Análise de Tráfego de Rede com *Machine Learning* para Identificação de Ameaças a Dispositivos IoT

Marcelo Vinícius Cysneiros Aragão, Samuel Baraldi Mafra e Felipe Augusto Pereira de Figueiredo

***Resumo*— Dispositivos IoT estão cada vez mais presentes em nosso dia-a-dia, tanto em contextos particulares quanto em ambientes públicos. Consequentemente, a segurança deles também deve ser tratada com atenção. Após revisar diversas técnicas propostas, este trabalho propõe uma abordagem para detecção de ameaças baseada em análise de tráfego de rede, realizada por modelos de aprendizado de máquina. Após extensa experimentação e avaliação, foi possível produzir um modelo rapidamente treinável e altamente confiável, comprovando a eficácia da proposta e apontando direções para trabalhos futuros.**

***Palavras-Chave*— Internet das Coisas, cibersegurança, aprendizado de máquina.**

***Abstract*— IoT devices are increasingly present in our daily lives, both in private contexts and in public environments. Consequently, their security must also be thought with care. After reviewing relevant literature, this work proposes an approach to threat detection based on network traffic analysis, performed by machine learning models. After extensive experimentation and evaluation, it was possible to produce a quickly trainable and highly reliable model, proving the effectiveness of the proposal and pointing out directions for future work.**

***Keywords*— Internet of Things, cibersecurity, machine learning.**

I. Introdução

Com o crescente número de dispositivos IoT (*Internet of Things*) conectados à rede e desempenhando os mais diversos papeis, como monitoramento residencial e de ambientes críticos, a segurança emerge como uma questão de suma importância. Para tal, pode-se empregar soluções tanto a nível de *hardware* quanto de *software* (como *firewalls*, que permitem ou restringem a comunicação entre dispositivos baseando-se em regras). Uma abordagem inovadora, entretanto, vem chamando a atenção da comunidade científica: o emprego de modelos de aprendizado de máquina para analisar o tráfego de rede e identificar possíveis ameaças a dispositivos IoT.

Usando o conjunto de dados IoT-23 [1], que é apresentado em mais detalhes na seção II, Stoian [2] comparou modelos como redes neurais, árvores de decisão e máquina de vetores de suporte na identificação tanto de tráfego benigno quanto de varredura de portas, *download* de arquivos e até mesmo ataques de negação de serviço. Os resultados obtidos foram promissores (i.e. 99,5% de acurácia), mesmo não tendo realizado qualquer tipo de pré-processamento de dados.

No mesmo ano, os autores de Shafiq et al. [3] propuseram uma nova métrica para seleção de características denominada CorrAUC, baseada na combinação de outras duas métricas: CAE (*Correlation Attribute Evaluation*) e AUC (*Area Under the Curve*). A proposta foi avaliada no conjunto de dados Bot-IoT [4] com quatro modelos de aprendizado de máquina: árvore de decisão C4.5 [5], classificador Bayesiano ingênuo (*Naïve Bayes*), floresta aleatória (*Random Forest*) [6] e máquina de vetores de suporte [7]. A solução proposta apresentou precisão na detecção de tráfego malicioso superior a 96%.

Com um tipo de modelo diferente dos trabalhos citados anteriormente, os autores de Liang e Vankayalapati [8] avaliaram o emprego de redes neurais profundas (*Deep Neural Network*) na identificação de ameaças a dispositivos IoT, usando o já mencionado conjunto IoT-23. Além deste modelo, também foram considerados o classificador probabilístico Bayesiano ingênuo, a máquina de vetores de suporte e árvore de decisão. Apesar da promissora proposta em empregar redes neurais profundas, dada sua capacidade de generalização, os resultados não foram promissores, variando de 30% a 73% de acurácia.

Abdalgawad et al. [9] empregaram os modelos AAE (*Adversarial Autoencoders*) e BiGAN (*Bidirectional Generative Adversarial Networks*) de aprendizagem profunda generativa para classificar os ataques do conjunto IoT-23. Foram obtidas alta acurácia de classificação de ataques conhecidos (escore F_1 na ordem de 99%) e capacidade de detectar ataques *zero-day* (ou seja, ameaças recém descobertas, para as quais ainda não há solução) com escore F_1 variando de 85% a 100%.

