UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
DEPARTAMENTO DE COMPUTAÇÃO
PROGRAMA DE PÓS–GRADUAÇÃO EM INFORMÁTICA APLICADA

VALDIR CARVALHO DE SANTANA FILHO

AVALIAÇÃO DE DESEMPENHO DE SISTEMAS DE DETECÇÃO DE INTRUSÃO BASEADOS EM IA PARA AMBIENTES DE NUVEM E BORDA

RECIFE – PE

2023

VALDIR CARVALHO DE SANTANA FILHO

AVALIAÇÃO DE DESEMPENHO DE SISTEMAS DE DETECÇÃO DE INTRUSÃO BASEADOS EM IA PARA AMBIENTES DE NUVEM E BORDA

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada do Departamento de Computação - DC - Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

ORIENTADOR: Prof. Dr. Ermeson Carneiro de Andrade

CO–ORIENTADOR: Prof. Dr. Júlio Rodrigues De Mendonça Neto

RECIFE – PE

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

F481a Filho, Valdir Carvalho de Santana
Avaliação de Desempenho de Sistemas de Detecção de Intrusão Baseados em IA para Ambientes de Nuvem e Borda / Valdir Carvalho de Santana Filho. - 2023.
92 f. : il.

Orientador: Ermeson Carneiro de Andrade.
Coorientador: Julio Rodrigues de Mendonca Neto.
Inclui referências.

Dissertação (Mestrado) - Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Informática Aplicada, Recife, 2023.

1. Avaliação de Desempenho. 2. Computação de Borda. 3. Computação em Nuvem. 4. Inteligência Artificial. 5. Sistema de Detecção de Intrusão. I. Andrade, Ermeson Carneiro de, orient. II. Neto, Julio Rodrigues de Mendonca, coorient. III. Título

CDD 004

VALDIR CARVALHO DE SANTANA FILHO

AVALIAÇÃO DE DESEMPENHO DE SISTEMAS DE DETECÇÃO DE INTRUSÃO BASEADOS EM IA PARA AMBIENTES DE NUVEM E BORDA

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada do Departamento de Computação - DC - Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

Aprovada em: 18 de abril de 2023.

BANCA EXAMINADORA

Prof. Dr. Ermeson Carneiro de Andrade (Orientador)
Universidade Federal Rural de Pernambuco
Departamento de Computação

Prof. Dr. Gustavo Rau de Almeida Callou
Universidade Federal Rural de Pernambuco
Departamento de Computação

Prof. Dr. Carlos Julian Menezes Araújo
Universidade Federal Rural de Pernambuco
Departamento de Computação

À Débora, Josefa, Valdir (*in memorian*), Geize, Priscila e Gustavo por todo apoio, amor e carinho.

Agradecimentos

A Deus por me dar a persistência necessária para não desistir diante das dificuldades encontradas no caminho, por me guiar e permitir vencer mais este sonho. Ao meu orientador, Professor Ermeson Andrade, por aceitar me orientar, pelo ótimo acolhimento, pela paciência, suporte e orientações que foram tão importantes nesta jornada. Agradeço ao meu co-orientador, Professor Júlio Mendonça, pelo grande apoio nesse período, por estar sempre disponível para ajudar com paciência, aconselhando nos momentos mais importantes. Agradeço aos membros da banca, os professores Gustavo Callou e Carlos Julian, pelos valiosos comentários e revisões feitas nesta dissertação.

A minha amada esposa, Geize Maria, por ter a compreensão e a paciência durante este período. Ela me incentivou, apoiou e me motivou a permanecer firme nos momentos mais difíceis. A minha mãe Débora e meu pai Valdir (*in memoriam*), por me apoiar, dar coragem e força para conseguir trilhar este caminho. A minhã mãe Josefa que me criou durante toda minha infância com todo amor. Minha irmã Priscila e meu irmão Gustavo por sempre me apoiarem. A toda minha família, por todo apoio e encorajamento. Vocês, e todos os outros não citados aqui (são muitos), fazem parte da minha vida. Amo-os.

Aos meus amigos e irmãos de infância e juventude, da Escola Técnica SENAI Areias, da Zero Hum Informática, do IFPE e do IFRN por estarem ao meu lado sempre apoiando e dando força para seguir adiante. Ao Ewerton, do grupo de pesquisa, que me ajudou bastante no início dessa jornada. E a todos os meus amigos que sempre me apoiaram e torceram por mim.

(...) posso enfrentar o que for, eu sei quem luta por mim.

(–Davi Sacer e Luís Arcanjo)

Resumo

As Tecnologias da Informação e Comunicação (TIC) estão em constante evolução. Novas infraestruturas surgem buscando atenuar novos, e por vezes, antigos problemas. A exemplo disso, o paradigma da computação em nuvem busca centralizar tecnologias e oferecer serviços aos clientes cobrando taxas em cima da demanda utilizada. No entanto, a alta latência na comunicação entre os usuários e os *datacenters* na nuvem dificultam o desenvolvimento de sistemas que necessitam de respostas em tempo real. Com o objetivo de mitigar a latência na comunicação dos dados entre os usuários finais e as infraestruturas em nuvem, surge a computação de borda, que propõe processar, e por vezes armazenar, parte dos dados produzidos pelos usuários finais na origem, fazendo com que o processamento realize a entrega dos serviços considerando critérios acordados, porém os recursos computacionais na borda são, normalmente, limitados. Alinhado a essa evolução, a área de Segurança da Informação tem desenvolvido novos *Intrusion Detection Systems* (IDSs) baseados em Inteligência Artificial (IA), visando tornar os ambientes de borda mais seguros e com menos interação humana. No entanto, não há pesquisas que avaliem os *trade-offs* de desempenho e acurácia dos ambientes de nuvem e borda computacional, levando em consideração diferentes implementações de IDSs baseados em IA. Assim, esta dissertação tem como objetivo avaliar os *trade-offs* de desempenho de ambientes de nuvem e borda que implementam IDSs baseados em IA, usando técnicas de *Machine Learning* (ML) e *Deep Learning* (DL). Mais especificamente, neste trabalho, utilizamos o *Feedforward Neural Network* (FNN), *Naive Bayes* (NB) e *Random Forest* (RF) para criar os modelos de ML e DL por possuírem altas taxas de aprendizagem e bom desempenho. Os resultados revelaram que os IDSs baseados nos modelos de ML (NB e RF) obtiveram menor consumo computacional em comparação com os modelos de DL (FNN), além de alcançarem altas taxas de aprendizagem. Além disso, os IDSs baseados em DL com Cargas de Trabalho (CTs) mais intensas, consumiram todos os recursos computacionais do dispositivo de borda, levando a sua indisponibilidade. Assim, antes de implementar IDSs utilizando técnicas de ML e DL em ambientes de nuvem ou borda é crucial realizar uma análise completa, pois os IDSs podem levar a indisponibilidade do ambiente.

Palavras-chave: Avaliação de Desempenho, Computação de Borda, Computação em Nuvem, Inteligência Artificial, Sistema de Detecção de Intrusão.

Abstract

Information and Communication Technologies (ICT) are constantly evolving. New infrastructures emerge seeking to mitigate new, and sometimes old, problems. As an example, the paradigm of cloud computing seeks to centralize technologies and offer services to customers by charging fees based on the demand used. However, the high latency in communication between users and cloud datacenters makes it difficult to develop systems that require real-time responses. With the aim of mitigating latency in data communication between end users and cloud infrastructures, edge computing emerges, which proposes to process, and sometimes store, part of the data produced by end users at the origin, making the Processing performs service delivery considering agreed criteria, but computational resources at the edge are usually limited. In line with this evolution, the Information Security area has developed new Intrusion Detection Systems (IDSs) based on Artificial Intelligence (AI), aiming to make edge environments safer and with less human interaction. However, there is no research evaluating the performance and accuracy trade-offs of cloud and edge computing environments, taking into account different implementations of AI-based IDSs. Thus, this dissertation aims to evaluate the performance trade-offs of cloud and edge environments that implement AI-based IDSs, using Machine Learning (ML) and Deep Learning (DL) techniques. More specifically, in this work, we used the Feedforward Neural Network (FNN), Naive Bayes (NB) and Random Forest (RF) to create the ML and DL models because they have high learning rates and good performance. The results revealed that the IDSs based on the ML models (NB and RF) had lower computational consumption compared to the DL models (FNN), in addition to achieving high learning rates. In addition, DL-based IDSs with more intense Workloads (CTs) consumed all the computational resources of the edge device, leading to its unavailability. Thus, before implementing IDSs using ML and DL techniques in cloud or edge environments, it is crucial to perform a thorough analysis, as IDSs can lead to unavailability of the environment.

Keywords: Artificial intelligence, Cloud Computing, Edge Computing, Intrusion Detection System, Performance Evaluation.

Lista de Figuras

Lista de tabelas

Lista de Siglas

AUC	*Area Under the Curve*
CART	*Classification and Regression Tree*
CHAID	*CHi-squared Automatic Interaction Detector*
CB	Computação de Borda
CN	Computação em Nuvem
CNN	*Convolutional Neural Network*
CPS	*Cyber-Physics Systems*
CPU	*Central Processing Unit*
CT	Carga de Trabalho
DDoS	*Distributed Denial of Service*
DDPG	*Deep Deterministic Policy Gradient*
DL	*Deep Learning*
DNN	*Deep Neural Network*
DoS	*Denial of Service*
DT	*Decision Tree*
FC	*Fog Computing*
FNN	*FeedForward Neural Network*
GCP	*Google Cloud Platform*
GPS	*Global Positioning System*
GPU	*Graphics Processing Unit*
HIDS	*Host Intrusion Detection System*
HIDS	*Hybrid Intrusion Detection System*
IA	Inteligência Artificial
IaaS	*Infraestructure-as-a-Service*
IDS	*Intrusion Detection System*
IGA	*Improved Genetic Algorithm*

IoT	*Internet of Things*
IO	*Isolation Forest*
LISP	*Locator ID Separation Protocol*
LOF	*Local Outlier Factor*
ML	*Machine Learning*
MEC	*Mobile Edge Computing*
NB	*Naive Bayes*
NBA	*Network Behavior Analyzer*
NIDS	*Network Intrusion Detection System*
ObfNet	*Obfuscation Neural Network*
PaaS	*Platform-as-a Service*
RAM	*Random Access Memory*
RF	*Random Forest*
RNA	Rede Neural Artificial
RNN	*Recurrent Neural Network*
ROC	*Receiver Operating Characteristic*
SAA	*Simulated Annealing Algorithm*
SaaS	*Software-as-a Service*
SBC	*Single Board Computer*
SDPN	*Stacked-Deep Polynomial Network*
SMO	*Spider Monkey Optimization*
TF-IDF	*Term Frequency - Inverse Document Frequency*
VM	*Virtual Machine*
VPN	*Virtual Private Network*
WAN	*Wide Area Network*

Sumário

1 Introdução

O surgimento da computação em nuvem (QIAN et al., 2009) permitiu o desenvolvimento de modelos de serviços como a *Infraestructure-as-a-Service* (IaaS), a *Platform-as-a-Service* (PaaS), o *Software-as-a-Service* (SaaS), o *Hardware-as-a-Service* (HaaS), e do *Database-as-a-Service* (DaaS), ou seja, a entrega de recursos computacionais como serviço acessíveis de forma pública, privada ou híbrida (WANG et al., 2010). As nuvens computacionais permitem integração, elasticidade, escalabilidade e alta disponibilidade, além de armazenarem e processarem grandes massas de dados que atualmente estão ligadas ao advento da *Internet of Things* (IoT) (ASHTON et al., 2009). Os dispositivos de IoT nos conectam a outros dispositivos e pessoas, além de estarem presentes em nosso dia a dia (WEISER, 1993), buscando tornar o ambiente mais autônomo e ágil. A Computação de Borda (YU et al., 2017) surgiu da necessidade de baixa latência na comunicação entre os dispositivos de IoT e as nuvens computacionais, pois sistemas críticos necessitam de respostas rápidas para tomar decisões em pouco tempo (SHI et al., 2016). Além disso, vale ser ressaltado que os dispositivos característicos na computação de borda normalmente possuem recursos limitados. Assim, a Computação de Borda é usada como um recurso para realizar parte do processamento mais próximo da fonte dos dados para tentar diminuir a latência entre os dispositivos de IoT e a nuvem. Posteriormente, os dados podem ser transmitidos para armazenamento ou para finalizar o processamento na nuvem computacional.

A segurança da informação é algo que não se pode negligenciar no contexto da computação em nuvem e da computação de borda, pois podem causar grandes perdas financeiras. Incidentes relacionados à cibersegurança são reportados frequentemente, independente do país e do tamanho da empresa (TSESMELIS et al., 2022). A área de pesquisa referente à segurança da informação é ampla, e busca mitigar ameaças a equipamentos específicos ou infraestruturas complexas, garantindo, normalmente, a confidencialidade, integridade, disponibilidade, autenticidade, e irretratabilidade (AVIZIENIS et al., 2004). Novos conceitos e técnicas de segurança surgem para atender um universo com muitas possibilidades, cenários e diferentes sistemas. Dessa forma, ambientes seguros modernos utilizam anti-vírus, *firewalls*, IDSs, *proxies* e *Virtual Private Network* (VPN) (YAN et al., 2015), juntamente com técnicas de barreira

física, gestão de usuário e senha, análise de logs e gestão ativa dos equipamentos buscando minimizar possíveis ameaças cibernéticas. Assim, pesquisas recentes buscam desenvolver IDSs com maior nível de acerto e menor consumo de recursos computacionais, e assim, mitigar vulnerabilidades junto às demais técnicas.

Os IDSs são capazes de gerar alertas a ameaças detectadas no ambiente que se está monitorando. Esses sistemas podem ser responsáveis por analisar e monitorar uma rede de computadores (*Network Intrusion Detection System* (NIDS)), uma rede sem fio (*Wireless Intrusion Detection System* (WIDS)), um computador (*Host Intrusion Detection System* (HIDS)) ou usar mais de uma técnica (*Mixed Intrusion Detection System* (MIDS)). Normalmente, utilizam uma base de dados com assinaturas das anomalias e realizam comparações com os dados gerados pelos equipamentos em tempo real (LIAO et al., 2013). *Denial of Service* (DoS), *Phishing* (JAIN; GUPTA, 2022), *Port Scanning* (VIVO et al., 1999) e *Spoofing* (YILMAZ; ARSLAN, 2015) são exemplos de ataques que as infraestruturas atuais são acometidas diariamente. Assim, a integração de sistemas e equipamentos de segurança buscam bloquear ataques, ou evitá-los, a aplicativos e infraestruturas. Além disso, recentemente, pesquisas integram a IA aos novos sistemas que previnem ataques cibernéticos (SHAIKH; HAIDER, 2011; ZEYU et al., 2020).

A integração entre a IA e os IDSs busca tornar esses sistemas mais leves e mais precisos, por não usar uma grande base de dados. O ML e o DL são exemplos de subáreas da IA (MELLIT; KALOGIROU, 2008) que utilizam métodos de aprendizado supervisionado, e a partir da entrada de dados e dos parâmetros selecionados são capazes de predizer, por meio de diferentes técnicas, o valor provável da saída desejada. Assim, quando ML e DL são associados aos IDSs, busca-se um sistema mais eficiente, eficaz e com menos interação humana, capaz de detectar ataques com maior precisão e com menor índice de falsos positivos (GE et al., 2019; ESKANDARI et al., 2020). Logo, desenvolver, implementar e analisar os impactos de IDSs baseados em ML e DL capazes de serem usados na computação em nuvem e na computação de borda é fundamental para mensurar a viabilidade desses IDSs em ambientes de produção em futuras implementações.

1.1 Motivação

A computação em nuvem permite centralizar o processamento, as tecnologias e as técnicas, facilitando a gestão da infraestrutura e aumentando a escalabilidade, a flexibilidade, a redundância, a elasticidade, a integridade (WANG et al., 2010). Já a computação de borda permite levar o processamento dos dados para mais próximo do cliente, diminuindo assim a latência na comunicação entre os serviços disponíveis na computação em nuvem, considerando que os dados serão processados mais próximos da origem geradora. As principais diferenças entre os dois ambientes se complementam, por exemplo, alto *jitter* (flutuação ou variação da latência ao longo do tempo), uso do modelo centralizado e localização desconhecida dos *datacenters* são características da computação em nuvem. Por outro lado, o baixo *jitter*, modelo distribuído e localização conhecida dos dispositivos na borda são características da computação de borda (KHAN et al., 2019). Assim, a integração dessas tecnologias, permite usar o alto poder de armazenamento e processamento centralizado da computação em nuvem com baixo *jitter* (e consequentemente, baixa latência) e alta velocidade de comunicação da computação de borda, com parte do processamento e armazenamento mais próximo do usuário final.

A segurança da informação acompanha o crescimento tecnológico visando mitigar ameaças e vulnerabilidades presentes nos ambientes de nuvem e de borda. Os IDSs são programas que possuem uma base de conhecimento predeterminada que podem ser baseadas em assinatura, regra e estado de conexão (SABAHI; MOVAGHAR, 2008). Estudos recentes buscam tornar os IDSs mais eficientes e eficazes, aumentando a taxa de acerto e diminuindo a quantidade de falsos positivos. Para isso, técnicas de ML e DL são empregadas para criar novos classificadores capazes de atuarem como o *core* dos IDSs atuais (GE et al., 2019; ESKANDARI et al., 2020). Porém, diferentes IDSs implementados em diferentes equipamentos consomem diferentes recursos computacionais. Um IDS pode ser eficiente em um determinado equipamento, no entanto, pode esgotar todos os recursos disponíveis em outro, principalmente se este equipamento possuir recursos limitados.

Equipamentos usados pelo ecossistema IoT e computação de borda normalmente possuem recursos limitados, necessitando de atenção especial antes da implantação de novos sistemas, pois o mau dimensionamento pode acarretar em uma sobrecarga e, assim, causar a indisponibilidade do ambiente (FERNÁNDEZ et al., 2018). Um IDS deve ser capaz de realizar a detecção de intrusão de maneira eficiente sem impactar o funcionamento

do equipamento, mas modelos de DL complexos algumas vezes podem consumir recursos demasiadamente. Dessa forma, novos IDSs baseados em IA podem consumir recursos essenciais, levando inclusive ao travamento dos equipamentos. Além disso, novos IDSs devem ser interoperáveis, ou seja, ter a capacidade de atuar na nuvem computacional ou na borda da rede. Neste contexto, a avaliação de desempenho de IDSs baseados em IA podem ajudar na tomada de decisão de futuras implementações, pois a depender do ambiente e dos recursos disponíveis, esses IDSs podem causar a indisponibilidade dos equipamentos, e consequentemente, do ambiente.

1.2 Trabalhos Relacionados

Esta seção apresenta pesquisas encontradas na literatura que possuem relação com esta dissertação. Com o crescente estudo nas áreas de computação em nuvem e computação de borda, a segurança da informação, mais especificamente IDSs, torna-se assunto indispensável. Associado a isso, a IA está inserida nos IDSs buscando tornar o ambiente mais eficiente, e também, com menos interação humana. Dessa forma, para uma melhor compreensão, os trabalhos relacionados são organizados em três seções. Primeiro, trabalhos relacionados à **computação em nuvem** são apresentados. Em seguida, os trabalhos relacionados à **computação de borda** são detalhados. Ao final, **uma comparação dos trabalhos** é apresentada, destacando assim, a relevância da pesquisa realizada nesta dissertação.

1.2.1 Computação em Nuvem

O uso da computação em nuvem, assim como suas técnicas e infraestruturas, é bastante difundida atualmente. Alinhado a isso, novas tecnologias são desenvolvidas e implementadas para tal infraestrutura. A seguir, são apresentados os principais trabalhos que abordam a computação em nuvem, normalmente integrado à IA.

Wan et al. (2018) apresentaram uma arquitetura denominada CaSF (*Cloud-assisted Smart Factory*) baseado em *Cyber-Physics Systems* (CPS) que, aprimora um processo da industria usando IA implementada na nuvem. O estudo propôs a implementação de quatro camadas, são elas: camada de dispositivo inteligente, camada de rede, camada de nuvem e

a camada de aplicação. Essas, correspondem aos recursos físicos de fabricação inteligente, redes de sensores sem fio industriais, plataformas de nuvem e serviços de aplicativos do sistema, respectivamente. A facilidade no gerenciamento da nuvem computacional e o poder centralizado capaz de executar atividades distribuídas, paralelas e complexas integradas à IA, foi crucial para a eficiência do processamento dos dados e da qualidade do serviço da plataforma, segundo os autores. Ao final, a computação em nuvem integrada à IA (usando, por exemplo, métodos como Convolutional Neural Network (CNN), Generative Deep Neural Network (GDNN) e Deep Learning) proposta pela arquitetura CaSF buscou tornar a indústria tradicional em um ambiente dinâmico, extensível e reconfigurável, ou seja, altamente resiliente.

Algumas das ameaças e vulnerabilidades mais recorrentes (e comumente encontradas) na computação em nuvem foram apresentadas a partir de um estudo realizado por Suryateja (2018). Apesar da facilidade no gerenciamento, integração e centralização dos dados, e a entrega de serviços como IaaS, PaaS e SaaS, a segurança da informação é parte fundamental no momento de decidir sobre a migração para o ambiente em nuvem. Violação e perda dos dados, sistemas de APIs vulneráveis, autenticação fraca e gerenciamento de identidade, sequestro de conta são exemplos de ameças que foram detalhadas no estudo e podem acometer uma nuvem computacional. Cada ameaça pode ser explorada por diversas vulnerabilidades, necessitando assim de um acompanhamento rígido e constante na computação em nuvem. Ao final, o estudo afirma que a migração de algumas áreas de negócio para a computação em nuvem está diminuindo (considerando a segurança da informação aplicada no ambiente), e também, exibe uma compilação das ameças e vulnerabilidades discutidas em todo o trabalho buscando alertar sobre o ambiente em nuvem.

Chiba et al. (2019) propuseram o desenvolvimento de um IDS baseado em anomalias, que usa ML, capaz de funcionar em nuvens computacionais denominado de *Machine Learning Intrusion Detection Systems* (MLIDS) que usa três *datasets* conhecidos (CICIDS2017, NSL-KDD e CIDDS-001). A pesquisa utilizou um framework híbrido que combina técnicas de ML (Improved Genetic Algorithm (IGA) e Simulated Annealing Algorithm (SAA)) e *Deep Neural Network* (DNN). O ML buscando os melhores parâmetros para ser usado na DNN. Além disso, o simulador de computação em nuvem CloudSim 4.0 foi usado para simular e validar o IDS proposto. Todo o pré-processamento, tipos de

ataque, método de treinamento e validação do modelo foi descrito para os três *datasets* selecionados. Ao final, os IDSs obtiveram a métrica acurácia superior a 99%. Além disso, obtiveram melhor desempenho ao ser comparado com trabalhos semelhantes. Assim, apesar da complexidade, o estudo usou a seleção automática para os melhores atributos e obteve melhores acurácias em relação a trabalhos equivalentes.

