ANÁLISE DO USO DE INTELIGÊNCIA ARTIFICIAL EM SISTEMAS DE DETECÇÃO DE INTRUSÃO

Diogo dos Santos Fernandes [1] Rogério Casagrande [2]

Resumo: A evolução da tecnologia têm não só aumentado a qualidade de vida em geral como também proporcionado falhas na segurança do usuário comum. Essa pesquisa tem o intuito de demonstrar a utilização de algoritmos de inteligência artificial para a proteção da rede contra ataques maliciosos. Nela são utilizados algoritmos de aprendizado de máquina, bem como diferentes modelos de ataques com o intuito de obter maior número de resultados. Foram utilizados três algoritmos com dois grupos de ataques. Os resultados obtidos mostraram que a aplicação de diferentes algoritmos de inteligência artificial para sistemas de detecção de intrusão podem sim melhorar a segurança das redes.

Palavras-chave: inteligência artificial; algoritmos; aprendizado de máquina; detecção de intrusão.

ABSTRACT: The evolution of technology has not only increased the quality of life in general but also caused security gaps for the common user. This research aims to demonstrate the use of artificial intelligence algorithms to protect the network against malicious attacks. It uses machine learning algorithms, as well as different attack models in order to obtain a greater number of results. Three algorithms were used with two groups of attacks. The results obtained showed that the application of different artificial intelligence algorithms for intrusion detection systems can improve network security.

Keywords: artificial intelligence; algorithms; machine learning; intrusion detection.

1 INTRODUÇÃO

O rápido avanço da tecnologia nos últimos anos não só vem sendo o principal vetor para uma melhora da qualidade de vida, como também uma forma de expor o usuário a diversos problemas de segurança (Saranya et al., 2020).

[1] sfernandes.diogo@unesc.net
[2] roc@unesc.net

Combater ameaças de intrusão à rede é um campo exigente de pesquisa na área da segurança. O principal problema de estudo em sistemas de detecção de intrusão, que tem recebido cada vez mais atenção, é otimizar a sua eficiência (Gautam; Doegar, 2018).

O *World Internet Statistics* informa que, o crescimento da internet (2000-2019) atingiu 1.114%, já que mais de 2 quintilhões de bytes de dados são gerados todos os dias. Isso mostra que a taxa de crescimento de dados de várias fontes são extremamente rápidas e, ao mesmo tempo, o desenvolvimento de metodologias e ferramentas de *hacking* também crescem dessa mesma forma. Portanto, há uma necessidade de segurança da informação e análise de dados para protegê-los de intrusão. Devido ao grande volume e alta velocidade dos dados, sistemas de detecção tradicionais não são capazes de detectar intrusões de maneira rápida (Saranya et al., 2020).

A proteção dos dados e da infraestrutura de rede permanecem como uma preocupação primordial para as organizações modernas, devido ao aumento de ameaças que visam infiltrar sistemas, roubar dados sensíveis, ou causar distúrbios operacionais. Estas ameaças sublinham a importância de Sistemas de Detecção de Intrusão (SDIs) robustos, os quais funcionam para identificar e mitigar atividades não autorizadas e potencialmente prejudiciais dentro de uma rede (Sudha et al., 2023).

A detecção de intrusão é um processo que monitora e analisa o tráfego em uma rede ou computador em busca de sinais de intrusão. O objetivos principais das SDIs podem ser resumidos como: a) Monitoramento de hospedeiros e redes; b) Análise de comportamentos de redes de computadores; c) Geração de alertas; d) Respostas à comportamentos suspeitos (Abdallah; Eleisah; Otoom, 2022).

Existem diversos tipos de técnicas de detecção de intrusão, como por exemplo: detecção baseada em host e detecção baseada em rede. As baseadas em host são implantadas em um host individual e empregadas para monitorar o fluxo, entrada e saída de dados e comparar resultados com imagens pré-geradas da atividade do host. Detecção de intrusão baseada em rede examina o tráfego e monitora múltiplos hosts na rede para detectar qualquer atividade suspeita. Elas tentam identificar estas atividades analisando as camadas de rede, transporte, aplicação e hardware pelo tráfego capturado da rede (Saranya et al., 2020).

