Aplicação de Algoritmos de Aprendizado de Máquina para Detecção de Intrusão

Eliel Dos S.Bentes[1], **Yann Fabricio Cardoso de Figueiredo**[1], **Lidio M.L. de Campos**[1].

[1]Faculdade de Computacão - ICEN– Universidade Federal do Pará (UFPA)
Rua Augusto Corrêa 01, Cep 66075-110 – Belém – PA – Brasil

{lidio}@ufpa.br, yann.fabricio@hotmail.com, mateusmelo04@gmail.com

Abstract. *Data mining techniques are fundamental tools in an Intrusion Detection System. In this work, simulations were carried out in an intrusion detection environment where the performance of two machine learning algorithms, the Decision Tree and the Naive Bayes in the task of classifying normal or abnormal connections using the KDDCUP'99 dataset, were analyzed. The classification of the data set was carried out in two stages in the first with only two connection classes (normal and intrusion) using the cross-validation technique with a k value less than or equal to 10. In the second with five detection classes (four for attacks and a normal one) using the k value greater than or equal to 10. The Decision Tree algorithm is quite efficient in correctly classifying a connection as being normal or intrusion, however, Naive Bayes algorithm is much faster in classifying a connection. Considering that a detection system must have a response time almost in real time, it was the most suitable, in the task of intrusion detection, for both two and five classes.*

Resumo. *Técnicas de mineração de dados são ferramentas fundamentais em um Sistema de Detecção de Intrusão. Nesse trabalho, realizaram-se simulações em um ambiente de detecção de intrusão onde foram analisados os desempenhos de dois algoritmos de aprendizado de máquina, Árvore de Decisão e o Naive Bayes na tarefa de classificação de conexões normais ou intrusões, utilizando-se 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 intrusão) 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. O algoritmo Árvore de Decisão é bastante eficiente em classificar corretamente uma conexão como sendo normal ou intrusão. Entretanto, algoritmo Naive Bayes(NB) é muito mais rápido em classificar uma conexão, Considerando que um sistema de detecção deve fornecer respostas em tempo real, por isso o NB foi o mais indicado, na tarefa de detecção de intrusão, tanto para duas como para cinco classes de detecção.*

1. Introdução

Os sistemas vulneráveis podem ser atacados a qualquer momento, sendo imprescindível reconhecer e reportar intrusões, que devem ocorrer o mais rápido possível, de preferência em tempo real [Khraisat 2019]. Para detectar intrusões em um sistema de computação,

uma ferramenta chamada IDS (Introduction Detection System) é usada, a qual pode usar várias técnicas de aprendizado de máquina para classificar uma conexão de rede como normal ou intrusão. Portanto, a segurança de Redes de Computadores se tornou essencial à medida que o uso da tecnologia da informação passou a fazer parte de nosso dia a dia. Como resultado, vários países como Austrália e Estados Unidos foram significativamente afetados pelos ataques. De acordo com o Relatório de Ameaças à Segurança na Internet da Symantec de 2017, mais de três bilhões de ataques foram relatados em 2016, o volume e a intensidade dos ataques foram substancialmente maiores do que antes [Symantec 2021]. Conforme destacado nas Estatísticas de segurança de dados em 2017, aproximadamente nove bilhões de registros de dados foram perdidos ou roubados por hackers desde 2013 [BLI 2021].

Nas últimas décadas, o aprendizado de máquina esta sendo usado para melhorar a detecção de intrusão. Há um grande número de estudos relacionados usando o conjunto de dados KDDCUP'99 ou DARPA 1999 [Stolfo 2021] para validar o desenvolvimento de IDSs, no entanto, não há uma resposta clara para a questão de quais técnicas de mineração de dados são mais eficazes. Nesse trabalho realiza-se a simulação de um ambiente de detecção de intrusão, a fim de verificar qual algoritmo de classificação, dentre os dois, Árvore de Decisão ou Naive Bayes, é o mais recomendado para detecção de intrusão usando o dataset KDDCUP'99.

A seção 2 apresenta a revisão de literatura, a secção 3 apresentam-se os algoritmos utilizados apara classificação e as métricas para avaliação dos modelos. Na seção 4, os experimentos e resultados são detalhados e finalmente, na seção 5, as principais conclusões são descritas.