Usando análise de dados e ML (mais especificamente, aprendizagem em conjunto), Li et al. (2021) propuseram um método capaz de detectar código de mineração malicioso em nuvem por meio da fusão dos algoritmos de *Bagging* e *Boosting*. O mecanismo de mineração maliciosa é composto por um script incorporado na página de internet, e ao carregá-la o navegador executa o *script*, fazendo com que ocupe muitos recursos computacionais para a mineração (o script pode inclusive conter a execução de um Trojan). Inicialmente, há o pré-processamento dos dados, e posteriormente, o método extrai e vetoriza as características aplicando o algoritmo TF-IDF (Term Frequency - Inverse Document Frequency) combinado com o N-GRAM, assim, o modelo de detecção de código de mineração malicioso é construído. Após os experimentos, o estudo pôde comprovar, baseado nas métricas utilizadas no trabalho, que o modelo proposto obteve boa precisão na detecção de código de mineração malicioso.

Qureshi et al. (2021) propuseram um sistema de detecção de anomalias baseado em redes definidas por software (SDN-ADS) para arquiteturas baseadas na computação em nuvem para redes de IoT. Além disso, também propuseram uma autoridade confiável para computação de borda (TA-Edge) capaz de garantir a confiabilidade dos dispositivos de borda que encaminham os dados pela infraestrutura. A descrição da arquitetura, APIs, fluxogramas e algoritmos são detalhados, do SDN-ADS até a TA-Edge. Os sistemas propostos são avaliados por meio de diversas métricas de desempenho como atraso, probabilidade de detecção, tempo de processamento e taxa de transferência de dados, e indicam que, comparando com os sistemas existentes (FFFSM e NIDPs) o SDN-ADS possui melhor desempenho em todos os cenários avaliados.

1.2.2 Computação de Borda

Apesar do paradigma da computação de borda ser, relativamente, recente, diversas pesquisas buscam expandir, aprimorar e desenvolver novas aplicações que mitigam a

latência na comunicação entre a borda da rede e os *datacenters*. Assim, esta seção destaca os trabalhos que estão inseridos no contexto da computação de borda, normalmente integrado à IA.

Xu et al. (2020) apresentaram o paradigma da computação de borda com três principais desafios (ambientes distribuídos, ambientes heterogêneos e a computação limitada na borda da infraestrutura). Além disso, destacaram o surgimento da IA atrelada à segurança da informação, salientando três grandes áreas de atuação (detecção de intrusão, preservação da privacidade e o controle de acesso). Os autores descrevem os cenários e serviços utilizados em uma infraestrutura de IoT, mostrando as vantagens na utilização da computação de borda, e também, exibiram um *framework* dividido em camadas, integrando os serviços de IoT à computação de borda. Assim, os autores apresentaram como preservar a privacidade para os serviços de IoT na computação de borda usando IA, descrevendo métodos de criptografia tradicional e métodos usando IA (mais especificamente técnicas de *Obfuscation Neural Network* (ObfNet), CNN e DNN). Por fim, também foi contextualizada a utilização do *Blockchain* (que é um serviço distribuído o qual fornece uma nova abordagem para preservar e transmitir dados) a serviço da IoT utilizando computação de borda e IA.

Eskandari et al. (2020) apresentaram um IDS, denominado PASSBAN, baseado em IA e capaz de ser implementado em dispositivos IoT (normalmente em um *gateway* IoT) com o objetivo de proteger o ambiente mais próximo ao cliente, conforme preconiza o paradigma da computação de borda. O estudo usou duas técnicas de ML denominadas de *Isolation Forest* (IF) e *Local Outlier Factor* (LOF) para treinar os modelos de IA. Dois experimentos utilizando uma Raspberry Pi 3 executando o software *gateway* AGILE conectados aos sensores (usados em infraestruturas IoT) foram realizados buscando avaliar o desempenho do IDS proposto. Ao final, os resultados mostraram que, o PASSBAN IDS obteve 79% e 99% no pior e no melhor caso na detecção de ameças. Além disso, obteve bom desempenho de hardware (considerando o consumo de *Central Processing Unit* (CPU) e da memória *Random Access Memory* (RAM)). Vale ser ressaltado que os experimentos foram realizados em um dispositivo de baixo poder computacional, dessa forma, IDSs baseados em ML implementados em equipamentos bom baixo poder computacional podem impactar em seus recursos disponíveis;

Arshad et al. (2020) conduziram um estudo em IDSs existentes para IoT, com a finalidade de avaliar métricas relacionadas à sobrecarga computacional, consumo de

energia e privacidade. O estudo divide as principais ameaças em dois grandes grupos, em que o primeiro está relacionado aos ataques que podem ocorrer diretamente no tráfego de dados de uma infraestrutura IoT. Já o segundo, está relacionado às vulnerabilidades que podem ser exploradas nas aplicações executadas nos dispositivos IoT. Os autores também conduziram um estudo baseado nos IDSs existentes para IoT que considera o desempenho, o tipo de detecção, os tipos de ataques, a escalabilidade e outras métricas identificando desafios ainda não explorados considerando a restrição de recursos, complexidade dos ataques e dados de segurança relevantes.

Dai et al. (2019) propuseram uma nova arquitetura para orquestrar dinamicamente recursos da computação de borda usando cache e algoritmos de IA voltados para redes veiculares. A arquitetura proposta busca realizar uma comunicação entre as estações de comunicação associada ao uso de cache, e consequentemente, diminuir a latência na comunicação entre os veículos e as estações. A IA foi utilizada para realizar a orquestração inteligente para a computação de borda veicular usando dados em tempo real e dados armazenados em cache. Assim, o *Deep Deterministic Policy Gradient* (DDPG) é proposto para maximizar o ambiente de aprendizagem e a memória cache na borda da rede, tornando a alocação de recursos mais rápida e eficiente. Ao final, os resultados mostram que a proposta obteve eficácia, considerando a análise dos modelos propostos.

Já Ge et al. (2019) propuseram a criação de dois modelos de DL usando a técnica de FNN para o desenvolvimento de dois IDSs capazes de realizar a classificação binária e multiclasse de tráfego de dados em dispositivos de IoT. O trabalho detalha como realizou a seleção dos dados e realizou o pré-processamento dos atributos usando o *dataset* Bot-IoT (por ser um *dataset* recente e trabalhar com IoT). Além disso, propõem um *framework* para análise dos dados. Todo o experimento foi conduzido na plataforma do Google Colaboratory usando as bibliotecas do *Tensorflow*, que são capazes de executar modelos de DL. O *framework* descreveu dois modelos com três camadas, sendo as duas primeiras camadas contendo 512 neurônios e utilizando a função de ativação *Rectified Linear Unit* ($ReLU$) e a última camada com a função de ativação *Sigmoid* e *Softmax*, ambos com o algoritmo de otimização *Adam*. Ao final, os autores obtiveram uma acurácia de 99,41% para ataques de DoS e *Distributed Denial of Service* (DDoS), os quais podem direcionar pesquisas futuras no desenvolvimentos de IDSs inteligentes.

Otoum et al. (2022) propuseram um *framework* com um IDS baseado em DL

para atuar e prevenir ataques no contexto de IoT, denominado DL-IDS (*Deep Learning-based Intrusion Detection System*). O estudo utilizou o *dataset* NSL-KDD, o algoritmo *Spider Monkey Optimization* (SMO) para selecionar as características mais relevantes e o algoritmo *Stacked-Deep Polynomial Network* (SDPN) para realizar a classificação dos dados (se tráfego malicioso ou não), considerando os ataques de DoS, *Probing*, *U2R* e *R2L*. Ao final dos experimentos, o modelo proposto obteve 99,02%, 99,38%, 98,91% e 99,14% para as métricas acurácia, precisão, revocação ou sensibilidade (do inglês, *recall*) e pontuação F1 (do inglês, *F1 Score*).

1.2.3 Comparação

Os trabalhos apresentados anteriormente estão diretamente conectados à computação em nuvem, à computação de borda e à segurança da informação. Mais especificamente à IDSs, usando a IA. As propostas de implementação vão de aplicações na indústria, como a proposta de Wan et al. (2018), até as redes veiculares, como apresentado no trabalho de Dai et al. (2019). Suryateja (2018) analisou as ameaças e vulnerabilidades mais comuns em nuvens computacionais e Xu et al. (2020) conduziram um estudo capaz de identificar as principais ameaças ao ecossistema de computação de borda e IoT. Chiba et al. (2019) propuseram um IDS baseado em ML denominado de MLIDS (os experimentos nesse estudo foram reproduzidos em simuladores) ao tempo em que Eskandari et al. (2020) desenvolveram um IDS baseado em ML denominado PASSBAN, cujo primeiro (MLIDS) foi criado especificamente para atuar na computação em nuvem e o segundo para atuar na computação de borda (PASSBAN). Além disso, técnicas de DL são usadas para propor novos IDSs, como é o caso dos estudos conduzidos por Ge et al. (2019) e Otoum et al. (2022) que, puderam apresentar uma proposta utilizando FNN e SDPN, respectivamente. No entanto, é notável a falta de experimentos em dispositivos reais, assim como faz-se necessário avaliar métricas computacionais (como o consumo de CPU e memória RAM) para avaliar o desempenho dos modelos de IA de modo mais amplo.

Ademais, para uma melhor visualização, a Tabela 1 exibe e compara os trabalhos relacionados nesta dissertação em função de algumas características (IDS, contexto da computação, técnica de IA e ambiente de experimentos). Na Tabela 1, a coluna Sistemas de detecção de intrusão aponta se o estudo propôs a criação ou melhoria no referido sistema

de segurança. As colunas Computação em nuvem e Computação de borda assim como as colunas *Machine Learning* e *Deep Learning* indicam em qual contexto o estudo foi proposto. A coluna Experimentos em ambiente real define se o estudo executou experimentos em ambientes reais. Ao final da tabela, características referentes a esta dissertação também são adicionadas, a fim de facilitar a comparação com os trabalhos relacionados que foram selecionados.

Tabela 1 – Comparação entre os trabalhos relacionados.

Trabalhos	**Sistema de detecção de intrusão**	**Nuvem computacional**	**Borda computacional**	**Técnica de Machine Learning**	**Técnica de Deep Learning**	**Experimentos em ambiente real**
(WAN et al., 2018)	-	Não Especificada	-	Método Supervisionado, Semi e Não Sup.	CNN, DNN	Não
(SURYATEJA, 2018)	-	Não Especificada	-	-	-	Não
(CHIBA et al., 2019)	Baseado em Anomalia	GCP	-	IGA, SAA	DNN	Não
(LI et al., 2021)	-	Não Especificada	-	TF-IDF, N-GRAM	-	Não
(QURESHI et al., 2021)	Baseado em Anomalia	Não Especificada	Não Especificada	-	-	Não
(XU et al., 2020)	-	-	Não Especificada	-	ObfNet, CNN, DNN	Não
(ESKANDARI et al., 2020)	Baseado em Anomalia	-	Não Especificada	IF, LOF	-	Sim
(ARSHAD et al., 2020)	-	-	Não Especificada	-	-	Não
(DAI et al., 2019)	-	-	Borda Veicular	-	DDPG	Não
(GE et al., 2019)	Baseado em análise de rede	-	Não Especificada	-	FNN	Não
(OTOUM et al., 2022)	Baseado em Anomalia	-	Não Especificada	-	SMO, SDPN	Não
Este trabalho	Baseado em análise de rede	Sim	Sim	NB, RF	FNN	Sim

1.3 Objetivos

As ameaças cibernéticas em torno da computação em nuvem e computação de borda tornaram pesquisas referente à segurança da informação fundamentais, pois ambientes grandes e complexos normalmente possuem ameças e vulnerabilidades (SURYATEJA, 2018; ESKANDARI et al., 2020). As implementações de IDSs inteligentes buscam ser soluções de fácil manutenção, com maior precisão e menor consumo de recursos, ou seja, buscam ser mais eficientes, eficazes, robustas e com menos interação humana. Dessa forma, este trabalho tem como objetivo principal avaliar o desempenho de diferentes sistemas de detecção de intrusão baseados em IA em ambientes de computação em nuvem e computação de borda, que sejam voltados ao ecossistema de IoT. Mais especificamente, analisar o desempenho de tais ambientes, levando em consideração o consumo de CPU e memória RAM, com diferentes CTs e avaliar os *trade-offs* entre desempenho e acurácia utilizados em tais ambientes. Dessa forma, espera-se que esta dissertação auxilie futuras implementações de IDSs baseados em ML e DL para o ecossistema de IoT, pois dependendo da complexidade e recursos disponíveis tais IDSs podem impactar negativamente o funcionamento dos dispositivos de IoT (consumindo todos os recursos e levando a indisponibilidade do dispositivo).

Os objetivos específicos desta dissertação, são:

- Implementar IDSs usando modelos de ML e DL nos ambientes de computação em nuvem e computação de borda;
- Exibir e executar experimentos em ambientes reais de computação em nuvem e computação de borda;
- Analisar os resultados e avaliar os *trade-offs* entre desempenho dos dispositivos e a acurácia de modelos de ML e DL nos ambientes propostos.

1.4 Organização da Dissertação

A dissertação está organizada conforme segue. O Capítulo 2 introduz os principais conceitos utilizados neste trabalho. O Capítulo 3 apresenta a metodologia adotada. O Capítulo 4 exibe as arquiteturas experimentais adotadas e o método de execução dos experimentos. O Capítulo 5 avalia os *trade-offs* dos sistemas de detecção de intrusão adotados nos ambientes de computação em nuvem e computação de borda. Por fim, o

Capítulo 6 apresenta as contribuições, limitações e os possíveis trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta os principais conceitos utilizados nesta dissertação. Inicialmente, os fundamentos sobre a computação em nuvem e a computação de borda são apresentados. Em seguida, os conceitos sobre a segurança da informação são detalhados, aprofundando o conhecimento sobre os sistemas de detecção de intrusão. Posteriormente, os conceitos base da Inteligência Artificial são expostos. Mais especificamente, as técnicas de Machine Learning (*Naive Bayes* e *Random Forest*) e Deep Learning (*Feedforward Neural Network*) são apresentados. Por fim, uma explanação sobre avaliação de desempenho é realizada.

2.1 Computação em Nuvem

O paradigma da computação em nuvem permite integrar tecnologias e oferecer serviços aos clientes cobrando taxas pela demanda utilizada. Os conceitos para os modelos de negócio da computação em nuvem definido pelo NIST (BADGER et al., 2012), de modo sucinto, são:

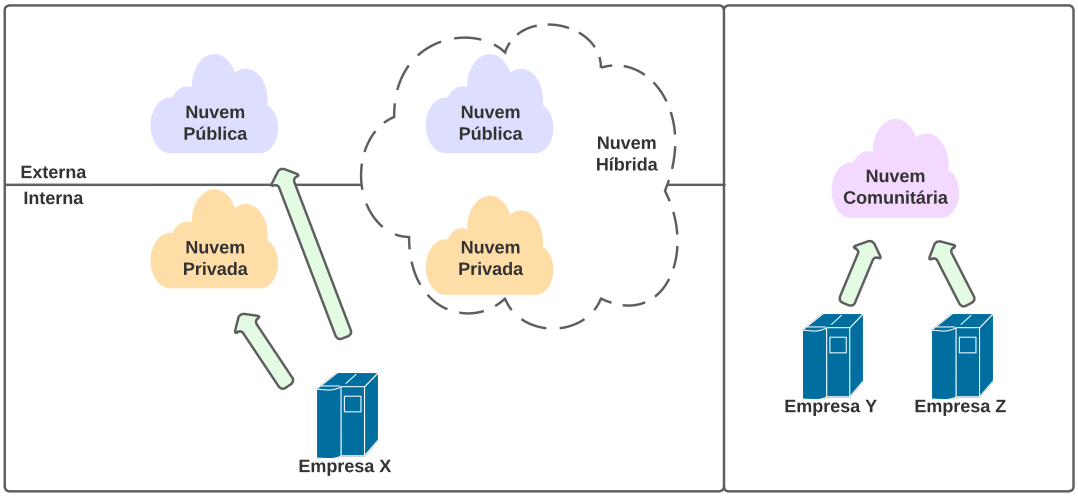
- **SaaS.** O usuário ou consumidor do serviço utiliza a infraestrutura do provedor para consumir aplicações disponíveis na nuvem;
- **PaaS.** O usuário ou consumidor do serviço utiliza a infraestrutura da nuvem por meio de bibliotecas de linguagens de programação suportadas pelo provedor para utilizar os recursos da nuvem, a qual a aplicação foi desenvolvida. Mas, há limitações de modo que o usuário não pode administrar e nem controlar a rede, servidores e outros recursos do *datacenter*, por exemplo;
- **IaaS.** O usuário ou consumidor do serviço utiliza o acesso ao provisionamento de processamento, armazenamento, redes e outros recursos de computação fundamentais.

Além disso, o conceito de computação em nuvem permite entregar um serviço: (i) **escalável**, por possuir a capacidade de crescimento estável e programado, em que normalmente é mais gradual e previsível; (ii) **elástico**, por se adaptar em situações momentâneas e abruptas; (iii) **disponível**, pois o esperado para uma nuvem computacional é estar operante vinte e quatro horas por dia e sete dias por semana; (iv) **confiável**, por

entregar exatamente o que é esperado satisfatoriamente por um período de tempo específico; e (v) **seguro**, por obter técnicas e sistemas de segurança capazes de fornecer ao contratante da nuvem computacional, ou ao cliente, um ambiente seguro (DANTAS, 2018).

As nuvens computacionais podem ser tipificadas como (ver Figura 1): **Pública**, que é uma infraestrutura de nuvem que pode ser controlada e administrada por uma empresa ou organização, porém fornece recursos para uso aberto ao público geral. Além disso, a empresa provedora determina as normas e regulamenta o uso. **Privada**, que é uma infraestrutura de nuvem que pode ser controlada, gerenciada e administrada integralmente por e para uma empresa. **Híbrida**, que é a combinação de mais de uma infraestrutura acima citada (Privada e Pública) com forte integração de padrões e normas, garantindo assim a portabilidade dos dados. **Comunitária**: que é a infraestrutura de nuvem que pode ser controlada e administrada por mais de uma empresa ou organização e atende a uma comunidade específica (BADGER et al., 2012).

Figura 1 – Tipos de Nuvens.



Fonte: do Autor.

Virtualização, Orquestração de serviços, *Web Services* e Serviços Orientados a Arquitetura (SOA) são tecnologias que as nuvens computacionais agregam e centralizam em seus *datacenters* (WANG et al., 2010). Essa centralização permite entregar aos clientes alto poder de processamento, armazenamento e taxa de transferência. Atualmente, existem *datacenters* replicados em diferentes continentes que podem ser acessados por meio de um dispositivo com um navegador conectado à Internet. Tal replicação busca minimizar a latência na comunicação cliente/servidor e aumentar a redundância dos dados, aplicações

e outros recursos quando o ambiente a ser configurado necessitar desses requisitos. No entanto, novas aplicações necessitam de alta velocidade na comunicação, exigindo assim que parte das aplicações sejam processadas ou armazenadas mais próximas à fonte dos dados, ou seja, mais próximo do cliente.

2.2 Computação de Borda

A computação de borda pode ser dividida em três grandes áreas de pesquisa: (i) **Fog Computing** (FG); (ii) **Cloudlets**; e (iii) **Mobile Edge Computing** (MEC). A primeira, foi proposta pela Cisco (SOLUTIONS, 2015) e preconiza que os aplicativos sejam executados diretamente na borda da rede por meio de outros dispositivos interconectados como uma grande rede interligada. A segunda tem por objetivo colocar equipamentos com grande poder de processamento e armazenamento próximo aos usuários (ou dos dispositivos de borda), habilitando serviços da nuvem próximos aos usuários. E a terceira, tem por objetivo permitir que softwares, processamento e armazenamento de dados possam ser realizados próximo aos usuários finais (mas sem a necessidade de grandes equipamentos como preconiza o *Cloudlets*) (MACH; BECVAR, 2017; KHAN et al., 2019).

O paradigma da computação de borda ganhou notoriedade após o alto número de dispositivos e aplicações associados à IoT. Esses dispositivos geram grandes massas de dados para serem armazenados ou processados. No entanto, algumas aplicações podem exigir curtos tempos de resposta no processamento, o que inviabiliza a transmissão dos dados pela Internet, necessitando assim, de um armazenamento ou processamento mais próximo do cliente.

Logo, a computação de borda (mais especificamente a MEC) permite processar os dados na borda da rede, mais próximo da fonte de dados, buscando maior agilidade e menor latência aos sistemas, pois a computação em nuvem nem sempre é eficiente para o processamento de dados quando os dados são produzidos na borda da rede (SHI et al., 2016). Um exemplo prático e simples da computação de borda é o uso de *Content Delivery Network* (CDN) por provedores de conteúdo, de modo que eles buscam armazenar os vídeos mais assistidos em um *cache* na borda da rede.

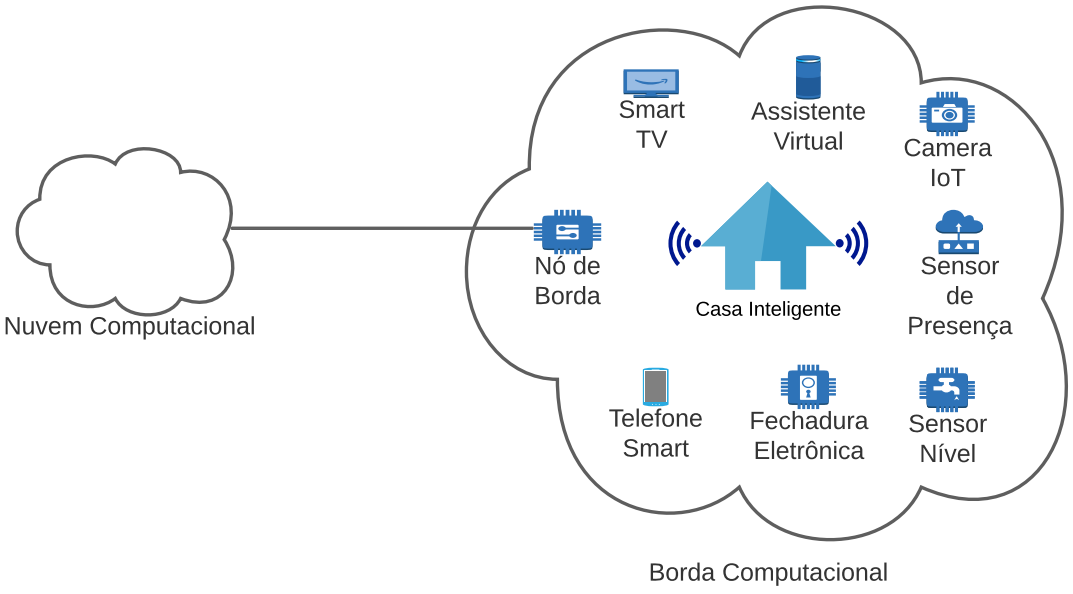
Khan et al. (2019) define as principais características da computação de borda, que a tornam única:

- **Distribuição geográfica densa:** a densa distribuição geográfica da infraestrutura pode auxiliar os administradores de redes, que por sua vez, podem realizar ações próximo ao usuário sem precisar atravessar a *Wide Area Network* (WAN). A análise de grandes volumes de dados pode ser realizado com maior velocidade e precisão na borda da rede, permitindo, inclusive, análise em tempo real em larga escala.
- **Suporte a mobilidade:** o protocolo *Locator ID Separation Protocol* (LISP) dá suporte à mobilidade, permitindo a comunicação direta entre os dispositivos móveis. Implementando assim, um sistema de diretório distribuído, desvinculando a identidade do local da identidade do host (princípio chave que possibilita o suporte à mobilidade).
- **Conhecimento do local:** permite que usuários móveis acessem serviços de um servidor de borda mais próximo de sua localização. Infraestruturas de telefonia, *Global Positioning System* (GPS) ou pontos de acesso sem fio podem encontrar a localização de dispositivos na infraestrutura.
- **Proximidade:** a disponibilidade de recursos computacionais na vizinhança local permite que usuários aproveitem as informações de contexto da rede para tomar decisões de uso de um determinado serviço. Assim, o sistema pode aproveitar as informações do usuário no local, extraindo informações do dispositivo e analisando o comportamento do mesmo buscando melhorar os serviços e alocação de recursos.
- **Baixa latência:** considerando a proximidade dos recursos e serviços, a baixa latência permite que os usuários executem aplicativos com uso intensivo de recursos e serviços com pouco atraso (ou atraso aceitável) no dispositivo de borda.
- **Consciência do contexto:** a percepção do contexto é uma característica dos dispositivos móveis que pode ser definida independentemente do conhecimento do local. Ou seja, dependendo do contexto em que o usuário esteja em um determinado momento, poderá ocorrer uma descarga de informações massivas, independente de sua localização, mas que é conveniente pelo contexto em que o dispositivo está inserido. Ademais, a localização pode ser usada para fornecer serviços com o reconhecimento de contexto, alinhando assim as duas características.
- **Heterogeneidade:** a computação de borda é heterogênea pois permite plataformas, arquiteturas, infraestruturas, tecnologias da computação e comunicação variadas. A implementação bem sucedida da computação de borda parte do desafio da

interoperabilidade desse conjunto tecnológico. Logo, a heterogeneidade refere-se à diversidade de tecnologias que impactam a entrega do serviço.

