Defesa Inteligente: Detectando Ataques em Rede de Computadores com Aprendizado de Máquina

Wellington Resende de Araújo Júnior[1], Gabriel Vinicios Moreira Fernandes[2], Agnaldo José da Rocha Reis[1]
[1]Universidade Federal de Ouro Preto, Ouro Preto, Brasil
[2]Itaipu Parquetec, Foz do Iguaçu, Brasil
wellington.araujo@aluno.ufop.edu.br, gabrielvinicios02@gmail.com, reis@ufop.edu.br

Abstract—One proposes in this study an approach to detecting intrusions in computer networks through machine learning techniques, applying classification models, including decision trees, random forests and artificial neural networks, to the CICIDS2017 real dataset. The main objective is to identify and classify different types of cyber attacks, such as DoS, Web Attacks and infiltrations, by analyzing network traffic via Machine Learning models. The methodology employed comprises data pre-processing steps, selection of relevant attributes and model training using specific libraries, such as TensorFlow. The results obtained indicate that the DT model achieved superior performance levels, demonstrating high accuracy and recall close to 1, while the remaining models did not perform as good as the DT-based models.

Index Terms—Cybersecurity, Artificial Neural Networks, Intrusion Detection, Machine Learning, Classification Models

I. Introdução

Nos últimos anos, foi possível testemunhar uma acelerada transformação digital em nossa sociedade. As formas de se comunicar e armazenar informações mudaram drasticamente. Com isso, empresas e governos passaram a se preocupar com a segurança das informações que circulam no meio digital, tornando a cibersegurança uma pauta muito necessária para as nações [1]. Desde o primeiro vírus desenvolvido por [2], as ameaças cibernéticas ganharam cada vez mais força, com evoluções crescentes de diversidade e complexidade.

Os vírus são acoplados a programas ou arquivos seguros e se replicam, podendo causar danos como perda de dados ou corromper o sistema operacional por completo. Por outro lado, os worms são malwares com alta capacidade de proliferação em redes, infectando diversas máquinas. Na maioria dos casos, os worms são utilizados para instalar bots e criar um exército para realizar outros tipos de ataque, como os de Negação de Serviço (DoS). Já os ransomwares são ataques que visam extorquir as vítimas. Na maioria dos casos, esse tipo de ameaça criptografa todos os arquivos da máquina infectada e exige um pagamento para que seja possível recuperar os arquivos. Todas essas ameaças podem causar danos irreversíveis para pessoas e organizações, levando a enormes perdas financeiras e indisponibilidade de serviços [3].

O uso de ferramentas de inteligência artificial para combater essas ameaças vem ganhando força à medida que temos mais recursos computacionais e ferramentas de fácil implementação. É neste contexto que se insere este trabalho. Propõe-se aqui uma abordagem para detectar intrusões em redes de computadores via Aprendizado de Máquina. Para tanto, três modelos distintos foram desenvolvidos e tiveram os resultados comparados.

Por fim, este artigo está organizado como se segue. Na Seção II, Revisão da Literatura, os principais trabalhos consultados são brevemente descritos. Já na Seção III, apresenta-se a Metodologia empregada. Os resultados obtidos são analisados em detalhes na Seção IV e as principais conclusões e sugestões para trabalhos futuros são assunto na Seção IV. O artigo é concluído com as referências bibliográficas.

II. Revisão da Literatura

Em [4] é abordado o problema de detecção de ameaças na rede através da construção de traços de comportamento para criar bases de comportamento regular do sistema, e utilizam de Alocação Latente de Dirichlet (LDA) para criar modelos estatísticos generativos que permitem encontrar relações entre dados observados, mencionando também o uso de inteligência sobre os dados para detecção de ameaças. Além disso, cita sobre a dificuldade de trabalhar com registros de texto não estruturado para identificação de ataques em um sistema crítico e a participação importante de recursos cognitivos humanos na proteção de computadores.

