Estudo Comparativo de Métodos de Aprendizagem de Máquina Aplicados em Sistemas de Detecção de Intrusão

Lucas C. C. Silva[1], **Alexandre W. S. Silva**[1], **Aridson N. Fernandes Filho**[1], **Alex O. Barradas Filho**[1]

[1]Universidade Federal do Maranhão (UFMA)
São Luís, MA – Brazil

{lcleopark99, alexandreacad, aridsonfilho, barradas.alex}@gmail.com

Abstract. *With the advent of technology, information traffic is growing faster and the invasion of this scope has become commonplace. However, there are mechanisms to avoid and / or identify this type of intrusion, that are the Intrusion Detection Systems. And with the advances of the Artificial Intelligence tools verified the viability of having IDS based on Machine Learning, starting from the resizing done in the NSL-KDD dataset using the same methods of an earlier study done in another dataset, maintaining the accuracy and decreasing the computational cost.*

Resumo. *Com o advento da tecnologia, o tráfego de informações cresce cada vez mais rápido e a invasão neste escopo se tornou cotidiana. Porém há mecanismos para evitar e/ou identificar este tipo de intrusão que são os Sistemas de Detecção de Intrusão (IDS – Intrusion Detection System). E com os avanços das ferramentas de Inteligência Artificial, verificou a viabilidade de se ter IDS baseados em Aprendizagem de Máquina, partindo do redimensionamento feito no dataset NSL-KDD utilizando os mesmos métodos de um estudo anterior feito em outro dataset, mantendo a acurácia e diminuindo o custo computacional.*

1. Introdução

Nas últimas décadas, o tráfego de dados na rede de computadores mundial cresceu exponencialmente motivado pelo desenvolvimento e consumo constante de novas tecnologias conectadas à internet (*smartwatch, smart glasses, video games*, entre outros dispositivos) [Hamzei and Navimipour 2018]. Tal necessidade de conectar diferente dispositivos, com o desígnio de compartilhar informações e recursos entre si para proporcionar um ambiente tecnológico mais sensível ao contexto e útil aos usuários, também proporcionou a indispensabilidade do desenvolvimento de tecnologias eficazes no controle e na gestão da transmissão dos dados [Alkhalil and Ramadan 2017].

Nesse contexto, alguns trabalhos têm direcionados esforços para a detecção de intrusão, que podem ser classificados em três categorias: detecção por assinatura ou mau uso, detecção por anomalia e detecção baseada em especificação [Hajiheidari et al. 2019]. O presente trabalho direciona o estudo para a detecção por anomalia, a qual entende-se pela prática de identificar itens ou eventos que não estão em conformidade com um comportamento esperado conforme um conjunto de dados analisados [Zhang et al. 2019]. Em geral, o tema de detecção por anomalia, com a aplicação de técnicas de aprendizado de máquina, tem sido amplamente explorado na literatura científica [Viegas et al. 2017].

No trabalho de [Zhang et al. 2019], descreve-se um novo método para detectar um comportamento anômalo em dados de desempenho de rede, reunidos pela *Open Science Grid* e utilizando servidores *perfSONAR*. Enquanto na pesquisa de [Chouhan et al. 2019], propõe-se uma nova arquitetura de *channel boosting and residual learning based deep convolutional neural network (CBR-CNN)* para a detecção de intrusões de rede.

Na pesquisa de [Viegas et al. 2017], um novo esquema de avaliação para o campo de detecção de intrusão por métodos de aprendizagem de máquina. Para tanto, o artigo propõe uma abordagem para a criação de banco de dados de intrusão, que contemple as características de fácil atualização e reprodução, tráfego real e válido, representatividade e disponibilidade ao público.

No artigo de [Tavallaee et al. 2009], uma análise estatística é realizada sobre uma base de dados denominada de KDDCup99, com o objetivo de identificar questões importantes que afetam fortemente o desempenho dos métodos aplicados na detecção de intrusão por anomalia. Como resultado da pesquisa, os autores propuseram um novo conjunto de dados (NSL-KDD).

Dessa forma, a proposta do presente trabalho está primeiramente em redimensionar a base de dados NSL-KDD para reduzir a quantidade de atributos e aplicar os métodos de aprendizagem de máquina para a detecção de intrusão por anomalia, com o objetivo de melhora do custo computacional e sem a perda tanto da acurácia como da representatividade dos dados.