Por fim, a implementação de Ullah e Mahmoud [10] baseou-se em redes neurais recorrentes, como LSTM (*Long Short Term Memory*), BiLSTM (*Bidirectional* LSTM) e GRU (*Gated Recurrent Unit*). O modelo foi avaliado em sete conjuntos de dados, incluindo Bot-IoT e IoT-23, e chegou a apresentar escore F_1 próximo a 100% para diversas ameaças conhecidas.

Comprovada a relevância do tema, este trabalho visa escolher, treinar, otimizar e avaliar diferentes modelos de aprendizado de máquina aplicados à identificação de ameaças a dispositivos IoT. Adicionalmente, sugere-se diversas técnicas de pré-processamento de dados que podem agilizar o treinamento e melhorar a acurácia dos modelos em questão.

O trabalho está organizado da seguinte forma: na seção I, o problema é introduzido e são visitadas outras abordagens da literatura para resolvê-lo. Em seguida, na seção II, é detalhada a metodologia da proposta. Os resultados experimentais são apresentados e discutidos na seção III. Por fim, a seção IV conclui o trabalho, recapitulando seus pontos-chave.

Marcelo Vinícius Cysneiros Aragão, Departamento de Engenharia de Computação e Software, Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí-MG, e-mail: marcelovca90@inatel.br; Samuel Baraldi Mafra, Departamento de Engenharia de Telecomunicações, Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí-MG, e-mail: samuelbmafra@inatel.br; Felipe Augusto Pereira de Figueiredo, Departamento de Engenharia de Telecomunicações, Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí-MG, e-mail: felipe.figueiredo@inatel.br.

II. METODOLOGIA

Resumidamente, este trabalho visa treinar e otimizar modelos de aprendizado de máquina, utilizando amostras de tráfego de rede, produzindo um mecanismo capaz de identificar possíveis ameaças a dispositivos conectados em ambientes reais.

O diagrama da Figura 1 ilustra as etapas do modelo, que são apresentadas em mais detalhes nas subseções a seguir.

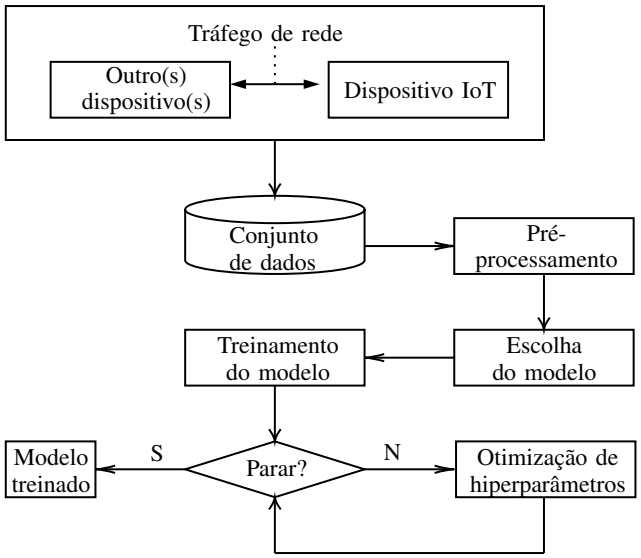

Fig. 1: Diagrama de blocos do modelo experimental.

A. Conjunto de dados

O conjunto de dados utilizado neste trabalho foi o Aposemat IoT-23 [1], capturado no laboratório Stratosphere, da Universidade Técnica Checa (Praga), com incentivo do Avast AI & Cybersecurity Laboratory. O conjunto consiste em vinte e três cenários de capturas de tráfego de rede envolvendo dispositivos IoT, sendo eles:

- 20 capturas de tráfego malicioso tais como varreduras de portas, ataques de negação de serviço distribuída e *malwares* executados em mini-computadores Raspberry Pi (tais como as *botnets* Mirai, Okiru e Torii [11]);
- 03 capturas de tráfego benigno em dispositivos IoT reais (*honeypots*) – assistente virtual Amazon® Echo, lâmpada inteligente Philips® Hue e fechadura com abertura remota Somfy® Door Lock.

É possível obter tanto uma versão reduzida ($8,8\ GBytes$) do conjunto de dados, contendo apenas as amostras rotuladas (ou seja, cujo tipo de tráfego foi previamente identificado), quanto a versão completa ($21\ GBytes$), que inclui amostras sem o rótulo associado. Neste trabalho, foi utilizada a primeira versão, tanto pela natureza da abordagem (classificação), quanto por limitações de poder computacional.