Em [5], é apresentado uma solução de detecção de invasões em sistemas Security Information and Event Management (SIEM) majoritários, baseados em confiabilidade. O modelo proposto combina diversas redes neurais feedfoward simples como weak learners, proporcionando boa capacidade de acerto utilizando poucos recursos computacionais. A pesquisa também abordou o desafio em minimizar a necessidade de recursos de processamento exigidos por sistemas desse tipo, além da exigência de técnicas de pré-processamento adequadas para cada contexto. Ao fim, o modelo se provou bastante eficaz e com alta precisão, sendo uma solução promissora para detecção de invasões.

No trabalho de [6], apresenta-se um sistema que utiliza técnicas de aprendizado de máquina como Naive Bayes e

KNN, sendo o primeiro para classificação e o segundo para agrupamento, a fim de identificar e classificar os ataques cibernéticos. A pesquisa também trás a necessidade de generalizar os algoritmos de detecção baseando-se na coleta de recursos e, por fim, ressalta o potencial do aprendizado de máquina para essa tarefa.

Entre os modelos apresentados no trabalho de [6], o que mais se destaca em relação a abordagem do presente trabalho seria o modelo de Naive Bayes, que realiza classificação dos diversos tipos de ataques. O modelo apresentou ótimos resultados entre os 3 tipos de ataques analisados (1).

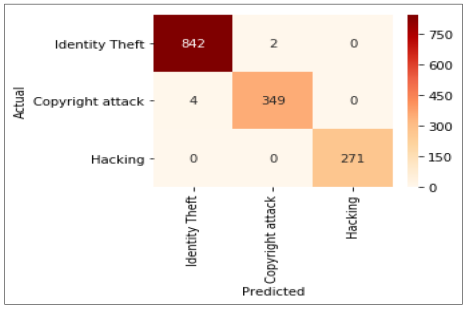

Fig. 1. Matriz de confusão do modelo Naive Bayes de acordo com o tipo de ataque. Fonte: [6]

Em [6] também é abordado um detalhamento de métricas similar ao utilizado no presente trabalho, onde foram calculados F1-Score, precisão e recall (2).

	precision	recall	f1-score	support
Identity Theft	1.00	0.99	0.99	241
Copyright attack	0.97	1.00	0.98	95
Hacking	1.00	1.00	1.00	84
avg / total	0.99	0.99	0.99	420

Fig. 2. Tabela de métricas do modelo de Naive Bayes de acordo com o tipo de ataque. Fonte: [6]

Além do modelo de classificação principal também foram desenvolvidos outros 4 modelos a efeito de comparação, entre eles o que apresentou melhor acurácia foi a Regressão Logística com 99,38%, em contra partida o modelo Random Forest apresentou 80,69%, sendo o pior método, como mostra a figura 3.

```
model_name
LinearSVC                0.992371
LogisticRegression       0.993803
MultinomialNB            0.989516
RandomForestClassifier   0.806923
Name: accuracy, dtype: float64
```

Fig. 3. Taxa de acurácia para cada modelo. Fonte: [6]

No trabalho proposto por [7], fez-se uso da base de dados CICIDS 2017, e [7] sugeriu algumas tratativas de dados para melhorar a eficiência dos modelos de aprendizado, como seleção das melhores variáveis para reduzir a quantidade de argumentos, tratativas para dados faltantes e, por fim, apresentou o desempenho de alguns tipos de modelos para predição dos dados do conjunto.

Neste trabalho, um modelo de RNA foi treinado com 500 épocas, o qual obteve uma acurácia de 96,53%, utilizando como fonte de dados a base CICIDS-2017, já mencionada. O mesmo modelo, treinado com 50 épocas apresentou acurácia de 87,79%. O estudo também trouxe uma visão detalhada do resultado do modelo de acordo com o tipo de ataque, onde foi possível notar uma maior dificuldade em detectar corretamente ataques do tipo DoS Hulk, SQL Injection e Heartbleed, nesta ordem. Por fim, os autores realizaram um comparativo com o método de Random Forest, que apresentou acurácia geral de 96,24%, e se percebeu um desempenho mais consistente deste tipo de modelo para os diferentes ataques presentes na base.