2. Preparação da base de dados

O NSL-KDD, contém mais de 140.000 instâncias e mantém as 42 características da base de dados antiga (KDDCup99), onde a mesma está isenta de dados redundantes – a saída de cada instância indica se o tráfego é anômalo ou normal.

Os atributos da base de dados NSL-KDD estão classificados em categóricos e numéricos. Categóricos: *protocol_type*, *service*, *flag*, *land*, *logged_in*. Numéricos: *duration*, *src_bytes*, *dst_bytes*, *wrong_fragment*, *urgent*, *hot*, *num_failed_logins*, *num_compromised*, *root_shell*, *su_attempted*, *num_root*, *num_file_creations*, *num_shells*, *num_access_files*, *num_outbound_cmds*, *is_host_login*, *is_guest_login*, *count*, *srv_count*, *serror_rate*, *srv_serror_rate*, *rerror_rate*, *srv_rerror_rate*, *same_srv_rate*, *diff_srv_rate*, *srv_diff_host_rate*, *dst_host_count*, *dst_host_srv_count*, *dst_host_same_srv_rate*, *dst_host_diff_srv_rate*, *dst_host_same_src_port_rate*, *dst_host_srv_diff_host_rate*, *dst_host_serror_rate*, *dst_host_srv_serror_rate*, *dst_host_rerror_rate*, *dst_host_srv_rerror_rate*, *class*.

Para a verificação da hipótese levantada, pegou-se a base de dados NSL-KDD no formato *CSV (Comma Separated Values)*, transformou-se as instâncias nominais dos atributos categóricos em números, duplicou-se este conjunto de dados, redimensionou-se a cópia e removeu-se algumas de suas características – chamou-se o novo conjunto de dados de NSL-KDD-R.

O critério de remoção das características baseia-se em duas observações: atributos altamente correlacionados e atributos que contém valores únicos [Campos and Lima 2012]. Atributos altamente correlacionados são linearmente dependentes, ou seja, estes atributos contribuem igualmente para o modelo quando o ocorre

a classificação das instâncias, podendo causar um aumento no tempo de espera e no custo computacional. Entende-se por atributos altamente correlacionados aqueles que tiverem uma pontuação maior ou igual que 0.8 [Morettin and Bussab 2010]. A segunda observação feita na base dados é que há atributos que contêm valores únicos ou perto da unicidade, ou seja, estes atributos não irão contribuir significativamente para o modelo no momento da classificação, visto que irão ser os mesmos valores para cada valor do atributo de saída (*class*). A Figura 1 ilustra as etapas realizadas nas bases de dados.

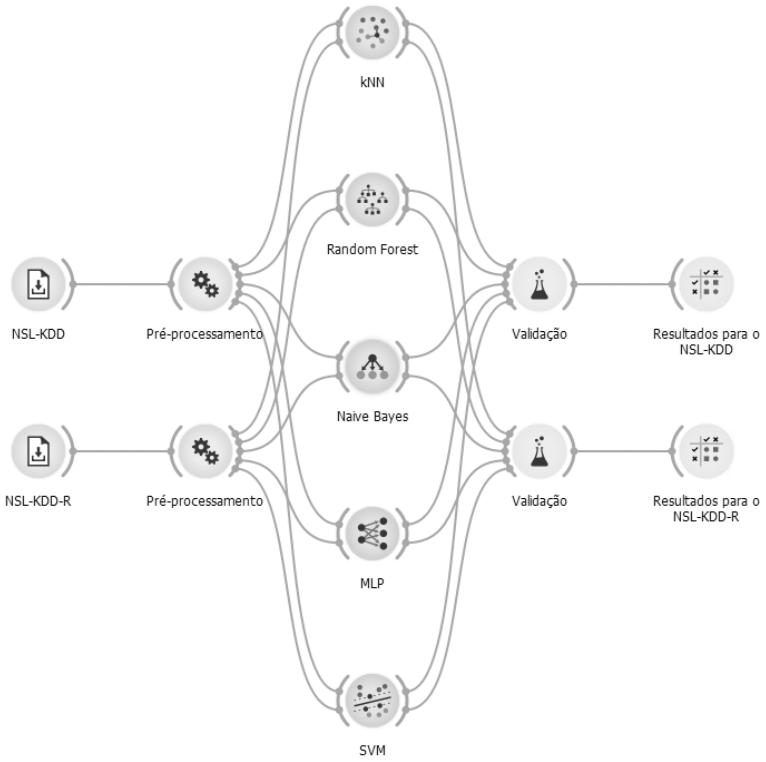

Figura 1. *Fluxo da metodologia deste artigo*

3. Métodos de classificação

Utilizou-se cinco métodos para classificar as instâncias das duas bases de dados apresentadas acima. Fez-se uma busca em grade para determinar os melhores parâmetros de um intervalo para cada um destes modelos.