2. Trabalhos Relacionados

Neste artigo, [Wenjuan Lian 2020] usa-se o algoritmo de aprendizagem combinando a árvore de decisão (DT) com métodos de eliminação de característica recursiva (RFE). Finalmente, uma série de experimentos de comparação por validação cruzada no dataset dados KDDCUP' 99 indicam que a abordagem pode melhorar melhor o desempenho do IDS e a precisão para todos os tipos de recursos é de 99%, exceto no caso da precisão do U2R, que é de 98%.

[Lin et al. 2015] apresenta uma abordagem híbrida para detecção de intrusão que utiliza a técnica k-Nearest Neighbors(KNN) em conjunto com a clusterização. Tal método utiliza um valor baseado na soma de duas distâncias para representar cada amostra em um classificador KNN, onde a primeira consiste na distância entre cada amostra e o centro de seu cluster e a segunda é a distância entre cada amostra e o seu vizinho mais próximo no mesmo cluster.

O trabalho realizado por [Raman et al. 2017] consiste em um sistema de detecção de intrusão evolutivo que usando Algoritmos Genéticos baseada em Hipergrafos (HG-GA) para seleção de atributos e configuração de parâmetros de uma SVM . Os resultados experimentais mostram que a abordagem tem um desempenho melhor que as técnicas existentes em relação a acurácia, taxa de detecção e taxa de falsos alarmes.

Nessa pesquisa, a análise comparativa dos estudos tomou por base o valor de k utilizado na validação cruzada. Em teoria, uma escolha comum é selecionar k = 10,

no entanto, isso tem um custo computacional maior, pois quanto mais subconjuntos de dados utilizados nessa técnica, mais modelos você precisa treinar. Os resultados gerais mostraram que nem sempre podemos esperar resultados mais precisos quando se aumenta o valor de k na validação cruzada. Em segundo lugar, o tempo gasto para construir IDSs não é considerado na avaliação de algumas técnicas de IDSs, apesar de ser um fator crítico para a eficácia de IDSs 'on-line'. Dessa forma, analisou-se, também, qual algoritmo é o mais rápido em classificar uma conexão como normal ou intrusão. Nesse sentido, esse trabalho contribui para avançar ou conhecimento e/ou complementar os trabalhos da literatura.

3. Materiais e Métodos

3.1. Algoritmos Usados

3.1.1. Árvore de Decisão

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 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 ótima. Dado um conjunto S, com instâncias pertencentes a classe i, com probabilidade p_i, a entropia é dada pela equação 1:

$$Entropia(S) = \sum p_i . \log_2^{p_i} \tag{1}$$

O ganho (gain) define a redução na entropia. Ganho(S,A) define a redução esperada na entropia de S, ordenando pelo atributo A. O ganho é dado pela equação 2.

$$GANHO(S, A) = Entropia(S) - \sum_{v=values(A)} \frac{|S_v|}{S} Entropia(S_v) \tag{2}$$

3.1.2. Naive Bayes

Classificadores bayesianos comumente são usados para predizer a probabilidade de pertinência de um objeto a determinada classe. O algoritmo da seguinte forma, cada objeto é representado por um vetor de características n-dimensional x=(F1, F2,.,Fn). 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 C para o objeto x se, e somente:

$$P(C|F1, ...Fn) = \frac{1}{Z} P(C) \prod_{i=1}^{n} p(Fi|C) \tag{3}$$

onde: Z é um fator de escala dependente apenas de F1,...,Fn, ou seja, uma constante se os valores das caraterísticas são conhecidas.

3.2. Considerações sobre o dataset KDDCUP'99

O KDDCUP'99 usa uma versão do conjunto de dados DARPA'98. O DARPA'98 tem cerca de 4 gigabytes de dados brutos (binários) compactados, coletados em tráfego de rede de 7 semanas, que consistem em cerca de 5 milhões de registros, cada um com cerca de 100 bytes. Os dados de teste têm cerca de 2 milhões de registros. O conjunto de dados de treinamento KDD consiste em aproximadamente 4.900.000 vetores de conexão, cada um com 41 recursos, possuindo por rótulo dois valores : conexão normal ou um ataque, com exatamente um tipo específico de ataque. Os tipos de ataque foram divididos em 4 categorias principais, como segue: Probing Attack, Denial of Service (DOS), User to root (U2R) e Remote to user (R2L). No datasets KDDCUP'99, os ataques (DoS, U2R, R2L e probe) são divididos em 22 tipos diferentes que estão apresentados na Tabela 1 [Sajid 2020].

