ou distribuições de classes.

Hosokawa (2011), resume esta estrutura como se fosse uma estrutura que, através de testes de decisão divide sucessivamente uma grande coleção de registros em conjuntos menores. Dessa forma, tem-se que os nós internos representam os testes feitos sobre o atributo e cada teste gera um valor que corresponde a uma ramificação da árvore levando a outro nó interno, ou a um nó folha. Os nós folhas são onde a classificação é definida, sendo muitas vezes chamados de nós terminais.

Duas importantes etapas do processo de criação de uma árvore de decisão são o particionamento e a condição de parada. “A Árvore de Decisão usará um algoritmo para definir quais atributos comporão um nodo, e qual a condição de particionar.” (AMARAL, 2016, pag. 99).

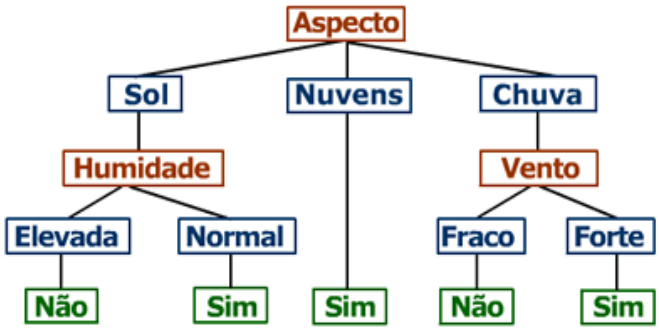
O particionamento é basicamente a distribuição do atributo de acordo com teste feito no nó da vez. Dessa forma, a partir do nó raiz o atributo em questão segue um caminho, uma ramificação, de acordo com valor obtido no teste do atributo. O teste geralmente segue uma expressão lógica que pode ser nominal ou numérica.

Caso seja nominal, o teste pode ser, por exemplo, se *aspecto*=sol, segue pelo nó da esquerda, se *aspecto* =chuva, segue pelo nó da direita, a Figura 1 ilustra como estes testes são feitos. Nesse caso, com atributos nominais o número de ramos vai ser igual ao número de possíveis valores do atributo. Já atributos numéricos podem ser testados em intervalos de valores ou com condições lógicas do tipo “maior” ou “menor igual” a uma constante predefinida para decidir que caminho percorrer.

Outra etapa importante do processo de geração de uma arvore de decisão é a condição de parada, ou seja, o ponto onde não haverá mais particionamento e o modelo vai levar a classe (AMARAL, 2016). Essa condição está relacionada com o grau de pureza dos nós, onde essa medida de pureza é que irá determinar quantos particionamentos posteriores serão necessários para classificar a instância, quanto mais puro menos partições serão necessárias para se chegar a classe.

As árvores de decisão classificam instâncias partindo da raiz da árvore, para algum nodo folha que fornece a classe da instância. Cada nodo da árvore especifica o teste de algum atributo da instância, e cada arco alternativo que desce daquele nodo corresponde a um dos possíveis valores de atributo. Uma instância é classificada começando no nodo raiz da árvore e testa o atributo relacionado a este nodo e segue o arco que corresponde ao valor do atributo na instância em questão. Este processo é repetido então para a sub-árvore abaixo até chegar a um nodo folha. A seguir apresenta-se uma árvore de decisão típica. A mesma classifica os dias, conforme eles são satisfatórios ou não, para jogar tênis.

Figura 1 - Árvore de decisão para jogar tênis



Fonte: (FREITAS, 2008)

A classificação de um exemplo de acordo com a esta árvore é feita da seguinte forma, analisando-se a Figura 2. O atributo Aspecto tem o valor Sol e a Humidade tem o valor Elevada. O exemplo é classificado como Não, ou seja, quando esteve Sol e Humidade elevada não se jogou ténis. Os atributos Temperatura e Vento não são considerados, pois são desnecessários para classificar este exemplo.

Figura 2 - Classificação utilizando a árvore de decisão para jogar tênis

Fonte: (FREITAS, 2008)

2.5.1.1-Entropia

A entropia de um conjunto pode ser definida como sendo o grau de pureza desse conjunto. Este conceito emprestado pela Teoria da Informação define a medida de "falta de informação", mais precisamente o número de bits necessários, em média, para representar a informação em falta, usando codificação óptima.

Dado um conjunto **S**, com instâncias pertencentes à classe **i**, com probabilidade **pi**, tem-se:

$$\boldsymbol{Entropia(S)} = \sum \ \boldsymbol{p_i log_2^{pi}} \qquad (1)$$

O ganho (*gain*) é define a redução na entropia. *Ganho(S,A)* significa a redução esperada na entropia de S, ordenando pelo atributo A. O ganho é dado pela seguinte equação:

$$\boldsymbol{GANHO(S,A)} = \boldsymbol{Entropia(S)} - \sum_{\boldsymbol{v=values(A)}} \frac{|\boldsymbol{S_V}|}{|\boldsymbol{S}|} \boldsymbol{Entropia(S_V)} \qquad (2)$$

2.5.2-Naive Bayes

Outro algoritmo utilizado para fazer a simulação com o *dataset KDDCUP'99* foi algoritmo *Naive Bayes*, por ser bastante conhecido e utilizado para tarefas que envolvem a classificação de novos objetos. O mesmo pertence a categoria dos algoritmos bayesianos, sendo baseado em princípios estatísticos do *Teorema de Bayes*. Esse Teorema foi formulado pelo

ministro britânico presbiteriano que viveu século XVIII, chamado Thomas Bayes, o mesmo utiliza de probabilidades condicionais para realizar a classificação.

Classificadores bayesianos comumente são usados para predizer a probabilidade de pertinência de um objeto a determinada classe, e possuem desempenho comparável a redes neurais artificiais e arvores de decisão para alguns problemas (CASTRO e FERRARI, 2016).

Também em Castro e Ferrari (2016), temos a descrição do funcionamento do algoritmo da seguinte forma, cada objeto é representado por um vetor de características m-dimensional **x** = (x1, x2, ..., xm). Assumindo se que a base de dados possui **c** classes, C1, C2, ...,Cc. Dado um objeto **x** com classe desconhecida, o classificador deve ser usado para predizer a classe à qual esse objeto pertence com base na maior probabilidade a posteriori encontrada, dado **x**, ou seja, o classificador bayesiano especifica uma classe $\mathbf{c_i}$ para o objeto **x** se, e somente se: $\mathbf{P(Ci|x) > P(Cj|x), \vee\ j \neq i}$ **.**

Ou seja, baseado nas probabilidades o algoritmo vai avaliar quanto o valor do atributo contribui para classificar a instancia como pertencente a uma das classes, dessa forma uma tabela de probabilidades é construída e o valor da classe com maior índice é escolhido para determinar a qual classe o novo registro pertence.

Para exemplificar o processo de classificação de um novo objeto utilizando o algoritmo *Naive Bayes*, será utilizada a base de dados de Carros. A descrição da base está disposta na Tabela 2. Trata-se do mesmo exemplo dado por Castro e Ferrari (2016), considerando como entrada a matriz de probabilidades apresentada na Tabela 2 e as probabilidades para cada classe sendo P(inaceitável) = 0,700, P(aceitável)= 0,222, P(bom) = 0,040 e P(ótimo) = 0,038. O objetivo é classificar o objeto x descrito pelos seguintes atributos: compra = médio; manutenção = médio; portas = 4; passageiros = 4; bagageiro = médio; segurança = média.

Partindo dos dados de entrada, o algoritmo calcula a probabilidade de o objeto x pertencer a cada classe:

P(x|inaceitável) x P(inaceitável) = (0,221 x 0,221 x 0,241 x 0,258 x 0,324 x 0,295) x 0,700 = $2{,}04 \times 10^{-4}$

P(x|aceitável) x P(aceitável) = (0,299 x 0,229 x 0,266 x 0,516 x 0,352 x 0,469) x 0,222 = $4{,}50 \times 10^{-4}$

P(x|bom) x P(bom) = (0,333 x 0,333 x 0,261 x 0,522 x 0,348 x 0,565) x 0,040 =

= $1{,}19 \times 10^{-4}$

P(x|ótimo) x P(ótimo) = (0,400 x 0,400 x 0,308 x 0,462 x 0,385 x 0,000) x 0,038 =
= 0,000

Para determinar a classe do objeto x, basta encontrar a probabilidade máxima dada pelo algoritmo, que é $4{,}50 \times 10^{-4}$ para a classe aceitável.

Tabela 2 - Probabilidade para o algoritmo *Naive Bayes* para base de dados carros

Atributo	Valor	Classe			
		Inaceitável	Aceitável	Bom	Ótimo
Compra	muito alto	0,298	0,188	0,000	0,000
Compra	Alto	0,268	0,281	0,000	0,000
Compra	Médio	0,221	0,299	0,333	0,400
Manutenção	muito alto	0,298	0,188	0,000	0,000
Manutenção	Alto	0,260	0,273	0,000	0,000
Manutenção	Médio	0,221	0,299	0,333	0,400
Manutenção	Baixo	0,221	0,240	0,667	0,400
Portas	2	0,269	0,211	0,217	0,154
Portas	3	0,248	0,258	0,261	0,231
Portas	4	0,241	0,266	0,261	0,308
Portas	5mais	0,241	0,266	0,261	0,308
Passageiros	2	0,476	0,000	0,000	0,000
Passageiros	4	0,258	0,516	0,522	0,462
Passageiros	Mais	0,266	0,484	0,478	0,538
Bagageiro	Pequeno	0,372	0,273	0,304	0,000
Bagageiro	Médio	0,324	0,352	0,348	0,385
Bagageiro	Grande	0,304	0,375	0,348	0,385

Atributo	Valor	Classe			
		Inaceitável	Aceitável	Bom	Ótimo
Segurança	Baixa	0,476	0,000	0,000	0,000
Segurança	Média	0,295	0,469	0,565	0,000
Segurança	Alta	0,229	0,531	0,435	1,000

Fonte: (CASTRO e FERRARI, 2016)

2.6-FERRAMENTA WEKA

Weka é um *software* gratuito de código aberto emitido sob a licença pública geral (GNU – General public license), desenvolvido em java e mantido pela Universidade de Waikato, da Nova Zelandia.

O *Software* possui interface gráfica que permite ao usuário realizar tarefas de pré-processamento, classificação, regressão agrupamento e visualização dos dados, assim como planejar e executar analise ou experimentos mais complexos por meio da construção de fluxogramas que encadeiam as tarefas de mineração de dados (CASTRO e FERRARI, 2016).

Figura 3 - Tela inicial do software *WEKA*

Fonte: Elaborada pelo autor (2018)

O *Weka* usa um formato de arquivo próprio chamado de formato *ARFF* (*Atribute-Relation File Format*). Trata se de um arquivo com a estrutura formada por um cabeçalho e uma seção com os dados separados por vírgula.