A base CICIDS-2017 tem sido amplamente empregada em estudos de detecção de ameaças cibernéticas por meio de técnicas de machine learning, sendo aplicada em conjunto com diversos algoritmos. Em Fathima et al., foi proposto um sistema de detecção de intrusão baseado em autoencoders empilhados em duas etapas, combinados com Random Forest, alcançando zero falso positivo e alta acurácia a partir da base CICIDS-2017 [8]. Em outra linha, Yin et al. utilizaram clustering como etapa de pré-agrupamento, adicionando pseudorótulos à base antes de treinar um MLP, obtendo precisão de 99,73% em classificação multiclasse [9]. Em Soltani et al. desenvolveram um método de detecção profunda que incorpora conteúdo bruto de pacotes via LSTM, alcançando precisão igual ou superior a 99% com a base de dados do CICIDS-2017 [10].

No trabalho de Jairu et al. Foram avaliados diversos classificadores supervisionados (como Random Forest, SVM, KNN e Naïve Bayes), selecionando os parâmetros mais relevantes por meio da correlação de Pearson e observando que o Random Forest apresentou desempenho superior com acurácia de até 99,93% [11]. Em síntese, estes trabalhos ilustram uma diversidade de metodologias baseadas em aprendizado profundo, clustering, técnicas clássicas de ensemble e seleção de atributos, todos fazendo uso da base CICIDS-2017. Tais características evidenciam o seu uso como uma base confiável para detecção de intrusão em conjunto com a aplicação de machine learning.

Em [12] foi proposto a integração de um SIEM e um sistema de detecção de intrusão que utilizava aprendizado de máquina por meio de Support Vector Machine para detecção de ataques baseada em análises em tempo real dos dados de rede. O sistema foi construído com ferramentas open-source e preparado para aplicações industriais. O mesmo possui monitoramento em tempo real e envio de alertas para os times responsáveis quando alguma anomalia era detectada.

III. Metodologia

Inicialmente, foi necessário definir qual seria o conjunto de dados utilizado para construção dos modelos propostos. Para isso, foi escolhido o conjunto de dados CICIDS2017 para treinamento e teste dos modelos de classificação utilizados no trabalho. Este dataset foi desenvolvido por [13] e forma uma série de estudos relacionados a cibersegurança da Universidade de New Brunswick. Os dados que formam o conjunto foram obtidos a partir da captura de tráfego de rede de diversos computadores durante 5 dias. Foram capturados e classificados tráfego benigno e mais 14 tipos de ataques, sendo eles: Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet e DDoS, resultando em 2,83 milhões de pacotes capturados com 80 parâmetros registrados. Desta forma, a base CICIDS2017 foi escolhida por sua robustez e variedade.

Definido o conjunto de dados, o próximo passo foi realizar a análise exploratória dos dados e a escolha das variáveis para treinamento dos modelos, visto que a base possui mais de 70 informações sobre o tráfego, e muitas dessas não seriam relevantes na predição do tipo de tráfego. Com isso em mente, foram selecionadas as 10 principais variáveis do tráfego conforme [7], que realizou um estudo de quais variáveis apresentavam maior relevância nos resultados de classificação. Desse modo, foram selecionados os seguintes atributos:

1) Init Window Bytes Forward: Tamanho médio das janelas de pacotes iniciais enviados;
2) ACK Flag Count: Flag ACK presente no pacote;
3) Forward Packets per second: Quantidade de pacotes enviados por segundo;
4) Flow Packets per second: Quantidade de pacotes entre dois pontos por segundo, independente de sentido;
5) Flow IAT Max: A maior diferença de tempo entre pacotes trocados por dois pontos;
6) Flow IAT Min: A menor diferença de tempo entre pacotes trocados por dois pontos;
7) Flow duration: A diferença de tempo entre o primeiro e o último pacote trocado entre dois pontos;
8) Init Window Bytes Backward: Tamanho médio das janelas de pacotes iniciais recebidos;
9) Subflow Backward Bytes: Quantidade total de bytes recebidos;
10) Flow IAT Mean: A diferença de tempo média entre pacotes trocados por dois pontos.