B. Pré-processamento dos dados

Mesmo utilizando a versão reduzida do conjunto, o volume de dados ainda é grande e demanda tratamento. Portanto, foram realizadas diversas etapas de pré-processamento, visando simultaneamente minimizar a quantidade de dados (e a consequente demanda por poder computacional) e maximizar a quantidade de informação. Os seguintes processos foram conduzidos usando as ferramentas `scikit-learn` [12], `pandas` [13] e `scipy` [14]:

1) Concatenação dos 23 conjuntos de dados, visando produzir um único arquivo para facilitar o processamento nas etapas posteriores;
2) Padronização dos rótulos das classes, pois em algumas amostras, os identificadores de um mesmo tipo de tráfego não eram consistentes entre si;
3) Remoção de amostras que não correspondessem aos cenários mais relevantes. Neste caso, considerou-se os quatro mais frequentes no conjunto de dados, sendo eles:
   - Tráfego benigno[1]: $48{,}59\%$ ($\approx 4{,}3M$ amostras);
   - DDoS[2]: $24{,}76\%$ ($\approx 2{,}2M$ amostras);
   - *Horizontal Port Scan*[3]: $26{,}48\%$ ($\approx 2{,}3M$ amostras);
   - Okiru[4]: $0{,}17\%$ ($\approx 15{,}2k$ amostras).
4) Preenchimento de valores ausentes com caracteres indicadores padronizados (ex.: quando uma quantidade de tráfego recebido ou enviado era nula, usava-se o indicador $-$, que foi substituído por 0.;
5) Remoção de amostras duplicadas, pois não trazem informação inédita e, consequentemente, não contribuem com o treinamento dos modelos de aprendizado de máquina;
6) Codificação de características categóricas usando *one-hot encoding* (ex.: a característica categórica `proto`, referente ao protocolo utilizado (cujos valores poderiam ser as cadeiras de caracteres "TCP" ou "UDP"), foi codificada em duas características numéricas (`proto_tcp` e `proto_udp`), que poderiam assumir valor 0 ou 1 [15];
7) Definição dos tipos de dados de cada característica, para assegurar que não seja demandada mais memória do que o necessário (ex.: representar desnecessariamente uma porta de comunicação, que naturalmente trata-se de um número inteiro, como um número de ponto flutuante);
8) Separação dos dados em subconjuntos de treinamento (80% das amostras) e teste (20%), usados para treinamento e avaliação dos modelos de aprendizado de máquina, respectivamente;
9) Seleção de características, mantendo apenas aquelas que apresentam alta variância, por tipicamente apresentarem maior potencial preditivo;
10) Escalonamento dos valores numéricos para a faixa $[0, 1]$, visando estabilizar os processos de treinamento e evitar estouro de variáveis (*overflow*);
11) Codificação dos rótulos das amostras usando números ordinais (em vez de cadeias de caracteres), para agilizar a identificação dos dados pertencentes a cada cenário.

Neste ponto, os dados estão prontos para serem usados no treinamento de modelos de aprendizado de máquina, cuja seleção dar-se-á na próxima etapa.

[1]Tráfego convencional, que não apresenta qualquer comportamento maligno ao dispositivo, usuário ou sistema.

[2]*Distributed Denial of Service*: ataque no qual dispositivos infectados realizam requisições simultâneas a um serviço, tornando-o indisponível devido ao intenso tráfego de rede.

[3]Varredura horizontal que busca uma mesma porta aberta em diferentes *hosts* para preparação de um ataque em massa a um determinado alvo.

[4]Variante do *malware* Mirai capaz de infectar dispositivos com processador ARC (*Argonaut RISC Core*) para torná-los parte de uma *botnet*.

C. Identificação dos modelos de aprendizado de máquina

Em posse dos dados já processados, deve-se identificar modelos de aprendizado de máquina para realizar a tarefa de classificação. Optou-se por tal abordagem pois a capacidade de aprender por meio de exemplos permite que os modelos adaptem-se à distribuição subjacente dos dados e realizem o processo de inferência rapidamente. Tais modelos também podem ser configurados para indicar *outliers* e/ou anomalias nos dados [16]. Para tal, foram considerados trabalhos da literatura que comparam diversos modelos [17, 18, 19], bem como as documentações das bibliotecas `scikit-learn` [12], `tensorflow` [20] e `keras` [21].