3.1. *KNN (K-Nearest Neighbors)*

O algoritmo *KNN (K-Nearest Neighbors)* armazena instâncias de treinamento e, quando uma nova instância é apresentada a ele, o algoritmo calcula a distância dessa nova instância com todas as outras armazenadas no modelo, e são retornados os k vizinhos mais próximos da mesma. E essa nova instância é classificada com a classe mais frequentes dentre os k vizinhos retornados. E para este artigo foram utilizados os parâmetros da Tabela 1.

3.2. *Random Forest*

No algoritmo *Random Forest*, são criadas muitas árvores de decisões de forma aleatória com o objetivo de melhorar a acurácia na classificação das instâncias [Farnaaz and Jabbar 2016]. Na Tabela 2 contém os parâmetros utilizados.

Tabela 1. *KNN*

Parâmetros	Especificações
Vizinhos	1, 3, 5, 7 e 9
Distância	Euclideana e Manhattam
Peso	Distância e Uniforme

Tabela 2. *Random Forest*

Parâmetros	Especificações
Nº de Árvores	60 a 200

3.3. *Naive Bayes*

Este algoritmo é um classificador probabilístico baseado na aplicação do Teorema de Bayes e que desconsidera a correlação entre os atributos do conjunto de dados. Implicando em uma inferência ingênua (*naive*).

3.4. *MLP (Multi-layer Perceptron)*

A *Multi-layer Perceptron (MLP)* é uma Rede Neural Artificial contendo um conjunto de neurônios combinados em camadas, cada uma contendo determinada quantidade de neurônios [Shenfield et al. 2018]. Os parâmetros utilizados estão na Tabela 3.

Tabela 3. *MLP*

Parâmetros	Especificações
Função de ativação	Tangente Hiperbólica (*TanH*) e Linear Retificada (*ReLu*)
Solver	*Adam*
Épocas	50, 100, 150, 200, 250, 300, 350 e 400
Taxa de aprendizado	Constante

3.5. *SVM (Support Vector Machine)*

SVM (Support Vector Machine) é um tipo de Aprendizagem de Máquina supervisionado, indicado para conjuntos de dados com muitos atributos. O *SVM* cria hiperplanos que cortam o conjunto de dados que classificam as instâncias. Na Tabela 4 contém os parâmetros utilizados neste artigo.

Tabela 4. *SVM*

Parâmetros	Especificações
Kernel	*RBF (Radial Basis Function)*
Cost	*1, 5 e 10*
Gamma	*auto*

4. Validação dos métodos de classificação

Para a avaliação das classificações feitas, utilizou-se validação cruzada com 10 pastas e os seguintes indicadores para cada modelo: Acurácia, Precisão, *Recall*, *F1 Score* e o tempo de execução (*Wall time*). Todos estes indicadores derivam da Tabela Matriz de Confusão.

Tabela 5. Matriz de Confusão

	Normal	Ataque
Normal	Verdadeiro Positivo	Falso Positivo
Ataque	Falso Negativo	Verdadeiro Negativo

4.1. Acurácia

A Acurácia indica uma performance geral do modelo testado. Dentre todas as classificações feitas, quantas o modelo classificou corretamente.

$$\frac{VP + VN}{VP + VN + FP + FN} \tag{1}$$

4.2. Precisão

A Precisão informa dentre todas as classificações de classe Positivo que o modelo fez, quantas estão corretas.

$$\frac{VP}{VP + FP} \tag{2}$$

4.3. *Recall*

O *Recall* ou Sensibilidade, indica dentre todas as situações de classe Positivo como valor esperado, quantas estão corretas.

$$\frac{VP}{VP + FN} \tag{3}$$

4.4. *F1 Score*

O *F1 Score* é definido como duas vezes a média harmônica entre *Recall* e a Precisão, ou seja, é um equilíbrio entre as duas métricas anteriores.