Além da definição dos atributos utilizados no treinamento, também se decidiu por tornar os rótulos binários, ou seja, manter apenas as classes 'BENIGN' e 'MALIGN', convertendo todos os rótulos de ataque para o último mencionado. Ainda na tratativa dos dados, notou-se que existiam dados com valores infinitos e nulos, os primeiros foram convertidos para nulos. Os quais tiveram suas respectivas linhas removidas do conjunto, tratando-os como outliers. Para encerrar as manipulações, todos as colunas de atributos foram normalizadas e o conjunto de dados foi dividido entre treinamento e teste, respeitando a proporção de 60% dos dados para treinamento e o restante para testes. Todos os 4 datasets resultantes foram convertidos em arquivos Valores Separados por Vírgula (CSV) para consumo nas seguintes etapas.

A. Treinamento dos modelos de Decision Tree e Random Forest

Ao realizar o treinamento, o primeiro passo é importar os dados dos arquivos CSV gerados na etapa anterior. Para tanto foi utilizada a biblioteca Pandas [14] do Python [15] e os dados foram armazenados em 4 variáveis do tipo dataframe, denominadas por: dtX_train, dtX_test, dtY_train e dtY_test. De posse desses dados, se iniciou o desenvolvimento dos primeiros modelos de classificação: Decision Tree e Random Forest. Ambos os classificadores foram desenvolvidos com a biblioteca Scikit-Learning [16] na linguagem Python, e sua escolha se deu pela facilidade de implementação e pela grande documentação disponível para consulta.

O modelo Decision Tree foi implementado utilizando o módulo DecisionTreeClassifier da biblioteca, que disponibiliza um modelo de árvore de decisão previamente configurado, sendo necessário apenas realizar as funções de treinamento e validação, já que optou-se por manter os parâmetros padrões do modelo.

Desta forma, com o modelo já criado, realizou-se o treinamento utilizando os dados dos conjuntos dtX_train e dtY_train. Após treinado fez-se a predição dos dados contidos no dtX_test, resultando no conjunto de rótulos que foi comparado com o dtY_test, obtendo assim, as métricas do modelo. Para tanto, utilizou-se alguns módulos do pacote sklearn.metrics, o accuracy_score e o confusion_matrix, que retornam a acurácia e a matriz de confusão (que por si só possui diversas métricas incorporadas). Para concluir utilizaram-se as bibliotecas Seaborn [17] e Matplotlib [18] para obter a figura da matriz de confusão do modelo.

A metodologia para o modelo de Random Forest é similar. Primeiramente, fez-se a importação do módulo RandomForestClassifier do sklearn. Porém, para este os treinamentos foram realizados variando o parâmetro max_depth entre os valores 2, 3, 4, 5, 8 e 12, escolhidos de forma empírica. Este atributo define a profundidade máxima de cada árvore que será criada na floresta randômica. Os demais passos da criação da Random Forest são idênticos aos descritos no modelo de Decision Tree: Realizou-se o treinamento dos modelos utilizando os conjuntos de dados dtX_train e dtY_train. Com o modelo obtido, realizamos os testes de desempenho utilizando os conjuntos dtX_test e dtY_test. A partir das predições realizadas, obteve-se as métricas de desempenho à partir da matriz de confusão.