Os modelos considerados são: análise discriminante linear e quadrática [22], árvores de decisão [5], *ensembles* (*AdaBoost*, *Extra Trees*, *Histogram Gradient Boosting*, *Random Forest* e *XGBoost*) [6, 23], modelos baseados em distância entre amostras (*Nearest Centroid* [24]), máquinas de vetores de suporte (com *kernels Additive* CHI2 e RBF – *Radial Basis Function*) [7, 25, 26], modelos lineares (*Passive Aggressive*, *Perceptron*, *Ridge* e SGD – *Stochastic Gradient Descent* [27, 28, 29], modelos probabilísticos (*Complement*, *Gaussian* e *Multinomial Naïve Bayes*) [30, 31] e redes neurais (*Multilayer Perceptron* e *Deep Neural Network*) [32, 33].

Mesmo concebidos a partir fundamentos teóricos distintos e apresentando particularidades de treinamento e operação, estes modelos servem ao mesmo propósito da classificação. Feita a escolha, pode-se proceder para a etapa de treinamento.

D. Treinamento dos modelos e otimização de hiperparâmetros

Primeiramente, deve-se "treinar" os modelos, utilizando (ou seja, utilizar uma fração dos conjuntos de dados para ajustar seus parâmetros) para que apresentem resultado satisfatório na classificação de dados que não foram usados no treinamento.

O sucesso ou fracasso deste processo, frequentemente realizado por meio da minimização iterativa de uma função de erro, está associado à configuração inicial de hiperparâmetros que do modelo. Portanto, deve-se avaliar diferentes combinações de valores de modo a otimizar o desempenho do modelo (ou seja, maximizar sua acurácia na classificação). Para tal, foi utilizada a biblioteca `optuna` [34, 35], e os espaços de hiperparâmetros foram especificados manualmente, considerando suas particularidades ao determinar as faixas e/ou conjuntos de valores aceitáveis para cada hiperparâmetro.

Cada modelo foi submetido a uma rodada de otimização, realizada pela biblioteca `optuna`. Foi estipulado que cada modelo seria treinado quantas vezes fosse possível durante uma hora, com diferentes configurações de hiperparâmetros. Caso o primeiro treinamento extrapolasse tal limite, aguardava-se até sua conclusão. Ao final de cada execução, era armazenada a configuração que produzia o melhor resultado preditivo.

Na seção a seguir, são apresentados e discutidos os resultados experimentais, e apontadas as conclusões obtidas.

III. Experimentos e Discussão

Os experimentos foram conduzidos em um computador com processador AMD Ryzen™ 7 5800X, 32 GB de RAM DDR4-3200, placa de vídeo Nvidia GeForce® RTX 2060 6GB GDDR6 (*driver* versão 511.79) e sistema operacional Microsoft Windows 10 Professional x64 (versão 21H1).

Como primeiro resultado esperado, deseja-se um classificador com a maior precisão de discernimento entre os tipos de ameaças aos dispositivos. Para tal, foi utilizada a métrica "acurácia", que contabiliza exatamente quantas previsões feitas pelo classificador coincidem com o tipo de ameaça em questão. Um *ranking* dos classificadores, dispostos em ordem decrescente de "acurácia" (ou seja, do melhor para o pior), está disposto na tabela I.

TABELA I: *Ranking* de acurácia dos classificadores.

Posição	Classificador	Acurácia (%)
1	*Decision Tree*	99,86
	Extra Trees	99,86
	Random Forest	99,86
2	*Histogram Gradient Boosting*	99,78
3	*Deep Neural Network*	99,38
4	*XGBoost*	99,37
5	*Support Vector Machine* (CHI2)	97,46
6	*AdaBoost*	93,46
7	*Multilayer Perceptron*	78,98
8	*Support Vector Machine* (RBF)	74,50
9	*Passive Aggressive*	74,05
10	*Stochastic Gradient Descent*	73,55
11	*Perceptron*	73,40
12	*Ridge*	72,98
13	*Quadratic Discriminant Analysis*	72,70
14	*Linear Discriminant Analysis*	72,64
15	*Gaussian Naïve Bayes*	72,53
16	*Complement Naïve Bayes*	72,23
17	*Multinomial Naïve Bayes*	72,01
18	*Nearest Centroid*	59,51

Além de uma alta acurácia, também é desejável que o classificador possa ser (re)treinado rapidamente. Isto porque, em cenários dinâmicos, as características dos ataques podem mudar, demandando uma rápida adaptação (ou mesmo uma recriação) do modelo em questão para que ele continue sendo útil à tarefa de detecção de ameaças. Neste sentido, um *ranking* dos classificadores, dispostos em ordem decrescente de "quantidades de treinamentos realizados em uma hora com, no mínimo, uma convergência" está disposto na tabela II.