A Figura 2 exibe uma casa inteligente com um equipamento (nó de borda) capaz de executar tarefas localmente, conforme preconiza a computação de borda. Pode ser observado que a residência pode estar demograficamente em qualquer local que tenha Internet. Na residência haverá baixa latência na comunicação entre os dispositivos de IoT. Assim, os dados mais importantes serão processados localmente no equipamento de borda (ou nó de borda) e posteriormente transmitidos para os *datacenters* ou nuvens computacionais. Os dispositivos de IoT buscam tornar a residência *"inteligente"* (por automatizar algumas ações), mas, para isso, eles geram grandes volumes de dados (EREMIA et al., 2017). Associado a esse novo contexto, novos sistemas de segurança precisam surgir (ou sistemas antigos precisam se adaptar) para tornar o ambiente em questão seguro.

Figura 2 – Computação de borda.



Fonte: do Autor.

2.3 A Segurança da Informação

A conexão entre milhões de dispositivos com alta velocidade, interoperando ambientes e sistemas faz com que criminosos possam se aproveitar de vulnerabilidades,

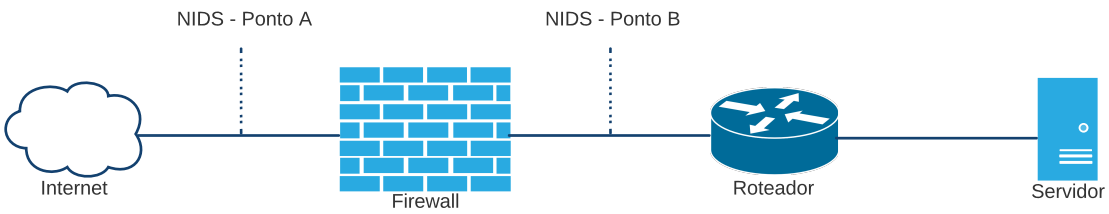
explorando, por exemplo, o ataque DDoS que busca tornar uma infraestrutura indisponível (LAU et al., 2000). Outro exemplo é o *Ransonware*, que encripta os dados dos usuários, por meio de um *malware*, e o grupo criminoso extorque a vítima (normalmente usando *bitcoin* ou outra criptomoeda) para decriptar os dados e assim liberá-los aos proprietários (BREWER, 2016).

Tais ameaças podem explorar as vulnerabilidades de protocolos bem estabelecidos ou aplicações inseguras. No entanto, normas técnicas (MERIAH; RABAI, 2019) e boas práticas (MOTII; SEMMA, 2017) buscam criar um modelo de gestão de tecnologia da informação íntegro e seguro. Dessa forma, integrar equipamentos e *softwares* (ex.: *firewall*, anti-vírus, IDS ou VPN) às normas técnicas e boas práticas na computação em nuvem, na computação de borda e em dispositivos de IoT buscando mitigar os riscos é fundamental para tornar um ambiente seguro (AHMAD et al., 2021; ESKANDARI et al., 2020).

A IA integrada à segurança da informação busca tornar o ambiente mais eficiente, eficaz, robusto e com menos interação humana, garantindo assim menor manutenção e maior precisão (GE et al., 2019). IDSs inteligentes integram os mais variados contextos de segurança da informação e são capazes de alertar possíveis ameaças, impedindo que uma vulnerabilidade possa ser explorada. Além disso, a localização de tais sistemas na infraestrutura pode variar, e assim, o resultado nas ações também.

Um IDS localizado antes de um *firewall* (ver Figura 3, ponto A) irá capturar muitos falsos positivos, mas trará uma informação bruta dos ataques (sem o bloqueio realizado pelo *firewall*). Já um IDS localizado após o *firewall* (ver Figura 3, ponto B) será capaz de detectar um possível ataque, e assim, um alerta será, provavelmente, uma ameaça real (pois o possível ataque passou pelo filtro do *firewall*). A localização no ponto A é muito utilizada para coleta e análise dos dados, e no ponto B é muito utilizada em produção. Logo, a localização estratégica do equipamento e as configurações fazem parte planejamento de um ambiente seguro e eficiente.

Figura 3 – Localização do IDS.



Fonte: do Autor.

2.3.1 Sistemas de Detecção de Intrusão

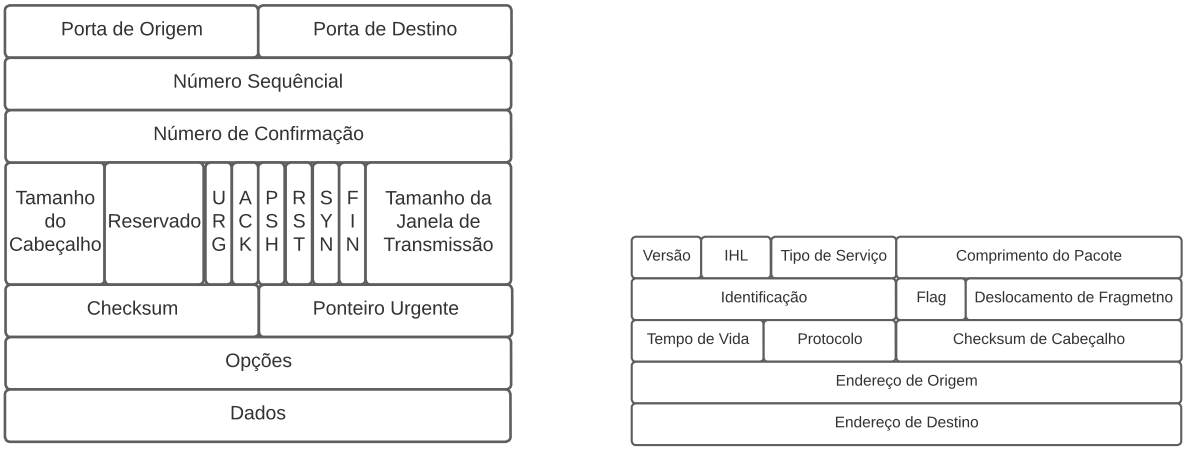
Os sistemas de detecção de intrusão podem ser baseados em: (i) ***host* (HIDS)**, em (ii) **rede (NIDS)**, (iii) **híbrido (HIDS)** ou (iv) **baseado no comportamento de rede (NBA)**. O primeiro, avalia as ações do computador para eventos suspeitos nesse dispositivo. O segundo, monitora o tráfego e os protocolos de rede a fim de identificar atividades suspeitas. O terceiro, utiliza as duas primeiras técnicas simultaneamente. O último caso, examina o tráfego de rede para identificar ameaças que geram fluxos de tráfego incomuns como ataques DDoS (SABAHI; MOVAGHAR, 2008).

As detecções podem ser baseadas em: (i) ***assinatura***, onde uma assinatura é um padrão que corresponde a um ataque ou ameaça conhecido. Dessa forma, padrões são capturados e comparados com eventos para reconhecer possíveis intrusões. (ii) ***Anomalia***, que é baseada em algum desvio do comportamento conhecido, ou seja, o monitoramento identificará um consumo demasiado do processador, falhas contínuas de login ou muitos e-mails enviados em um curto período de tempo, por exemplo. Por fim, (iii) ***na análise do estado de protocolos de rede***, onde normalmente, o estado da conexão de protocolos conhecidos são monitorados, ou seja, um possível alerta será realizado caso um dispositivo esteja realizando ações fora do padrão descrito nos protocolos (LIAO et al., 2013).

Os protocolos de rede possuem camadas que se comunicam entre si, onde cada camada possui campos com informações importantes para o estabelecimento de conexão, controle, fluxo de dados. Ao analisar o cabeçalho do protocolo TCP (POSTEL, 1981) (ver Figura 4(a)), pode-se observar que há campos de informações específicas (porta de origem, porta de destino, número sequencial) que são responsáveis para o funcionamento do protocolo TCP. Assim como o TCP, o protocolo IP (POSTEL, 1981) (ver Figura 4(b)) também possui campos responsáveis por fornecer informações específicas para seu funcionamento. A seleção desses campos são muito importantes para o desenvolvimento de um IDS baseado em rede e em comportamento de rede, pois é com base em tais campos que os IDS, normalmente, identificam padrões e são desenvolvidos.

A seleção de atributos (campos das camadas dos protocolos de redes) de maneira eficiente é fundamental para desenvolver um IDS baseado em IA com boa precisão (MOUSTAFA et al., 2018). Soe et al. (2019) utilizaram técnicas de IA para automatizar a seleção dos melhores atributos, minimizando assim a interação humana, gerando menos programação e buscando maior acurácia e precisão no treinamento dos

Figura 4 – Cabeçalho dos protocolos TCP e IP.



(a) Protocolo TCP. (b) Protocolo IP.

Fonte: Adaptado de (POSTEL, 1981) e (POSTEL, 1981).

modelos que poderão se tornar um IDS baseado em IA. Dessa forma, o desenvolvimento de novos IDSs integrados às técnicas de ML e DL são constantes. Busca-se maior precisão e menor consumo de recursos (computacionais ou humanos), seja com classificadores ágeis ou redes neurais complexas, sempre com o objetivo de mitigar ameaças a grandes infraestruturas com múltiplas camadas em ambientes atuais (ESKANDARI et al., 2020).

2.4 Inteligência Artificial

Pesquisas remetem a primeira definição de IA na Conferência de Dartmouth, em 1956 (MCCORDUCK; CFE, 2004; CREVIER, 1993). Antes disso, a inteligência humana sempre foi objeto de estudo, mas, a partir desse momento, o paradigma da inteligência humana foi estudada com o objetivo de reproduzir ações humanas em um computador, algumas vezes, buscando simular o comportamento humano, e assim, aprimorar a compreensão da cognição humana (NORMAN, 1991).

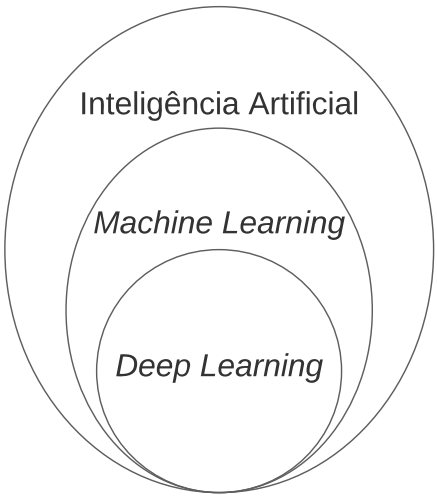
Arquiteturas cognitivas (que podem ser definidas como um sistema único capaz de produzir todos os aspectos do comportamento) são interdisciplinares e integradas, abrangendo normalmente as áreas de IA, psicologia cognitiva e neurobiologia (CHONG et al., 2007). No entanto, até 2012 o alcance dos estudos relacionados a IA estavam limitados a grandes empresas de tecnologia, governos e agências de pesquisa (ONGSULEE, 2017). Assim, apenas recentemente a grande área da IA ganhou força e está dentre as de maior interesse entre os pesquisadores da computação.

Atualmente, a IA está intimamente integrada a programas e sistemas computacionais modernos, tornando-os mais eficientes, fáceis de manusear e podendo auxiliar no dia a dia das pessoas que usam esses programas. Sistemas especialistas, visão computacional, mineração de dados e algoritmos genéticos são subáreas da IA que, no decorrer do tempo, foram aprimorando a grande área. Kumar e Venkataram (1997), por exemplo, propuseram usar IA no auxílio do gerenciamento de redes de computadores, ou seja, integrar um sistema baseado em IA para ajudar na gestão dos equipamentos de rede.

O ML é uma subárea da IA que possui técnicas e modelos capazes de aprender um determinado conteúdo a partir de grandes massas de dados (*datasets*), sem a necessidade de codificar um programa explícito (ONGSULEE, 2017). Posteriormente, as Redes Neurais Artificiais (RNA) surgiram (como uma subárea do ML) para tentar simular as redes neurais humanas. Em seguida, surgiu o DL que usa RNA com camadas ocultas integradas e são capazes de resolver problemas mais amplos e complexos (POUYANFAR et al., 2018).

A IA possui o ML como subárea, que por sua vez possui o DL como subárea (ver Figura 5). Tanto o ML como o DL utiliza um conjunto de dados iniciais para gerar correlação entre esses dados, e assim, produzir um modelo de IA capaz de predizer um possível resultado, criando dessa forma, um modelo com *"inteligência"*. Cada subárea usa uma técnica diferente e específica para o desenvolvimento do modelo de IA.

Figura 5 – A inteligência artificial e suas subáreas.

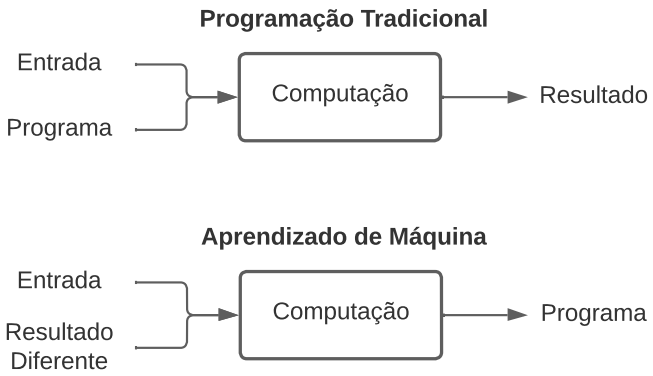


Fonte: do Autor.

2.4.1 Machine Learning

A principal diferença entre a programação tradicional e a programação usando ML pode ser observado na Figura 6, onde na programação tradicional, um programador escreve explicitamente um programa com várias condições e respostas. Por outro lado, o algoritmo baseado em ML infere as condições baseado nos exemplos de entrada, e assim, as regras são aprendidas buscando a melhor decisão. Logo, inferir automaticamente as ações sem a intervenção manual é a principal diferença entre o ML e a programação tradicional (CHINNAMGARI, 2019). O ML preditivo pode classificar um valor (A em detrimento de B, por exemplo), ou pode usar a regressão e predizer um determinado valor (um valor aproximado).

Figura 6 – Diferença entre ML e a programação tradicional.



Fonte: Adaptado de (CHINNAMGARI, 2019).

Ayodele (2010) afirma que os algoritmos são organizados em taxonomia, com base no resultado desejado do algoritmo, onde os mais comuns incluem:

- **Aprendizado supervisionado:** os dados de entrada são fornecidos com as soluções desejadas, ou seja, o modelo receberá os dados (com as características) de treinamento com o resultado esperado (haverá um rótulo indicando a solução desejada);
- **Aprendizado semi-supervisionado:** uma parte dos dados de entrada (normalmente poucos dados em relação ao todo) possuem a solução desejada, ou seja, parte da solução é rotulada com o que é esperado;
- **Aprendizado não supervisionado:** a correlação dos dados de entrada são encontradas sem a necessidade do rótulo indicando a solução;
- **Aprendizado por reforço:** um agente observará o ambiente, selecionará e executará as ações em troca de recompensas ou penalidades, ou seja, o agente irá aprender

baseado em uma política para obter o máximo de recompensas possíveis ao longo do tempo. A política descreverá a ação que o agente tomará em determinada situação;

- **Transdução:** é semelhante ao processo de aprendizado supervisionado. Mas, esse tenta prever novas saídas com base nas entradas de treinamento, saídas de treinamento e novas entradas;
- **Aprendizado baseado no aprendizado:** onde o algoritmo aprende, de maneira indutiva, com base na experiência anterior. Também conhecida com aprendizagem indutiva.

Não há um algoritmo de ML perfeito para ser usado em todas as situações. A escolha do algoritmo será feita baseada no contexto em que o problema está inserido. Ou seja, um algoritmo A pode ser mais eficiente que um algoritmo B para resolver determinado problema, mas ter pior desempenho em outros tipos de problemas. Este trabalho utiliza os algoritmos *Naive Bayes* e *Random Forest* por possuírem altas taxas de aprendizagem e bom desempenho (em relação ao tempo) na classificação de dados. Ambos são considerados algoritmos que usam o método de aprendizado supervisionado preditivo. A seguir, os dois algoritmos são detalhados.

2.4.1.1 Naive Bayes

Thomas Bayes (estatístico e filósofo) desenvolveu um teorema que foi publicado por Richard Price (matemático e filósofo), o qual foi denominado de *Teorema de Bayes*, no século XVIII. O teorema é capaz de predizer, usando a probabilidade, o resultado de um evento dado que outro evento tenha ocorrido anteriormente. O algoritmo NB usado em ML é baseado no teorema de bayes, que é escrito conforme a Equação 2.1 (ZHANG, 2005):

$$P(c|X) = \frac{P(X|c) \times P(c)}{P(X)} \tag{2.1}$$

onde,

- P(c|X) é a probabilidade do evento c dado o evento X ter ocorrido (probabilidade a posteriori);
- P(X|c) é a probabilidade do evento X dado o evento c ter ocorrido (probabilidade a

posteriori);

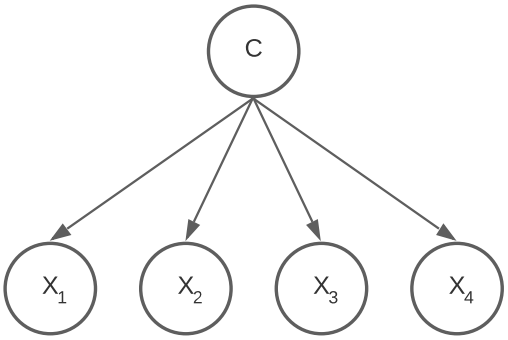
- P(c) é a probabilidade de c acontecer (a priori); e
- P(X) é a probabilidade de X acontecer (a priori).

A Equação 2.1 pode ser expandida, resultando na Equação 2.2:

$$P(c|X) = \frac{\prod_i P(X_i|c) \times P(c)}{\prod_i P(X_i)} \tag{2.2}$$

Dessa forma, o algoritmo de NB é considerado ingênuo por assumir a independência dos atributos relacionado à classe (o resultado) que se deseja predizer. Ou seja, todo atributo possui o mesmo peso. Além disso, é considerado descritivo com aprendizado supervisionado. Logo, recebe um conjunto de dados com atributos e seus valores, $(x_1, x_2, ..., x_n)$ onde o x_i é o valor do atributo X_i, e seja C a variável de classe, onde c é o valor de C (ZHANG, 2005). Isto é, por meio dos atributos X $(x_1, ..., x_n)$ o algoritmo poderá predizer o resultado (valor de c) que está contido em C. Normalmente, busca-se encontrar a classe mais provável de c $\in$ C (ver Figura 7).

Figura 7 – Um exemplo do Naive Bayes.



Fonte: Adaptado de (ZHANG, 2005).

O algoritmo de NB realiza o treinamento usando um *dataset* (conjunto de dados usados pelos algoritmos de IA), e assim, um modelo baseado no algoritmo é criado, capaz de predizer um resultado a partir da entrada de dados. Tal modelo cria uma tabela com as probabilidades baseada no teorema de Bayes (ver Figura 2, onde Ca significa Característica e Co, Condição).

Tabela 2 – Exemplo de tabela de probabilidades criada pelo NB.

Condição / Característica	**Característica01**	**Característica02**	**Característica03**	**Resultado**
Condição 01	Prob(Ca01;Co01)	Prob(Ca02;Co01)	Prob(Ca03;Co01)	Resultado Final
Condição 02	Prob(Ca01;Co02)	Prob(Ca02;Co02)	Prob(Ca03;Co02)	Resultado Final
Condição 03	Prob(Ca01;Co03)	Prob(Ca02;Co03)	Prob(Ca03;Co03)	Resultado Final
Condição 04	Prob(Ca01;Co04)	Prob(Ca02;Co04)	Prob(Ca03;Co04)	Resultado Final

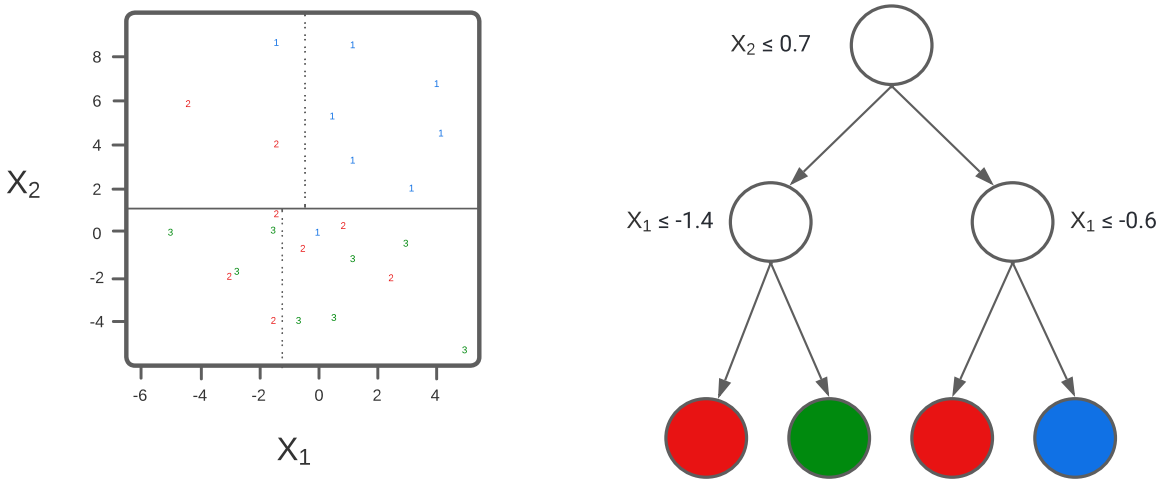
2.4.1.2 Random Forest

Com o objetivo de apresentar o algoritmo *Random Forest* (RF), faz-se necessário explicar primeiramente o algoritmo *Decision Tree* (DT). O algoritmo DT é capaz de realizar a classificação de um determinado conteúdo baseado em uma amostra (n) com classes (Y) e variáveis (X) com os respectivos valores ($X_1, ..., X_n$). O objetivo do algoritmo é prever os valores para as classes Y a partir dos novos valores de X. O método de classificação consiste em criar uma árvore de decisão a partir de condições específicas. Pode-se, inclusive, visualizar os dados por meio de um plano com os dados e as condições definidas na árvore de decisão, tal ato permite facilitar a interpretação dos dados (LOH, 2011).