Muitas pesquisas propuseram algoritmos de aprendizado de máquina (*machine learning*) para sistemas de detecção de intrusão para redu-

zir possibilidades de resultados falso-negativos e produzir SDIs mais precisas. Entretanto, para lidar com *Big Data* (grande volume de dados), técnicas tradicionais de *machine learning* levam muito tempo para aprender e classificar esses dados. Utilizando técnicas de *Big Data* e *machine learning* para SDIs pode resolver diversos desafios, como a velocidade da aplicação, tempo de computação e o desenvolvimento de SDIs mais precisas (Othman et al., 2018).

Machine Learning (ML) é um subconjunto da inteligência artificial. ML faz o sistema aprender e melhorar sua habilidade automática da experiência sem ser explicitamente programada. Para sistemas de detecção de intrusão, algoritmos de *machine learning* funcionam mais precisamente para detectar ataques em grandes quantidades de dados em menos tempo (Saranya et al., 2020).

Este trabalho tem como objetivo apresentar os benefícios da aplicação de algoritmos de *machine learning* em sistemas de detecção de intrusão, visto que conforme a tecnologia avança, a quantidade de dados que trafegam por segundo tende a aumentar exponencialmente, dificultando a capacidade de identificação de anomalias e ataques em sistemas.

2 Materiais e Métodos

Essa pesquisa caracteriza-se por ser aplicada e de base tecnológica. Nela, foram treinados e aplicados três diferentes modelos de inteligência artificial, sendo eles, *Logistic Regression*, *Random Forest* e *Decision Tree*, a partir de cinco arquivos CSV (valores separados por vírgulas ou ponto e vírgulas) divididos e trabalhados em dois grupos diferentes de ataques originados do *dataset* CSE-CIC-IDS2018 (Brunswick, 2018), sendo um deles composto por *FTP-BruteForce*, *SSH-BruteForce*, *DoS attacks-GoldenEye*, *DoS attacks-Slowloris*, *DoS attacks-SlowHTTPTest* e *DoS attacks-Hulk*, e o outro composto por *Bot* e *Infiltration*.

2.1 Google Colab

Para a realização dos treinamentos dos modelos foi utilizado o Google Colab Pro (Google Research, 2023), que permite ao usuário conexão com diferentes máquinas virtuais provenientes do Google Compute Engine. Para esta pesquisa foi utilizado o acelerador de hardware TPU v2, que no momento dos testes, forneceu aproximadamente 330GB de memória *RAM*.

Toda a pesquisa e testes foram feitos em um ambiente de exe-

cução que roda Python 3, juntamente com as bibliotecas *pandas*, *numpy* e *joblib* para auxílio. Além destas, também foi utilizada a biblioteca *sklearn*, responsável por fornecer os modelos de inteligência artificial que foram aplicados, além de ser a responsável pela obtenção dos parâmetros avaliativos.

2.2 Preparação do Dataset

Para a realização dos testes, foram utilizados cinco arquivos CSV distintos. Para facilitar os testes, eles foram divididos em grupos de dois e três arquivos e unidos em dois arquivos separados.

Eles foram organizados ambos da mesma forma, afim de mantê-los o mais limpos possível para que não tenham valores irregulares influenciando no resultado final.

2.3 Modelos de Inteligência Artificial

A pesquisa conta com três diferentes modelos da biblioteca *sklearn*, sendo eles: *Decision Tree Classifier*; *Random Forest Classifier*; e *Logistic Regression*; todos treinados e testados com os mesmos arquivos previamente preparados e no mesmo acelerador de hardware TPU v2. Os parâmetros utilizados para avaliar a possibilidade de se aplicar algum destes modelos em sistemas de detecção de intrusão são o tempo decorrido em comparação com a quantidade de registros, a precisão do modelo e a matriz de confusão.