Classes	22 Tipos de Ataques
DoS	back,land,neptune,pod,smurt,teardrop
R2L	ftp-write,guess-passwd,imap,multihop,phf,spy,warezclient,warezmaster
U2R	buffer-overflow,perl,loadmodule,rootkit
Probing	ipsweep,nmap,portsweep,satan

Tabela 1. Diferentes tipos de ataques no Dataset KDDCUP'99

3.3. Avaliação do Desempenho dos Classificadores

Existem diversas métricas para avaliar os resultados obtidos de classificação. Entretanto, quatro conceitos são necessários para calcular essas métricas: Verdadeiros Positivos (VP), Falsos Negativos (FN), Falsos Positivos (FP) e Verdadeiros Negativos (VN).

3.4. Métricas Utilizadas

A seguir apresentam-se as métricas utilizadas para avaliação dos modelos de classificação para duas e cinco classes de intrusão.

3.4.1. Métricas para Duas Classes

Nos experimentos para duas classes utilizou-se a equação 4, para a taxa de acurácia (Tacurácia), para a Taxa de erros por falsos positivos para a classe "normal" (TEnormal), a equação 5, para a Taxa de erros por falsos positivos para a classe "anormal", (TEanormal), a equação 6.

$$T_{acuracia} = \frac{VP + VN}{VP + VN + FP + FN} \tag{4}$$

$$T_{Enormal} = \frac{FP}{FP + VN} \tag{5}$$

$$T_{Eanormal} = \frac{FN}{VP + FN} \tag{6}$$

3.4.2. Métricas para Cinco Classes

Nos experimentos para cinco classes para a taxa de acurácia $T_{acuracia2}$, usou-se a equação 8, para a taxa de erro total, $T_{Etotal2}$ a equação 9, para a taxa de erro para a classe normal, TE_{normal} a equação 10, as demais taxas de erros, para as classes de intrusão: Dos, r2l, u2r e probing, TE_{Dos}, TE_{r2l}, TE_{u2r}, $TE_{probing}$ são obtidas de forma similar a equação 10.

$$V_{Ptotal} = VP_{noNo} + VP_{dosDOS} + VP_{r2lR2l} + VP_{u2rU2r} + VP_{ProbProbing} \tag{7}$$

$$T_{acuracia2} = \frac{VP_{total}}{VP + VN + FP + FN} \tag{8}$$

$$T_{Etotal2} = 1 - T_{acuracia2} \tag{9}$$

$$T_{Enormal} = \frac{TotaldeFP_{Normal}}{TotaldeFP_{Normal} + TotaldeVN_{dos}, r2l, u2r, probing} \tag{10}$$

$$TotaldeFP_{Normal1} = FP_{nodos} + FP_{nor2l} + FP_{nou2r} + FP_{noprobing} \tag{11}$$

$$Total_{VN} = VN_{dosDos} + VN_{r2lR2l} + VN_{u2rU2r} + VN_{probProb} \tag{12}$$

4. Experimentos e Resultados

Nesta seção 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.

4.1. Resultados com Árvore de Decisão e Naive Bayes (2 classes)

Os dados da Tabela 2 mostram os resultados obtidos com os classificadores Árvore de Decisão e Naive Bayes. O primeiro valor apresentado na coluna acertos, erros, precisão e tempo, corresponde ao valor obtido pelo classificador Árvore de Decisão e o segundo ao Naive Bayes. Com base nesses dados, percebe-se que aumentando-se o valor de k, o valore da acurácia e precisão do melhor modelo, varia muito pouco quando o classificador Árvore de decisão é utilizado. Mas, essa diferença se torna mais notável quando medida em números de registros (instâncias) classificadas corretamente. Quando k=2 o número de instancias classificadas com precisão foi de 493828, que corresponde a soma dos registros classificados corretamente como verdadeiros positivos(VP = 97223) e os registros classificados corretamente como verdadeiros negativos (VN = 396605), quando k=10 essa soma (VP = 97230 e VN = 396660) resultou em 493890 registros classificados corretamente. Totalizando uma diferença de 62 registros a mais que foram classificados corretamente quando k=10. Mesmo quando se alterou o número de partições no conjunto de dados para treino, verificou-se que isso, não afetou tanto o seu desempenho, mas quando se diminui o número de partições aumentou-se o tempo de construção do modelo, o que não é desejável.