A Figura 8(a) exibe os pontos de dados e as partições baseado em algumas condições, e a Figura 8(b) exibe a estrutura da árvore de decisão correspondente (LOH, 2011). Nesse exemplo, a árvore possui um nó raiz (a condição $X_2 \leq 0.7$), dois nós de decisão (as condições $X_1 \leq -1.4$ e $X_1 \leq -0.6$) e quatro nós folhas (as classes 1, 2 e 3, que são os resultados esperados). Além disso, vale ser ressaltado que há mais de um método para realizar a classificação por meio do algoritmo DT, como o *CHi-squared Automatic Interaction Detector* (CHAID) (KASS, 1980), o *Classification and Regression Tree* (CART) (LI et al., 1984) e o algoritmo C4.5 (XIAOLIANG et al., 2009). Neste trabalho, usamos o CART como método de implementação.

Com base no algoritmo DT, o algoritmo RF foi desenvolvido. Isto é, ao invés de usar apenas uma árvore de decisão, utiliza-se várias árvores, criando assim uma floresta. Pode-se considerar que o algoritmo RF é um melhoramento do algoritmo DT. O algoritmo RF seleciona aleatoriamente as variáveis que servirão de base para as árvores de decisão. Note que a quantidade de variáveis e árvores podem ser configuradas no momento da

Figura 8 – Exemplo de classificação das classes 1, 2 e 3

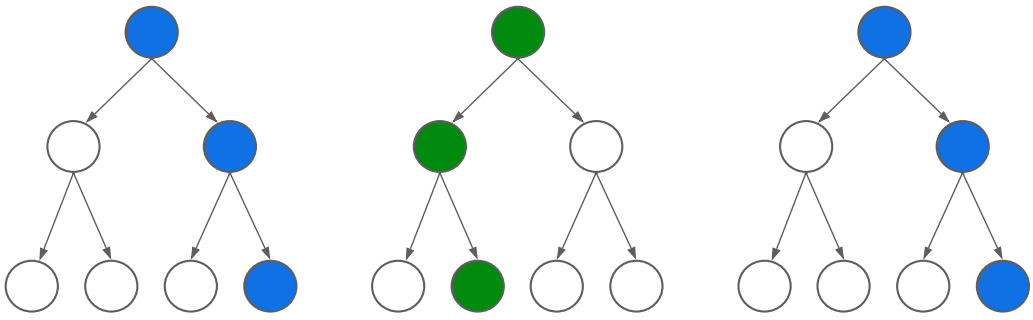


(a) Baseado em partições. (b) Baseado na árvore de decisão.

Fonte: Adaptado de (LOH, 2011).

utilização do algoritmo. Assim, o algoritmo RF criará a quantidade especificada de árvores de decisão para realizar a classificação de determinado conteúdo. Após a classificação, haverá um somatório dos resultados individualmente para cada árvore, e ao final, haverá a análise do resultado final. A Figura 9 exibe o conjunto de duas classes (azul e verde), duas características, e o resultado de três árvores. Nesse exemplo, ao final, o algoritmo RF obterá como resultado a classe azul, considerando que houve dois resultados para essa classe e um resultado para a classe verde.

Figura 9 – Resultado de um algoritmo RF.



Fonte: do Autor.

Os algoritmos DT e RF podem obter um resultado muito bom no treinamento do modelo. No entanto, ao executá-lo com dados de teste, o resultado pode não ser tão efetivo. Tal evento é chamado de *overfitting*. Esse fato ocorre quando o modelo treinado aprende muito bem com os dados de treinamento, mas não obteve um bom resultado com

os dados de teste ou os dados reais. Logo, após o treinamento dos modelos, é fundamental verificar a acurácia dos modelos com dados de teste e dados de validação. A divisão entre os dados de treinamento, teste e validação é utilizada nesta dissertação e será exibida nos próximos capítulos.

2.4.2 *Deep Learning*

As primeiras redes neurais remontam à década de 40, porém, os modelos não conseguiam aprender a contento. Os estudos relacionados às redes neurais com aprendizado supervisionado podem ser encontrados entre as décadas de 50 e 60. Os trabalhos de Ivakhnenko e Lapa (1965) e Ivakhnenko et al. (1967) produziram, possivelmente, os primeiros algoritmos de DL do tipo *Feedforward Multilayer Perceptron* (SCHMIDHUBER, 2015).

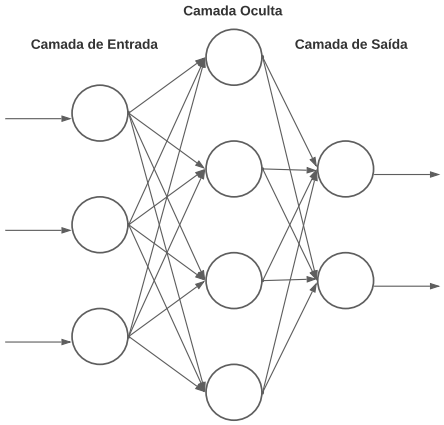
Um neurônio biológico é composto pelo corpo celular, dendritos, núcleo, axônio e ramificações terminais do axônio, enquanto um neurônio artificial é composto de diversos cálculos matemáticos. O neurônio artificial foi desenvolvido para simular a função do neurônio biológico, fazendo assim parte fundamental da RNA. Os sinais (os dados) que chegam, também podem ser chamados de entradas, são multiplicados pelos pesos de conexão (*weight* ou pesos sinápticos) e, em seguida, passam por uma função de transferência (também chamados de função de ativação) para produzir uma saída com o resultado do neurônio. Apesar de um único neurônio artificial poder executar tarefas simples de processamento de informação, o poder dos cálculos neurais vem de uma rede neural com múltiplas conexões, podendo criar uma RNA (AGATONOVIC-KUSTRIN; BERESFORD, 2000).

Assim como os algoritmos de ML, não há um algoritmo perfeito de DL capaz de resolver todo e qualquer problema. O contexto ao qual o problema está inserido será determinante para o desenvolvimento do algoritmo de DL. Este trabalho utiliza o algoritmo baseado em uma FNN por possuírem estudos com altas taxas de aprendizagem no contexto de IDSs específicos ao ecossistema IoT (GE et al., 2019).

Uma rede neural *feedfoward* (FNN) possui uma camada de entrada, uma camada oculta, e uma camada de saída com neurônios distribuídos em cada camada totalmente interligados (conectadas), conforme pode ser observado na Figura 10. O termo *feedforward*

é usado, pois os dados iniciais entram na camada de entrada passam pela camada oculta e saem na camada de saída apenas uma única vez. Em outras palavras, não há recorrência na passagem dos dados entre os neurônios, logo o modelo recebe uma informação, processa e gera uma saída. Por outro lado, um algoritmo de DL denso e profundo é caracterizado pela alta quantidade de neurônios e camadas ocultas, e pela ligação entre as camadas, normalmente, representada por um grafo.

Figura 10 – Um exemplo de uma *Feedfoward Neural Network*.



Fonte: Adaptado de (SVOZIL et al., 1997).

Além dos neurônios e das camadas, os pesos sinápticos, *bias* e funções de ativação também compõem a RNA, conforme foi falado acima. Os pesos, são números que expressam a importância da entrada (o dado) no neurônio. Já o *bias* é um parâmetro adicional que é usado para ajustar a saída do neurônio. Isto é, uma constante que ajudará o modelo a se adaptar melhor aos dados fornecidos. Após os dados passarem pelos neurônios, a função de ativação recebe os dados encaminhados pelos neurônios e transformam os dados não-lineares, tornando-o capaz de aprender e executar tarefas mais complexas. *Sigmoid*, *tanh* (KALMAN; KWASNY, 1992) e *ReLU* (HARA et al., 2015) são exemplos de funções de ativação utilizadas em redes neurais artificiais. Após o treinamento e aprendizado da RNA, o algoritmo é capaz de predizer um determinado resultado, como os algoritmos de Machine Learning NB e RF.

2.5 Avaliação de Desempenho

A avaliação de desempenho pode ser realizada através de duas diferentes técnicas: (i)**modelagem de desempenho**; e (ii) **medição de desempenho**. A modelagem de

desempenho é normalmente usada no início de algum projeto, quando os sistemas reais não estão em produção para poder haver uma medição. A modelagem de desempenho pode ser, ainda, subdividida em: (a) **simulação**; e (b) **modelagem analítica**. Os **simuladores** são um modelo do sistema o qual necessita ser reproduzido, normalmente escrito em uma linguagem de computador e executado em um computador. Já a **modelagem analítica** usa princípios matemáticos para criar modelos probabilísticos como os modelos de fila, modelos de Markov ou redes de Petri (JAIN, 2008).

Na **medição de desempenho**, considera-se um ambiente real em produção, ou um protótipo, para realizar a medição do ambiente ou sistema, por meio de hardwares ou softwares. Avaliar o desempenho (ou medir o desempenho) de um ambiente real é fundamental para entender o seu comportamento, além de ser mais preciso, porém o custo dos experimentos é mais alto. Além disso, a medição de desempenho de ambientes reais, ou de um protótipo, pode ajudar no desenvolvimento de projetos maiores que podem ser implementados no futuro (JOHN; EECKHOUT, 2018).

Erros comuns podem ocorrer na medição de desempenho, já que cada sistema a ser medido consiste em um projeto único. Alinhado a isso, Jain (2008) apresenta etapas comuns a todos os projetos de avaliação de desempenho que podem ajudar a evitar tais erros comuns. Selecionar as métricas desejadas, escolher técnica de avaliação, a Carga de Trabalho (CT), analisar e interpretar os dados, e apresentar os resultados são algumas etapas que podem ajudar a mitigar os erros. Vale ser ressaltado, ainda, que softwares e hardwares podem ser usados para realizar a medição de desempenho e salvar os resultados em arquivos para posterior análise.

No contexto de ML e DL, estudos (KOCHER; KUMAR, 2021; CHOUDHURY; BHOWAL, 2015; KHAN; GUMAEI, 2019) buscam avaliar o desempenho através de métricas que aferem a precisão dos modelos (além do tempo gasto para realizar a referida atividade), são elas: acurácia, precisão, revocação, pontuação F1, *Receiver Operating Characteristic* (ROC), e *Area Under the Curve* (AUC). Evidenciando assim que a seleção das métricas são parte fundamental no desenvolvimento de um projeto que busca avaliar o desempenho de um ambiente ou sistema. No entanto, vale ser destacado ainda que não é interessante negligenciar as demais etapas.

Considerando novos IDSs baseados em IA, medir o desempenho em ambientes reais pode ajudar no constante desenvolvimento e futuras implementações. Acurácia, precisão,

consumo de memória RAM e CPU são métricas que podem nortear a eficiência e eficácia dos sistemas e ambientes avaliados. Além disso, a descrição de como as métricas são obtidas e calculadas é fundamental para a análise e replicação dos ambientes. As médias de acurácia dos modelos de IA, consumo de CPU e memória RAM serão abordadas em detalhes nesta dissertação.

2.6 Considerações Finais

Este capítulo apresentou os principais conceitos relativos a esta dissertação. Inicialmente, os temas de **computação em nuvem** e **computação de borda** foram apresentados e descritos, considerando que esta dissertação usa ambos os ambientes. Posteriormente, os conceitos básicos de **segurança da informação**, e mais especificamente, de **IDSs**, também foram apresentados e discutidos. Além disso, a área de **IA**, as subáreas de **ML** e **DL**, e alguns algoritmos foram expostos e discutidos. Por fim, conceitos introdutórios relacionados à **avaliação de desempenho** também foram apresentados, uma vez que esta dissertação avalia o desempenho de sistemas de IDSs baseados em IA em ambientes de nuvem e borda, integrando assim todos os conceitos acima elencados.

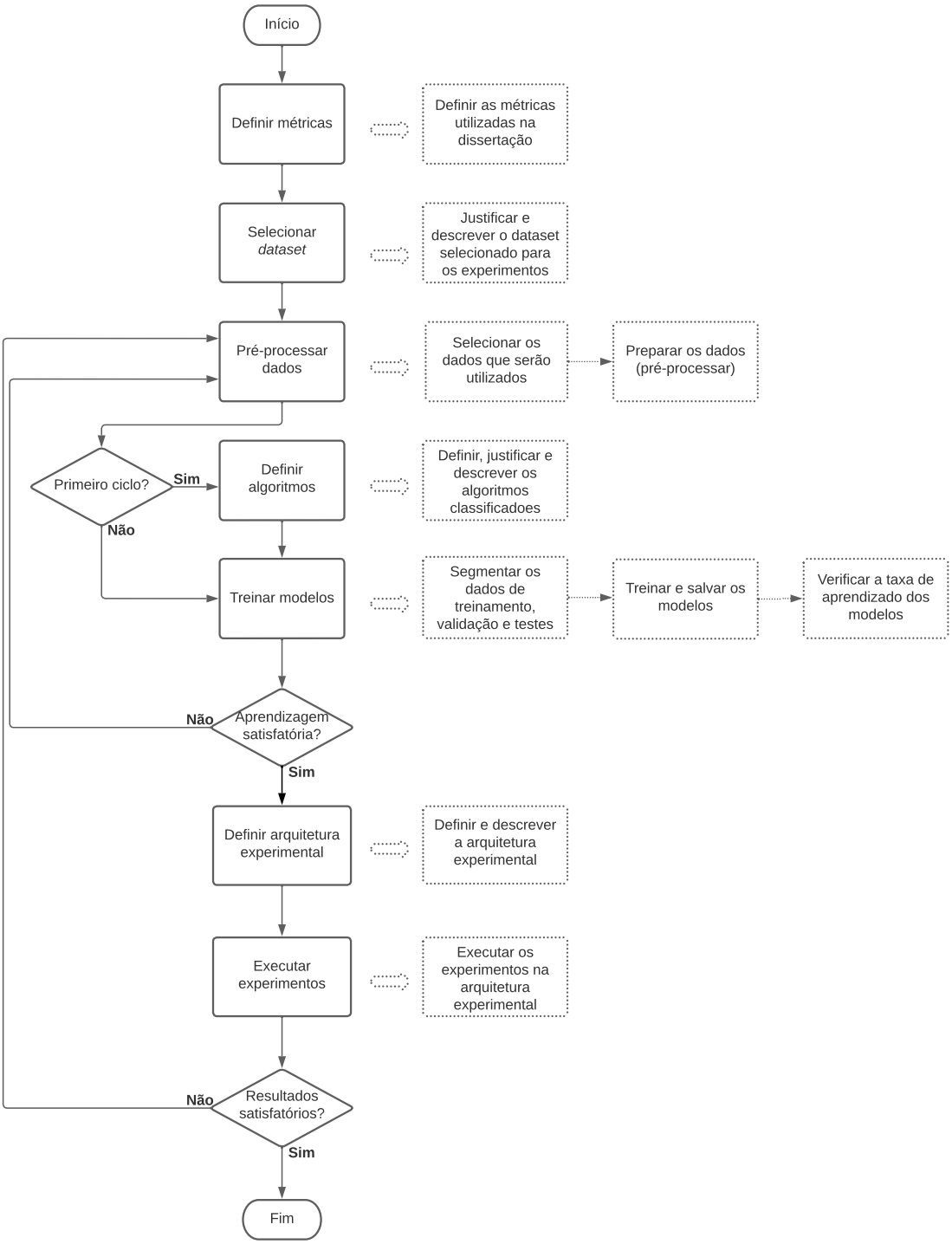
3 Metodologia

Este capítulo apresenta a metodologia utilizada nesta dissertação, sendo descritos os passos realizados desde a definição das métricas de desempenho até a definição da arquitetura e execução dos experimentos. O principal objetivo deste capítulo é possibilitar a replicação dos experimentos futuramente, facilitando aprimoramentos na avaliação de desempenho de IDSs baseados em IA desenvolvidos e implementados para os ambientes em nuvem e ambientes de borda computacional.

O fluxograma exibido na Figura 11 representa de maneira visual a metodologia proposta desta dissertação. Os retângulos representam cada atividade a ser executada e desenvolvida, em que as setas apontam a ordem de execução. Os blocos tracejados (ao lado de cada atividade) representam as possíveis etapas a serem executadas, em que as setas tracejadas também obedecem a ordem de execução. Apenas quando as etapas de uma atividade são finalizadas, é possível seguir para a próxima atividade. O losango é usado para determinar uma decisão em que apenas um caminho pode ser tomado (dos dois em questão), podendo alterar o fluxo de execução das atividades.

Dessa forma, a metodologia proposta consistem em 7 atividades principais, são elas: *Definir métricas, Selecionar o dataset, Pré-processar os dados, Definir os algoritmos, Treinar os modelos, Definir a arquitetura experimental, e por fim, Executar experimentos.* Conforme apresentado a seguir, cada uma dessas atividades são subdivididas em etapas, com o objetivo proporcionar um melhor entendimento.

Figura 11 – Metodologia proposta para guiar da seleção do *dataset* a execução dos experimentos.



Fonte: do Autor.

3.1 Definir Métricas

Avaliar a taxa de aprendizagem permite validar a eficiência de um modelo, além de possibilitar o ajuste de algum parâmetro para torná-lo ainda melhor. Por meio da matriz de confusão (ver Tabela 3), é possível visualizar a quantidade de acertos e erros dos modelos criados após o treinamento. Ou seja, a matriz resume o desempenho, pois pode fornecer o valor (as quantidades) que o modelo classificou (valor real) um determinado dado e o valor que deveria classificar (valor previsto), no momento dos testes e validação.

Existem quatro componentes (variáveis) na matriz de confusão, são eles: **Verdadeiros Positivos** (*True Positive* (TP)), **Falsos Positivos** (*False Positive* (FP)), **Falsos Negativos** (*False Negative* (FN)) e **Verdadeiros Negativos** (*True Negative* (TN)). Logo, há quatro hipóteses para as referidas classificações dos modelos de IA, de maneira geral são elas: (i) o modelo classificar um dado como positivo, e o resultado esperado ser positivo (TP); (ii) o modelo classificar um dado como negativo, e o resultado esperado ser positivo (FP); (iii) o modelo classificar um dado como positivo, e o resultado esperado ser negativo (FN); (iv) o modelo classificar um dado como negativo e o resultado esperado ser negativo (TN) (FERRAG et al., 2020).

Tabela 3 – Exemplo de matriz de confusão.

		Valor Previsto	
		Positivo	**Negativo**
Valor Atual	**Positivo**	**TP**	**FN**
	Negativo	**FP**	**TN**

A matriz de confusão é uma ferramenta poderosa na classificação de dados baseados em IA, no entanto, outras métricas de desempenho também podem ser usadas para avaliar o desempenho por meio das variáveis usadas na matriz. Tais métricas fornecem saídas em forma de valor numérico (porcentagens), o que facilita o entendimento e comparação (diferentemente das quantidades postas pela matriz de confusão). Acurácia, Precisão, Revocação e Pontuação F1 são métricas bastante usadas no contexto de IA. A Tabela 4 exibe as referidas métricas e como são calculadas (DALIANIS, 2018). Nesta dissertação, a acurácia foi a métrica adotada para as comparações relacionadas ao aprendizado dos modelos de ML e DL.

Tabela 4 – Definição de métricas.

Acurácia	$\frac{TP+TN}{TP+TN+FP+FN}$
Precisão	$\frac{TP}{TP+FP}$
Revocação	$\frac{TP}{TP+FN}$
Pontuação F1	$2 \times \frac{Precisão \times Revocação}{Precisão\ +\ Revocação}$

Métricas relacionadas ao consumo dos equipamentos também foram coletadas e analisadas com o objetivo de avaliar o desempenho dos IDSs baseados em ML e DL, além das métricas para avaliar a taxa de aprendizagem dos modelos de IA. No momento da execução dos experimentos, o consumo de CPU e de memória RAM foram salvos para analisar o comportamento dos dispositivos, e assim, comparar o impacto dos IDSs em cada experimento e ambiente avaliados, permitindo assim avaliar os *trade-offs* entre desempenho e acurácia dos modelos de IA utilizados nos ambientes propostos. A biblioteca PSUtil [1] foi utilizada por um *script* escrito em Python para coletar o consumo de CPU e de memória RAM dos IDSs.

3.2 Selecionar *Dataset*

Existem diversos *datasets* no meio acadêmico que servem de base para o desenvolvimento de novos modelos de IA e, aos poucos, *datasets* mais atuais são desenvolvidos para novos cenários, buscando extrair dados específicos com o objetivo de aumentar a taxa de aprendizagem dos modelos. NSL-KDD (2009) e CICIDS (2017) são exemplos de *datasets* difundidos no universo acadêmico que possuem dados os quais são usados para o desenvolvimento de modelos de IA voltados aos IDSs (REVATHI; MALATHI, 2013; DHANABAL; SHANTHARAJAH, 2015).

Ferrag et al. (2020) conduziram um estudo que avaliaram modelos de DL voltados ao desenvolvimento de IDSs com diversos *datasets*. O estudo demonstrou que apenas o *datasets* BoT-IoT [2], desenvolvido por Koroniotis et al. (2019), tinha sido desenvolvido considerando o ecossistema IoT. Dessa forma, esta pesquisa utilizou o *datasets* BoT-IoT um dos *datasets* mais completos na academia para sistemas IoT. Além disso, o referido

[1] Disponível em: <https://psutil.readthedocs.io/en/latest/>
[2] Disponível em: <https://research.unsw.edu.au/projects/bot-iot-dataset>

dataset usou softwares consolidados no mercado como o VMWare ESXi (VMWARE, 2022b) e o VSphere (VMWARE, 2022a). Também considerou cenários compostos por *Virtual Machines* (VM) e equipamentos reais (por exemplo, um *firewall* pfsense[3]).

Adicionalmente, o *dataset* Bot-IoT adota diferentes protocolos relevantes em uma infraestrutura ampla, com sistemas e dispositivos diversos (dentre eles, protocolos e simuladores de dispositivos IoT), além de conter ataques cibernéticos atuais. Os dados do BoT-IoT foram capturados usando o programa TShark (TSHARK, 2022), os quais foram posteriormente inseridos em um arquivo de extensão *Packet Capture* (PCAP). Após tal etapa, os arquivos PCAP foram transformados em arquivos de extensão *Comma-Separated Values* (CSV), tendo em vista que essa extensão permite a manipulação dos dados capturados no desenvolvimento dos modelos de IA. Por fim, os arquivos CSV foram marcados com rótulos de tráfego normal (ou seja, tráfego válido) e tráfego malicioso com os referidos ataques, são eles: *Distributed Denial of Service* e *Denial of Service* sobre HTTP, TCP e UDP, *Service Scan*, *OS Fingerprinting*, *Keystroke Logging* e *Exfiltration*.