2.3.1 Decision Tree Classifier

Decision Trees são um método de aprendizagem supervisionado e não paramétrico usado para classificação e regressão. O objetivo é criar um modelo que preveja o valor de uma variável alvo, aprendendo regras de decisão simples inferidas a partir dos recursos dos dados. Uma árvore pode ser vista como uma aproximação constante por partes (Scikit-learn developers, 2024).

2.3.2 Random Forest Classifier

O método *Random Forest* (RF) é introduzido como um dos conjuntos de algoritmos de aprendizado de máquina, visto que diversas árvores de decisão são construídas quando se treina o modelo, e os resultados das predições de todas as árvores são coletados para se obter o resultado final. Este é um dos métodos complexos não lineares supervisionados de aprendizagem de máquina usados para regressão e classificação (Samawi;

Yousif; Al-Saidi, 2022).

2.3.3 Logistic Regression

Este método é um algoritmo de classificação de aprendizado de máquina supervisionado usado para observar o conjunto discreto de classes. A função logística faz uso da função de custo que é chamada de função *sigmoid*. Esta função mapeia predições e probabilidades (Saranya et al., 2020). Ajustando os dados à função logística, a probabilidade de ocorrência do evento pode ser prevista (Belavagi; Muniyal, 2016).

3 RESULTADOS

A proposta dos testes envolvidos foi parcialmente satisfatório. Levando em conta o primeiro grupo de ataques, todos os modelos apresentaram uma precisão de predições superior a 99%. Já no segundo grupo de ataques, a precisão varia de 92% a 94%. O primeiro grupo é composto da maior quantidade de linhas no total e foi dividido em treino, que foi realizado com 70% do arquivo, para criar um modelo de maior precisão, e na parte de teste com os 30% restantes, para que se possa verificar o treinamento com dados provenientes de uma mesma base dede dados, como consta na tabela 1 a seguir.

Tabela 1 – Primeiro grupo de arquivos

	Linhas Benignas	Linhas Maliciosas	Linhas Totais
Arquivo Inteiro	1.662.717	290.951	1.953.668
Arquivo Treino	1.163.900	203.667	1.367.567
Arquivo Teste	498.817	87.284	586.101

Fonte: Elaborado pelo autor.

Já o segundo grupo é composto pelo menor número de linhas no total e foi dividido em treino, que foi realizado da mesma forma que o primeiro grupo. Consequentemente, a parte de testes possui a mesma proporção, sendo 30% do arquivo completo, como mostrado na tabela 2 seguinte.

Tabela 2 – Segundo grupo de arquivos

	Linhas Benignas	Linhas Maliciosas	Linhas Totais
Arquivo Inteiro	802.592	206.664	1.009.256
Arquivo Treino	561.865	144.614	706.479
Arquivo Teste	240.727	62.050	302.777

Fonte: Elaborado pelo autor.

3.1 Decision Tree Classifier

Este modelo, no grupo 1 de arquivos, apresentou uma precisão de aproximadamente 99,993%, sendo o maior dentre os três modelos. Seu treinamento durou em torno de 21 minutos. O modelo recebeu hiper-parâmetros, limitando a profundidade da árvore de decisões. Na matriz de confusão gerada, nota-se que houveram 498.796 verdadeiros positivos, 21 falsos negativos, 19 falsos positivos e 87.265 verdadeiros negativos, como mostrado na tabela 3 a seguir.

Tabela 3 – Matriz de Confusão Grupo 1 Decision Tree Classifier

Valor Real /	Valor predito	
	Sim	Não
Sim	498.796	21
Não	19	87.265

Fonte: Elaborado pelo autor.

Já no segundo grupo de arquivos, foi registrada uma precisão de aproximadamente 94,261%, sendo também o maior dentre os três modelos. Seu treinamento durou em torno de 12 minutos. O modelo recebeu os mesmos hiper-parâmetros. Na matriz de confusão gerada, nota-se que houveram 240.157 verdadeiros positivos, 570 falsos negativos, 16.728 falsos positivos e 45.322 verdadeiros negativos, como informado na tabela 4 a seguir.

Tabela 4 – Matriz de Confusão Grupo 2 Decision Tree Classifier

Valor Real /	Valor predito	
	Sim	Não
Sim	240.157	570
Não	16.728	45.322

Fonte: Elaborado pelo autor.