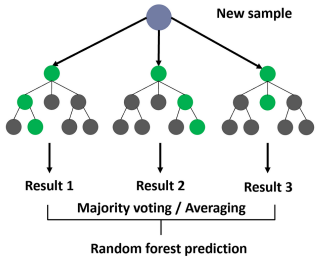

Fig. 4. Representação gráfica do algoritmo de Random Forest. Fonte: [19]

B. Treinamento das Redes Neurais Artificiais

As Redes Neurais Artificias foram desenvolvidas utilizando, assim como nos modelos já mencionados, a linguagem Python. Porém, neste caso, utilizou-se o pacote Tensorflow [20] para construir os modelos. Ao utilizar essa biblioteca, exige-se uma etapa de pré-processamento adicional. Deve-se fazer a importação dos 4 conjuntos de dados e em seguida a transformação dos dataframes Pandas obtido para uma estrutura de vetores Numpy [21] utilizando a função to_numpy. A transformação deverá ser aplicada nos quatro conjuntos de dados importados.

Na etapa de treinamento das redes neurais foram avaliadas 6 diferentes composições de redes, para que se possa comparar os resultados e encontrar a melhor escolha de arquitetura da rede. Para tanto, foram utilizadas redes com 10 parâmetros de entrada, com variações de 128, 64, 32 e 16 nós na camada oculta, com valores escolhidos de forma empírica.

Feito todo o pré-processamento necessário, iniciou-se o desenvolvimento da rede neural. O primeiro passo é definir a estrutura de camadas. Para tal finalidade, utilizou-se o módulo keras.layers embarcado no Tensorflow. Começando pela camada de entrada, criou-se um layer do tipo Input com a quantidade de nós igual a de parâmetros de entrada da rede (10), que equivale a quantidade de atributos que nosso modelo receberá para usar nas predições. Logo, criou-se um nó receptor para cada um deles.

Em seguida, definiu-se uma única camada oculta utilizando um layer do tipo Dense com a variação da quantidade de nós já mencionada e função de ativação 'Relu'. A camada de saída foi a última a ser definida e também utiliza o tipo Dense. Porém, para a camada de saída teremos 2 nós, respeitando a quantidade de possíveis rótulos do conjunto de dados. Para ativação desta camada foi selecionada a função 'Softmax', que retorna como valor de saída de cada nó a probabilidade de que aquele rótulo seja o resultado verdadeiro.

Com todas as camadas bem definidas é feita a compilação do modelo que define alguns parâmetros em relação ao treinamento. Nesses casos, os parâmetros foram definidos da seguinte forma: optimizer igual a 'adam', que utiliza o algoritmo de Adam para fazer a convergência da rede, loss igual a 'sparse_categorical_crossentropy', que calcula a entropia cruzada entre rótulos e predições e, por fim, metrics como 'accuracy'.

Em seguida foi definido um Early Stopping para o treinamento da rede. Esse recurso permite que a rede interrompa o treinamento caso as perdas aumentem consecutivamente e restaura a época que apresentou o melhor desempenho no treinamento. O Early Stopping foi implementado utilizando o módulo callbacks do keras e utiliza acurácia como métrica analisada para avaliar a rede treinada, permitindo até 5 pioras consecutivas no desempenho antes de interromper o processo. Caso não aconteça nenhuma interrupção, o treinamento acontece até o número máximo de épocas definido.

O próximo passo é fazer o treinamento do modelo. Para isso foi definido o número máximo de 500 épocas, pelo parâmetro epochs, e o uso do Early Stopping no parâmetro callback, para não permitir que o processo de treinamento rode infinitamente. Além disso, o parâmetro 'shuffle' foi definido como verdadeiro, fazendo com que os dados fossem apresentados para a rede em ordens diferentes em cada época, para garantir que a ordem dos dados não fosse determinante no resultado final do modelo.

Realizadas todas essas etapas, pôde-se utilizar o modelo para predição dos conjuntos de teste e validar seu desempenho por meio das mesmas métricas e ferramentas trabalhadas com os modelos descritos em III-A. É importante ressaltar que por se tratar de um rede neural de 2 nós de saída, o resultado de uma predição é um vetor de tamanho 2, contendo valores entre 0 e 1, que correspondem a probabilidade de cada rótulo ser a saída real. Para tornar esse vetor em um valor único utilizou-se a função numpy.argmax, que retorna o index do argumento com maior valor, logo, o mais provável entre as 2 opções.