3.3 Pré-Processar os Dados

O ato de pré-processar (tratar) os dados para torná-los *legíveis* para os modelos de IA é uma fase fundamental que busca aumentar a taxa de aprendizagem dos modelos. Remover *outliers* (dados mal formatados ou que de alguma forma estejam fora dos padrões - pontos fora da curva) é essencial, pois os modelos podem aprender com dados incorretos, e assim, se tornam ineficazes. Classificar os dados de forma correta (quando podem ser categorizados), eliminar dados duplicados (ou linhas duplicadas) e campos em branco (campos vazios) em um *dataset* são exemplos de ações que buscam pré-processar os dados.

A fase de pré-processamento dos dados nesta dissertação foi realizada no Google Colaboratory[4] por meio de um ambiente padrão e gratuito disponibilizado pela plataforma com a seguinte configuração: 12 GB de memória RAM, processador de marca Intel, modelo Xeon e frequência 2.20GHz. Os dados foram armazenados e carregados para o Google Colaboratory por meio do Google Drive.

Primeiramente, as características (ou colunas) desnecessárias relacionados aos endereços de rede foram removidas. Isto é, os campos ip.src, ip.dst, ipv6.src, ipv6.dst,

[3] Disponível em: <https://www.pfsense.org/>
[4] Disponível em: <https://colab.research.google.com/>

arp.src.proto_ipv4 e arp.dst.proto_ipv4 que estavam no *dataset* foram removidos, pois os endereços são genéricos em cada ambiente, e consequentemente, esses campos não foram utilizados. Os campos relacionados à verificação *tcp.checksum* e *udp.checksum* também foram removidos por não haver relação aparente com os ataques disponíveis no *dataset*.

Posteriormente, as colunas com dados categóricos foram identificadas e passaram pelo processo de *One-Hot-Encoding* (ou seja, as categorias foram transformadas em colunas binárias) na busca de tornar o modelo mais eficiente. Portas mais comuns do protocolo TCP/IP na camada de aplicação foram utilizadas (20, 21, 22, 23, 25, 42, 43, 53, 80, 161, 433), pois representam protocolos conhecidos (como FTP, SSH, Telnet, DNS, SMTP, HTTP/HTTPS, SNMP) e estão disponíveis no *dataset* BoT-IoT. Vale ressaltar que novas portas podem ser investigadas a fim de aprimorar a eficiência no momento do treinamento modelo, além de cobrir mais serviços usados nas redes de computadores.

As colunas de portas de origem ou destino foram divididas em duas colunas, a primeira no intervalo de portas conhecidas (de 0 até 1023) e a segunda com o intervalo de portas registradas (acima de 1023). Para a coluna tcp.ttl, houve uma separação entre as colunas tcp.ttl_1 e tcp.ttl_2, pois há dois valores separados por vírgulas. Adicionalmente, os campos com valores vazios foram preenchidos com NAs. Para as colunas tcp.flags e stream, obteve-se o valor máximo em cada coluna e os NAs foram preenchidos com o valor máximo mais 1.

Para as outras colunas, os NAs foram preenchidos com 0. Considerando o pré-processamento realizado, linhas foram duplicadas neste processo e, ao final, as linhas redundantes foram removidas, pois o conjunto de dados não deve conter dados duplicados para o treinamento do modelo. Vale destacar que cabeçalhos IPv6 não foram utilizados nesse experimento, pois o protocolo IPv4 ainda é o mais usado atualmente na Internet. A Tabela 5 exibe a quantidade de registros categorizados após o pré-processamento dos dados.

3.4 Definir os Algoritmos

Estudos avaliam o desempenho de modelos de IA, seja ele ML ou DL, por meio de métricas relacionadas à taxa de aprendizagem como: acurácia, precisão, revocação, pontuação F1, e etc. No contexto de IDSs baseados em IA, escolher um modelo em

Tabela 5 – Informações do *dataset* após pré-processamento dos dados.

Categoria do tráfego	Subcategoria	Quantidade de registros
DDoS	HTTP	194417
	TCP	999444
	UDP	1000054
DoS	HTTP	287480
	TCP	998703
	UDP	999977
Reconnaissance	OS fingerprinting	827058
	Service scanning	999895
Information theft	Data exfiltration	301710
	Keylogging	11387
Normal	Normal	2543626
Total		9163751

detrimento de outro se torna uma análise complexa, pois depende da finalidade do trabalho, dos equipamentos que serão utilizados, e das métricas a serem avaliadas.

Considerando modelos de ML, Choudhury e Bhowal (2015) realizaram um estudo com diversos classificadores e métricas por meio do WEKA (JENITHA; VENNILA, 2014) com o objetivo de avaliar os melhores modelos que realizam classificação. Ao final, um modelo baseado no teorema de bayes e o RF obtiveram os melhores resultados e, segundo os autores, são os mais indicados para esta finalidade. Já Khan e Gumaei (2019) realizaram um estudo comparando diferentes classificadores de ML desenvolvidos para IDSs com dois *datasets* distintos. Eles concluíram que o modelo baseado no RF esteve entre os melhores resultados, assim como o NB obteve excelente tempo na construção do modelo. Isto é, os modelos NB e RF obtiveram bons resultados no referido estudo frente a outros algoritmos de ML que visam ser utilizados em um IDS.

Por outro lado, considerando modelos de DL, Kocher e Kumar (2021) realizaram um estudo que destacou os desafios e as descobertas relacionadas à IA, mais especificamente ML e DL, voltadas ao contexto de IDSs. Nesse trabalho, os autores também apresentam técnicas mais usadas em ML (dentre elas NB e RF) e DL (*Convolutional Neural Network* (CNN) e *Recurrent Neural Network* (RNN)). Ao final, um comparativo entre os métodos de ML e DL são apresentados considerando os diferentes modelos, *datasets* e métricas

usadas em cada estudo, enfatizando assim a complexidade de selecionar a melhor técnica para desenvolver um IDS baseado em IA. Além disso, Ge et al. (2019) propõe a criação de um IDS por meio de um modelo de DL (usando uma FNN) capaz de realizar classificação binária e multiclasse de ciberataques com altas acurácias.

Dessa forma, nesta dissertação, os algoritmos NB e RF foram selecionados e utilizados por haver estudos que evidenciam bons resultados frente a outros algoritmos de ML voltados à construção de IDSs. E dois modelos de DL que usam uma FNN foram utilizados, pois o classificador multiclasse obteve alta taxa de aprendizado (acurácia) nos testes realizados. Diferentemente dos demais estudos, neste, além das métricas conhecidas (acurácia, precisão, revocação e pontuação F1), métricas relacionadas ao consumo dos recursos dos dispositivos de nuvem e borda (CPU e memória RAM) também serão avaliadas levando em conta experimentos reais.

Os modelos NB e RF utilizaram as bibliotecas disponíveis no scikit-learn[5], porém o modelo FNN-Binary (FNN-B) foi configurado com 512 neurônios na primeira e segunda camadas, ambos com a função de ativação ReLU (SHARMA et al., 2017), enquanto a última camada foi configurada com 5 neurônios e com a função de ativação *Sigmoid* (SHARMA et al., 2017). Já o modelo FNN-Multiclass (FNN-M) também foi configurado com 512 neurônios nas três primeiras camadas com a função de ativação ReLU. A última camada foi configurada com 4 neurônios e com a função de ativação *Softmax* (SHARMA et al., 2017). Ambos os modelos de DL utilizaram o otimizador *Adam* (KINGMA; BA, 2014). No entanto, o modelo FNN-MultiClass recebeu mais entradas que os demais, pois também foram considerados os cabeçalhos do protocolo TCP.

3.5 Treinar Modelos

Após a fase de pré-processamento e definição dos algoritmos, os modelos foram treinados, e assim, criados. Porém, apenas 60% dos dados foram usados para o treinamento, visto que 20% foram separados para realizar testes e os demais 20% foram usados para a validação.

Os modelos NB e RF foram instanciados e criados por meio do *scikit-learn*, e posteriormente, foram treinados e salvos. O treinamento do modelo FNN-Binary obteve

[5] Disponível em: <https://scikit-learn.org/stable/>

20 *epochs*, *batch size* de 128 e a função de perda foi configurada com *Binary Crossentropy*, uma vez que esse modelo usa a classificação binária. Já o modelo FNN-Multiclass, obteve 10 *epochs*, *batch size* de 128 e a função de perda foi configurada com o parâmetro *Sparse Categorical Crossentropy*, visto que esse modelo utiliza a classificação multiclasse. Pesos aleatórios foram usados inicialmente para as camadas do modelo, mas a função Adam foi usada para atualizar os pesos e aplicar a função de perda de maneira mais eficiente.

Após o treinamento, os modelos foram salvos em disco para servirem de base para os IDSs. Seus arquivos (treinados e salvos) possuem tamanhos de 2KB, 1MB, 15MB e 18MB para os modelos NB, RF, FNN-Binary e FNN-MultiClass, respectivamente. A diferença entre os tamanhos está relacionada à técnica e à tecnologia empregada em cada modelo de IA. A partir disso, pôde-se verificar as métricas de acurácia, precisão, revocação e pontuação F1, bem como analisar o aprendizado de cada modelo proposto. Caso os modelos não obtivessem uma acurácia satisfatória, haveria uma nova análise na seleção de atributos e um novo pré-processamento, e assim, posteriormente, os algoritmos seriam treinados mais uma vez, buscando aumentar a sua taxa de aprendizado. Este ciclo deve ocorrer até atingir uma taxa de aprendizagem satisfatória para os modelos.

3.6 Definir Arquitetura Experimental

Após a criação dos modelos, duas arquiteturas experimentais são propostas para avaliar o desempenho dos IDSs executando os modelos de ML e DL, são elas: o (i) **ambiente em nuvem** e o (ii) **ambiente de borda**. No primeiro ambiente, o objetivo é executar e avaliar o desempenho dos IDSs em uma nuvem computacional. Já no segundo ambiente, busca-se executar e avaliar o desempenho dos IDSs em um equipamento de baixo poder computacional, considerando que o ambiente de borda utiliza normalmente equipamentos com recursos limitados. As duas arquiteturas são consideradas por serem bastante utilizadas no contexto de IoT.

No ambiente de nuvem, duas VMs são criadas. A VM cliente foi criada para gerar tráfego normal ou malicioso, enquanto uma VM servidor é criada para executar os IDSs e tratar as requisições. Por outro lado, no ambiente de borda, um equipamento de baixo poder computacional (Raspberry Pi) é utilizado como dispositivo de borda (servidor), onde os IDSs são instalados e um computador *desktop* (cliente) envia tráfego normal ou

malicioso. Em ambos os casos, a comunicação entre os dispositivos cliente e servidor é estabelecia por meio de um socket TCP, buscando não interferir no desempenho dos experimentos. Além disso, diferentes CTs podem ser consideradas com o objetivo de analisar os diferentes impactos nos equipamentos.

Tais infraestruturas permitem a execução dos experimentos com a finalidade de coletar e analisar a quantidade do consumo computacional que os IDSs baseados em ML e DL podem exercer sobre um determinado equipamento, e assim, influenciar a infraestrutura computacional como um todo. Além de permitir a comparação com as métricas relacionadas às taxas de aprendizagem dos modelos. Essa ação pode ajudar no desenvolvimento, implementação e execução de IDSs baseados em IA em infraestruturas computacionais de borda e nuvem.

3.7 Executar Experimentos

Após a definição das métricas, processamento dos dados, definição dos algoritmos, treinamento dos modelos e da definição da arquitetura experimental, os experimentos podem ser executados. Um programa responsável por gerar diferentes CTs dos clientes para a VM servidor e para a *Single Board Computer* (SBC) (um computador de placa única de baixo poder computacional) foi desenvolvido. Tais CTs buscam estressar os ambientes, e assim, avaliar o consumo computacional executando cada IDS. Um *script* foi responsável por coletar dados computacionais como o consumo de CPU e memória RAM na VM servidor e na SBC. Após a execução dos experimentos, uma análise para verificar se os resultados foram coletados e estão corretos é realizada. Caso os resultados apresentem alguma inconsistência, há uma revisão, e se necessário uma atualização, no pré-processamento de dados e nas fases posteriores até uma nova execução. Esse ciclo deve ocorrer até os dados apresentarem dados satisfatórios. As arquiteturas experimentais e a execução dos experimentos serão apresentadas em detalhes no próximo capítulo.

3.8 Considerações Finais

Este capítulo apresentou a metodologia proposta para desenvolver e implementar modelos de ML e DL capazes de classificar tráfegos (maliciosos ou não) e serem

implementados como IDSs em ambientes de borda e nuvem computacional, armazenando inclusive informações relacionados aos equipamentos avaliados, como o consumo de CPU e de memória RAM. Vale ser ressaltado que a diferença entre o tráfego malicioso ou tráfego normal não impacta no consumo dos dispositivos. Os dados permitirão avaliar os *trade-offs* entre o desempenho computacional dos ambientes e a taxa de aprendizagem dos modelos, pois a implementação em ambientes com recursos computacionais limitados pode impactar, e por vezes impossibilitar, o dispositivo o qual executará o IDS. Tais implementações e análises podem auxiliar empresas, profissionais ou pessoas interessadas no processo de desenvolvimento e execução de IDSs baseados em IA em ambientes de nuvem e borda computacional.

4 Arquitetura Experimental

Neste capítulo, as arquiteturas experimentais do ambiente em nuvem e do ambiente de borda são apresentadas em detalhes. Tais arquiteturas foram consideradas para executar os IDSs baseados em ML e DL nos ambientes em nuvem e de borda. Em seguida, a execução dos experimentos é descrita para nortear como os IDSs foram implementados e como os dados foram coletados. Uma vez montadas as arquiteturas e os experimentos realizados foi possível avaliar os *trade-offs* entre o desempenho dos ambientes e a taxa de aprendizagem dos modelos.

4.1 Ambiente em Nuvem

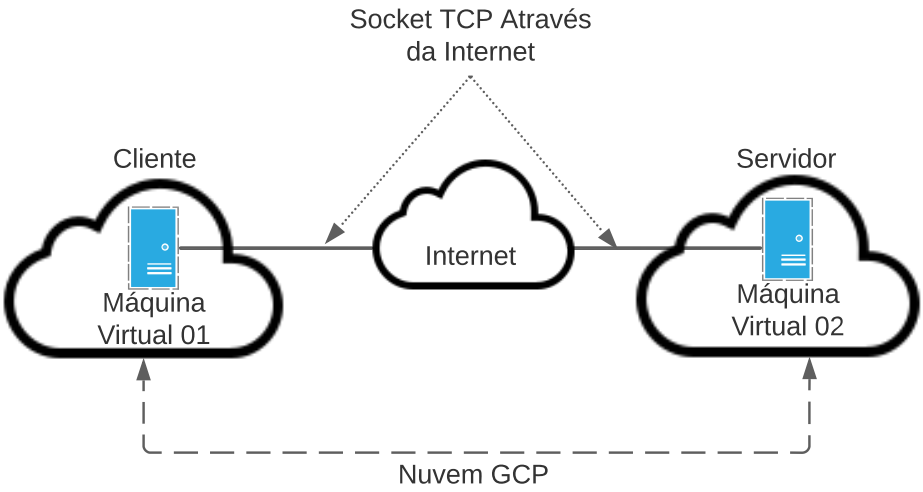
No ambiente em nuvem, a *Google Cloud Platform* (GCP) foi selecionada para realizar os experimentos por possuir um ambiente estável, conter um vasta documentação de maneira acessível, ser bastante difundida no mercado e disponibilizar um ambiente para novos experimentos gratuitamente, antes de contratar o serviço integral. Dessa forma, duas VMs (cliente e servidor) foram criadas e utilizadas em regiões geograficamente distintas, onde a VM cliente ficou localizada no *datacenter* denominado de Southamerica-East1-A e a VM servidor ficou localizada no *datacenter* denominado de US-Central1-A.

As VMs foram criadas por meio de um acesso gratuito de até 3 meses de uso ou até atingir um consumo máximo de recursos permitidos, liberando assim usar a VM da série E2, do tipo E2-MEDIUM, composta por uma vCPU (CPU virtual) compartilhada, 4 GB de memória RAM e um disco de 10 GB. Após a criação das VMs, fez-se necessário realizar uma configuração de roteamento da rede externa para a rede interna a fim de que as VMs pudessem se comunicar por meio da internet.

A distância entre os *datacenters* tem o objetivo de simular ambientes mais próximos da realidade, fazendo o uso da Internet, pois o uso do mesmo *datacenter* aumentaria o risco da conexão ser realizada localmente. Nesta arquitetura experimental, a VM cliente simulou os usuários enviando requisições normais ou maliciosas, com diferentes CTs para a VM servidor. A Figura 12 apresenta o ambiente adotado para execução dos experimentos na nuvem.

O socket TCP foi utilizado para realizar a comunicação entre as duas VMs no

Figura 12 – Infraestrutura experimental na nuvem.



Fonte: do Autor.

ambiente em nuvem. As VMs foram configuradas com o sistema operacional GNU/Linux Debian Buster (sem interface gráfica) e sem recursos adicionais, apenas com a instalação das bibliotecas necessárias para o experimento. Na VM servidor foram instalados os IDSs baseados nos modelos de ML e DL propostos com o objetivo de classificar o tráfego, em normal ou malicioso, gerado pelo cliente.

4.2 Ambiente de Borda

A execução dos experimentos no ambiente de borda foi realizada em uma SBC de marca Rasbperry Pi modelo 4 B[6], onde os IDSs foram instalados. Vale ser destacado que há no mercado atual outras SBCs que podem ser utilizadas, como a Jetson da NVIDIA[7], GA-SBCAP3350 da Gigabyte[8] e a N2+ da ODROID[9]. No entanto, a marca Raspberry Pi foi escolhida por possuir o melhor custo-benefício. Vale ser ressaltado ainda que, a Raspberry Pi 4 Modelo B, especificamente, foi escolhida devido a sua arquitetura 64 bits, valor de mercado acessível e disponibilidade no mercado brasileiro.

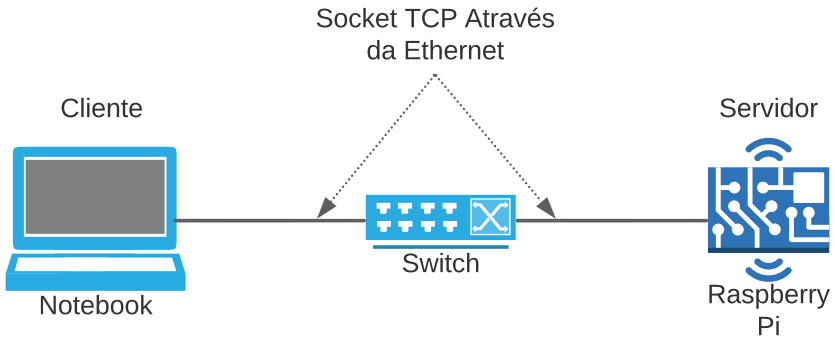
Similar a um computador, uma SBC possui um processador, uma memória RAM, uma placa de rede ethernet e wifi, um slot para receber um cartão de memória do tipo *Secure Digital* (SD) que é utilizado como o disco rígido, além de possuir entradas USB e

[6] Disponível em: <https://www.raspberrypi.com/>
[7] Disponível em: <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/>
[8] Disponível em: <https://www.gigabyte.com/Motherboard/GA-SBCAP3350-rev-20>
[9] Disponível em: <https://www.odroid.co.uk/index.php?route=product/category&path=246_239>

saídas de vídeo. Outras placas de expansão também podem ser adicionadas para agregar novas funcionalidades, permitindo adequar a SBC a diversas aplicações.

Além da SBC, um computador foi usado para gerar tráfego (requisições normais ou maliciosas com diferentes intensidades) e enviar por meio de sockets TCP para a SBC, similiar ao experimento no ambiente em nuvem. Um switch (ethernet 10/100mbps) foi utilizado para interconectar o computador (cliente) e a SBC (servidor). Dessa forma, o computador cliente enviou requisições para a SBC que por meio dos IDSs instalados (baseados em ML e DL) classificaram os dados. Isto é, se o tráfego era normal ou malicioso. A Figura 13 apresenta o ambiente adotado.

Figura 13 – Infraestrutura experimental na SBC.



Fonte: do Autor.

O cliente utilizou o sistema operacional GNU/Linux Debian Stretch e a Raspberry Pi utilizou o sistema operacional Pi OS Buster[10] (padrão para o dispositivo). Os dois dispositivos foram instalados com a configuração mínima padrão. Apenas as bibliotecas necessárias para o experimento foram instaladas. Além disso, sua arquitetura também permite executar modelos de DL mais atuais, pois o TensorFlow (principal *framework* para execução de modelos de DL) nas versões mais recentes (> 2.2) necessita de processadores com arquitetura 64 bits como requisito mínimo (TENSORFLOW, 2022). A Tabela 6 detalha as configurações de hardware dos equipamentos utilizados.

4.3 Execução dos Experimentos

Após a definição das arquiteturas, experimentos foram executados considerando os ambientes adotados. Assim, esta seção descreve como os experimentos foram executados e

[10] Disponível em: <https://www.raspberrypi.com/software/operating-systems/>

Tabela 6 – Configuração dos equipamentos utilizados nos experimentos.

Equipamento	**Processador**	**Qtd. núcleos**	**RAM**	**Arquitetura**
Raspberry Pi4 Model B	BCM2711 - 1.5 GHz	4	4 GB	64 bits
Lenovo G40-80	I5-5200U - 2.2 Ghz	4	4 GB	64 bits
VMs	Intel Xeon - 2.2 GHz	1 (VCPU)	4 GB	64 bits

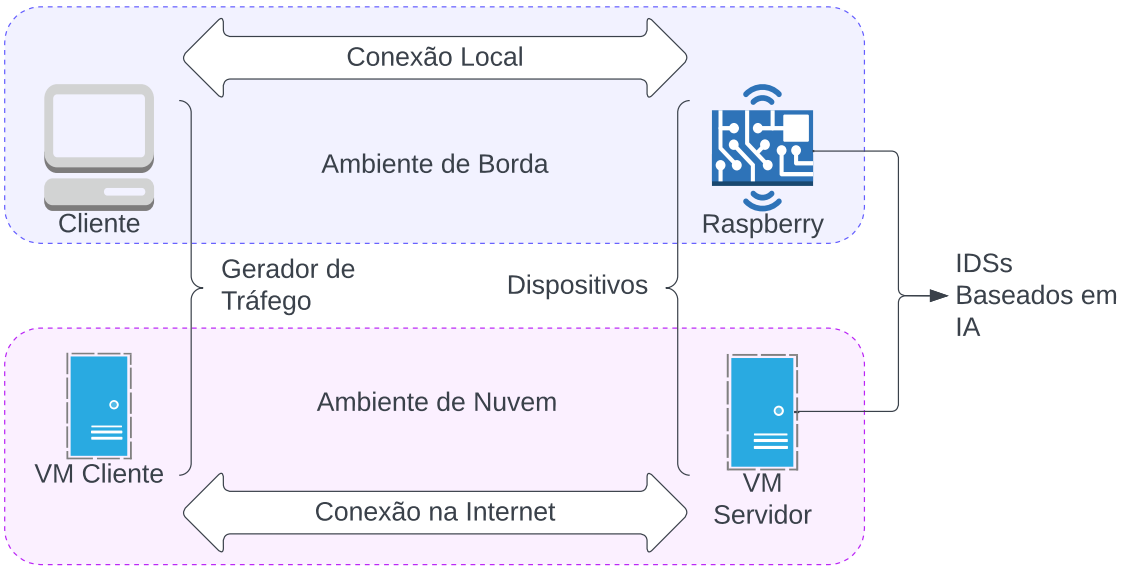
quais programas foram necessários para tais execuções. Inicialmente, um programa escrito em Python versão 3 foi desenvolvido para gerar CTs constantes nos ambientes de borda e em nuvem. No primeiro ambiente (ver Figura 13), o computador envia as requisições, seguindo as CTs, para o dispositivo de borda (SBC) realizar a classificação dos dados (se tráfego malicioso ou tráfego normal) por meio do IDS e modelo configurado. No segundo (ver Figura 12), a VM cliente envia as CTs para a VM servidor, e assim, o IDS e o modelo configurado realizará a classificação dos dados, similar ao primeiro ambiente.