3.2 Random Forest Classifier

Este modelo, no grupo 1 de arquivos, apresentou uma precisão de aproximadamente 99,990%. Seu treinamento durou em média de 14 minutos. O modelo recebeu hiper-parâmetros que simbolizam a quantidade de árvores no modelo. Na matriz de confusão gerada, nota-se que houveram 498.780 verdadeiros positivos, 37 falsos negativos, 19 falsos positivos e 87.265 verdadeiros negativos, como mostrado na tabela 5 a seguir.

Tabela 5 – Matriz de Confusão Grupo 1 Random Forest Classifier

Valor Real /	Valor predito	
	Sim	Não
Sim	498.780	37
Não	19	87.265

Fonte: Elaborado pelo autor.

Este modelo demonstrou resultados muito próximos ao modelo *Decision Tree Classifier*, tendo apenas 16 falsos negativos a mais. Essa diferença pode ter acontecido por conta da simplicidade com o qual foi tratado o modelo, visto que o mesmo representa uma versão mais complexa do próprio *Decision Tree Classifier*, que apresentou os melhores resultados da pesquisa.

Já no segundo grupo de arquivos, foi registrada uma precisão de aproximadamente 92,986%. Seu treinamento durou em torno de 12 minutos. O modelo recebeu os mesmos hiper-parâmetros. Na matriz de confusão gerada, nota-se que houveram 234.241 verdadeiros positivos, 6.486 falsos negativos, 14.748 falsos positivos e 47.302 verdadeiros negativos, como consta na tabela 6 seguinte.

Tabela 6 – Matriz de Confusão Grupo 2 Random Forest Classifier

Valor Real /	Valor predito	
	Sim	Não
Sim	234.241	6.486
Não	14.748	47.302

Fonte: Elaborado pelo autor.

3.3 Logistic Regression

Este modelo, no grupo 1 de arquivos, apresentou uma precisão de aproximadamente 99,532%. Seu treinamento durou em média 33 minutos. O modelo recebeu hiper-parâmetros para regularização e para testar

qual o melhor *solver*, no caso do teste, foi o *liblinear*. Na matriz de confusão gerada, nota-se que houveram 496.882 verdadeiros positivos, 1.935 falsos negativos, 751 falsos positivos e 86.533 verdadeiros negativos, como mostrado na tabela 7 a seguir.

Tabela 7 – Matriz de Confusão Grupo 1 Logistic Regression

Valor Real /	Valor predito	
	Sim	Não
Sim	496.882	1.935
Não	751	86.533

Fonte: Elaborado pelo autor.

Já no segundo grupo de arquivos, foi registrada uma precisão de aproximadamente 93,363%. Seu treinamento durou em torno de 21 minutos. O modelo recebeu os mesmos hiper-parâmetros para regularização e para testar qual o melhor *solver*, no caso do teste, foi o *saga*. Na matriz de confusão gerada, nota-se que houveram 238.411 verdadeiros positivos, 2.316 falsos negativos, 17.778 falsos positivos e 44.272 verdadeiros negativos, como consta na tabela 8 a seguir.

Tabela 8 – Matriz de Confusão Grupo 2 Logistic Regression

Valor Real /	Valor predito	
	Sim	Não
Sim	238.411	2.316
Não	17.778	44.272

Fonte: Elaborado pelo autor.

4 CONCLUSÃO

Este trabalho buscou treinar alguns modelos de inteligência artificial e colocá-las à prova contra ataques artificiais provenientes do *dataset* CSE-CIC-IDS2018, como forma de simular a função de um sistema de detecção de intrusão. Os ataques estavam separados em diferentes arquivos CSV, que foram unidos em dois grupos diferentes, com o objetivo de verificar a dificuldade que os algoritmos de aprendizado de máquina teriam em diferentes cenários.