$$2 \cdot \frac{Precisão \cdot Recall}{Precisão + Recall} \tag{4}$$

5. Resultados e Discussão

No que se refere ao tratamento da base de dados, foram removidos os atributos com índice de correlação acima de 0.8 e atributos com valores únicos ou com uma distinção muito pequena [Campos and Lima 2012]. Dentre os 42 atributos da base de dados NSL-KDD, foram removidos 14, para assim formar nossa nova base de dados redimensionada, NSL-KDD-R. A seguir estão os atributos que foram removidos: *land*, *num_root*, *num_outbound_cmds*, *is_host_login*, *is_guest_login*, *srv_count*, *serror_rate*, *srv_serror_rate*, *rerror_rate*, *srv_rerror_rate*, *same_srv_rate*, *dst_host_srv_count*, *dst_host_srv_serror_rate*, *dst_host_srv_rerror_rate*.

Os melhores parâmetros encontrados para cada um dos modelos de classificações estão na Tabela 5.

Tabela 6. Melhores parâmetros encontrados

Modelo	Base de dados	Parâmetros	Especificações
KNN	NSL-KDD	Vizinhos	1
		Distância	Manhattam
		Peso	Uniforme
	NSL-KDD-R	Vizinhos	1
		Distância	Manhattam
		Peso	Uniforme
Random Forest	NSL-KDD	Nº de Árvores	180
	NSL-KDD-R	Nº de Árvores	169
MLP	NSL-KDD	Função de ativação	*TanH*
		Solver	*Adam*
		Épocas	350
		Taxa de aprendizado	Constante
	NSL-KDD-R	Função de ativação	*TanH*
		Solver	*Adam*
		Épocas	400
		Taxa de aprendizado	Constante
SVM	NSL-KDD	*Kernel*	*RBF*
		Cost	5
		Gamma	*auto*
	NSL-KDD-R	*Kernel*	*RBF*
		Cost	5
		Gamma	*auto*

A Tabela 6 mostra as pontuações obtidas para as duas base de dados (NSL-KDD e NSL-KDD-R) nos modelos de Aprendizagem de Máquina e o tempo de execução com utilizando a mesma máquina para todos.

Tabela 7. Resultados das classificações

Modelo	Base de dados	Acurácia	Precisão	*F1 Score*	*Recall*	*Wall time*
KNN	NSL-KDD	0.9926	0.9931	0.9929	0.9927	17min 4s
	NSL-KDD-R	0.9922	0.9921	0.9925	0.9929	14min 20s
Random Forest	NSL-KDD	0.9952	0.9950	0.9954	0.9958	5min 36s
	NSL-KDD-R	0.9955	0.9952	0.9956	0.9961	4min 43s
Naive Bayes	NSL-KDD	0.8733	0.8466	0.8848	0.9291	7.7s
	NSL-KDD-R	0.8755	0.8611	0.8849	0.9131	5.51s
MLP	NSL-KDD	0.9734	0.9716	0.9747	0.9784	3min 56s
	NSL-KDD-R	0.9711	0.9664	0.9728	0.9800	2min 30s
SVM	NSL-KDD	0.9743	0.9571	0.9761	0.9963	14h 28min
	NSL-KDD-R	0.9783	0.9640	0.9796	0.9958	12h 50min

É possível observar na Tabela 7 que as variações nas pontuações dos indicadores são pequenas e favorecem a base de dados redimensionada (NSL-KDD-R) na maioria dos modelos. A base de dados original (NLS-KDD) apresenta um tempo de execução

superior em quatro dos cinco algoritmos de Aprendizagem de Máquina, ou seja, obteve-se uma melhora no custo computacional nesses quatro modelos. O *KNN* treinado com a base de dados original é superior na maioria dos indicadores e o tempo de execução não favorece a classificação em relação ao modelo treinado com a base de dados reduzida. Mostrando que mesmo quando não há uma melhora no custo computacional, pode haver um aumento na acurácia.

O modelo *Random Forest* com a base de dados reduzida tem resultados melhores quando comparado ao modelo que utiliza a base de dados original. O primeiro modelo contém 11 árvores a menos que o segundo, resultando em um menor tempo processamento, pode-se perceber que a acurácia é mantida e o tempo de execução é menor.