Figura 4 - Exemplo de arquivo *ARFF* usado como entrada de dados no software WEKA

```
@relation kdd_cup_1999

@attribute duration numeric
@attribute protocol_type {tcp,udp,icmp}
@attribute land {1,0}
@attribute wrong_fragment numeric
@attribute urgent numeric
@attribute hot numeric
@attribute num_failed_logins numeric
@attribute logged_in {1,0}
@attribute label {normal,anormal}

@data
0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,9,0,0.11,0,0,0,0,normal
0,tcp,http,SF,239,486,0,0,0,0,0,1,0,0,0,0,0,0,0,19,0,0.05,0,0,0,0,normal
0,tcp,http,SF,235,1337,0,0,0,0,0,1,0,0,0,0,0,0,0,29,0,0.03,0,0,0,0,normal
0,tcp,http,SF,219,1337,0,0,0,0,0,1,0,0,0,0,0,0,0,39,0,0.03,0,0,0,0,normal
0,tcp,http,SF,217,2032,0,0,0,0,0,1,0,0,0,0,0,0,0,49,0,0.02,0,0,0,0,normal
0,tcp,http,SF,217,2032,0,0,0,0,0,1,0,0,0,0,0,0,0,59,0,0.02,0,0,0,0,normal
0,tcp,http,SF,212,1940,0,0,0,0,0,1,0,0,0,0,0,0,1,69,0,1,0.04,0,0,0,normal
0,tcp,http,SF,159,4087,0,0,0,0,0,1,0,0,0,0,0,0,0,79,0,0.09,0.04,0,0,0,normal
0,tcp,http,SF,210,151,0,0,0,0,0,1,0,0,0,0,0,0,0,89,0,0.12,0.04,0,0,0,normal
0,tcp,http,SF,212,786,0,0,0,1,0,1,0,0,0,0,0,0,0,99,0,0.12,0.05,0,0,0,normal
0,tcp,http,SF,210,624,0,0,0,0,0,1,0,0,0,0,0,0,0,109,0,0.06,0.05,0,0,0,normal
0,tcp,http,SF,177,1985,0,0,0,0,0,1,0,0,0,0,0,0,0,119,0,0.04,0.04,0,0,0,normal
0,tcp,http,SF,222,773,0,0,0,0,0,1,0,0,0,0,0,0,0,129,0,0.03,0.04,0,0,0,normal
0,tcp,http,SF,256,1169,0,0,0,0,0,1,0,0,0,0,0,0,0,139,0,0.25,0.04,0,0,0,normal
0,tcp,http,SF,241,259,0,0,0,0,0,1,0,0,0,0,0,0,0,149,0,0.07,0.04,0,0,0,normal
0,tcp,http,SF,260,1837,0,0,0,0,0,1,0,0,0,0,0,0,0,159,0,0.04,0.04,0,0,0,normal
0,tcp,http,SF,241,261,0,0,0,0,0,1,0,0,0,0,0,0,0,169,0,0.03,0.04,0,0,0,normal
0,tcp,http,SF,257,818,0,0,0,0,0,1,0,0,0,0,0,0,0,179,0,0.02,0.03,0,0,0,normal
0,tcp,http,SF,233,255,0,0,0,0,0,1,0,0,0,0,0,0,0.25,189,0,0.02,0.03,0,0,0,normal
0,tcp,http,SF,233,504,0,0,0,0,0,1,0,0,0,0,0,0,0,199,0,0.02,0.03,0,0,0,normal
0,tcp,http,SF,256,1273,0,0,0,0,0,1,0,0,0,0,0,0,0,209,0,0.01,0.03,0,0,0,normal
```

Fonte: Elaborada pelo autor (2018)

Pode-se observar que o arquivo *ARFF* (em inglês: *Atribute-Relation File Format*) é composto de três campos, onde o nome do campo é precedido pelo símbolo @, são eles: relação (*@relation*), atributos (*@atribute*) e dados (*@data*). O campo relação geralmente é usado para referir-se ao nome do conjunto de dados.

O campo de atributos, refere-se conjunto de linhas onde cada linha apresenta o nome do atributo seguido do seu tipo, que pode ser nominal ou numérico. Depois da linha com o elemento *@data* segue se os dados brutos, ou registros, ou também chamados de instâncias onde cada linha refere se aos valores dos atributos separados por vírgulas (SANTOS, 2005).

2.6-SISTEMAS DE DETECÇÃO DE INTRUSÃO

No contexto de segurança da informação uma das ameaças mais divulgadas entre os profissionais de segurança é o *intruso*, que é um agente usuário ou software que comete uma transgressão a um sistema ou rede de computadores.

O ataque, ou a transgressão por usuários pode tomar a forma de acesso não autorizado a uma máquina ou, no caso de usuário autorizado, aquisição de privilégios ou execução de ações, além das que lhe foram autorizadas (STALLINGS, 2014). Uma das formas de combater essa transgressão é com um Sistema de Detecção de Intrusão (IDS – Intrusion Detection System).

Pode-se definir Detecção de Intrusão como um processo de monitoramento e análise de eventos em sistemas de computação ou redes de computadores, visando identificar intrusões prováveis as quais podem comprometer a integridade, disponibilidade de recursos e a confidencialidade do sistema computacional (FAGUNDES, 2016). Com a utilização dos métodos de detecção de intrusão pode-se guardar informações a respeito classes de ataques conhecidas e detectar iminentes tentativas de ataques à rede. Essas informações coletadas, podem ter utilidade para a composição de uma base informativa e também para o processo de tomada de decisões.

"Detecção de Intrusão é um serviço de segurança que monitora e analisa eventos de sistema com a finalidade de descobrir e avisar em tempo real ou quase em tempo real que estão ocorrendo tentativas de acesso a recursos de sistema de modo não autorizado." (STALLINGS, 2014 apud RFC 2828).

Esse é um conceito genérico para definir o que é um IDS, porém a tarefa de reconhecer acessos ditos anormais em um ambiente de rede de computadores se reduz a eficiência com que o computador, tomado de autonomia, seja capaz de distinguir estes acessos de forma mais eficiente possível.

A detecção de intrusão é baseada na suposição de que o comportamento do intruso difere daquele de um usuário legítimo de maneira que podem ser quantificadas (STALLINGS, 2014).

"Os Sistemas de Detecção de Intrusão (SDI) são elementos passivos da rede e trabalham como um coletor e analisador de tráfego, procurando por evidências de uso indevido geralmente baseados em regras ou comportamentos." (MAIA, 2005). Dessa forma, aplica-se o processo de mineração de dados para, através da análise dos dados históricos obtidos, classificar

estes dados e prever comportamentos de acessos que se distinguem de um comportamento normal.

2.7 – TRABALHOS RELACIONADOS

Recentemente nota-se o aumento do interesse na técnica de detecção de intrusão, estudos são feitos utilizando os mais variados conceitos computacionais aderentes a área da inteligência artificial envolvendo seus subcampos como aprendizado de máquina e técnicas como redes neurais, lógica fuzzy, e também técnicas de mineração de dados, através da tarefa de classificação.

Lemos, Assis e Souza (2010) em seu trabalho fazem um comparativo entre três tipos de árvores de decisão, mostrando qual mais eficiente com base no conceito de entropia, que é o conceito que mede o grau de pureza das árvores geradas. Apesar da variação de modelos de árvores geradas, verifica-se a eficiência do algoritmo árvore de decisão, apesar de sua variação. E utilizando o ambiente WEKA, e a técnica de *k-fold cross validation* com valor um fixo para k (k=10), em seu resultado também apresentou um desempenho de 99% de acurácia (acertos) usando apenas o algoritmo de árvore de decisão.

Maia (2005), propõe o desenvolvimento de um classificador baseado em indução probabilística e inferência Bayesiana, para implementar um sistema de detecção que pudesse identificar três tipos diferentes de ataques (*Guest, Neptune e portsweep*). Reforça a simplicidade e baixo custo computacional que o método Bayesiano apresenta. Em seus resultados apenas a classificação do ataque *Neptune* foi bem sucedida com 100% de acerto, e atribuiu o resultado não tão satisfatório nas outras classificações, ao fato de a base de dados apresentar características pouco correlacionadas, o que afetou o treinamento das classes.

Mohaisen e Alrawi (2013) estudaram a detecção do *malware Zeus banking Trojan* utilizando quatro tipos de aprendizado de máquina: SVM, Regressão Logística, árvore de decisão. A base de dados utilizada foi gerada pelos próprios autores. Esta base contém 1980 amostras do *Zeus Banking Trojan* identificadas manualmente por analistas. Com o auxílio de um sistema automatizado para análise de *malwares*, foram capturados os comportamentos característicos dessas amostras. Esse sistema captura atividades de, por exemplo, sistema de arquivos e de rede.

Narang et al. (2013), trata da extração de características relevantes de bases de dados, com a finalidade de reduzir o número de informações menos relevantes, redução do tempo de construção do classificador, melhorar a eficiência do classificador e a taxa de detecção de intrusão. A base de dados foi gerada por meio do monitoramento de atividades não maliciosas, reproduzidas em laboratório.

Stevens e Lowd (2013), aborda o emprego do aprendizado de máquina para detecção de comportamentos maliciosos. O trabalho é relevante, abordando, principalmente, o fato de que o atacante consegue escapar do detector quando estuda e testa os mecanismos de defesa, descobrindo assim, qual o comportamento deve ser utilizado para ser detectado, indevidamente, como legítimo.

Mantere et al. (2014) propõe um mecanismo complementar para auxiliar no monitoramento e detecção de anomalias em redes de computadores. A pesquisa contribui para o monitoramento de redes de ambientes industriais, melhorando, assim, a sua segurança por meio da criação de um módulo para detecção de anomalias na rede, utilizando a técnica de aprendizado dos Mapas Auto Organizáveis (SOM).

3 MATERIAL E MÉTODOS

3.1-INTRODUÇÃO

Neste Capítulo, apresenta-se a descrição detalhado do *Dataset* KDDCUP´99, utilizado para classificação de conexões normais e anormais, analisado no Capítulo 4. Além disso, discute-se a etapa de pré-processamento, realizada no mesmo antes da realização dos experimentos, bem como a técnica utilizada para validação de resultados e as métricas utilizadas para avaliação do desempenho dos classificadores. Por fim, apresentam-se os dois classificadores utilizados na classificação dos dados: Árvore de Decisão e Naive Bayes.

3.2-APRESENTAÇÃO DETALHADA DO *DATASET* KDDCUP'99

O *dataset* utilizado nos processos de classificação foi o *KDDCUP'99*, o mesmo foi usado para a Terceira Competição Internacional de Ferramentas de Descoberta de Conhecimento e Ferramentas de Mineração de dados, realizada em conjunto com a KDD-99, a Quinta Conferência Internacional sobre Descoberta de Conhecimento e Mineração de Dados. A tarefa da competição era construir um detector de intrusão de redes, um modelo preditivo capaz de distinguir entre conexões "anormais", chamadas intrusões ou ataques e conexões "normais". É baseado na iniciativa da *DARPA (Defense Advanced Research Projects Agency*) que forneceu a base de dados para projetista de Sistemas de Detecção de Intrusão.

Apesar do tempo que foi lançado o *dataset* ainda é muito utilizado por pesquisadores para análise de tráfego de rede. Sendo mais acessível e mais próximo de um cenário real de invasão a redes de computadores, seus registros apresentam os ataques mais conhecidos para a área de segurança de computadores.