O primeiro grupo de arquivos, composto por ataques do tipo *FTP-BruteForce*, *SSH-BruteForce*, *DoS attacks-GoldenEye*, *DoS attacks-Slowloris*, *DoS attacks-SlowHTTPTest* e *DoS attacks-Hulk*, apresentou resultados de maior precisão em todos os modelos, tendo o menor número de

erros registrados, já o segundo grupo, composto por ataques de *Bot* e *Infiltration*, apresentaram maiores dificuldades para a previsão dos algoritmos. Os testes do primeiro grupo foram mais demorados porém mais precisos, enquanto o segundo grupo apresentou treinamentos e testes mais rápidos porém menos precisos, devido a sua diferença de ataques e número de linhas, contando com aproximadamente metade do tamanho do primeiro grupo.

Todos os testes obtiveram resultados de precisão pós-treinamento superior a 92%, sendo o melhor dos resultados 99,993% em suas previsões. Os testes levaram tempos variados, indo de 12 a 33 minutos de duração para a leitura de arquivos de mais de um milhão de linhas diversas vezes, visto a utilização dos hiper-parâmetros. Foram usados modelos com base em classificação e regressão, sendo o modelo de classificação com base em árvores de decisão o mais rápido e preciso dentre os testes feitos. Com base nos conhecimentos adquiridos, bem como nos resultados obtidos, propõe-se para futuros trabalhos: aumentar a abrangência de algoritmos de aprendizado de máquina; trabalhar com diferentes hiper-parâmetros; utilizar diferentes grupos de ataques, bem como diferentes *datasets*.

REFERÊNCIAS

ABDALLAH, E. E.; ELEISAH, W.; OTOOM, A. F. Intrusion detection systems using supervised machine learning techniques: A survey. *Procedia Computer Science*, Elsevier BV, v. 201, p. 205–212. Disponível em: <https://doi.org/10.1016/j.procs.2022.03.029>.

BELAVAGI, M. C.; MUNIYAL, B. Performance evaluation of supervised machine learning algorithms for intrusion detection. *Procedia Computer Science*, Elsevier BV, v. 89, p. 117–123. Disponível em: <https://doi.org/10.1016/j.procs.2016.06.016>.

BRUNSWICK, U. of N. *CICIDS 2018 Dataset*. 2018. <https://www.unb.ca/cic/datasets/ids-2018.html>. Accessed: 2024-06-03. Disponível em: <https://www.unb.ca/cic/datasets/ids-2018.html>.

GAUTAM, R. K. S.; DOEGAR, E. A. An ensemble approach for intrusion detection system using machine learning algorithms. In: *2018 8th International Conference on Cloud Computing, Data Science Engineering (Confluence)*. [S.l.: s.n.], 2018. p. 14–15.

Google Research. *Google Colaboratory*. 2023. Accessed: 2024-06-03. Disponível em: <https://colab.research.google.com>.

OTHMAN, S. M. et al. Intrusion detection model using machine learning algorithm on big data environment. *Journal of Big Data*, Springer Science and Business Media LLC, v. 5, n. 1. Disponível em: <https://doi.org/10.1186/s40537-018-0145-4>.

SAMAWI, V. W.; YOUSIF, S. A.; AL-SAIDI, N. M. G. Intrusion detection system: An automatic machine learning algorithms using auto- WEKA. In: *2022 IEEE 13th Control and System Graduate Research Colloquium (ICSGRC)*. IEEE, 2022. Disponível em: <https://doi.org/10.1109/icsgrc55096.2022.9845166>.

SARANYA, T. et al. Performance analysis of machine learning algorithms in intrusion detection system: A review. *Procedia Computer Science*, Elsevier BV, v. 171, p. 1251–1260. Disponível em: <https://doi.org/10.1016/j.procs.2020.04.133>.

Scikit-learn developers. *Decision Trees — scikit-learn 1.5.0 documentation*. 2024. <https://scikit-learn.org/stable/modules/tree.html#tree>. Accessed: 2024-05-30.

SUDHA, M. et al. Optimizing intrusion detection systems using parallel metric learning. *Computers and Electrical Engineering*, Elsevier BV, v. 110, p. 108869. Disponível em: <https://doi.org/10.1016/j.compeleceng.2023.108869>.