Para simular ambientes distintos, considerando diferentes intensidades de ataque, CTs foram utilizadas para o envio dos dados. Assim, foram adotadas as CTs de 1,0, 0,5, 0,4, 0,3. Isso significa que, o cliente enviou ao servidor, em média, uma requisição a cada 1,0, 0,5, 0,4 e 0,3 segundos, respectivamente. Cada combinação da infraestrutura adotada, modelo implantado nos IDSs (NB, RF, FNN-Binary e FNN-MultiClass) e CTs (1,0, 0,5, 0,4 e 0,3) constitui um cenário experimental analisado. Ao final, 32 experimentos foram executados e analisados, considerando uma rodada de experimento para cada ambiente, IDS e CT. Além disso, cada cenário foi executado por um período de 40 minutos, sendo que os cinco minutos iniciais e finais buscaram capturar o comportamento do sistema em estado ocioso. Essa ação evidencia a saída da ociosidade do sistema quando o envio das CTs é iniciado e o retorno à ociosidade.

Outro *script*, também escrito em Python 3, coletou, a cada 10 segundos, o consumo de CPU e de memória RAM no dispositivo de borda e na VM servidor através da biblioteca PSUtil[11] do Python. A Figura 14 exibe e compara os ambientes adotados.

[11] Disponível em: <https://pypi.org/project/psutil/>

Figura 14 – Comparativo entre as infraestruturas.



Fonte: do Autor.

4.4 Considerações Finais

Este capítulo apresentou as arquiteturas experimentais e como os experimentos foram conduzidos. Tais arquiteturas buscam reproduzir um ambiente em nuvem e de borda executando IDSs baseados em ML e DL para coletar o consumo de recursos dos ambientes e, posteriormente, analisar seus impactos. Assim, avaliar os *trade-offs* entre desempenho e taxa de aprendizagem nos referidos ambientes propostos. Essa análise é crucial tendo em vista que IDSs podem causar impactos consideráveis quando instalados em equipamentos com recursos limitados, podendo levar a indisponibilidade.

5 Resultados e Discussão

Este capítulo apresenta os resultados dos estudos realizados considerando a taxa de aprendizagem dos modelos de IA e o desempenho considerando as métricas de CPU e de memória RAM nos ambientes. Primeiro, os resultados relacionados à acurácia dos modelos de IA são apresentados e discutidos. Posteriormente, os resultados relacionados aos consumos dos IDSs nos ambientes de nuvem e borda são detalhados. Em seguida, um comparativo de desempenho entre os ambientes é realizado. Por fim, as considerações finais são apresentadas.

5.1 Aprendizado dos Modelos

Nesta seção, serão apresentados os resultados obtido pelos modelos de ML e DL por meio das métricas acurácia, precisão, revocação e pontuação F1. Os modelos realizaram a classificação dos dados de modo binário (NB, RF, FNN-Binary) ou de modo multiclasse (FNN-Multiclasse) e, assim, a classificação gerou um rótulo normal ou malicioso para cada registro (ou tráfego). Após o treinamento, os modelos NB, RF, FNN-Binary e FNN-Multiclass obtiveram os resultados da aprendizagem baseados nos dados de validação. Tais resultados podem ser observados na Tabela 7.

Tabela 7 – Resultado das métricas após o treinamento.

Métricas	**Modelos**			
	NB	**RF**	**FNN-B**	**FNN-M**
Acurácia	98,79%	99,99%	45,58%	99,61%
Precisão	97,00%	100%	50,09%	72,40%
Revocação	99,00%	100%	91,33%	100%
Pontuação F1	98,00%	100%	64,69%	83,97%

Considerando a acurácia, é possível observar que os modelos NB e RF (baseados em ML) obtiveram bons resultados em comparação com o modelo FNN-Binary (baseado em DL). Os modelos NB e RF obtiveram resultados próximos ao modelo FNN-Multiclass. No entanto, vale ser ressaltado que o FNN-Multiclass recebeu mais dados de entrada e usou

a classificação multiclasse. Considerando as métricas Precisão e Pontuação F1, pode-se observar que os modelos de ML (Naive Bayes e Random Forest) obtiveram melhores resultados em comparação com os modelos de DL (FNN-Binary e FNN-Multiclass). Ao levarmos em consideração a métrica revocação, é possível observar uma igualdade entre os dados apresentados pelos modelos Naive Bayes, Random Forest e FNN-Multiclass (destoando apenas o modelo FNN-Binary), mas a métrica pontuação F1 reafirma a superioridade dos modelos de ML em comparação aos modelos de DL, para os exemplos e experimentos adotados.Vale ser destacado ainda que modelos de ML e DL desenvolvidos para o contexto de IDSs, há, em média, uma taxa de aprendizagem superior a 95%, considerando diferentes métricas (acurácia, precisão, revocação, pontuação F1) (CHIBA et al., 2019; ESKANDARI et al., 2020; GE et al., 2019; KOCHER; KUMAR, 2021). Ao final, observa-se uma constância de bons resultados entre os modelos NB e RF, apesar do modelo FNN-Multiclass também alcançar altos índices. Por outro lado, o modelo FNN-Binary superou o índice de 90% apenas na métrica revocação, ou seja, mostrou-se um modelo que não obteve boa aprendizagem.

5.2 Ambiente em Nuvem

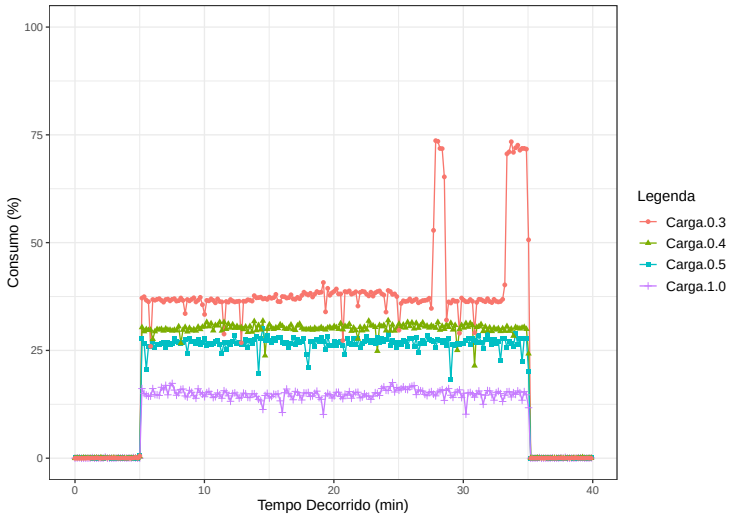
Nesta seção, serão apresentados os resultados coletados por meio dos experimentos realizados no ambiente computacional de nuvem pública. Nesse ambiente, a infraestrutura hospedada na GCP é utilizada para simular um ambiente de nuvem capaz de enviar tráfego (normal ou malicioso) por meio de uma VM cliente através da Internet para uma VM servidor, a qual executará um IDS baseado em ML ou DL. Busca-se assim, avaliar o desempenho dos IDSs no ambiente de nuvem, considerando que é um ambiente comumente usado em diferentes contextos na atualidade.

As Figuras 15(a) e 15(b) exibem o consumo de CPU e memória RAM dos experimentos realizados na nuvem computacional. Isto é, o consumo da VM servidor ao executar o IDS baseado no modelo FNN-Binary com todas as cargas de trabalho, da mais intensa para a menos intensa (de 0,3 até 1,0) em relação ao período de execução do experimento (40 minutos).

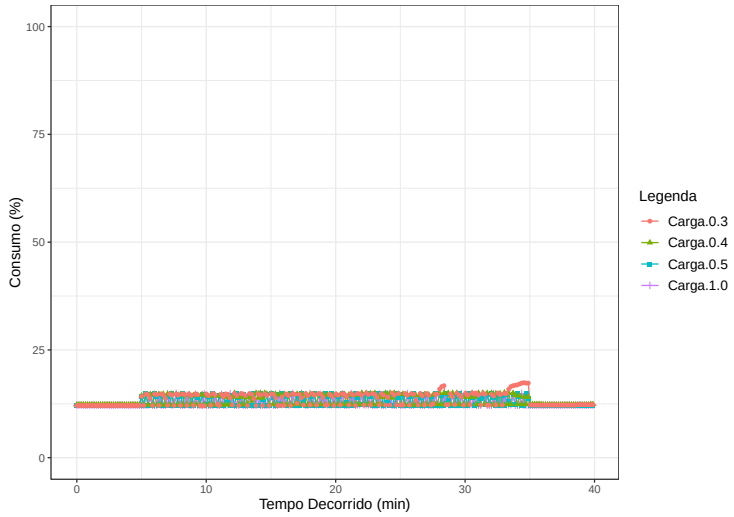
Pode ser observado que a VM servidor conseguiu processar e classificar todas as requisições enviadas, independente da CT utilizada. Vale ser destacado, ainda, que o

consumo médio de CPU foi de 14,86%, 26,68%, 30,07% e 39,73% para as CTs 1,0, 0,5, 0,4 e 0,3, respectivamente. Por outro lado, o consumo médio de memória RAM foi de 12,88%, 13,31%, 13,58% e 13,98%, considerando as mesmas cargas de trabalho. Isso mostra que independente das CTs, o modelo FNN-Binary não gerou alto impacto nos ambientes que foram avaliados.

Figura 15 – Consumo do IDS FNN-Binary na VM servidor.



(a) Consumo de CPU.



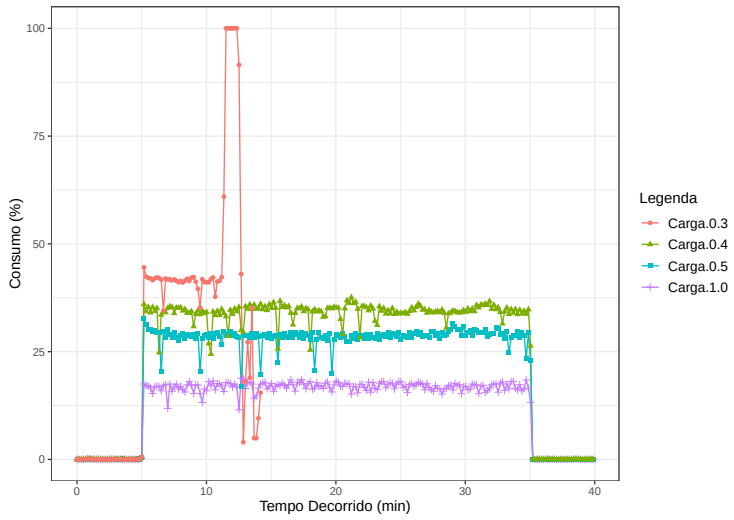
(b) Consumo de Memória RAM.

Fonte: do Autor.

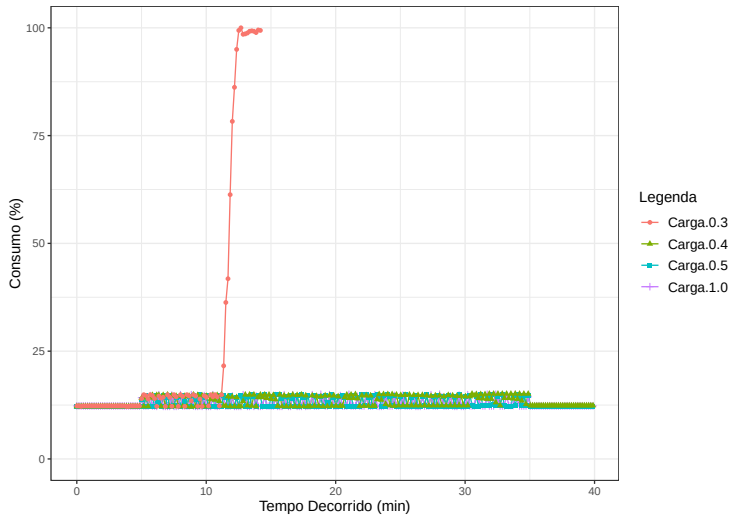
As Figuras 16(a) e 16(b) exibem o consumo de CPU e memória RAM dos experimentos executados na VM servidor usando o modelo FNN-Multiclass. O consumo médio de CPU foi de 16,87%, 28,51%, 34,18% e 49,70%, considerando as CTs 1,0, 0,5, 0,4 e 0,3, respectivamente. Já o consumo médio de memória RAM foi de 12,93%, 13,36%,

13,72% e 21,30%, considerando as mesmas CTs. Vale salientar que no experimento realizado com a CT 0,3, o IDS consumiu totalmente a memória RAM disponível na VM servidor, aumentando o consumo do processador até o travamento do equipamento (a média do consumo fez com que os índices de CPU e memória RAM, usando a CT 0,3, se mantivessem baixos). Em outras palavras, com a carga de trabalho mais intensa, o equipamento não conseguiu executar todo o experimento, pois os recursos computacionais foram exauridos no **minuto 12** após o início do experimento. Após o travamento do equipamento, aguardou-se a finalização do tempo do experimento (40 minutos) para efetuar o desligamento do equipamento. A referida ação foi feita nos momentos em que houve travamento do dispositivo.

Figura 16 – Consumo do IDS FNN-Multiclass na VM servidor.



(a) Consumo de CPU.

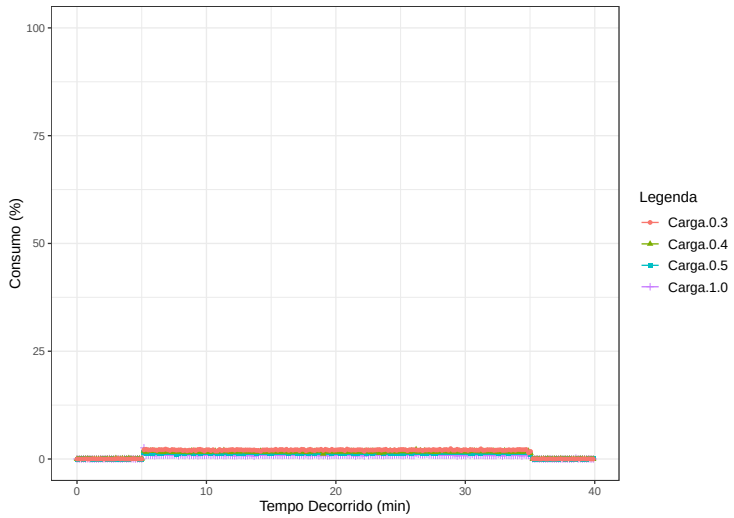


(b) Consumo de Memória RAM.

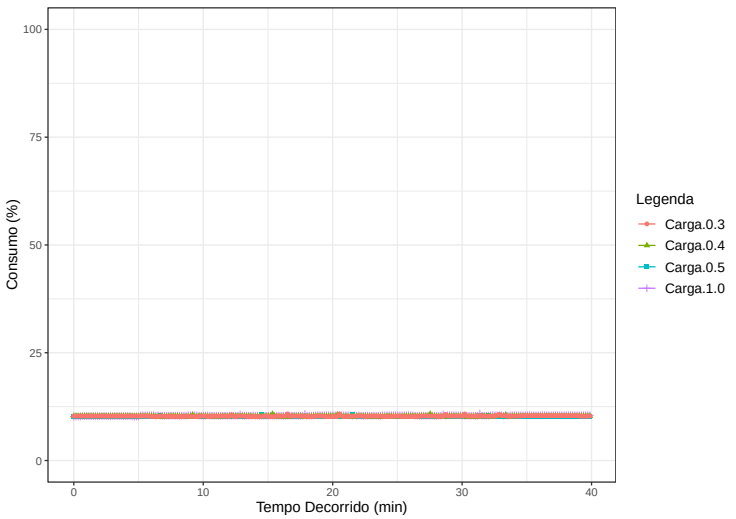
Fonte: do Autor.

Considerando o IDS baseado no modelo NB, as Figuras 17(a) e 17(b) exibem o consumo de CPU e de memória RAM da VM servidor. O consumo médio de CPU foi de 0,89%, 1,42%, 1,74% e 2,09% para as CTs 1,0, 0,5, 0,4 e 0,3, respectivamente. Já para a memória RAM, o consumo médio foi de 10,45%, 10,23%, 10,26% e 10,29%, considerando as mesmas CTs. Nesse experimento e ambiente, nota-se um baixo consumo de CPU e memória RAM. Da CT mais intensa para a CT menos intensa, há um aumento do consumo médio de 1,20% para a CPU e uma baixíssima variação no consumo de memória RAM.

Figura 17 – Consumo do IDS Naive Bayes na VM servidor.



(a) Consumo de CPU.



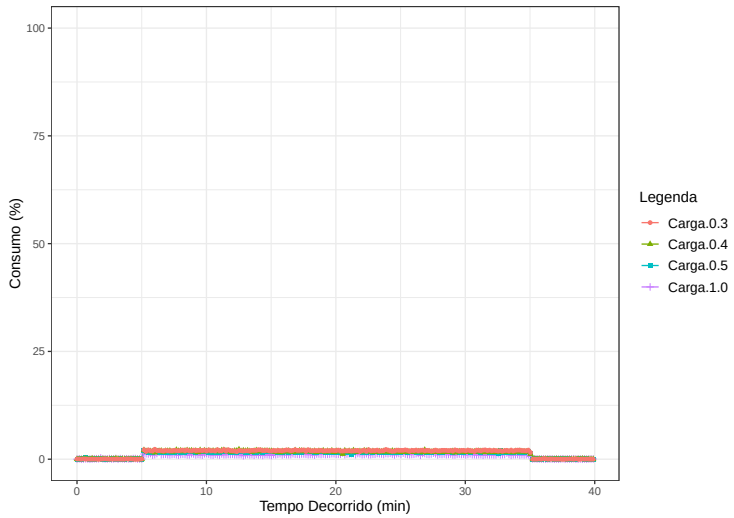
(b) Consumo de Memória RAM.

Fonte: do Autor.

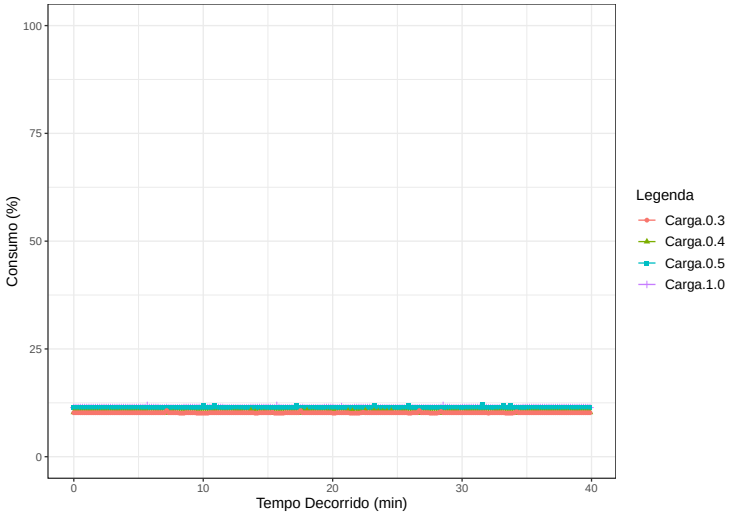
Levando em consideração o IDS baseado no modelo RF, as Figuras 18(a) e 18(b) apresentam os resultados obtidos para o consumo de CPU e memória RAM,

respectivamente, executando na VM servidor. O consumo médio de CPU foi de 1,11%, 1,69%, 1,82% e 2,00%, enquanto o consumo médio de memória RAM foi de 11,35%, 11,43%, 10,25% e 10,19% para as cargas de trabalho 1,0, 0,5, 0,4 e 0,3, respectivamente. Logo, pode ser observado uma oscilação de 0,89% e 1,16% de consumo médio de CPU e de memória RAM, respectivamente, entre a CT mais e menos intensa. Uma diferença muito pequena, similarmente ao IDS baseado no modelo NB.

Figura 18 – Consumo do IDS Random Forest na VM servidor.



(a) Consumo de CPU.



(b) Consumo de Memória RAM.

Fonte: do Autor.

Considerando o ambiente de nuvem, os resultados mostraram que os IDSs baseados em ML consumiram menos recurso computacional quando comparados aos IDSs baseados em DL. Vale ser destacado, ainda, que o modelo FNN-Multiclass consumiu todos os

recursos do equipamento no **minuto 12** após o início do experimento com a CT mais intensa (0,3). Logo, a execução desse IDS em um ambiente de nuvem pode comprometer a aplicação que estiver em execução, ou até mesmo todo o ambiente caso o IDS esteja localizado na borda da infraestrutura da nuvem (quando o IDS está localizado à frente de todos os serviços de nuvem). A Tabela 8 exibe um resumo dos consumos médios de CPU e de memória RAM para os experimentos realizados na nuvem para os diferentes IDSs e CTs.

Tabela 8 – Comparativo do consumo médio de CPU e memória RAM na VM servidor.

Modelo	CPU/RAM	CT 1,0	CT 0,5	CT 0,4	CT 0,3
FNN-B	CPU	14,86%	26,67%	30,07%	39,72%
	RAM	12,87%	13,31%	13,58%	13,98%
FNN-M	CPU	16,87%	28,51%	34,18%	49,70%
	RAM	12,93%	13,36%	13,72%	21,30%
NB	CPU	0,88%	1,42%	1,74%	2,09%
	RAM	10,44%	10,23%	10,27%	10,28%
RF	CPU	1,11%	1,68%	1,81%	2,00%
	RAM	11,35%	11,43%	10,25%	10,19%

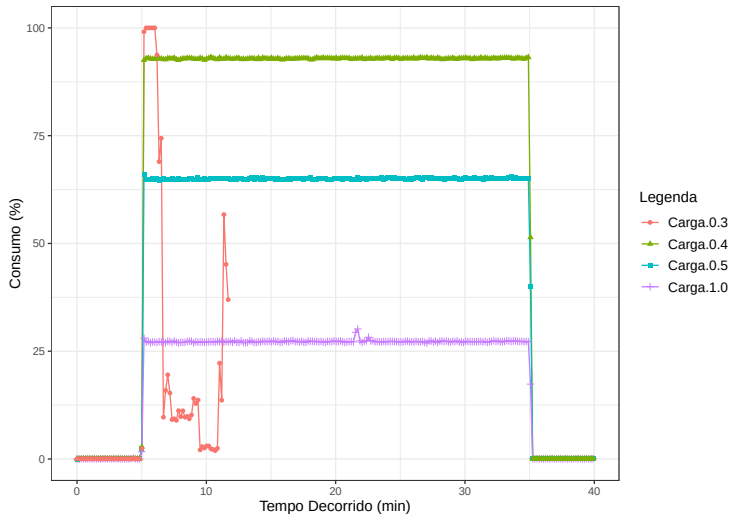
5.3 Ambiente de Borda

Nesta seção, serão apresentados os resultados coletados por meio dos experimentos realizados no ambiente de borda computacional. Nesse ambiente, a infraestrutura é utilizada para simular um ambiente de borda capaz de enviar tráfego (normal ou malicioso) por meio de um computador através de uma rede local para uma SBC, a qual executará um IDS baseado em ML ou DL. Busca-se, assim, avaliar o desempenho dos IDSs no dispositivo de borda, que é a base de estudos para minimizar a latência entre a comunicação das nuvens computacionais e os ambientes do ecossistema de IoT.