Os dados do *dataset* publicado no site do evento (*www.kdd.org)* tem aproximadamente quatro *Giga Bytes* de arquivo comprimido de dados brutos de conexão *TCP*, de sete semanas de tráfego de rede, que pode ser transformado em cerca de cinco milhões de registros de conexões cada um com 100bytes.

Além disso, o site disponibiliza uma partição de 10% do total do *dataset*, que corresponde a uma base de dados com 494.020 registros, cada registro possui 42 instancias onde 41 instâncias correspondem a características da conexão que determinam o tipo de acesso

e uma última instancia que rotula a classe. No *dataset* estão simulados 22 tipos de ataques mais comuns encontrados em um ambiente de redes de computadores, de acordo com as instruções do site para o evento *KDDCUP*, pode-se dividir esses ataques em quatro categorias, conforme vê-se na Tabela 3.

Tabela 3 - Tipos de ataques e classes

4 classes principais de ataques	22 classes de ataques
Denial Of Service (DOS)	Back, land, neptune, pod, smurf, teardrop.
Remote to Local (R2L)	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
User to Root (U2R)	buffer_overflow, perl, loadmodule, rootkit
Probing	İpsweep, nmap, portsweep, satan

Fonte: (www.kdd.org)

A Tabela 4 descreve brevemente as quatro categorias de ataques em que foram dividas as classes de ataques.

Tabela 4 - Categorias de ataque

Categoria do Ataque	Descrição
DOS (Negação de Serviço)	Atacante envia um grande número de mensagens que esgote algum dos recursos da vítima, como CPU, memória, banda,etc. Ex: “syn flood”
U2R (User to Root attack)	Atacante acessa o sistema como usuário normal (ganho por : sniffing password, um dicionário local ou engenharia social) e passa a explorar vulnerabilidades para ganhar acesso como root ao sistema. Ex: “buffer overflow”
R2L (Remote to local attack)	Ocorre quando um atacante tem a habilidade de enviar pacotes para uma máquina através da rede, mas não tem uma conta nessa máquina e explora alguma vulnerabilidade para ganhar acesso local como usuário da máquina. Ex: “guessing password”
Probing	É uma tentativa de reunir informações sobre uma rede de computador com o propósito de burlar os controles de segurança. Ex: “port scanning”

Fonte: (www.kdd.org)

A divisão das 22 classes em 4 categorias é importante para que fazer a simulação onde o classificador será treinado para prever 5 classes de detecção. Sendo 4 classes de ataque e uma classe normal.

Após reconhecimento das classes de ataques que compõem a base de dados, e a categorização dela, apresentam-se a seguir os 42 atributos que fazem parte desta base. A Tabela 5 apresenta os campos de atributos que representam as características clássicas de uma conexão em redes *TCP/IP*, sendo por isso denominadas características intrínsecas de conexões *TCP/IP*. No total são nove atributos iniciais que fazem parte do *dataset.*

Tabela 5 - Características intrínsecas de conexões TCP/IP

Nome	Descrição	Tipo
"Duration"	Duração em segundos da conexão	Contínua
"Protocol_type"	Tipo de protocolo usado na conexão, i.e. tcp, udp, etc.	Discreta
"Service"	Serviço de rede no destino, por exemplo, http, telnet, etc.	Discreta
"Src_bytes"	Número de bytes enviados da fonte para o destino	Contínua
"Dst_bytes"	Número de bytes enviados do destino para a fonte	Contínua
"Flag"	Status da conexão (normal ou erro)	Discreta
"Land"	1 se conexão é de/para o mesmo host; 0 caso contrário	Discreta
"Wrong_fragment"	Número de fragmentos com erro	Contínua
"Urgent"	Número de pacotes com flag urgente habilitado	Contínua

Fonte: (www.kdd.org)

A Tabela 6 ilustra as características baseadas em conhecimento do especialista, as mesmas foram obtidas através da análise de informações contidas, primordialmente na área de dados dos pacotes *TCP*, *UDP* e *IP*. Nesta área, encontram-se normalmente, cabeçalhos de

aplicações de nível superior tais como *Telnet, FTP, http,* etc. São 13 atributos que formam essa área.

Tabela 6 - Características de conexão por conhecimento do especialista

Nome	Descrição	Tipo
"Hot"	Número de indicadores chaves ("hot")	Contínua
"Num_failed_logins"	Tentativas de login sem sucesso	Contínua
"Logged_in"	1 se login efetuado com sucesso; 0 caso contrário	Discreta
"Num_compromised"	Número de condições de "comprometimento"	Contínua
"Root_shell"	1 se shell root foi obtido; 0 caso contrário	Discreta
"Su_attempted"	1 se comando "su root" foi tentado; 0 caso contrário	Discreta
"Num_root"	Número de acessos como root	Contínua
"Num_file_creations"	Número de operações de criação de arquivos	Contínua
"Num_shells"	Número de "shells prompts" obtidos	Contínua
"Num_access_files"	Número de operações em arquivos de controle de acesso	Contínua
"Num_outbound_cmds"	Número de comandos externos em uma sessão ftp	Contínua
"Is_hot_login"	1 se o login pertence a lista "hot"; 0 caso contrário	Discreta
"Is_guest_login"	1 se o login usou a conta guest; 0 caso contrário	Discreta

Fonte: (www.kdd.org)

A Tabela 7 ilustra as características temporais de cada conexão. Tem-se nessa tabela mais 19 atributos que compõem a lista de atributos do *dataset.*

Tabela 7 - Características temporais: janela de 2 segundos

Nome	Descrição	Tipo
“Count”	Número de conexões iguais a esta para este mesmo “host” nos últimos 2 segundos	Contínua
“Srv_count”	Número de conexões para o mesmo serviço que o usado nesta conexão nos últimos 2 segundos	Contínua
“Serror_rate”	% de conexões que possuem erro “SYN”	Contínua
“Srv_serror_rate”	% de conexões que possuem erro “SYN” para este serviço	Contínua
“Rerror_rate”	% de conexões que possuem erro “REJ”	Contínua
“Srv_rerror_rate”	% de conexões que possuem erro “REJ” para este serviço	Contínua
“Same_srv_rate”	% de conexões para um mesmo serviço	Contínua
“Diff_srv_rate”	% de conexões para serviços diferentes	Contínua
“Srv_diff_host_rate”	% de conexões deste mesmo serviço para hosts diferentes	Contínua
“Dst_host_count”	Número de conexões com mesmo host de destino que esta	Contínua
“Dst_host_srv_count”	Número de conexões com mesmo host de destino e mesmo serviço que esta	Contínua
“Dst_host_serror_rate”	% de conexões para o mesmo host que o da conexão atual e que possui um erro S0.	Contínua
“Dst_host_srv_serror_rate”	% de conexões para o mesmo host e serviço que o da conexão atual e que possui um erro S0.	Contínua
“Dst_host_rerror_rate”	% de conexões para o mesmo host que apresentem flag RST	Contínua
“Dst_host_srv_rerror_rate”	% de conexões para o mesmo host e serviço da conexão atual que apresentem flag RST.	Contínua

Nome	Descrição	Tipo
“Dst_host_same_srv_count”	% de conexões com o mesmo host de destino e mesmo serviço que esta	Contínua
“Dst_host_diff_srv_rate”	% de conexões com o mesmo host de destino e services diferentes que esta	Contínua
“Dst_host_same_src_port_rate”	%conexões c/mesmo host de destino e mesma porta de origem que a conexão atual	Contínua
“Dst_host_srv_diff_host_rate”	% de conexões para o mesmo serviço vindo de diferentes hosts.	Contínua

Fonte: (www.kdd.org)

As Tabelas 5, 6 e 7 descrevem os 41 atributos que fazem parte do *dataset KDDCUP’99,* o ultimo atributo, dentre os 42 do total, é aquele que define o rótulo da classe, podendo ser um dos tipos de ataques descritos ou um acesso normal.

3.3-PRÉ-PROCESSAMENTO REALIZADO NO KDDCUP’99

A primeira etapa do processo de descoberta de conhecimento em base de dados consistiu em selecionar o *dataset KDDCUP’99* o qual será submetido ao processo de mineração de dados. Selecionou-se o conjunto de dados alvo, composto de 42 variáveis do *dataset* original. Na etapa de pré-processamento reduziu-se o número de atributos de 42 para 27. Utilizando-se estatísticas do software *WEKA* foram eliminados atributos que possuíam valor único dentre eles *num_outbound_cmds* e *is_host_login*. Foram eliminados atributos com alto valor de correlação, convencionou-se chamar atributos fortemente correlacionados aqueles que possuíam coeficientes de correlação maiores ou iguais a 0.8. O interesse era fazer seleção de atributos e não síntese por isso eliminou-se os atributos altamente correlacionados que menos ajudavam na classificação. Atributos altamente correlacionados influenciam um ao outro e trazem pouca informação, então não é interessante ter atributos correlacionados em qualquer problema, usamos PCA (*Principal Components Analysis*): *lnum_compromised, lnum_file_creations, lnum_access_files, lnum_outboud_cmds, is_host_login, count, srv_count,*

serror_rate,srv_serror_rate, rerror_rate,srv_rerror_rate, same_srv_rate, dst_host_count, dst_host_same_srv_rate, dst_host_rerror_rate.

Após a seleção de atributos normalizaram-se alguns atributos: *duration, wrong_fragment enum_failed_logins*, foram normalizados de forma que todos os valores fiquem no intervalo [0,1], usou-se o filtro normalize do ambiente *WEKA*. Ela se torna necessária para que os dados tenham a mesma ordem de grandeza. Se não houvesse essa normalização poderiam existir grandezas que teriam mais importância que outras**.**

3.4-PARTIÇÃO DO DATASET*KDDCUP'99* UTILIZANDO A TÉCNICA *K-FOLDCROSS VALIDATION*

Basicamente a técnica *K-fold cross validation*, funciona da seguinte maneira: os dados originais são divididos em ***K*** partes semelhantes, de forma aleatória e sem substituição. O processo então faz ***K*** iterações: a cada iteração ***K-1*** partes são utilizadas para treino e uma para teste do classificador. Na próxima iteração, uma nova parte é usada para teste e as outras ***K-1*** partes para treino, de tal forma que ao final as ***K*** partes são utilizadas para testar o modelo.

Uma das etapas mais importantes em um projeto de classificadores é a etapa de validação dos resultados. Muitas das técnicas mais eficientes de aprendizado de máquina apresentam uma grande quantidade de parâmetros e quanto menos restrições colocarmos no nosso modelo maior a probabilidade de encontrarmos um super ajustamento, ou como é mais conhecido, o *overfitting*.

Overfitting ocorre quando o método que estimamos consegue bons resultados apenas nos dados que eles foram treinados. Já na presença de novas observações, percebemos uma perda da eficiência da predição. Ou seja, o *overfitting* ocorre quando o método de aprendizado não consegue generalizar os resultados para dados que não foram utilizados no processo de treino.