Todo o fluxo de desenvolvimento do processo é apresentado na figura 5.

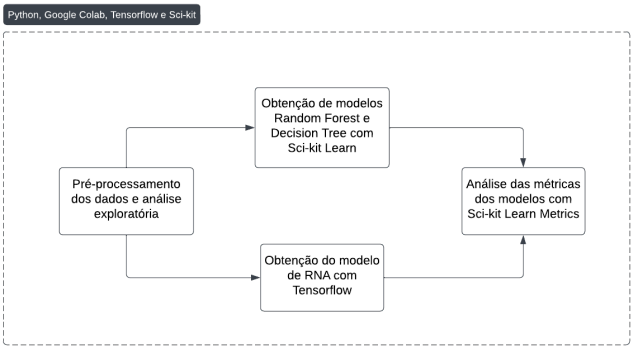

Fig. 5. Fluxograma do processo de desenvolvimento.

IV. Resultados

Ao fim do trabalho foram desenvolvidos 11 modelos de aprendizado de máquina (4 RNAs, 1 Decision Tree

e 6 Random Forests) utilizando os dados do conjunto CICIDS2017. Esses modelos apresentaram bons resultados, onde foi possível entender o comportamento de cada uma das técnicas escolhidas neste trabalho.

O modelo Decision Tree apresentou ótimo desempenho, com acurácia de 99,80% e F1-score de 99,49%, como apresentado na matriz de confusão na figura 6.

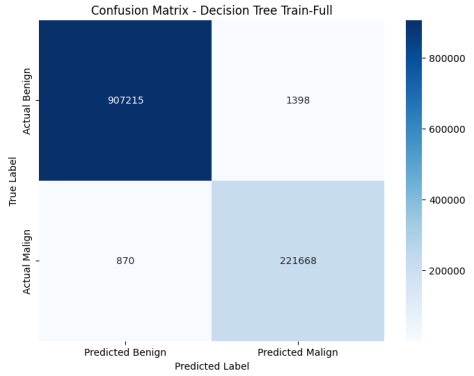

Fig. 6. Matriz de Confusão para Decision Tree com dados do CICIDS2017.

Ao treinar o primeiro modelo de Random Forest, inicialmente definindo o parâmetro max_depth como 2, foram obtidos resultados abaixo do esperado. Desta forma, realizou-se 5 novos treinamentos variando esse parâmetro e avaliando os resultados, que foram melhorando significativamente. A exatidão do modelo variou de 86,01%, com max_depth igual a 2, até 99,54%, com o parâmetro igual a 12, como detalhado na tabela I.

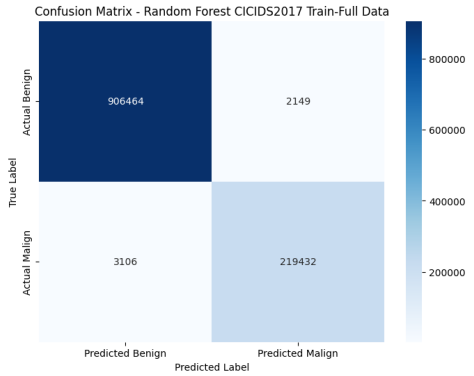

Fig. 7. Matriz de Confusão para Random Forest com dados da base CICIDS2017.

Por fim, foi realizado o treinamento dos modelos de Rede Neural Artificial, que apresentaram bons resultados, porém abaixo dos modelos de árvore. A melhor Rede Neural Artificial desenvolvida possuía 32 neurônios na camada oculta e função de ativação 'Relu', levando em conta o modelo com o maior valor da taxa de acerto. O modelo teve acurácia de 97,39% e F1-score de 93,52%, como apresentado na figura 8.