TABELA II: *Ranking* de treinamentos dos classificadores.

Posição	Classificador	Treinamentos
1	*Complement Naïve Bayes*	100
	Decision Tree	100
	Gaussian Naïve Bayes	100
	Linear Discriminant Analysis	100
	Multinomial Naïve Bayes	100
	Nearest Centroid	100
	Passive Aggressive	100
	Quadratic Discriminant Analysis	100
	Ridge	100
2	*Perceptron*	83
3	*Stochastic Gradient Descent*	26
4	*Extra Trees*	25
5	*Random Forest*	18
6	*AdaBoost*	10
7	*Histogram Gradient Boosting*	7
8	*Support Vector Machine* (CHI2)	4
9	*Support Vector Machine* (RBF)	3
	Multilayer Perceptron	3
10	*Deep Neural Network*	1
	XGBoost	1

Analisando os resultados das tabelas I e II, pode-se fazer as seguintes afirmações:

- Modelos simples são treinados mais rapidamente, atingindo a convergência múltiplas vezes dentro da janela de tempo estabelecida (ex.: modelos probabilísticos e de análise discriminante), colocando-os no topo do *ranking*;
- Modelos mais complexos (como no caso das máquinas de vetores de suporte e da rede neural profunda) atingiram a quantidade mínima aceitável de convergências (uma) e, portanto, ficaram nas últimas posições do *ranking*;
- *Ensembles* e árvore(s) de decisão produziram os melhores resultados (tanto em termos de taxas de acerto quanto de tempo de treinamento). Outra vantagem destes modelos é que são do tipo "caixa branca", ou seja, é possível compreender o que foi aprendido pelo modelo (diferentemente das redes neurais, por exemplo);
- Em termos de acurácia, redes neurais profundas também podem ser utilizadas para a tarefa, mas demandam longos períodos de treinamento;
- Classificadores probabilísticos e baseados em análise discriminante apresentaram baixo tempo de treinamento, mas são desconsiderados devido às baixas taxas de acerto;
- Classificadores lineares, mesmo com funções de *kernel* não-lineares, não apresentaram bons resultados, e seus tempos de treinamento foram longos.

Conclui-se, então, que é possível detectar ameaças a dispositivos IoT usando modelos de aprendizado de máquina, por meio da análise do tráfego de rede. Em particular, árvores de decisão desempenhariam tal tarefa com maestria, pois apresentam alta acurácia e baixo tempo de treinamento.

O código e as instruções para reprodução dos experimentos, bem como todos os resultados obtidos, estão disponíveis em um repositório[5] público do GitHub.

IV. Conclusão

O advento da IoT vem demandando atenção a questões de segurança, já que tais dispositivos estão inseridos no dia-a-dia das pessoas em contextos particulares e ambientes públicos.

Neste sentido, autores vêm propondo formas de detectar ameaças a tais dispositivos. Uma das técnicas consiste na análise do tráfego de rede, por meio da captura dos pacotes trocados na comunicação e, em seguida, classificação do tráfego em benigno ou maligno, identificando a ameaça em questão com modelos de aprendizado de máquina, por exemplo.

O trabalho propôs uma metodologia de treinamento, avaliação e aprimoramento de modelos de aprendizado de máquina, sugerindo e detalhando desde o pré-processamento de dados (para redução do volume de dados, maximização da quantidade de informação e redução dos tempos de treinamento) até a otimização de hiperparâmetros (para potencialização da capacidade preditiva dos modelos).

Diante da análise dos resultados experimentais, foi possível comprovar a eficácia da metodologia proposta, já que foi capaz de produzir rapidamente um modelo cuja capacidade de detecção aproxima-se de 99,9% de acurácia.

As técnicas de pré-processamento empregadas também são de grande valor, pois aumentam a relevância dos dados, reduzem o custo computacional do treinamento e potencializam a capacidade preditiva dos modelos.