A técnica *K-fold Cross Validation* estima o erro do método de aprendizado em observações não utilizadas no treino, ou seja, estima como o modelo construído irá se comportar com novos dados, claro que isto é válido apenas se mantivermos a mesma probabilidade conjunta das variáveis explicativas e da variável resposta utilizada durante o treino. O método *K-fold Cross Validation* consiste em dividir a base em k pedaços. Para cada pedaço, estimamos

o método sem a presença desta parte e verificamos o erro médio no pedaço não utilizado durante o treino. Abaixo, no Quadro 1, apresenta-se a descrição desse algoritmo.

Quadro 1 - Algoritmo k-foldcross-validation

1.Arranjar os exemplos de treinamento em ordem randômica 2.Dividir os exemplos de treinamento em "k" conjuntos não sobrepostos D_1,D_2,...,D_k. (K pedaços de aproximadamente D/k exemplos cada). 3.Para **i=1….k;** Treine o classificador usando todos os exemplos que não pertencem ao conjunto i ($D \backslash D_i$). Teste classificador em todos os exemplos no conjunto i (D_i). Computar $\boldsymbol{n_i}$, o número de exemplos no conjunto **i** que foram classificados erradamente. 4. Retorne a seguinte estimativa de erro para o classificador: $\boldsymbol{E} = \frac{\sum_{i=1}^{k} n_i}{D}$ *100% OBS: Para **t** execuções do algoritmo: $\boldsymbol{E} = \frac{\sum_{i=1}^{t} E_i}{t}$

Fonte: Elaborado pelo autor

3.5 AVALIAÇÃO DE DESEMPENHO DO CLASSIFICADOR

Uma vez obtidos os classificadores e os agrupamentos correspondentes à execução dos algoritmos, o desempenho de cada um deles deve ser avaliado. Existem diversas métricas para avaliar os resultados obtidos. Entretanto, quatro conceitos são necessários para calcular essas métricas: **Verdadeiros Positivos** (**VP**), **Falsos Negativos** (**FN**), **Falsos Positivos** (**FP**) e **Verdadeiros Negativos** (**VN**).

Verdadeiros Positivos (VP): são os registros reais da amostra de conexões normais que foram corretamente previstas como conexões normais;

Falsos Negativos (FN): são os registros reais da amostra de conexões normais que foram previstos como conexões anormais;

Falsos Positivos (FP): são registros da amostra de conexões anormais que foram previstas como sendo normal;

Verdadeiros negativos (VN): são registros de amostra de conexões anormais que foram previstas como sendo anormal;

Uma forma usual de apresentar os conceitos anteriores é por meio de uma tabulação cruzada entre a classe predita pelo modelo e a classe verdadeira (real) das instâncias. Essa tabulação é denominada matriz de confusão e será apresentada na próxima seção.

3.6-MATRIZ DE CONFUSÃO PARA OS EXPERIMENTOS COM DUAS E CINCO CLASSES

No ambiente do WEKA, os resultados dos experimentos de predição são apresentados através de matrizes de confusão com *layouts* específicos. A Tabela 8 é apresentado uma formatação especifica para os experimentos com duas classes: normal e anormal.

Tabela 8 - Matriz de confusão utilizada no WEKA para 2 classes de predição

		PREDITO	
		NORMAL	**ANORMAL**
REAL	**NORMAL**	**VP**	**FN**
	ANORMAL	**FP**	**VN**

Fonte: Elaborado pelo autor.

Na tabela 8, VP (Verdadeiros Positivos) representa a quantidade de instâncias classificadas corretamente como “normal”. FN (Falsos negativos) a quantidade de instâncias pertencentes a classe “normal” que foram classificadas incorretamente como “anormal”. FP (Falsos Positivos) a quantidade de instâncias que são classificadas incorretamente como “normais” e que pertencem à classe “anormal”. VN (Verdadeiros Negativos) a quantidade de instancias classificadas corretamente como “anormal”. Para uma boa classificação é necessário que os valores de VP e VN sejam os maiores possíveis e que os valores de FP e FN sejam os menores.

Na Tabela 9 apresenta-se o esquema da matriz de confusão para cinco classes NORMAL(NO) e ANORMAL (DOS, R2L, U2R e PROBING) que são as cinco classes presentes no KDDCUP'99.

Tabela 9 - Matriz de confusão utilizada no WEKA para 5 classes de predição

		PREDITO				
		NO	**DOS**	**R2L**	**U2R**	**PROBING**
REAL	**NO**	**VP**	**FP**	**FP**	**FP**	**FP**
		NO→NO	**DOS→NO**	**R2L→NO**	**U2R→NO**	**PR→NO**
	DOS	**FP**	**VP**	**FP**	**FP**	**FP**
		NO→NO	**DOS→DOS**	**R2L→DO**	**U2R→DOS**	**PR→DOS**
	R2L	**FP**	**FP**	**VP**	**FP**	**FP**
		NO→R2L	**DOS→R2L**	**R2L→R2L**	**U2R→R2L**	**PR→R2L**
	U2R	**FP**	**FP**	**FP**	**VP**	**FP**
		NO→U2R	**DOS→U2R**	**R2L→U2R**	**U2R→U2R**	**PR→U2R**
	PROBING	**FP**	**FP**	**FP**	**FP**	**VP**
		NO→PR	**DOS→PR**	**R2L→PR**	**U2R→PR**	**PR →PR**

Fonte: Elaborado pelo autor.

Na Tabela 9 os elementos da diagonal principal, representam os verdadeiros positivos, ou seja, **VP (NO→NO**) correspondem as instâncias classificadas corretamente como "normal". **VP (DOS→DOS)** representam as instâncias classificadas corretamente como "DOS"**, VP (R2L→R2L)** correspondem as instancias classificadas corretamente como (R2L), as instâncias **VP (U2R→U2R)** correspondem as U2R classificadas corretamente como U2R, e as instâncias **VP (PR →PR)** representam as instâncias classificadas corretamente como "probing". Os demais elementos fora da diagonal principal na tabela 9 são todos falsos positivos, onde a ordem das classes é seguinte **(PREDITO→REAL),** ou seja, **FP (DOS→NO)** representam as instâncias "normais", classificadas incorretamente como "DOS" e **FP**

(U2R→PR) as instâncias da classe “probing” classificadas incorretamente como “U2R” e assim sucessivamente para as demais.

3.7-MÉTRICAS UTILIZADAS

Visando avaliar a precisão do classificador, nas simulações para duas classes, foram usadas as métricas seguintes:

- Taxa de acurácia (**Tacurácia**) que indica a proporção de acertos ocorridos na classificação.
- Taxa de erros por falsos positivos para a classe “normal” (**TEnormal**), que representa a taxa de erros ocorridos na classificação da classe normal
- Taxa de erros por falsos positivos para a classe “anormal” (**TEanormal**), que representa a taxa de erros ocorridos na classificação da classe anormal.
- Taxa de erro total (**TEtotal1**) que indica a proporção total de erros ocorridos na classificação.

Nos experimentos para duas classes utilizou-se a equação 3, para a taxa de acurácia (**Tacurácia),** para a Taxa de erros por falsos positivos para a classe “normal” (**TEnormal**), a equação 4, para a Taxa de erros por falsos positivos para a classe “anormal”, (**TEanormal**), a equação 5 e para a Taxa de erros total (**TEtotal1**) a equação 6.

$$\mathbf{Tacurácia} = \frac{VP+VN}{VP+VN+FP+FN} \tag{3}$$

$$\mathbf{TEnormal} = \frac{FP}{FP+VN} \tag{4}$$

$$\mathbf{TEanormal} = \frac{FN}{VP+FN} \tag{5}$$

$$\mathbf{TEtotal1} = \frac{FP+FN}{VP+VN+FP+FN} \tag{6}$$

Para avaliar a precisão do classificador, nas simulações para cinco classes, foram usadas as métricas seguintes:

- Taxa de acurácia (**Tacurácia2**) que indica a proporção de acertos ocorridos na classificação.
- Taxa de erro total (**TEtotal2**) que indica a proporção total de erros ocorridos na classificação.
- Taxa de erros por falsos positivos para a classe "DOS" (**TEdos**), que representa a proporção de erros ocorridos na classificação da classe "DOS".
- Taxa de erros por falsos positivos para a classe "R2L" (**TEr2l**), que representa a proporção de erros ocorridos na classificação da classe "R2L".
- Taxa de erros por falsos positivos para a classe "U2R" (**TEu2r**), que representa a proporção de erros ocorridos na classificação da classe "U2R".
- Taxa de erros por falsos positivos para a classe "probing" (**TEprobing**), que representa a proporção de erros ocorridos na classificação da classe "probing".

No experimento para cinco classes para a **Tacurácia2,** usou-se a equação 7, para **TEtotal2** a equação 8**, TEnormal** a equação 9, **TEdos** a equação 10, **TEr2l** a equação 11**, TEu2r** a equação 12**, TEprobing** a equação 13.

$$\textbf{Tacurácia2} = \frac{VPno \rightarrow no + VPdos \rightarrow dos + VPr2l \rightarrow r2l + VPu2r \rightarrow u2r + VPprobing \rightarrow probing}{Total\_de\_Intâncias} \quad (7)$$

$$\textbf{TEtotal2=1- Tacurácia2} \quad (8)$$

$$\textbf{TEnormal} = \frac{Total\ de\ FP\ Normal}{Total\ de\ FP\ Normal + Total\ de\ VNdos,r2l,u2r,probing} \quad (9)$$

Onde:

$$Total\ de\ FP\ Normal = (FPno \rightarrow dos + FPno \rightarrow r2l + FPno \rightarrow u2r + FPno \rightarrow probing)$$

$$Total\ de\ VNdos, r2l, u2r, probing = (VNdos \rightarrow dos + VNr2l \rightarrow r2l + VNu2r \rightarrow u2r + VNprobing \rightarrow probing)$$

$$\textbf{TEdos} = \frac{Total\ de\ FP\ \ dos}{Total\ de\ FP\ dos + Total\ de\ VNno,r2l,u2r,probing} \quad (10)$$

Onde:

$$Total\ de\ FP\ dos = (FPdos \rightarrow no + FPdos \rightarrow r2l + FPdos \rightarrow u2r + FPdos \rightarrow probing)$$

$$\boldsymbol{Total\ de\ VNno, r2l, u2r, probing = (VNno \rightarrow no + VNr2l \rightarrow r2l + VNu2r \rightarrow u2r + VNprobing \rightarrow probing)}$$

$$\mathbf{TEr2l} = \frac{\boldsymbol{Total\ de\ FP\ \ r2l}}{\boldsymbol{Total\ de\ FP\ r2l + Total\ de\ VNno,do,u2r,probing}} \quad (11)$$

Onde:

$$\boldsymbol{Total\ de\ FP\ r2l = (FPr2l \rightarrow no + FPr2l \rightarrow dos + FPr2r \rightarrow u2r + FPr2l \rightarrow probing)}$$

$$\boldsymbol{Total\ de\ VNno, dos, u2r, probing = (VNno \rightarrow no + VNdos \rightarrow dos + VNu2r \rightarrow u2r + VNprobing \rightarrow probing)}$$

$$\mathbf{TEu2r} = \frac{\boldsymbol{Total\ de\ FP\ \ u2r}}{\boldsymbol{Total\ de\ FP\ u2r + Total\ de\ VNno,do,r2l,probing}} \quad (12)$$