TABLE I
Métricas do treinamento dos modelos de Random Forest com variação do parâmetro max_depth utilizando a base CICIDS2017

max_depth	Acurácia	Precisão	Recall	F1
2	0,8601	1	0,2887	0,4481
3	0,8606	1	0,2915	0,4514
4	0,9385	0,9972	0,6894	0,8152
5	0,9525	0,9965	0,7611	0,8631
8	0,9878	0,9895	0,9479	0,9683
12	0,9954	0,9903	0,986	0,9882

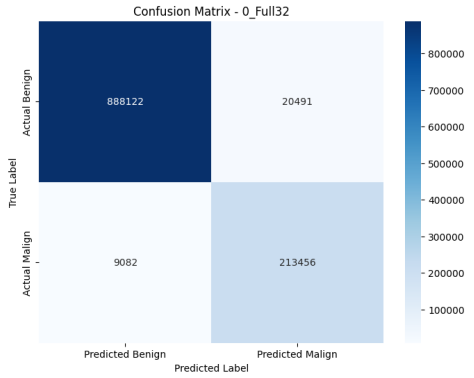

Fig. 8. Matriz de Confusão para Rede Neural Artificial com dados do CICIDS2017.

TABLE II
Métricas do treinamento dos modelos de RNA com variação da quantidade de nós na camada escondida utilizando a base CICIDS2017

Neurônios	F-Score	Acurácia	Precisão	Recall
128	0.965900	0.972200	0.900200	0.931900
16	0.964200	0.959200	0.848900	0.902900
32	0.959200	0.973900	0.912400	0.935200
64	0.933200	0.970400	0.917800	0.925500

Conclusões

Avaliou-se neste trabalho o desempenho de diferentes modelos de aprendizado de máquina na detecção de alguns tipos de ataques cibernéticos. Para isso, buscou-se comparar as Redes Neurais Artificiais com os modelos de árvore, Decision Tree e Random Forest que, via de regra, exigem custos de treinamento e processamento reduzidos.

O destaque em relação ao desempenho ficou para o modelo Decision Tree que apresentou os melhores resultados tanto nos testes, com acurácia e recall próximos de 1. Em contrapartida, o modelo de Rede Neural Artificial, apresentou bons resultados, porém, inferiores aos modelos de árvore.

Como recomendações para trabalhos futuros, se sugerem:

1) Explorar diferentes métodos e combinações de atributos de entrada dos modelos.
2) Utilizar técnicas de validação cruzada para melhor avaliar a capacidade de generalização dos modelos.
3) Utilizar rótulos relacionados ao tipo de ataque, a fim

de conseguir informar ao usuário com mais clareza o que se passa na aplicação monitorada.

Por fim, considera-se que o trabalho atingiu o objetivo esperado, sendo possível visualizar o comportamento de cada um dos modelos avaliados e abrir portas para melhorias e implementações no futuro.

References

[1] O. V. Sviatun, O. V. Goncharuk, C. Roman, O. Kuzmenko, and I. V. Kozych, "Combating cybercrime: economic and legal aspects," WSEAS Trans. Bus. Econ., vol. 18, pp. 751–762, 2021.

[2] F. Cohen, "Computer viruses: theory and experiments", Computers & Security, vol. 6, no. 1, pp. 22–35, 1987.

[3] Microsoft Security, "O que é malware?," 2024. [Online]. Disponível Em: https://www.microsoft.com/pt-br/security/business/security-101/what-is-malware. [Acessado em: Sep. 28, 2024].

[4] M. Cinque, D. Cotroneo, and A. Pecchia, "Challenges and directions in security information and event management (SIEM)," in *Proc. 2018 IEEE Int. Symp. Softw. Reliab. Eng. Workshops (ISSREW)*, 2018, pp. 95–99.

[5] N. Moukafih, G. Orhanou, and S. El Hajji, "Neural network-based voting system with high capacity and low computation for intrusion detection in SIEM/IDS systems," Secur. Commun. Netw., vol. 2020, pp. 1–15, 2020.