Como sugestões para trabalhos futuros, pode-se sugerir o emprego de técnicas mais sofisticadas de pré-processamento de dados (como PCA – *Principal Component Analysis* – e RFE – *Recursive Feature Elimination*), outros modelos de aprendizado de máquina (como *autoencoders*, CNNs – *Convolutional Neural Networks* – e GANs – *Generative Adversarial Networks*), a avaliação de outros conjuntos de dados (como o Bot-IoT, IoT Network Intrusion e MQTT-IoT-IDS2020), a consideração de mais tipos de ataques (como Mirai e Torii) e uma exploração mais ampla do espaço de hiperparâmetros.

Agradecimentos

Os autores agradecem o apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), (403827/2021-3), Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) (2021/06946-0) e parcialmente financiado pela Rede Nacional de Ensino e Pesquisa (RNP) com recursos do Ministério da Ciência, Tecnologia e Inovações (MCTIC), concessão número 01245.010604/2020-14, sob o projeto Brasil 6G do Centro de Referência em Radiocomunicações (CRR) do Instituto Nacional de Telecomunicações (Inatel), Brasil.

Referências

[1] S. Garcia, A. Parmisano e M.J. Erquiaga. *IoT-23: A labeled dataset with malicious and benign IoT network traffic*. Zenodo, 2020. DOI: 10.5281/zenodo.4743746.

[2] N.A. Stoian. *Machine Learning for Anomaly Detection in IoT networks: Malware analysis on the IoT-23 Data set*. 2020.

[3] M. Shafiq, Z. Tian, A.K. Bashir, X. Du e M. Guizani. "CorrAUC: A Malicious Bot-IoT Traffic Detection Method in IoT Network Using Machine-Learning Techniques". Em: *IEEE Internet of Things Journal* 8.5 (2021), pp. 3242–3254. ISSN: 23274662. DOI: 10.1109/JIOT.2020.3002255.

[4] N. Moustafa. *The Bot-IoT dataset*. 2019. DOI: 10.21227/r7v2-x988.

[5] L. Breiman, J. Friedman, C.J. Stone e R.A. Olshen. *Classification and regression trees*. 1ª ed. June. Boca Raton, FL, USA: Chapman & Hall/CRC, 1984, p. 358. DOI: 10.1002/widm.8.

[6] L. Breiman. "Random forests". Em: *Machine Learning* 45.1 (2001), pp. 5–32. ISSN: 1098-6596. DOI: 10.1017/CBO9781107415324.004.

[7] C. Cortes e V. Vapnik. "Support-Vector Networks". Em: *Machine Learning* 20.3 (1995), pp. 273–297. DOI: 10.1023/A:1022627411411.

[8] Y. Liang e N. Vankayalapati. *Machine Learning and Deep Learning Methods for Better Anomaly Detection in IoT-23 Dataset Cybersecurity*. 2021.

[5] https://github.com/marcelovca90/Anomaly-Detection-IoT23

[9] N. Abdalgawad, A. Sajun, Y. Kaddoura, I. A. Zualkernan e F. Aloul. "Generative Deep Learning to Detect Cyberattacks for the IoT-23 Dataset". Em: *IEEE Access* 10 (2022), pp. 6430–6441. ISSN: 21693536. DOI: 10.1109/ACCESS.2021.3140015.

[10] I. Ullah e Q.H. Mahmoud. "Design and Development of a Deep Learning-Based Model for Anomaly Detection in IoT Networks". Em: *IEEE Access* 9 (2021), pp. 103906–103926. ISSN: 21693536. DOI: 10.1109/ACCESS.2021.3094024.

[11] H. Habibi Gharakheili, A. Hamza e V. Sivaraman. "Cyber-Securing IoT Infrastructure by Modeling Network Traffic". Em: *Security and Privacy in the Internet of Things* (2021), pp. 151–176. DOI: 10.1002/9781119607755.ch6.

[12] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". Em: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[13] The Pandas development team. *Pandas*. 2020. DOI: 10.5281/zenodo.3509134.

[14] W. McKinney. "Data Structures for Statistical Computing in Python". Em: *Proceedings of the 9th Python in Science Conference*. Ed. por Stéfan van der Walt e Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/majora-92bf1922-00a.

[15] C. Seger. *An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing*. 2018.