Onde:

$$\boldsymbol{Total\ de\ FP\ u2r = (FPu2r \rightarrow no + FPu2r \rightarrow dos + FPu2r \rightarrow r2l + FPu2r \rightarrow probing)}$$

$$\boldsymbol{Total\ de\ VNno, dos, r2l, probing = (VNno \rightarrow no + VNdos \rightarrow dos + VNr2l \rightarrow r2l + VNprobing \rightarrow probing)}$$

$$\mathbf{TEprobing} = \frac{\boldsymbol{Total\ de\ FP\ \ probing}}{\boldsymbol{Total\ de\ FP\ probing + Total\ de\ VNno,do,r2l,u2r}} \quad (13)$$

Onde:

$$\boldsymbol{Total\ de\ FP\ probing = (FPpr \rightarrow no + FPpr \rightarrow dos + FPpr \rightarrow r2l + FPpr \rightarrow u2r)}$$

$$\boldsymbol{Total\ de\ VNno, dos, r2l, u2r = (VNno \rightarrow no + VNdos \rightarrow dos + VNr2l \rightarrow r2l + VNu2r \rightarrow u2r)}$$

4 EXPERIMENTOS COMPUTACIONAIS E RESULTADOS

4.1 INTRODUÇÃO

Neste Capítulo são avaliadas as precisões dos classificadores Árvore de Decisão e *Naive Bayes* na tarefa de classificar uma conexão de rede como normal ou anormal utilizando o *dataset* KDDCUP'99. Conforme mencionado no Capítulo 3, Seção 3.3 que trata do pré-processamento dos dados, algumas modificações foram feitas no *dataset* original que possuía, inicialmente, 42 atributos, entre essas modificações, reduziu-se o número de campos para 27.

No entanto, avaliar simplesmente o percentual de acertos ou erros do classificador não é suficiente, sabem-se que existem quatro métricas adicionais para avaliação do desempenho de cada classificador, essas métricas foram descritas no Capítulo3, Seção 3.5, elas são: Verdadeiros Positivos (VP) que representam conexões normais que o classificador previu como normais. Falsos Negativos (FN) que significam que o classificador classificou erroneamente uma conexão normal como anormal. Verdadeiros Negativos (VN) que significam que o classificador classificou conexões anormais e de fato eram, e finalmente Falsos Positivos (FP) que representam conexões anormais que o modelo previu normais.

Para melhor visualização e comparação do desempenho de cada algoritmo utilizado, os experimentos foram divididos em duas etapas de simulação: na primeira etapa comparam-se os dois algoritmos (Árvore de Decisão e *Naive Bayes),* considerando apenas duas classes de detecção e a variação do valor do parâmetro *k* da técnica de validação cruzada com valores menor ou igual a 10. Na segunda etapa, comparam-se os mesmos dois algoritmos, considerando agora cinco classes de detecção e o parâmetro *k* com valores maiores ou igual a 10 até no máximo 20.

Dessa forma observa-se o desempenho de cada classificador quando há um número maior de partições na hora de treinamento do modelo, o que ocorre quando altera-se o valor do parâmetro *k*, e também quando há um número maior de classes para classificar, quando aumenta-se de duas para cinco classes de detecção.

Adicionalmente, foram utilizadas outras métricas, descritas no Capítulo 3, Seção 3.7, que são Taxa de Erros (TE), Taxa de Acertos (Tacurácia e Tacurácia2) e tempo gasto para construção do modelo.

4.2-RESULTADOS DE SIMULAÇÃO PARA CLASSIFICAÇÃO DE DUAS CLASSES

Nessa primeira etapa de simulações após as alterações feitas no conjunto de dados inicial, ele ficou com duas classes para detecção: normal e anormal. O valor do parâmetro *k* da validação cruzada foi definido inicialmente para 2 no primeiro experimento depois para 4, 6, 8 e 10, nos experimentos posteriores totalizando 5 experimentos.

Essa mudança no conjunto de dados tem como objetivo mostrar o desempenho dos algoritmos de classificação (Árvore de Decisão e *Naive Bayes*) quando a simulação é feita com apenas duas classes, como também para verificar esse desempenho quando o valor de *k* muda para valores menores do que 10. Os resultados obtidos para cada classificador são apresentados nos próximos tópicos.

4.2.1-Resultados para a classificação com Árvore de Decisão (2 classes)

O conjunto de dados com 494020 registros, duas classes de detecção (normal e anormal) foi submetido ao software *WEKA* para classificação com o algoritmo Árvore de Decisão. O valor do parâmetro *k* foi alterado para cinco valores diferentes de 2 a 10.

Diminuindo o número de partições, esperava-se que com um número menor de partições, o tempo para treinamento do modelo fosse também menor, porém, não foi, pois quando k=2 o modelo levou 35.66 segundos para ser gerado e quando k=10 o modelo levou 34.2 segundos, conforme se observa nos resultados apresentados na Tabela 10.

Tabela 10 - Resultados de simulação para 2 classes com classificador Árvore de Decisão

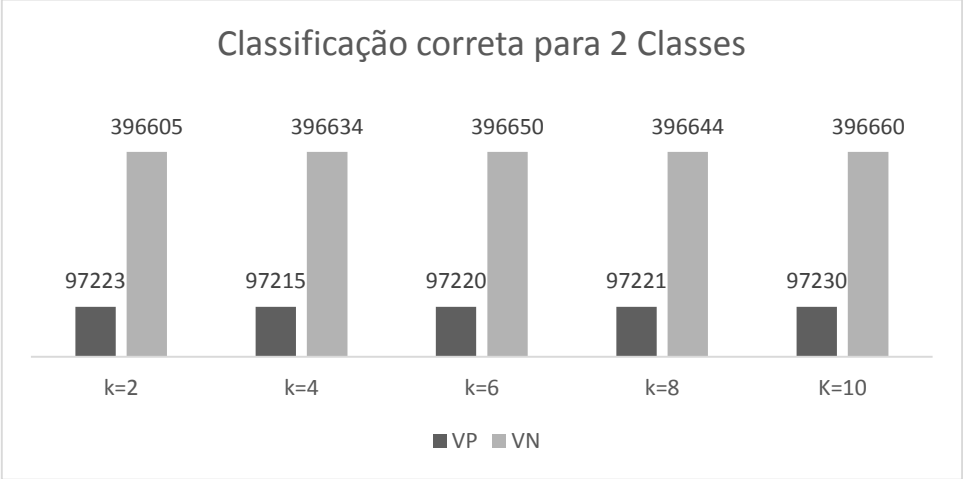
Exp/K	Acertos	Erros	Precisão	Tempo
1/K=2	**0,9996**	**0,0004**	**0,9986**	**35.66**
2/K=4	**0,9997**	**0,0003**	**0,9989**	**34.71**
3/K=6	**0,9997**	**0,0003**	**0,9990**	**37.15**
4/K=8	**0,9997**	**0,0003**	**0,9990**	**37.85**
5/K=10	**0,9997**	**0,0003**	**0,9991**	**34.2**

Fonte: Elaborada pelo autor.

Além disso, avalia-se nessa simulação variação da taxa de acertos e da taxa de erros, quando k=2 a taxa de acertos estava em 99,96% e a taxa de erros em 0,04%, e quando k=10 tem-se 99,97% e 0,03% respectivamente para taxa de acerto e taxa de erro. Uma diferença de valores muito baixa, porém essa diferença se torna mais notável quando medida em números de registros (instâncias).

Quando k=2 o número de instancias classificadas corretamente foi de 493828, que corresponde a soma dos registros classificados corretamente como positivos (VP = 97223) e os registros classificados corretamente como negativos (VN = 396605), quando k=10 essa soma (VP = 9730 e VN = 396660) resultou em 493890 registros classificados corretamente. Totalizando uma diferença de 62 registros a mais que serão classificados corretamente quando k=10. Esses valores são apresentados no Gráfico 1, onde resume-se os valores obtidos para classificação correta (VP+VN) das classes normal e anormal para cada experimento realizado na variação de *k (*k=2, k=4, k=6 e k=8).

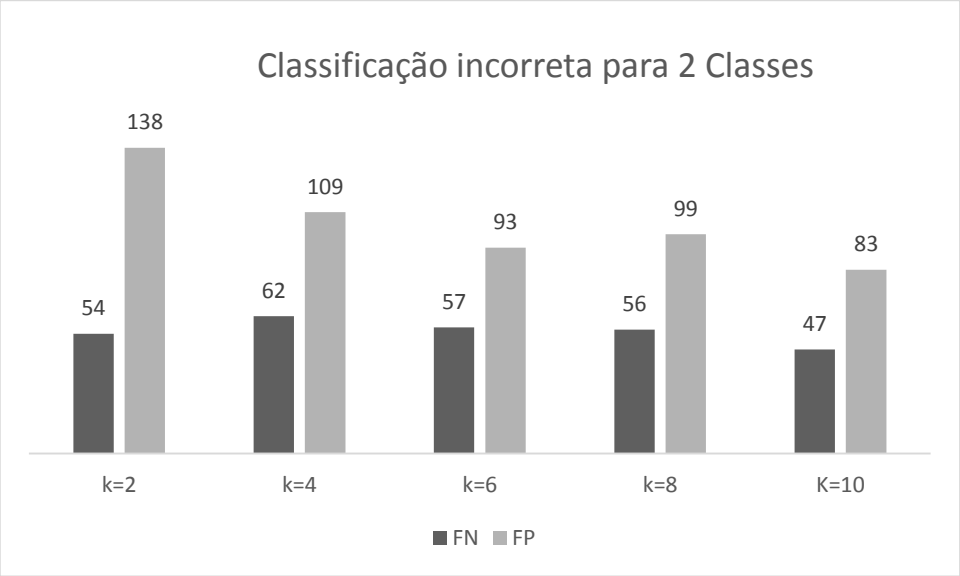
Gráfico 1 - Registros classificados corretamente (VP e VN) usando Árvore de Decisão



Percebe-se então que o melhor desempenho para esses experimentos ocorre quando *k*=10, no o número de registros classificados corretamente aumenta em 62 registros, de forma complementar, o número de registros classificados incorretamente diminui em 62. Conforme se observa no Gráfico 2 o total de instancias classificadas incorretamente, quando k=2 era de 192 (somando-se VP=54 e VN=138) e quando k= 10 diminuiu para 130 (soma de VP=47 e VN=83). O gráfico também apresenta os resultados obtidos a partir das matrizes de confusão do *WEKA*, dessa vez usando apenas os valores da diagonal secundária, que corresponde aos

registros classificados incorretamente, os falsos positivos (FP) e falsos negativos (FN) para cada experimento realizado com valor de *k*.

Gráfico 2 - Registros classificados incorretamente (FN e FP) com Árvore de Decisão



O classificador Árvore de Decisão é considerado bastante eficaz quando está analisando um grande conjunto de dados, e nessa primeira etapa das simulações fica comprovado seu ótimo desempenho, mesmo que o conjunto de dados seja considerado grande. Mesmo quando se alterou o número de partições no conjunto de dados para treino, verifica-se que não afeta tanto o seu desempenho, mas ao diminuir o número de partições aumentou-se o tempo de construção do modelo.