[6] R. Ch, T. R. Gadekallu, M. H. Abidi, and A. Al-Ahmari, "Computational system to classify cyber crime offenses using machine learning," *Sustainability*, vol. 12, no. 10, p. 4087, 2020.

[7] Z. Pelletier and M. Abualkibash, "Evaluating the CIC IDS-2017 dataset using machine learning methods and creating multiple predictive models in the statistical computing language R," Science, vol. 5, no. 2, pp. 187–191, 2020.

[8] Fathima, N., Pramod, A., Srivastava, Y., Thomas, A. M., Ibrahim, S. I. S. P., & Chandran, K. R. (2021). Two-stage deep stacked autoencoder with shallow learning for network intrusion detection system. IEEE.

[9] Yin, Y., Jang-Jaccard, J., Sabrina, F., & Kwak, J. (2022). Improving multilayer-perceptron(MLP)-based network anomaly detection with Birch clustering on CICIDS-2017 dataset. IEEE.

[10] Soltani, M., & Siavoshani, M. J. (2020). A content-based deep intrusion detection system. IEEE.

[11] Jairu, P. (2022). Network anomaly uncovering on CICIDS-2017 dataset: A supervised artificial intelligence approach. In IEEE International Conference on Electro/Information Technology.

[12] A. R. Muhammad, P. Sukarno, and A. A. Wardana, "Integrated Security Information and Event Management (SIEM) with Intrusion Detection System (IDS) for Live Analysis based on Machine Learning," Procedia Comput. Sci., vol. 217, pp. 1406–1415, 2023.

[13] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in Proc. Int. Conf. Inf. Syst. Secur. Privacy (ICISSP), vol. 1, pp. 108–116, 2018.

[14] The Pandas Development Team, "Pandas - Python Data Analysis Library," pandas.pydata.org. [Online]. Disponível em: https://pandas.pydata.org/. [Acessado em: Jun. 7, 2025].

[15] Python Software Foundation, "Python Documentation," Python.org. [Online]. Disponível em: https://www.python.org/doc/. [Acessado em: Jun. 7, 2025].

[16] The scikit-learn Developers, "scikit-learn: Machine Learning in Python," scikit-learn.org. [Online]. Disponível em: https://scikit-learn.org/stable/. [Acessado em: Jun. 7, 2025].

[17] M. Waskom and the seaborn Development Team, "seaborn: Statistical Data Visualization," seaborn.pydata.org. [Online]. Disponível em: https://seaborn.pydata.org/. [Acessado em: Jun. 7, 2025].

[18] The Matplotlib Development Team, "Matplotlib: Visualization with Python," matplotlib.org. [Online]. Disponível em: https://matplotlib.org/. [Acessado em: Jun. 7, 2025].

[19] R. Yehoshua, "Random Forest," *Medium*, 2023. [Online]. Disponível em: https://medium.com/@roiyeho/random-forests-98892261dc49. [Acessado em: Sep. 28, 2024].

[20] The TensorFlow Authors, "TensorFlow," tensorflow.org. [Online]. Disponível em: https://www.tensorflow.org/?hl=pt-br. [Acessado em: Jun. 7, 2025].

[21] The NumPy Developers, "NumPy" numpy.org. [Online]. Disponível em: https://numpy.org/. [Acessado em: Jun. 7, 2025].

[22] W. R. de Araújo Júnior, Sistema de monitoramento e detecção de ameaças em redes utilizando aprendizado de máquina, Monografia de Graduação, Curso de Engenharia de Controle e Automação, Escola de Minas, Universidade Federal de Ouro Preto, Ouro Preto, 2025.

[23] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. Bin Idris, A. M. Bamhdi and R. Budiarto, "CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection," in IEEE Access, vol. 8, pp. 132911-132921, 2020, doi: 10.1109/ACCESS.2020.3009843.