Diante dos resultados obtidos pelo classificador Naive Bayes, constatou-se que a precisão média do modelo gerado ficou em torno de 95,15%, abaixo do resultado obtido com a Árvore de Decisão. Porém, o tempo obtido para criação do modelo foi muito menor do que o obtido com a utilização da Árvore de Decisão, tendo o valor de 6.77 segundos para k=2 e diminuindo para 4.84 segundos quando k=10. Conforme observa-se na Tabela 2, quando k=2 a acurácia ficou em torno de de 96,23%, e a taxa de erro foi de 3,77%, e para k=10, em torno de 94,64% e 5,36% respectivamente. O que significa que quando se aumenta o número de partições o classificador aumenta o número de registros classificados incorretamente, o que não é desejável para um analisador de tráfego que procura identificar intrusões em uma rede de computadores.

4.2. Resultados de classificação com Árvore de Decisão e Naive Bayes (5 classes)

Na segunda etapa, dos experimentos, comparam-se os mesmos algoritmos, considerando agora cinco classes de detecção de intrusão e o parâmetro k com valores maiores ou igual a 10 até no máximo 20. Adicionalmente, foram realizadas análises sobre o tempo gasto para construção do modelo. Os dados da Tabelas 3 e 4 mostram os resultados obtidos com os classificadores Árvore de Decisão e Naive Bayes. Nas simulações realizadas com cinco classes, a cada experimento foi calculada a taxa de erro total. Porém, como acontece com mais de duas classes é interessante medir a taxa de erro obtida para cada classe. Os resultados para o classificador Árvore de Decisão, mostram que a taxa de erro para a classe normal ficou em torno de 0,02%. Na Tabela 4 apresentam-se as taxas de erro obtidas para cada classe (normal e intrusão). Para esse classificador, não houveram mudanças significativas na acurácia, quando se aumentou o valor de k. Entretanto, conforme mostra a Tabela 3, 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) que ficou em torno de 0,03% em todos os experimentos. O menor tempo de construção do modelo ocorreu quando k=14, 104.68 segundos, e o segundo menor quando k=10, 107.42 segundos. A Figura 1 mostra uma matriz de confusão para simulação com árvore de decisão com K=10. Analisando os resultados para o classificador Naives Bayes, nas Tabelas 3 e 4. Percebe-se

A	B	C	D	E	
390993	0	0	9	6	**A = DOS**
1	1085	6	0	34	**B = R2L**
0	0	496	0	6	**C = U2R**
7	0	1	4077	22	**D = probing**
19	29	8	10	97211	**E= normal**

Figura 1. Matriz de Confusão para simulação com árvore de Decisão com K=10

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 menor 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.

5. Conclusão

Pelos resultados obtidos com algoritmo Árvore de Decisão, para duas classes, percebe-se que utilizando a técnica de validação cruzada, com valores de k crescentes k=(2,4,6,8,10),

obteve-se pouca variação na acurácia que manteve-se na média em torno de 99.97%. Com relação ao classificador Naive Bayes, nota-se que a melhora de desempenho é inversamente proporcional ao valor de k. Quando k é baixo (k=2, por exemplo) a acurácia acerto foi a mais alta (96,23%) e a taxa de erro também foi a mais baixa (3,77%), porém o tempo de obtenção do modelo foi maior (6.77 segundos). O melhor tempo ocorreu quando k=10 (4.84 segundos), porém nesse experimento a acurácia foi a menor (94,64%) e a taxa de erro maior (5,36%). Portanto para duas classes de classificação o algoritmo árvore de decisão é mais preciso, entretanto o tempo de obtenção do modelo foi menor para o Naive Bayes.

Com relação aos resultados de classificação, para cinco classes, obtidos com o classificador Árvore de Decisão, percebe-se que a acurácia e a taxa de erro mantiveram-se praticamente em torno, de 99,97% e 0,03% na média, respectivamente. O menor tempo de geração do modelo ocorre quando k=14, 104.8 segundos e k=10, 107.42 segundos. Já para o classificador Naive Bayes, as menores acurácia e taxa de erro ocorrem quando k=20, 95,02% e 4,98%, respectivamente, porém para K=20, obteve-se o segundo maior tempo de geração do modelo (7.80 segundos).