4.2.1-Resultados para a classificação com *Naive Bayes* (2 classes)

Ainda utilizando o conjunto de dados com apenas duas classes (normal e anormal), novas simulações foram feitas agora com o algoritmo *Naive Bayes*, utilizando o software *WEKA* com o parâmetro *k,* da validação cruzada, para valores menores ou igual a 10 (2, 4, 6, 8 e 10).

Em cada variação do valor de *k* foi feito um experimento, onde a partir dos resultados busca-se avaliar o desempenho do algoritmo com base na taxa de acerto, taxa de erro e tempo de construção do modelo. Diante dos resultados obtidos constatou-se que o desempenho do algoritmo foi bom, porém a precisão média do modelo gerado ficou em torno de 95,15%, um pouco abaixo do resultado obtido com o algoritmo Árvore de Decisão. Porém

o tempo obtido para criação do modelo foi muito melhor do obtido com a utilização da Árvore de Decisão, tendo valor 6.77 segundos para k=2 e diminuindo para 4.84 segundos quando k=10.

Conforme observa-se na Tabela 11, quando k=2 a taxa de acertos ficou em torno de 96,23%, e a taxa de erro foi de 3,77%, e para k=10, foi de 94,64% para taxa de acerto e 5,36% para taxa de erro. O que significa que quando se aumenta o número de partições o classificador aumenta o número de registros classificados incorretamente, o que não é bom para um analisador de tráfego que procura identificar intrusões em uma rede de computadores.

Tabela 11 - Resultados de Simulação para 2 classes utilizando o classificador Naive Bayes

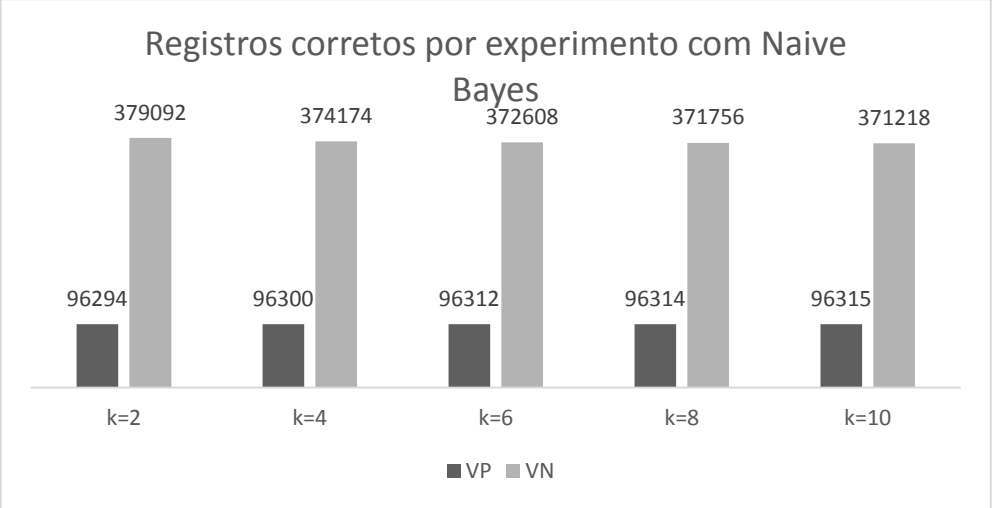
Exp/k	Tacuracia	TErro	Precisão	Tempo
1/K=2	**0,9623**	**0,0377**	**0,9899**	**6.77**
2/K=4	**0,9523**	**0,0477**	**0,9900**	**5.25**
3/K=6	**0,9492**	**0,0508**	**0,9901**	**4.84**
4/K=8	**0,9475**	**0,0525**	**0,9901**	**4.88**
5/K=10	**0,9464**	**0,0536**	**0,9901**	**4.84**

Fonte: Elaborada pelo autor.

Percebe-se nessa simulação que variação da taxa de acertos, da taxa de erros e tempo, conforme o valor de *k* vai aumentando de 2 para 10, gera uma diferença um pouco maior com relação a diferença encontrada nos experimentos feitos com Árvore de Decisão, resta saber como essa diferença se dará em número de registros.

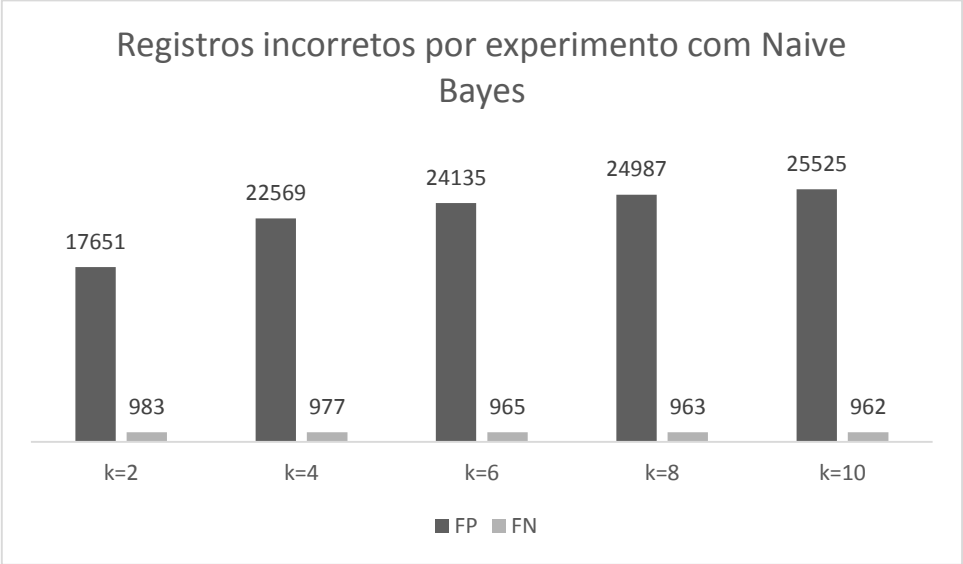
Quando *k*=2 tem-se na soma de VP e VN, que são os registros classificados corretamente, VP os classificados corretamente como normal e VN os classificados corretamente com anormal (intrusão), essa soma resulta em 475386 registros classificados corretamente e quando *k*=10 essa soma chega a 467533, ou seja, a diferença chega a 7853 registro classificados incorretamente quando *k*=10. O Gráfico 3 representa os valores dos registros classificados corretamente (VP e VN), para experimento realizado com a variação do valor *k*.

Gráfico 3 - Registros classificados corretamente (VP e VN) com Naive Bayes



No Gráfico 4 tem-se os valores dos registros classificados incorretamente (FP e FN) para cada variação do valor de *k*. percebe-se que o número de instâncias classificadas incorretamente é muito alto, principalmente quando a valor de *k* aumenta, e ainda bastante alto se comparado com a mesma análise feita com o algoritmo Árvore de Decisão.

Gráfico 4 - Registros classificados incorretamente (VP e VN) com Naive Bayes



.

4.3- RESULTADOS DE SIMULAÇÃO PARA CLASSIFICAÇÃO DE CINCO CLASSES

Nessa segunda etapa de simulações algumas alterações foram feitas no conjunto de dados inicial, de modo que ele ficou agora com cinco classes para detecção, quatro classes para ataque (DOS, R2L, U2R e probing) e uma classe normal. Essa divisão das classes de ataques em quatro categorias foi descrita na seção 3.2 que trata da apresentação detalhada do *dataset*. Dessa vez o valor do parâmetro *k* da validação cruzada foi definido inicialmente para 10 no primeiro experimento depois para 12, 14, 16, 18 e 20, nos experimentos posteriores.

Essa mudança no conjunto de dados tem como objetivo mostrar o desempenho dos algoritmos de classificação (Árvore de Decisão e *Naive Bayes*) quando a simulação é feita com mais de duas classes, como também para verificar esse desempenho quando o valor de *k* muda para valores acima do valor padrão 10. Os resultados obtidos para cada classificador são apresentados nos próximos tópicos.

4.3.1-Resultados para a classificação com Árvore de Decisão (5 classes)

Após submeter o conjunto de dados modificado para 5 classes de detecção ao *software WEKA*, realizaram-se 6 experimentos onde cada um teve um valor diferente para *k* (10, 12, 14, 16, 18 e 20) e o classificador árvore de decisão (Algoritmo J.48).

Os resultados desta simulação são resumidos na Tabela 12, onde constata-se que a taxa de acertos, definida como "Tacurácia2", ficou em torno de 99,97% para todos os experimentos, praticamente não houve melhora ao aumentar o valor de *k*, assim como também a taxa de erro total (TEtotal2) ficou em torno de 0,03% em todos os experimentos. O melhor tempo de construção do modelo ocorre quando k=14, 104.68 segundos, e o segundo melhor quando k=10, 107.42 segundos.

Tabela 12 - Resultados de Simulação para 5 classes utilizando o classificador Árvore de Decisão

Exp/k	Tacuracia2	TEtotal2	Tempo
1/k=10	0,99978	0,00032	107.42
2/k=12	0,99969	0,00031	108.43
3/k=14	0,99968	0,00032	104.68

Exp/k	Tacuracia2	TEtotal2	Tempo
4/k=16	0,99967	0,00033	109.93
5/k=18	0,99969	0,00031	110.19
6/k=20	0,99968	0,00032	110.63

Fonte: Elaborada pelo autor.

Nessas simulações realizadas com cinco classes, a cada experimento foi calculada a taxa de erro total. Porém, como acontece com classificação feita com mais de duas classes é interessante medir a taxa de erro obtida para cada classe. Diante disso observa-se que a taxa de erro para a classe normal quando comparada com mais de uma classe de ataque sofre uma redução quando comparada com apenas mais uma classe, para classificação com uma classe anormal foi de 0,03% e com 4 classes de ataque (DOS, R2L, U2R e probing) foi de 0,02%. Na Tabela 13 têm-se as taxas de erro obtidas para cada classe. Porém, como não houveram mudanças significativas em cada experimento, a taxa de erro por classe também não apresenta resultados notáveis, conforme observa-se na Tabela 13.