Na classificação feita com o *Naive Bayes*, pode-se observar que mesmo com uma inferência que desconsidera a correlação dos atributos, é possível nota uma diferença no tempo de execução entre as duas base de dados, favorecendo a NSL-KDD-R. Também há uma pequena diferença no que se refere aos indicadores que favorecem a base de dados não redimensionada, porém essa diferença é muito pequena.

O tempo de execução da *MLP* que utiliza a base de dados NSL-KDD tem uma diferença de 1 minuto e 26 segundos a mais em relação à que utiliza a base de dados NSL-KDD-R. E os indicadores favorecem levemente à primeira *MLP*. Porém, os resultados desses indicadores podem variar devido a geração aleatória dos pesos iniciais da rede, mas o tempo de execução da rede que contém a base de dados reduzida será menor que o tempo da rede que contém a base de dados original.

No *SVM*, levou-se 1 hora e 38 minutos a mais para o modelo que estava sendo treinado com o NSL-KDD concluir todo o processo. Isto significa que a redução feita na base de dados original tem um impacto muito grande no custo computacional quando se utiliza modelos de Aprendizagem de Máquina mais lentos, como é o caso do *SVM*. E a partir desses resultados apresentados, pôde-se observar que bases de dados com muitas dimensões, podem ser reduzidas e utilizadas em algum modelo de classificação e mesmo assim manter suas pontuações nas avaliações de desempenho do modelo.

6. Considerações Finais

Com base na acurácia dos resultados obtidos a partir dos métodos de classificação, observa-se que o *Random Forest* retornou um resultado mais satisfatório quanto à classificação em geral, pois este método é eficiente em bases de dados de altas dimensões e com atributos que não são altamente correlacionados. Logo se encaixa muito bem nos requisitos da base de dados reduzida (NSL-KDD-R), uma vez que foram retirados atributos de alta correlação. Temos que levar em consideração que em métodos de classificação onde a correlação entre dados não é fator determinante para os resultados, como o *Naive Bayes*, não houve uma grande diferença nas avaliações ao utilizar-se as duas bases de dados. Pôde-se observar também que ao realizar a retirada dos atributos, apesar de não ter uma melhoria em sua acurácia, tem-se uma diminuição em seu custo computacional, pois estão trabalhando com menos atributos que anteriormente, como observado no *Naive Bayes*.

Também é possível dizer que pode-se treinar um modelo de Aprendizagem de Máquina utilizando, com uma base de dados contendo dados de tráfego em Redes de

Computadores, para que possa-se implementá-lo em um *IDS* para que monitore e identifique comportamentos na rede que significam possíveis ameaças de forma autônoma.

Referências

Alkhalil, A. and Ramadan, R. A. (2017). Iot data provenance implementation challenges. *Procedia Computer Science*, 109:1134 – 1139.

Campos, L. M. L. and Lima, A. S. (2012). Sistema para detecção de intrusão em redes de computadores com uso de técnica de mineração de dados. In *V Congresso Tecnológico Infobrasil*, Fortaleza, Brasil.

Chouhan, N., Khan, A., and ur Rasheed Khan, H. (2019). Network anomaly detection using channel boosted and residual learning based deep convolutional neural network. *Applied Soft Computing*, 83:105612.

Farnaaz, N. and Jabbar, M. (2016). Random forest modeling for network intrusion detection system. *Procedia Computer Science*, 89:213 – 217.

Hajiheidari, S., Wakil, K., Badri, M., and Navimipour, N. J. (2019). Intrusion detection systems in the internet of things: A comprehensive investigation. *Computer Networks*, 160:165 – 191.

Hamzei, M. and Navimipour, N. J. (2018). Toward efficient service composition techniques in the internet of things. *IEEE Internet of Things Journal*, 5(5):3774–3787.

Morettin, P. A. and Bussab, W. d. O. (2010). *Estatística básica*. Saraiva, 6 edition.

Shenfield, A., Day, D., and Ayesh, A. (2018). Intelligent intrusion detection systems using artificial neural networks. *ICT Express*, 4(2):95 – 99.

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*.

Viegas, E. K., Santin, A. O., and Oliveira, L. S. (2017). Toward a reliable anomaly-based intrusion detection in real-world environments. *Computer Networks*, 127:200 – 216.

Zhang, J., Gardner, R., and Vukotic, I. (2019). Anomaly detection in wide area network meshes using two machine learning algorithms. *Future Generation Computer Systems*, 93:418 – 426.