Tabela 2. Resultados de simulação para 2 classes com os classificadores Árvore de Decisão e Naive Bayes.

Exp/k	Acertos	Erros	Precisão	Tempo
1/k=2	0.9996 —0.9623	0.0004 — 0.0377	0.9986 — 0.9899	35.36—6.77
2/k=4	0.9997 —0.9523	0.0003 —0.0477	0.9989 — 0.99	34.71—5.25
3/k=6	0.9997 —0.9492	0.0003 —0.0508	0.9990—0.9901	37.15—4.84
4/k=8	0.9997—0.9475	0.0003— 0.0525	0.9990—0.9901	37.85—4.88
5/k=10	0.9997—0.9464	0.0003—0.0536	0.9991—0.9901	34.2—4.84

Tabela 3. Resultados de simulação para 5 classes com os classificadores Árvore de Decisão e Naive Bayes.

k	TAcuracia2	TEtotal2	Tempo
k=10	0.9978 —0.9494	0.0032 — 0.0506	107.42 — 7.89
k=12	0.9969 —0.9496	0.0031 — 0.0504	108.43 — 6.09
k=14	0.9968 —0.9496	0.0032 — 0.0504	104.68 — 6.14
k=16	0.9967 —0.9499	0.0033 — 0.0501	109.93 — 7.53
k=18	0.9969 —0.95	0.0031 — 0.05	110.19 — 5.94
k=20	0.9968 —0.9502	0.0032 — 0.0498	110.63 — 7.80

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

Tabela 4. Resultados de simulação para 5 classes com os classificadores Árvore de Decisão e Naive Bayes.

Exp/k	TEnormal	TEdos	TEr2l	TEu2r	TEprobing
k=10	0.0002 —0.0065	0.0003 — 0.1193	0.0001 —0.0008	0.0—0.0101	0.0—0.0131
k=12	0.0002 —0.0065	0.0003 — 0.1182	0.0001 —0.0008	0.0—0.0101	0.0—0.0134
k=14	0.0002 —0.0065	0.0003 — 0.1189	0.0001 — 0.0008	0.0—0.0100	0.0—0.0133
k=16	0.0002 —0.0065	0.0003 —0.1173	0.0001 —0.0008	0.0—0.01001	0.0—0.0065
k=18	0.0002 —0.0065	0.0003 —0.1176	0.0001 —0.0008	0.0—0.01	0.0—0.0132
k=20	0.0002 —0.0065	0.0003 — 0.1162	0.0001 — 0.0008	0.0—0.0101	0.0—0.0131

Por fim, o algoritmo Árvore de Decisão é bastante eficiente em classificar corretamente uma conexão como sendo normal ou intrusão. Entretanto, o algoritmo Naive Bayes é muito mais rápido. Considerando que, um sistema de detecção deve fornecer respostas em tempo real, ele seria o mais indicado como IDS, mesmo que houvesse um número maior de classes de ataques a serem detectadas.

Referências

BLI (2021). Breach level index. data breach statistics. In *February, 2021, Available: http://breachlevelindex.com/*.

Khraisat, A., G. I. V. (2019). *Survey of intrusion detection systems: techniques, datasets and challenges*. Springer, 2th edition.

Lin, W.-C., Ke, S.-W., and Tsai, C.-F. (2015). Cann: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-Based Systems*, 78:13–21.

Raman, M. G., Somu, N., Kirthivasan, K., and Sriram, V. S. (2017). A hypergraph and arithmetic residue-based probabilistic neural network for classification in intrusion detection systems. *Neural Networks*, 92:89–97. Advances in Cognitive Engineering Using Neural Networks.

Sajid, A. (2020). A Taxonomy of Cyber-attacks on Computer Networks. Research report, university of bradford.

Stolfo, S. (2021). Kdd cup 1999 data set. kdd repository, university of california, irvine. In *February, 2021, Available: http://kdd.ics.uci.edu/*.

Symantec (2021). Internet security threat report 2017. In *February, 7017 2021, vol. 22 Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf*.

Wenjuan Lian, Guoqing Nie, B. J. D. S. Q. F. Y. L. (2020). An intrusion detection method based on decision tree-recursive feature elimination in ensemble learning. In *Mathematical Problems in Engineering*. Hindawi.