Tabela 13 - Taxa de erro para cada classe por experimento com Árvore de Decisão

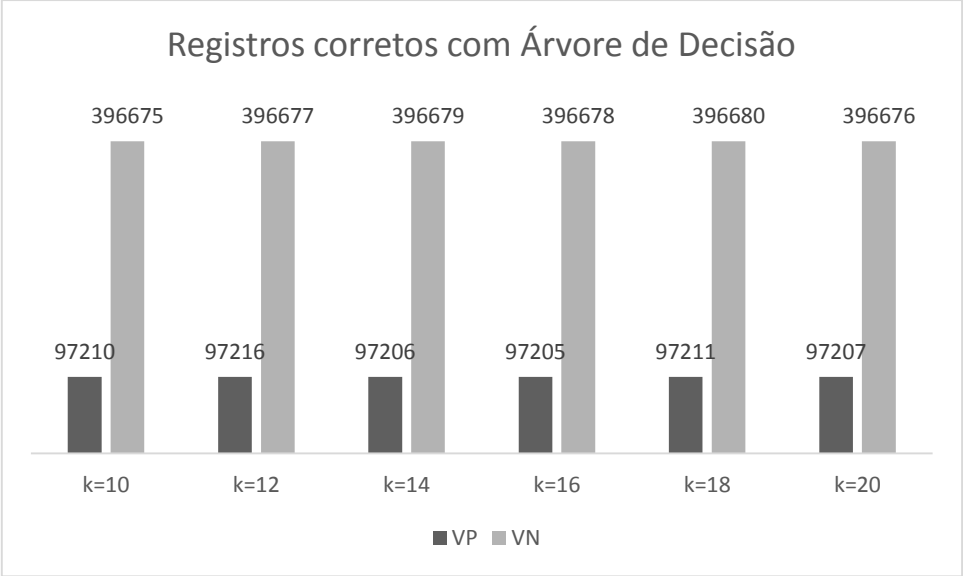
Exp/k	TEnormal	TEdos	TEr2l	TEu2r	TEprobing
1/k=10	0,0002	0,0003	0,0001	0,0000	0,0000
2/k=12	0,0002	0,0003	0,0001	0,0000	0,0000
3/k=14	0,0002	0,0003	0,0001	0,0000	0,0000
4/k=16	0,0002	0,0003	0,0001	0,0000	0,0000
5/k=18	0,0002	0,0003	0,0001	0,0000	0,0000
6/k=20	0,0002	0,0003	0,0001	0,0000	0,0000

Fonte: Elaborada pelo autor.

Após obter as taxas de acerto, de erro e o tempo de construção do modelo, para melhor visualização da quantidade de registros classificados corretamente e incorretamente, cada matriz de confusão gerada pelo *WEKA* para cada um dos experimentos com 5 classes, foi reduzida em outra matriz de confusão com apenas duas classes de detecção: normal e anormal.

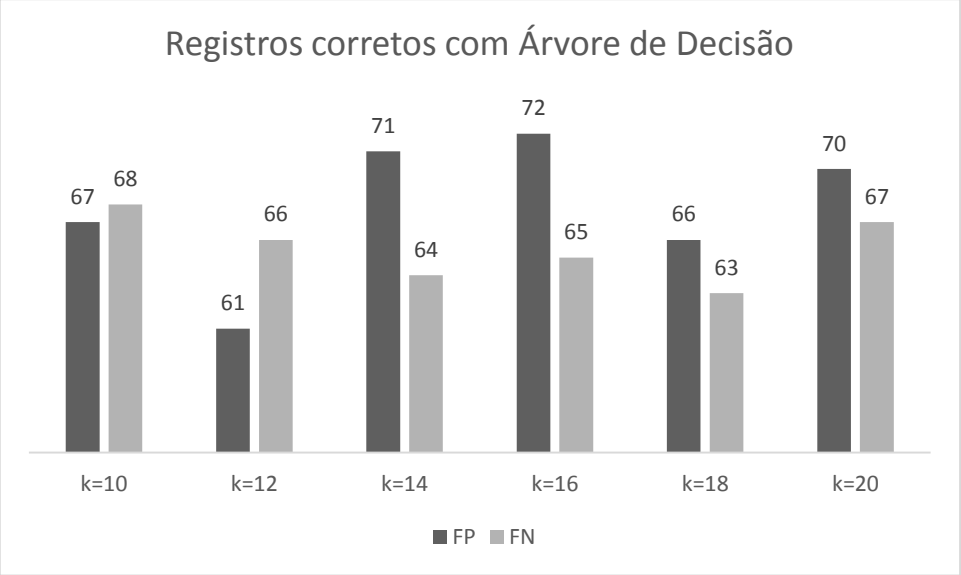
O Gráfico 5 apresenta a quantidade de registros classificado corretamente como sendo conexão normal ou conexão anormal para cada experimento realizado com a variação de *k*. Os valores de verdadeiros positivos (VP), registros que o modelo previu como conexões normal e realmente eram conexões normal, e verdadeiros negativos (VP), os registros que o modelo previu como sendo algum tipo de ataque e que realmente eram algum tipo de ataque. O maior número de registros classificados corretamente ocorre quando *k*=12, onde a soma de VP e VN resulta em 493893 registros, quando *k*=14 (melhor tempo) o número de registros classificados corretamente caí para 493885, que é o mesmo número de registros quando *k*=10.

Gráfico 5 - Registros classificados corretamente (VP e VN) com Árvore de Decisão (5 classes)



De forma complementar os registros classificados incorretamente, aqueles que o modelo previu como sendo conexões normais e não o eram (FP) e aqueles que o modelo previu como sendo anormais e eram normais (FN), são expostos no Gráfico 6. O menor número de registros classificados incorretamente ocorre quando *k*=12, onde a soma de FP e FN resulta em 127 registros. E o maior número ocorre quando *k*=16 e *k*=18, quando a soma de de FP e FN resulta 137 registros.

Gráfico 6 - Registros classificados incorretamente (FP e FN) com Árvore de Decisão (5 classes)



Nessas simulações feita com o *dataset* contendo 5 classes, nota-se uma diferença muito pequena nos resultados obtidos para cada experimento. Entretanto, o classificador conseguiu classificar corretamente em torno de 99%, como mostra a Tabela 12.

4.3.2-Resultados para a classificação com *Naive Bayes* (5 classes)

Ainda com o conjunto de dados modificado para 5 classes de detecção foi aplicado o classificador *Naive Bayes*. No *software WEKA*, realizaram-se com esse conjunto de dados, seis experimentos onde cada um teve um valor diferente para *k* (10, 12, 14, 16, 18 e 20), a fim de obter o melhor desempenho do algoritmo.

Os resultados desses experimentos demonstram que quando o valor de *k* aumenta, a taxa de acerto também aumenta proporcionalmente, de modo que quando *k*=10 a taxa de acerto (TACURACIA2) ficou em 94,94% e quando *k*=20 ficou em 95,02%.

O melhor tempo de criação do modelo ocorre quando *k*=18 com 5.94 segundos e o pior tempo foi quando *k*=10 com 7.89 segundos. A Tabela 14 mostra os resultados obtidos na simulação feita no conjunto de dados com 5 classes utilizando o algoritmo *Naive Bayes*.

Tabela 14 - Resultados de Simulação para 5 classes utilizando o classificador Naive Bayes

Exp/k	TACURACIA2	TEtotal2	Tempo
1/k=10	0,9494	0,0506	7.89
2/k=12	0,9496	0,0504	6.09
3/k=14	0,9496	0,0504	6.14
4/k=16	0,9499	0,0501	7.53
5/k=18	0,9500	0,0500	5.94
6/k=20	0,9502	0,0498	7.80

Fonte: Elaborada pelo autor.

De acordo com os resultados obtidos, percebe-se que a diferença entre a taxa de acerto no experimento 1 e no experimento 6 foi muito pouca. O que ocorreu também para taxa de erro total (TEtotal2). O mesmo acontece com as taxas de erros por classe (TEnormal, TEdos, TEr2l, TEu2r e TEprobing). A Tabela 15 exibe os resultados obtidos para as taxas de erros por classe nessas simulações.

Tabela 15 - Taxa de erro para cada classe por experimento com Naive Bayes

Exp/k	TEnormal	TEdos	TEr2l	TEu2r	TEprobing
2/k=10	0,0065	0,1193	0,0008	0,0101	0,0135
3/k=12	0,0065	0,1182	0,0008	0,0101	0,0134
4/k=14	0,0065	0,1189	0,0008	0,0100	0,0133
5/k=16	0,0065	0,1173	0,0008	0,0101	0,0065
6/k=18	0,0065	0,1176	0,0008	0,0100	0,0132
7/k=20	0,0065	0,1162	0,0008	0,0101	0,0131

Fonte: Elaborada pelo autor.

Assim como foi feito no tópico anterior, na simulação feita com Árvore de Decisão para cinco classes, nessa simulação após obter as taxas de acerto, de erro e o tempo de construção do modelo, cada matriz de confusão gerada pelo *WEKA* para cada um dos experimentos com 5 classes, foi consolidada em outra matriz de confusão com apenas duas classes de detecção: normal e anormal. Dessa forma pode-se observar a diferença no

desempenho do algoritmo *Naive Bayes*, baseado também na quantidade de registros classificados corretamente e incorretamente.

Tabela 16 - Matriz de Confusão Gerada pelo WEKA (para K=10)

A	**B**	**C**	**D**	**E**	
387916	0	21	601	2470	**A = DOS**
66	377	625	23	35	**B = R2L**
23	16	455	0	8	**C = U2R**
84	0	199	3788	36	**D = probing**
10809	338	3919	5725	76486	**E= normal**

Fonte: Elaborada pelo autor.

Tabela 17 - Matriz de Confusão consolidada para as classes Normal e Anormal (para K=10)

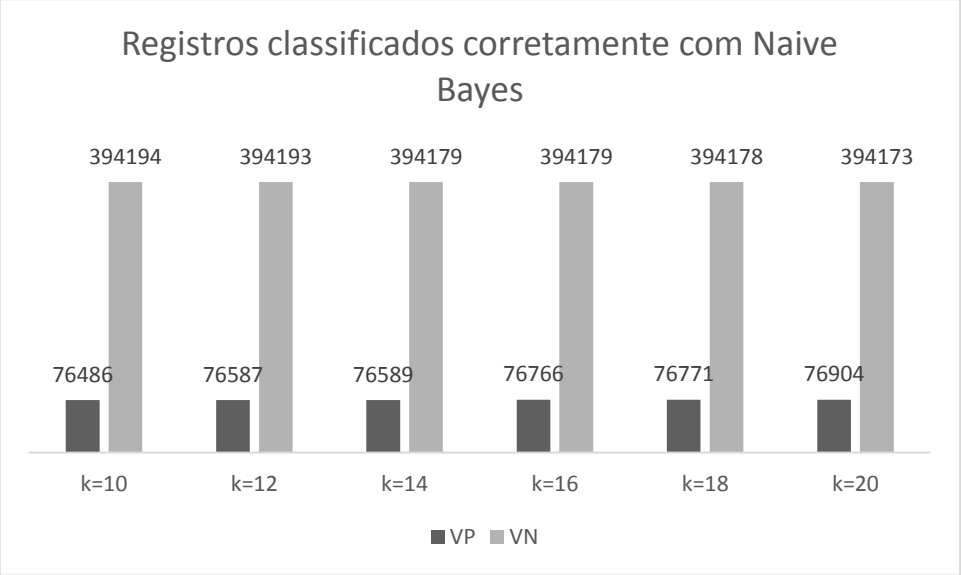
A	B	
76486	20791	A = Normal
2549	394194	B = Intrusão

Fonte: Elaborada pelo autor.

O total de registros classificados corretamente por experimento são obtidos da mesma forma que foi feito com os experimentos realizados com Árvore de Decisão, a soma entre o total de verdadeiros positivos (VP), que são registros que o modelo previu como conexões normal e realmente eram conexões normal, e os verdadeiros negativos (VP), os registros que o modelo previu como sendo algum tipo de ataque e que realmente eram algum tipo de ataque.