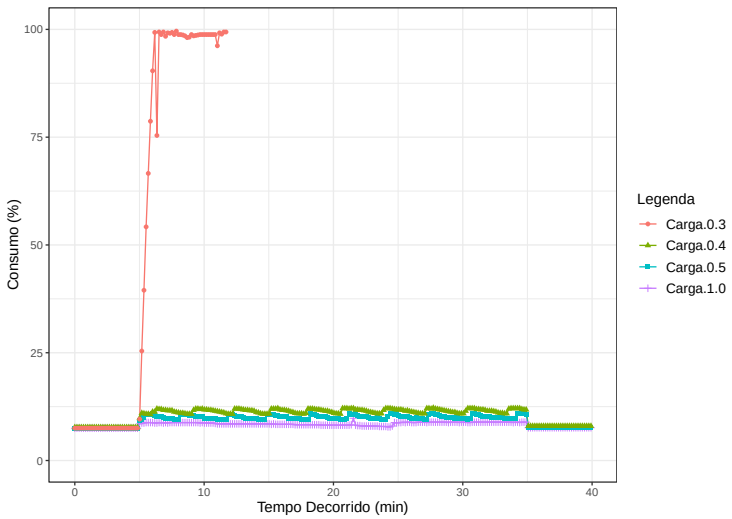
Considerando os IDSs baseado no modelo FNN-Binray, as Figuras 19(a) e 19(b) mostram o consumo de CPU e memória RAM, respectivamente, executando na SBC. O consumo médio de CPU foi de 27,20%, 64,90%, 92,67% e 99,85%, enquanto o consumo

médio de memória RAM foi de 8,59%, 10,08%, 11,48% e 59,13% para as CTs 1,0, 0,5, 0,4 e 0,3, respectivamente. Após o início do envio dos dados usando a CT 0,3, o dispositivo não conseguiu suportar o grande fluxo de dados, e no **minuto 6**, após o início do experimento, o equipamento consumiu todos os recursos de memória e de processador disponíveis, levando ao travamento do equipamento. Vale ser destacado, ainda, que no experimento considerando a CT 0,4, houve um consumo médio de CPU de 92,67%, o que pode inviabilizar a execução de outra aplicação no dispositivo.

Figura 19 – Consumo do IDS FNN-Binary na SBC.



(a) Consumo de CPU.



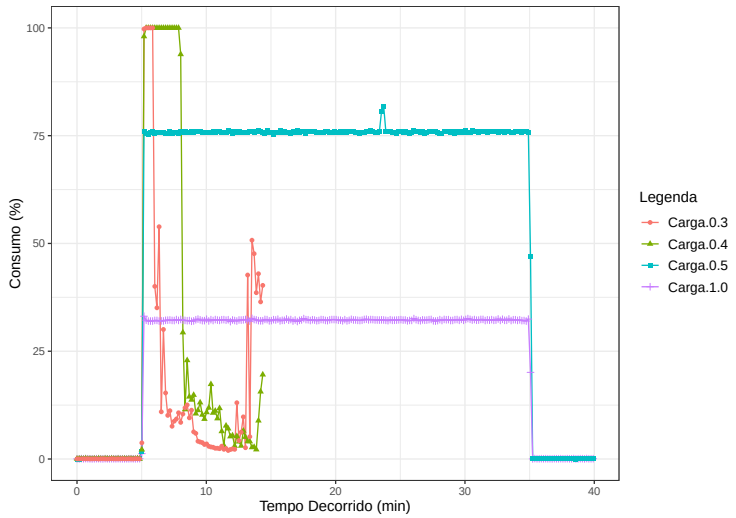
(b) Consumo de Memória RAM.

Fonte: do Autor.

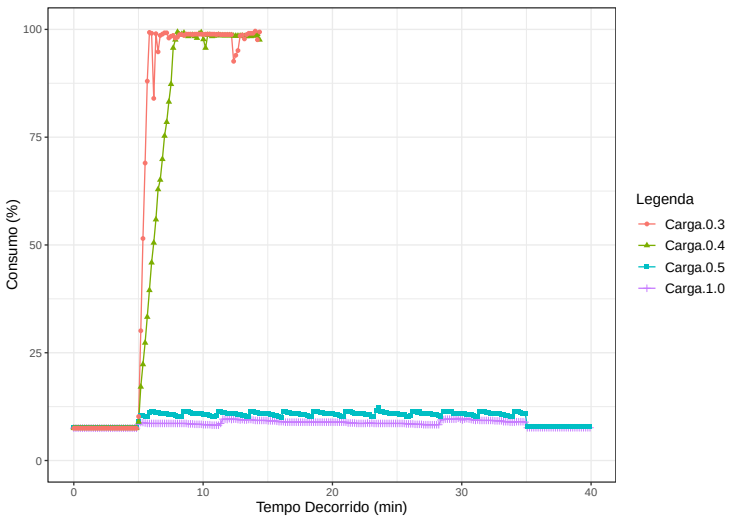
As Figuras 20(a) e 20(b) exibem os resultados dos experimentos executados na SBC pelo IDS baseado no modelo FNN-Multiclass. Considerando esse modelo, o consumo

médio de CPU foi de 32,16%, 75,74%, 99,88% e 99,95%, enquanto o consumo médio de memória RAM foi de 8,89%, 10,82%, 59,25% e 67,58% para as cargas de trabalho 1,0, 0,5, 0,4 e 0,3, respectivamente. Graficamente fica claro o alto consumo de CPU ao executar esse IDS na SBC, principalmente, com as cargas de trabalho mais intensas (0,4 e 0,3). Vale ser destacado que nessas cargas de trabalho a SBC consumiu todos os recursos de CPU e memória RAM, levando ao travamento do equipamento no **minuto 8 e 6** respectivamente, após o início do experimento.

Figura 20 – Consumo do IDS FNN-Multiclass na SBC.



(a) Consumo de CPU.



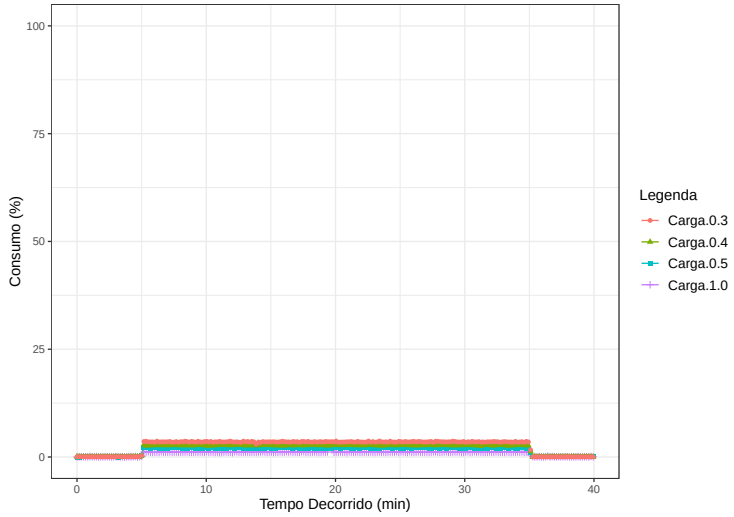
(b) Consumo de Memória RAM.

Fonte: do Autor.

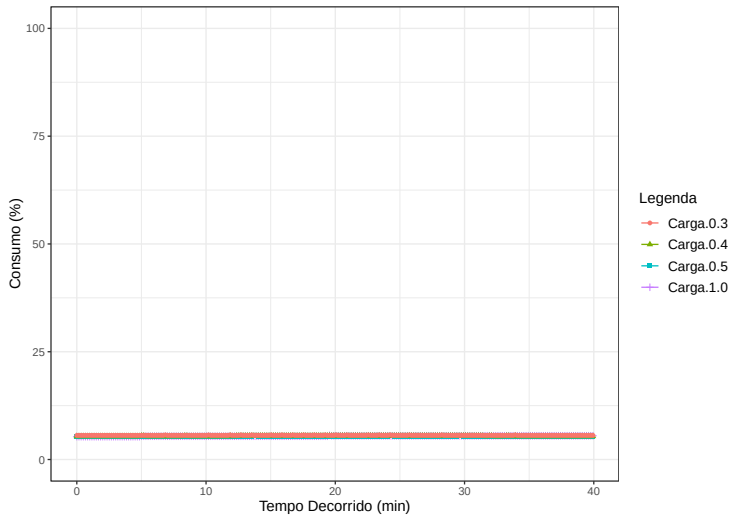
Já os resultados dos experimentos executando os IDSs baseados no modelo NB, podem ser observados nas Figuras 21(a) e 21(b). O consumo médio de CPU foi de 1,14%,

2,10%, 2,74% e 3,51%, enquanto o consumo de memória foi de 5,45%, 5,40%, 5,47% e 5,61% para as cargas de trabalho 1,0, 0,5, 0,4 e 0,3, respectivamente. Logo, houve um aumento de consumo médio de 2,36% para a CPU e 0,16% para a memória RAM, se comparados os consumos entre as CTs mais e menos intensas. Demonstrando assim, baixo consumo computacional ao executar os experimentos na SBC por meio do IDS baseado no modelo NB.

Figura 21 – Consumo do IDS Naive Bayes na SBC.



(a) Consumo de CPU.



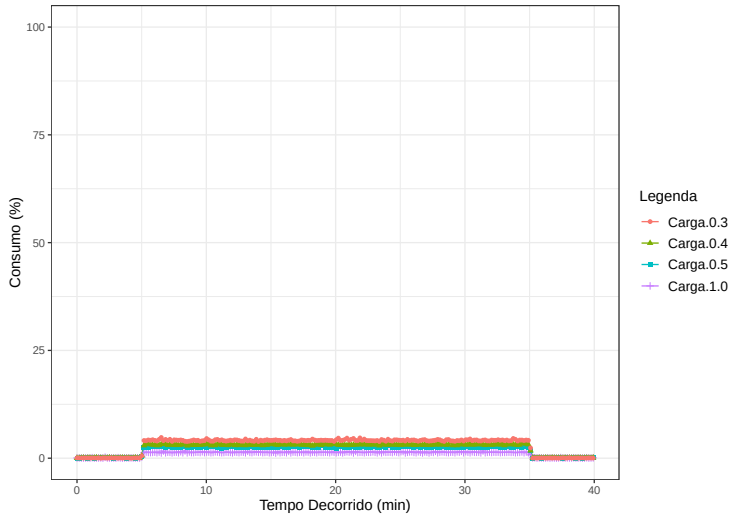
(b) Consumo de Memória RAM.

Fonte: do Autor.

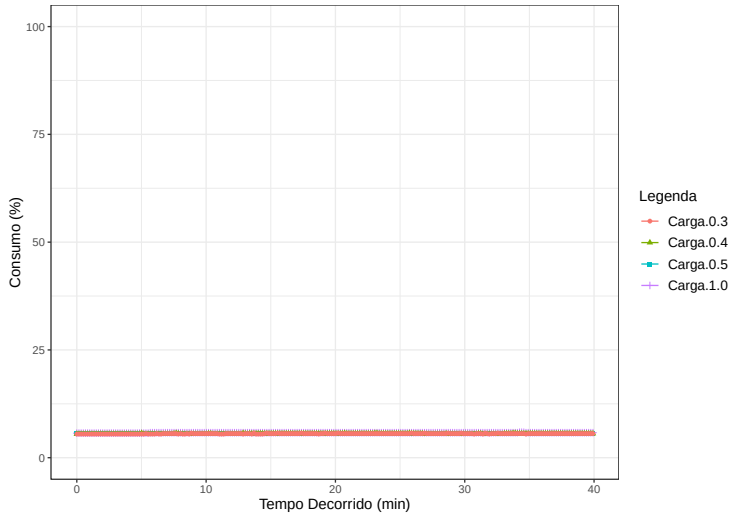
Os resultados obtidos através dos experimentos executados pelo IDS baseado no modelo RF são apresentados nas Figuras 22(a) e 22(b). O consumo médio de CPU foi de 1,37%, 2,62%, 3,16% e 4,14%, enquanto o consumo de memória RAM foi de 5,80%, 5,60%,

5,56%, 6,59% para as cargas de trabalho 1,0, 0,5, 0,4 e 0,3, respectivamente. Dessa forma, os dados mostram uma oscilação de 2,77% para o consumo médio de CPU e de 0,21% para o consumo médio de memória RAM entre as CTs de maior e menor intensidade. Evidenciando assim uma diferença relativamente pequena, semelhante ao IDS baseado no modelo NB.

Figura 22 – Consumo do IDS Random Forest na SBC.



(a) Consumo de CPU.



(b) Consumo de Memória RAM.

Fonte: do Autor.

Ao considerar o ambiente de borda, os resultados mostraram que os IDSs baseados em ML também consumiram menos recurso computacional frente aos IDSs baseados em DL, assim como no ambiente em nuvem. Diferente do ambiente de nuvem, no ambiente de borda os IDSs baseados em DL (FNN-Binary e FNN-Multiclass) consumiram os recursos

da SBC levando-a ao travamento. O IDS baseado no modelo FNN-Binary consumiu todos os recursos na CT 0,3 (a mais intensa) no **minuto 6** após o início do experimento. Já o IDS baseado no modelo FNN-Multiclass consumiu todos os recursos nas cargas de trabalho 0,4 e 0,3 nos **minutos 8 e 6**, respectivamente, após o início do experimento.

Dessa forma, a execução dos IDSs baseados em DL executando em um ambiente de borda, mais especificamente, em uma SBC, pode causar a indisponibilidade do ambiente, pois ao executar cargas de trabalhos mais intensas, há o consumo demasiado de recursos computacionais. A Tabela 9 exibe o consumo médio de CPU e de memória RAM para os experimentos realizados na SBC para os diferentes IDSs e cargas de trabalho.

Tabela 9 – Comparativo do consumo médio de CPU e memória RAM na SBC.

Modelo	CPU/RAM	CT 1,0	CT 0,5	CT 0,4	CT 0,3
FNN-B	CPU	27,20%	64,90%	92,67%	99,85%
	RAM	8,59%	10,08%	11,48%	59,13%
FNN-M	CPU	32,16%	75,74%	99,83%	99,95%
	RAM	8,89%	10,82%	59,25%	67,58%
NB	CPU	1,14%	2,10%	2,74%	3,50%
	RAM	4,45%	5,40%	5,46%	5,61%
RF	CPU	1,37%	2,62%	3,16%	4,14%
	RAM	5,80%	5,60%	5,56%	5,59%

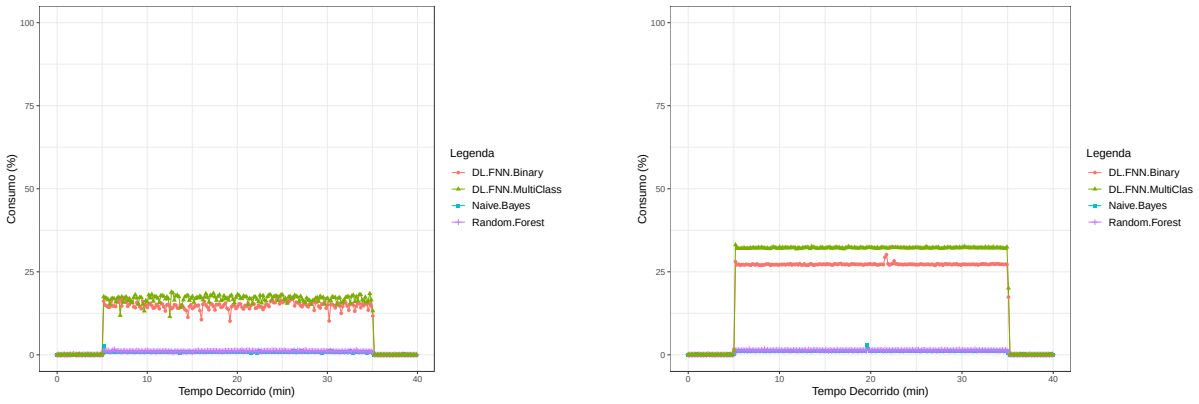
5.4 Comparação dos Resultados

Nesta seção, será apresentado um comparativo entre os resultados coletados executando os IDSs considerando as CTs da maior até a menor intensidade nos ambientes adotados, buscando analisar o consumo de CPU e memória RAM. Com isso, verificar o impacto dos IDSs em cada equipamento e, consequentemente, em cada contexto. Inicialmente, ao realizar o comparativo da memória RAM entre a SBC e a VM servidor, pode-se verificar que a VM servidor hospedada na nuvem consome mais recursos do que o dispositivo de borda. O consumo inicial da SBC parte de 7,50% frente a 12,20% do consumo da VM servidor. Logo, a VM servidor possui cerca de 62,66% a mais de consumo inicial de memória. Este fato está relacionado, a princípio, ao consumo dos recursos

essenciais de cada ambiente (por exemplo, o sistema operacional de cada dispositivo).

Considerando o consumo de CPU do ambiente em nuvem e do ambiente de borda executando os IDSs de ML e DL com a CT 1,0, pode-se observar (ver Figuras 23(a) e 23(b)) que ambos os equipamentos conseguem executar todos os experimentos sem grandes impactos. O consumo médio de CPU foi de 14,86%, 16,86%, 0,88% e 1,11% para a VM servidor e de 27,20%, 32,16%, 1,14% e 1,37% para a SBC, considerando os IDSs baseados em FNN-Binary, FNN-Multiclass, NB e RF, respectivamente. Como pode ser visto nos gráficos, os IDSs baseados em DL consumiram mais recursos em relação aos IDSs baseados em ML, de maneira constante, do início ao final do experimento, tanto na VM servidor quanto na SBC. Além disso, os IDSs executados na SBC consumiram em média 83,04%, 90,74%, 29,54% e 23,42% mais CPU em comparação com os IDSs executados na VM servidor, considerando os IDSs baseados em FNN-Binary, FNN-Multiclass, NB e RF, respectivamente. Logo, nota-se que os IDSs baseados em DL consumiram mais recursos quando comparamos aos baseados em ML. Ademais, também pode ser observado que todos os IDSs consumiram mais recursos computacionais na SBC quando comparados com a VM servidor.

Figura 23 – Consumo de CPU na VM servidor e na SBC com CT 1,0.



(a) Consumo na VM servidor.

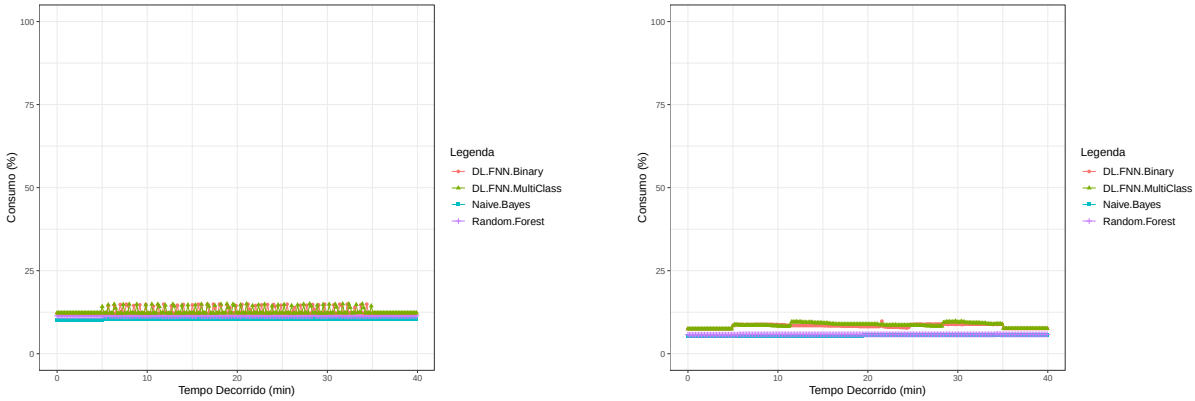
(b) Consumo na SBC.

Fonte: do Autor.

Considerando o consumo de memória RAM da VM servidor e da SBC com a CT 1,0, pode-se observar que nenhum experimento causou grande impacto (ver Figuras 24(a) e 24(b)), assim como o consumo de CPU. Ambos ambientes executaram todos os IDSs de maneira satisfatória, sem travamentos. O consumo médio de memória RAM da VM servidor foi de 12,87%, 12,93%, 10,44% e 11,35%, enquanto o consumo médio de memória

RAM da SBC foi de 8,59%, 8,89%, 4,45% e 5,80% para os IDSs baseados em FNN-Binary, FNN-Multiclass, NB e RF, respectivamente.

Figura 24 – Consumo de memória RAM na VM servidor e na SBC com CT 1,0.



(a) Consumo na VM servidor. (b) Consumo na SBC.

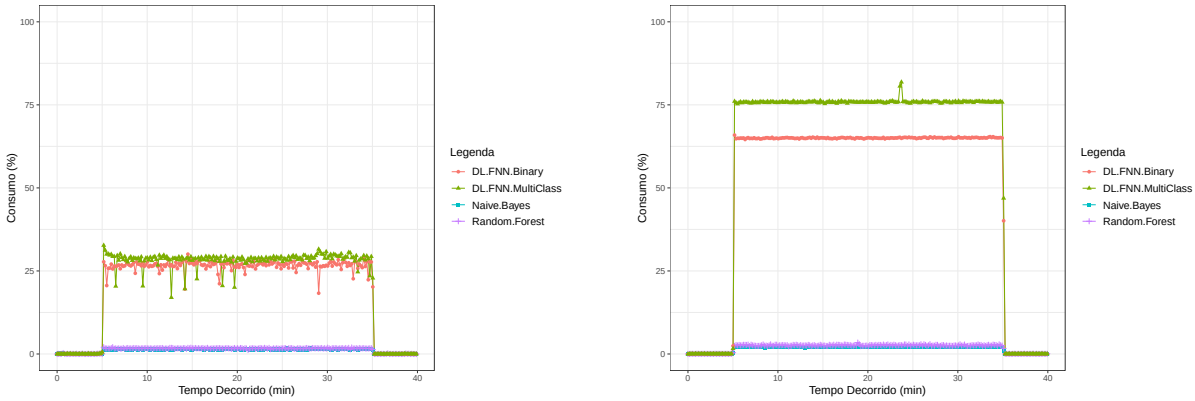
Fonte: do Autor.

Com base nos experimentos realizados no ambiente de nuvem e de borda considerando a CT 0,5, as Figuras 25(a) e 25(b) exibem um comparativo do consumo de CPU da VM servidor e da SBC, respectivamente. O consumo médio de CPU para a VM servidor foi de 26,68%, 28,51%, 1,42% e 1,69%, enquanto o consumo médio de CPU para a SBC foi de 64,90%, 75,74%, 2,1% e 2,61% para os IDSs baseados em FNN-Binary, FNN-Multiclass, NB e RF, respectivamente. Inicialmente, pode-se notar que houve um crescimento considerável do consumo médio de CPU quando comparado as CTs 1,0 e 0,5, principalmente, na SBC.