[16] P. Gogoi, D. K. Bhattacharyya, B. Borah e Jugal K. Kalita. "A survey of outlier detection methods in network anomaly identification". Em: *Computer Journal* 54.4 (2011), pp. 570–588. ISSN: 00104620. DOI: 10.1093/comjnl/bxr026.

[17] R. Caruana e A. Niculescu-Mizil. "An empirical comparison of supervised learning algorithms". Em: *International Conference on Machine Learning*. Vol. 148. 2006, pp. 161–168. DOI: 10.1145/1143844.1143865.

[18] Marcelo V.C. Aragao, Isaac C. Ferreira, Edvard M. Oliveira, Bruno T. Kuehne, Edmilson M. Moreira e Otavio A.S. Carpinteiro. "A Study and Evaluation of Classifiers for Anti-Spam Systems". Em: *IEEE Access* 9 (2021), pp. 157482–157498. ISSN: 21693536. DOI: 10.1109/ACCESS.2021.3129203.

[19] A. Churcher, R. Ullah, J. Ahmad, S. Ur Rehman, F. Masood, M. Gogate, F. Alqahtani, B. Nour e W.J. Buchanan. "An experimental analysis of attack classification using machine learning in IoT networks". Em: *Sensors* 21.2 (2021), p. 446.

[20] Martín∼A. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. https://tensorflow.org. 2015.

[21] F. Chollet et al. *Keras*. https://keras.io. 2015.

[22] A. Tharwat. "Linear vs. quadratic discriminant analysis classifier: a tutorial". Em: *International Journal of Applied Pattern Recognition* 3.2 (2016), p. 145. ISSN: 2049-887X. DOI: 10.1504/ijapr.2016.079050.

[23] T. Chen e C. Guestrin. "XGBoost: A scalable tree boosting system". Em: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785.

[24] R. Tibshirani, T. Hastie, B. Narasimhan e G. Chu. "Diagnosis of multiple cancer types by shrunken centroids of gene expression". Em: *Proceedings of the National Academy of Sciences of the United States of America* 99.10 (2002), pp. 6567–6572. ISSN: 00278424. DOI: 10.1073/pnas.082099299.

[25] A. Vedaldi e A. Zisserman. "Efficient additive kernels via explicit feature maps". Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.3 (2012), pp. 480–492. ISSN: 01628828. DOI: 10.1109/TPAMI.2011.153.

[26] A. Rahimi e B. Recht. "Random features for large-scale kernel machines". Em: *Advances in Neural Information Processing Systems 20 - Proceedings of the 2007 Conference* 20 (2009).

[27] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz e Y. Singer. "Online passive aggressive algorithms". Em: (2006).

[28] F. Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain". Em: *Psychological Review* 65.6 (1958), pp. 386–408. ISSN: 0033295X. DOI: 10.1037/h0042519.

[29] T. Zhang. "Solving large scale linear prediction problems using stochastic gradient descent algorithms". Em: *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*. 2004, pp. 919–926. ISBN: 1581138385. DOI: 10.1145/1015330.1015332.

[30] J.D.M. Rennie, L. Shih, J. Teevan e D. Karger. "Tackling the Poor Assumptions of Naive Bayes Text Classifiers". Em: *Proceedings, Twentieth International Conference on Machine Learning*. Vol. 2. 2003, pp. 616–623. ISBN: 1577351894.

[31] H. Schütze, C.D. Manning e P. Raghavan. *Introduction to information retrieval*. Vol. 39. Cambridge University Press Cambridge, 2008.

[32] D.E. Rumelhart, G.E. Hinton e R.J. Williams. "Learning representations by back-propagating errors". Em: *Nature* 323.6088 (1986), pp. 533–536. ISSN: 00280836. DOI: 10.1038/323533a0.

[33] I. Goodfellow, Y. Bengio e A. Courville. *Deep Learning*. MIT Press, 2016.

[34] T. Akiba, S. Sano, T. Yanase, T. Ohta e M. Koyama. "Optuna: A Next-generation Hyperparameter Optimization Framework". Em: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019, pp. 2623–2631. ISBN: 9781450362016. DOI: 10.1145/3292500.3330701.

[35] H. Jin, Q. Song e X. Hu. "Auto-Keras: An Efficient Neural Architecture Search System". Em: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2019, pp. 1946–1956.