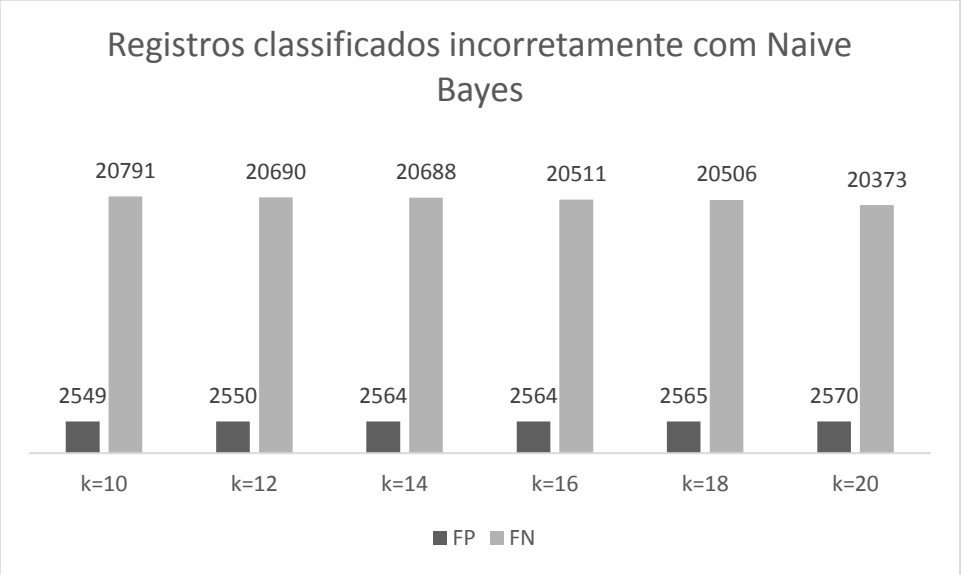
Dessa forma, o Gráfico 7 apresenta a quantidade de registros classificados corretamente como sendo conexão normal ou conexão anormal para cada experimento realizado com a variação de *k*. O experimento que teve um número maior de classificações bem-sucedidas foi quando *k*=20, no qual somando-se VP e FN resulta em 471077 registros. O pior caso ocorre quando *k*=10, no qual a soma de VP e VN resulta em 470680 registros.

Gráfico 7 - Registros classificados corretamente (VP e VN) com Naive Bayes (5 classes)



Complementarmente o número de registros classificados incorretamente tem o melhor caso, ou seja, uma quantidade menor de registros, quando *k*=20 no qual a soma de FP (aqueles registros que o modelo previu como sendo conexões normais e não o eram) e FN (aqueles que o modelo previu como sendo anormais e eram normais) resulta em 22943 registros. O Gráfico 8 apresenta esse resultado juntamente com o número de registros classificados incorretamente para cada experimento realizado com a variação de *k*.

Gráfico 8 - Registros classificados incorretamente (FP e FN) com Naive Bayes (5 classes)



5 CONCLUSÕES

Considerando-se os resultados para obtidos na primeira etapa de simulações para duas classes (normal e anormal), com algoritmo Árvore de Decisão, percebe-se que utilizando a técnica de validação cruzada, para valores de k crescentes k=(2,4,6,8,10), aumenta-se a taxa de acertos (valor médio em torno de 99.97%), e que a diferença encontrada entre os experimentos realizados é muito pequena. O experimento que teve *k*=10 como o melhor desempenho encontrado para o algoritmo, sendo a taxa de acerto (99,97%), taxa de erro (0,03%) e tempo de 34.2 segundos.

Ainda nessa primeira etapa de simulação, com o classificador *Naive Bayes*, percebe-se que a diferença de desempenho encontrada entre os experimentos foi maior do que a diferença encontrada com o algoritmo Árvore de Decisão. Também se nota que a melhora de desempenho é inversamente proporcional ao valor de *k*, quando *k* é baixo (k=2, por exemplo) a taxa de acerto foi a mais alta (96,23%), a taxa de erro foi mais baixa (3,77%), porém o tempo maior (6.77 segundos). O melhor tempo ocorreu quando k=10 (4.84 segundos), porém, nesse experimento a taxa de acertos foi a menor (94,64%) e a taxa de erro a maior (5,36%).

Portanto nessa primeira etapa de simulação feita com apenas duas classes o algoritmo de classificação que acertou na classificação foi o Árvore de Decisão com o valor do parâmetro *k* da técnica de validação cruzada igual a 10. E o classificador *Naive Bayes* foi o mais rápido.

Já para o segunda etapa de simulações para cinco classes com classificador Árvore de Decisão, o resultado obtido não foi muito diferente do que foi obtido na primeira etapa de simulação. Ao longo de todos os seis experimentos realizados a taxa de acerto e de erro mantiveram-se praticamente constante, 99,97% e 0,03% respectivamente. E o melhor tempo de criação do modelo ocorre quando *k*=14, com 104.8 segundos e o segundo melhor quando *k*=10, com 107.42 segundos.

Nessa mesma etapa para o classificador *Naive Bayes*, a melhor taxa de acerto ocorre quando *k*=20, ficando em 95,02% e a taxa de erro em 4,98%, porém o segundo maior tempo (7.80 segundos), dentre os experimentos.

Além disso, conclui-se nessa segunda etapa de simulações que o classificador *Naive Bayes* é bem mais rápido mesmo se aumentar-se o número de partições e o número de classes a serem analisadas. Também conclui-se que o classificador Árvore de Decisão necessita de um tempo maior para gerar o modelo quando é aumentado o número de classes.

Dessa forma, apesar de que o algoritmo Árvore de Decisão ser bastante eficiente em classificar corretamente uma conexão como sendo normal ou anormal, o algoritmo *Naive Bayes* é muito mais rápido em classificar essa conexão, considerando que um sistema de detecção tem de ter um tempo de resposta quase em tempo real, ele seria o mais indicado, mesmo que houvesse um número maior de classes de ataque a ser detectadas. A Tabela 18 ilustra esses resultados.

Tabela 18 - Analise de desempenho dos classificadores quando k=10

Classificador	**Taxa de Acertos**	**Taxa de Erros**	**Tempo**	**Valor de *k***
Árvore de Decisão (2 classes)	99,97%	0,03%	34.2	10
***Naive Bayes* (2 classes)**	96,23%	3,77%	4.84	10
Árvore de Decisão (5 classes)	99,97	0,03%	107.42	10
***Naive Bayes* (5 classes)**	95,02%	5,98%	7.80	10

Fonte: Elaborada pelo autor.

REFERÊNCIAS

AMARAL, Fernando **Introdução à ciência de dados: mineração de dados e big data**. Rio de Janeiro. Alta Books. 2016.

ALPAYDIN, Ethem; **Introduction to Machine Learning.** 3ª Ed. Massachusetts Institute of Technology (MIT). 2014.

CASTRO, Leandro Nunes de; FERRARI, Daniel Gomes; **Introdução à Mineração de Dados:** Conceitos Básicos. Algoritmos e Aplicações. 1ª Ed. São Paulo. Saraiva. 2016.

ELMASRI, Ramez; NAVATHE Shamkant B.; **Sistemas de Banco de Dados.** 6ª Ed. São Paulo: Pearson Education, 2011.

FAYYAD, Usama; PIATETSKI-SHAPIRO, Gregory; SMYTH, Padhraic. **The KDD Process for Extracting Useful Knowledge from Volumes of Data. In: Communications of the ACM.** p.27-34. 1996.

FERREIRA, Vinícius Oliveira. **Classificação de anomalias e redução de falsos positivos em sistemas de detecção de intrusão baseados em rede utilizando métodos de agrupamento.** 2016. Dissertação (Mestrado em Ciência da Computação) – Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual de São Paulo, São Paulo, 2016.

FREITAS, Ana T.; **Árvores de Decisão**; Disponível em: < http://web.tecnico.ulisboa.pt/ana.freitas/bioinformatics.ath.cx/bioinformatics.ath.cx/indexf23d.html?id=199>

GOEBEL, M.; Gruenwald L. **A survey of data mining and knowledge discovery software tools**. In: ACM SIGKDD Explorations Newsletter. 1. ed. vol. 1 1999.

GOLDSCHMIDT, Ronaldo Passos Emmanuel; BEZERRA, Eduardo. **Data Mining: Conceitos, técnicas, algoritmos, orientações e aplicações.** 2 ed. Rio de Janeiro. Elsevier, 2015.

HOSOKAWA, Eric Ossamu. **Técnica de Árvore de Decisão em Mineração de Dados -**; **Faculdade de Tecnologia de São Paulo. São Paulo. 2011.**

KAHWAGE, Cássia Maria Carneiro. **Ataques a redes de computadores e recomendações para sistemas de detecção de intrusos-IDS.** Dissertação (Mestrado em Ciência da Computação). Florianópolis. Universidade Federal de Santa Catarina. 2002.

KDD Cup 1999: Computer network intrusion detection. KDD. 1999. Disponível em: <http://www.kdd.org/kdd-cup/view/kdd-cup-1999/Tasks>. Acesso em: 10 mar. 2018.

LIMA, Christiane F. L.; ASSIS, Francisco M. de; SOUZA, Cleonilson P. de. **Árvores de Decisão baseadas nas entropias de Shannon, Rényi e Tsallis para Sistemas Tolerantes a Intrusão.** Instituto Federal do Maranhão. Disponível em: <

http://www.iiis.org/CDs2010/CD2010CSC/CISCI_2010/PapersPdf/CA554RJ.pdf>. Acesso em: 25 nov. de 2018.

LIMA, Felipe A. de. **Estudo do Sistema de Detecção de Intrusão.** Especialização em Redes e Segurança de Computadores. PUC-Paraná. Curitiba, 2012. Disponível em: <https://www.ppgia.pucpr.br/~jamhour/RSS/TCCRSS11/Felipe%20Alves%20de%20Lima%20_%20TCC%20v.1.pdf>

MAIA, Roberto Bomeny. **Detecção da intrusão utilizando classificação bayesiana.** Dissertação (Mestrado em Engenharia Elétrica). Rio de Janeiro. Universidade Federal do Rio de Janeiro – COPPE, 2005. Disponível em: <http://pee.ufrj.br/teses/textocompleto/2005042802.pdf>. Acesso em: 25 nov. de 2018.

MITCHELL, Tom M.; **Machine Learning.** McGraw-Hill, 1997.

MOLL, Vinícius. **Detecção de intrusão usando técnicas de aprendizagem de máquinas.** 2010. Dissertação (Mestrado em Engenharia de Automação e Sistemas) – Universidade Federal de Santa Catarina, Florianopólis, 2010.

SANTOS, Rafael. **Weka na Munheca:** Um guia para uso do Weka em scripts e integração com aplicações em Java. Instituto Nacional de Pesquisas Espaciais. 2005. Disponível em: <https://edisciplinas.usp.br/mod/resource/view.php?id=25867>

STALLINGS, William; BROWN, Lawrie - **Segurança de Computadores:** princípios e práticas; [tradução Arlete Simille Marques]. - 2. Ed. - Rio de Janeiro: Elsevier, 2014.