Considerando as CTs 1,0 e 0,5, o crescimento do consumo médio de CPU na VM servidor foi de 79,54%, 69,09%, 61,36% e 52,25%, enquanto o crescimento referente à SBC foi de 138,60%, 135,50%, 84,21% e 90,51% para os IDSs baseados nos modelos FNN-Binary, FNN-Multiclass, NB e RF, respectivamente. Ou seja, a execução da CT 0,5 (que é o dobro da CT 1,0) impactou, em alguns casos, em mais de 100% no consumo dos IDSs baseados em DL na SBC. Vale ser ressaltado que a VM servidor obteve menor consumo de CPU quando comparado ao consumo de CPU da SBC.

Considerando os resultados dos experimentos executados na VM servidor e na SBC com a CT 0,5, as Figuras 26(a) e 26(b) exibem os resultados do consumo da memória RAM da VM servidor e da SBC, respectivamente. O consumo médio de memória RAM executando os IDSs na VM servidor foi de 13,31%, 13,36%, 10,32% e 11,43%, enquanto

Figura 25 – Consumo de CPU na VM servidor e na SBC com CT 0,5.

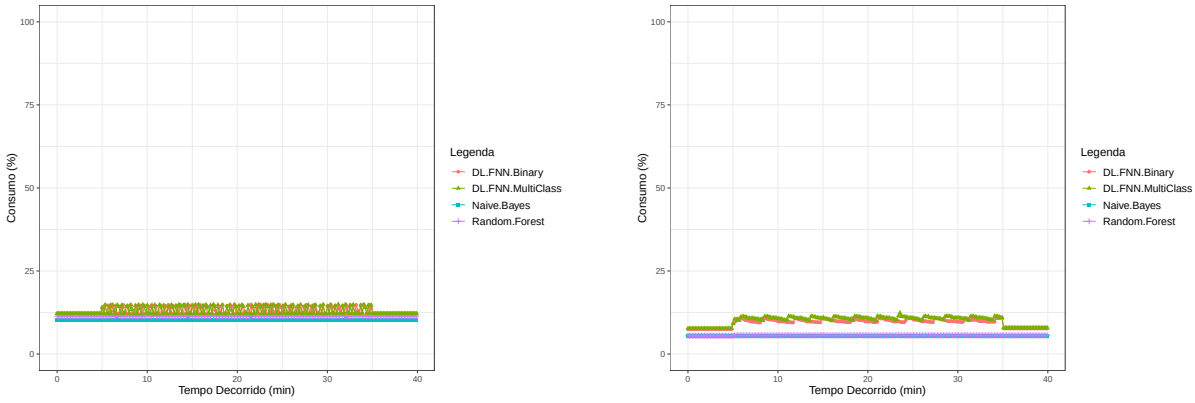


(a) Consumo na VM servidor. (b) Consumo na SBC.

Fonte: do Autor.

o consumo médio de memória RAM executando os IDSs na SBC foi de 10,08%, 10,82%, 5,40% e 5,60% para os IDSs baseados nos modelos FNN-Binary e FNN-Multiclass, NB e RF, respectivamente. Há de se notar que houve pouco crescimento no contexto da memória RAM, tendo em vista o aumento das cargas de trabalho de 1,0 para 0,5, diferentemente do que ocorre com o consumo de CPU. Neste momento, pode-se observar que os IDSs consomem pouca memória quando executam cargas de trabalho menos intensas.

Figura 26 – Consumo de memória RAM na VM servidor e na SBC com CT 0,5.



(a) Consumo na VM servidor. (b) Consumo na SBC.

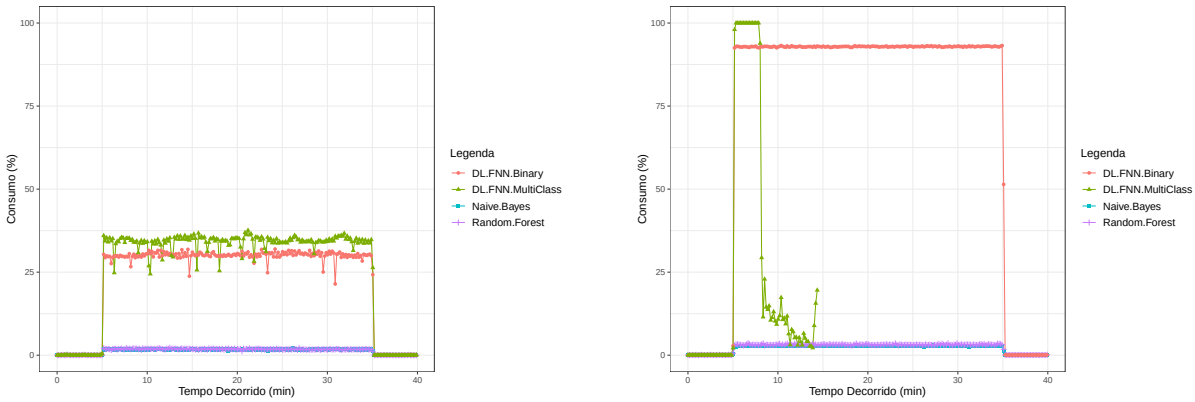
Fonte: do Autor.

Os resultados do consumo de CPU dos experimentos realizados na VM servidor e na SBC, considerando a CT 0,4, podem ser observados nas Figuras 27(a) e 27(b). A VM servidor pôde executar todo o experimento (com a CT 0,4) sem grandes impactos, assim como os experimentos executando as cargas de trabalho 1,0 e 0,5. Seu consumo médio de CPU foi de 30,07%, 34,18%, 1,74% e 1,82% para os IDSs baseados nos modelos

FNN-Binary, FNN-Multiclass, NB e RF, respectivamente. Por outro lado, o consumo médio de CPU para a SBC executando com a CT 0,4 foi de 92,67%, 99,88%, 2,74% e 3,16% para os mesmos IDSs.

Com base na Figura 27(b), pode ser observado que o experimento com o IDS baseado no modelo FNN-Multiclass executando na SBC consome todos os recursos de processador, levando ao travamento do equipamento no **minuto 8** após o início do experimento. Considerando as CTs 0,5 e 0,4 na VM servidor, há um aumento no consumo médio de CPU de 12,71%, 19,89%, 22,53%, 7,69%, enquanto na SBC o aumento do consumo médio de CPU foi de 32,57%, 31,87%, 30,48%, 20,61%, considerando os IDSs baseados nos modelos FNN-Binary, FNN-Multiclass, NB e RF, respectivamente. Evidenciando, assim, o maior impacto no consumo da SBC em relação à VM servidor, alinhado ao que ocorreu no comparativo entre os ambientes e a CT 0,5. Ademais, os dados demonstram o alto impacto das cargas de trabalhos mais intensas nos IDSs baseados em DL em dispositivos com baixo poder computacional (a exemplo da SBC), levando inclusive a sua indisponibilidade.

Figura 27 – Consumo de CPU na VM servidor e na SBC com CT 0,4.



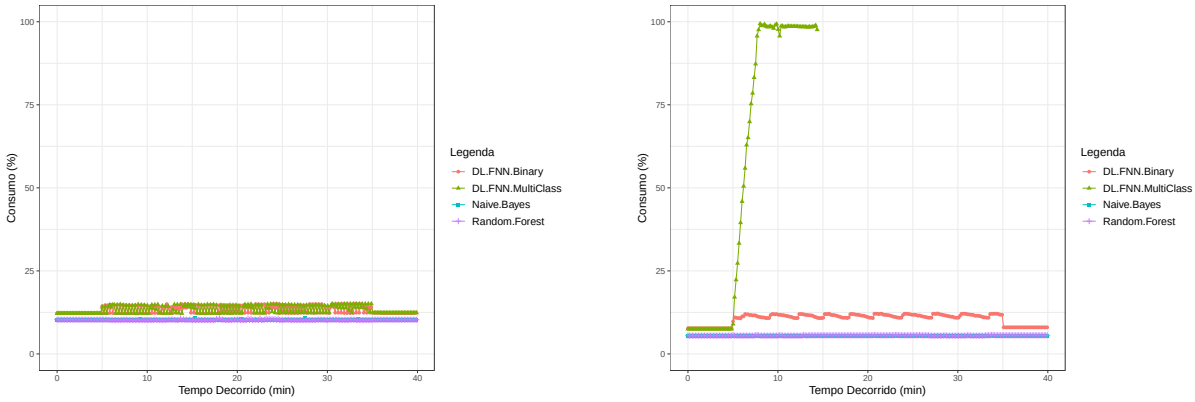
(a) Consumo na VM servidor. (b) Consumo na SBC.

Fonte: do Autor.

As Figuras 28(a) e 28(b) exibem os resultados dos experimentos considerando o consumo de memória RAM na VM servidor e na SBC, respectivamente. Em geral, o consumo foi relativamente baixo, destoando apenas o experimento executado na SBC com o IDS baseado no modelo FNN-Multiclass, o qual consumiu toda a memória RAM do equipamento, levando a seu travamento. O consumo médio de memória RAM para a VM servidor foi de 13,58%, 13,72%, 10,27% e 10,25%, enquanto o consumo médio de memória RAM para a SBC foi de 11,48%, 59,25%, 5,47% e 5,56%, considerando os IDSs baseados em

FNN-Binary e FNN-Multiclass, NB e RF, respectivamente. Logo, apenas o IDS baseado no modelo FNN-Multiclass levou ao consumo total de memória, e consequentemente, ao travamento do equipamento. Nota-se que a partir do momento em que a memória RAM não consegue mais armazenar os dados, o processador começa a aumentar seu consumo rapidamente até o travamento (ver Figuras 27(b) e 28(b)).

Figura 28 – Consumo de memória RAM na VM servidor e na SBC com CT 0,4.



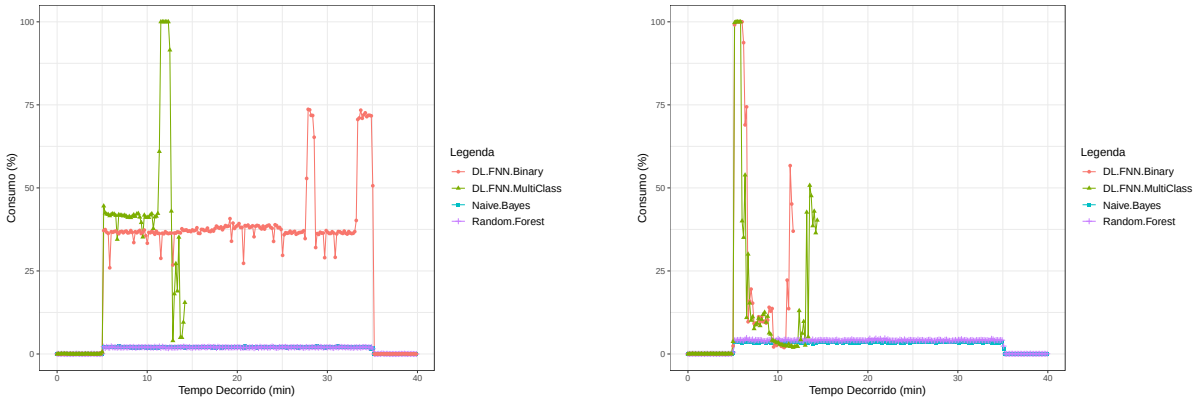
(a) Consumo na VM servidor. (b) Consumo na SBC.

Fonte: do Autor.

As Figuras 29(a) e 29(b) exibem o consumo de CPU dos experimentos executados na VM servidor e na SBC, considerando a CT 0,3, respectivamente. Considerando a VM servidor, pode ser observado que há o travamento do equipamento ao executar o experimento com o IDS baseado no modelo FNN-Multiclass no **minuto 12** após o início do experimento, enquanto na SBC há o travamento do equipamento ao executar o experimento com os IDSs baseados nos modelos FNN-Multiclass e FNN-Binary nos **minutos 6 e 8**, respectivamente, após o início do experimento. O consumo médio de CPU para a VM servidor foi de 39,73%, 49,70%, 2,10% e 2,01% ao passo que o consumo médio de CPU para a SBC foi de 13,98%, 21,30%, 10,28% e 10,19% considerando os IDSs baseados nos modelos FNN-Binary e FNN-Multiclass, NB e RF, respectivamente. Tais dados evidenciam o alto consumo de CPU nos experimentos executando cargas de trabalhos mais intensas, principalmente na SBC, visto que possui recursos mais limitados.

Já as Figuras 30(a) e 30(b) exibem o consumo de memória RAM dos experimentos executados na VM servidor e na SBC, considerando a CT 0,3, respectivamente. O consumo médio de memória RAM da VM servidor foi de 13,98%, 21,30%, 10,28% e 10,19% enquanto o consumo da SBC foi de 59,13%, 67,58%, 5,61% e 5,59% considerando os IDSs baseados

Figura 29 – Consumo de CPU na VM servidor e na SBC com CT 0,3.

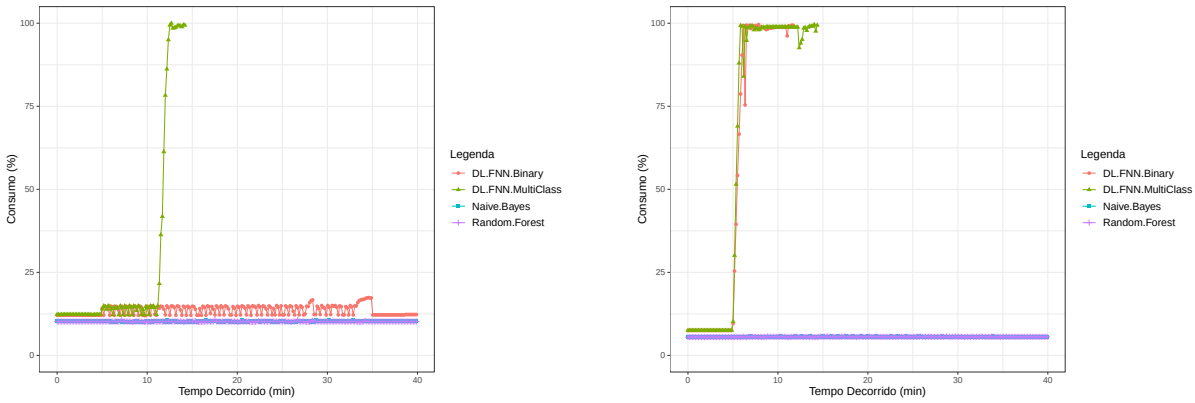


(a) Consumo na VM servidor. (b) Consumo na SBC.

Fonte: do Autor.

nos modelos FNN-Binary e FNN-Multiclass, NB e RF, respectivamente. Vale ser destacado que o alto consumo de memória RAM nos experimentos executando os IDSs baseados em DL na SBC está relacionado ao travamento dos equipamentos, pois o processador não conseguiu tratar as requisições e consumiu toda a memória RAM.

Figura 30 – Consumo de memória RAM na VM servidor e na SBC com CT 0,3.



(a) Consumo na VM servidor. (b) Consumo na SBC.

Fonte: do Autor.

Os comparativos de desempenho entre os ambientes de nuvem e borda computacional considerando os diferentes IDSs baseados em ML e DL, evidenciam o alto consumo de CPU nos experimentos executando os IDSs baseados nos modelos FNN-Binary e FNN-Multiclass em cargas de trabalhos mais intensas (0,4 e 0,3), levando inclusive à indisponibilidade dos ambientes. A VM servidor, por possuir maior poder de processamento, obteve melhor desempenho ao executar os experimentos, quando comparado à SBC. Cargas de trabalhos menos intensas (1,0 e 0,5) foram executadas sem grandes impactos em todos

os experimentos. Além disso, os IDSs baseados em ML obtiveram bons resultados nos experimentos executados, frente aos IDSs baseados em DL. Dessa forma, ao desenvolver e implementar IDSs baseados em IA, mais especificamente baseados em DL, considerando os experimentos executados, existe a possibilidade de grande consumo de CPU, possibilitando o travamento do equipamento e, consequentemente, do ambiente.

5.5 Considerações Finais

Este capítulo apresentou os resultados obtidos pelas métricas acurácia, precisão, revocação e pontuação F1 após o treinamento e validação dos modelos de IA, além de exibir o consumo de CPU e de memória RAM após a execução dos experimentos nos ambientes de nuvem e borda computacional pelas diferentes implementações de IDS e diferentes CTs. Ao considerar a métrica acurácia, os resultados foram 98,79%, 99,99%, 45,58% e 99,61%, ou seja, boa aprendizagem dos modelos NB, RF e FNN-Multiclass e baixa aprendizagem do modelo FNN-Binary.

Considerando a métrica CPU, os resultados evidenciaram alto consumo nos experimentos executando os IDSs baseados em DL, tanto no ambiente em nuvem quanto no ambiente de borda, levando até mesmo, em alguns casos, ao travamento do equipamento (VM servidor e SBC). Por outro lado, ao considerar o consumo da memória RAM, os resultados revelaram baixo consumo dos IDSs baseados nos modelos de ML. No entanto, os IDSs baseados nos modelos de DL consumiram totalmente a memória RAM disponível na SBC ao executarem cargas de trabalhos mais intensas (0,4 e 0,3), a depender do modelo, levando inclusive à indisponibilidade do equipamento.

Dessa forma, os IDSs baseados em ML obtiveram boas acurácias e bom desempenho na execução dos experimentos, considerando o ambiente e contexto. Os resultados podem direcionar o desenvolvimento de novos IDSs baseados em IA que serão implantados em nuvens computacionais e, principalmente, nos equipamentos usados na computação de borda, que possuem recursos computacionais limitados.

6 Conclusão

Este trabalho apresentou a avaliação de desempenho experimental de quatro IDSs, sendo dois utilizando modelos de ML (Naive Bayes e Random Forest) e dois utilizando modelos de DL (FNN-Binary e FNN-MultiClass). Tais sistemas são capazes de classificar se uma requisição é maliciosa ou não em um dispositivo de borda (Raspbery Pi 4 Modelo B) ou em uma máquina virtual utilizando o ambiente em nuvem da Google. Os experimentos consideraram diferentes cargas de trabalho para simular diferentes cenários em nuvem e de borda. Nossos experimentos constataram que os IDSs baseados nos modelos de ML obtiveram melhores desempenho em relação ao consumo médio de CPU e de memória RAM, quando comparados aos sistemas baseados nos modelos de DL. No cenário com uma carga de trabalho de maior intensidade (0,3), o ambiente de borda obteve uma média de processamento mais elevada que o ambiente em nuvem para os modelos adotados. Evidenciou-se, ainda, que os IDSs baseados em DL levaram ao travamento do dispositivo de borda em diferentes cenários devido ao consumo excessivo de recursos.

Além disso, os modelos de ML, nestes experimentos e contexto, obtiveram melhores acurácias que os modelos de DL, além de consumirem menos recursos computacionais nos ambientes em nuvem e de borda. Quando os IDSs baseados em modelos de DL obtiveram um bom desempenho na métrica acurácia, houve grande consumo CPU e memória RAM. Diferentemente dos IDSs baseados em ML, que obtiveram bom desempenho na métrica acurácia e baixo consumo de CPU e memória RAM. Também foi atestado que os experimentos utilizando o ambiente em nuvem (VM servidor) obtiveram melhores resultados em comparação aos experimentos realizados no dispositivo de borda, já que a VM possui um maior poder computacional em termo de processamento.

Logo, de acordo com a arquitetura usada na pesquisa, considerando os ambientes de nuvem e borda, os IDSs baseados em modelos de ML são mais adequados para ambientes que possuem recursos computacionais limitados (mas também podem ser usados em ambientes com grandes recursos computacionais, como na nuvem). Por outro lado, IDSs baseados em modelos de DL apresentaram um desempenho moderado, ou seja, necessita de ambientes com maior de poder de processamento, visto que em dispositivos de baixo poder computacional, pode ocasionar a indisponibilidade do dispositivo e até mesmo do ambiente (vale ser destacado aqui que a nuvem é altamente escalável). Portanto, antes de

implementar um IDS utilizando técnicas de ML e DL em ambientes de nuvem ou de borda é importante uma análise profunda, pois dependendo da complexidade e dos recursos disponíveis, os IDSs podem afetar diretamente no funcionamento do ambiente, levando inclusive à indisponibilidade do dispositivo.

6.1 Contribuições

Nesta seção são apresentadas as contribuições deste trabalho:

- A avaliação de desempenho de ambientes baseados em nuvem e borda, considerando IDSs baseados em ML e DL. Tal avaliação permite ajudar na tomada de decisão de futuras implementações, pois a depender do ambiente e dos recursos disponíveis, esses IDSs podem causar a indisponibilidade do ambiente.
- O desenvolvimento de IDSs baseados em ML e DL implantados em ambientes de nuvem e borda. A integração entre a segurança da informação e a IA permite o desenvolvimento de novos sistemas capazes de aumentar a eficiência, eficácia e nível de segurança nas organizações, considerando os ambientes mais atuais (nuvem e borda).
- O método adotado na execução dos experimentos para desenvolver os IDSs baseados em IA e analisar o impacto da implantação nos ambientes (de nuvem e de borda). Este método permite a replicação dos IDSs e dos ambientes em futuros experimentos.
- Aceitação de artigo no *workshop* em Desempenho de Sistemas Computacionais e de Comunicação (WPERFORMANCE) 2021:
  CARVALHO, V.; QUEIROZ, E.; MENDONÇA, J.; CALLOU, G.; ANDRADE, E. Avaliação de desempenho de modelos deep learning para detecção de intrusão em dispositivos iot. In: SBC. Anais do XX Workshop em Desempenho de Sistemas Computacionais e de Comunicação. [S.l.], 2021. p. 1–12.
- Submissão de artigo no *workshop* em Desempenho de Sistemas Computacionais e de Comunicação (WPERFORMANCE) 2023:
  CARVALHO, V.; QUEIROZ, E.; MENDONÇA, J.; CALLOU, G.; ANDRADE, E. Avaliação de Desempenho de Sistemas de Detecção de Intrusão Baseados em IA para Ambientes de Nuvem e Borda. In: SBC. Anais do XXII Workshop em Desempenho de Sistemas Computacionais e de Comunicação.

6.2 Limitações

As limitações desta dissertação são descritas a seguir:

- Este trabalho limitou-se a avaliar apenas dois modelos de ML (Naive Bayes e Random Forest) e dois modelos de DL (FNN-Binary e FNN-Multiclass), onde utilizou apenas uma técnica de DL, chamada de *FeedForward Neural Network*.
- Os experimentos foram conduzidos em apenas um dispositivo de borda (Raspberry Pi 4 Model B) e em uma nuvem computacional (GCP) devido ao custo envolvido para comprar outros dispositivos e alugar outros ambientes em nuvem.
- Os experimentos foram realizados através da simulação de envio envio de dados contidos em um *dataset* público (Bot-IoT). Dessa forma, não foi necessário a execução de ataques reais através de softwares como metasploit, hping, nmap.
- Apenas as métricas de consumo de CPU e memória RAM foram coletadas e não houve o uso de área de troca SWAP nos experimentos realizados. No entanto, a área de troca SWAP poderia auxiliar e aumentar o desempenho da memória.
- Houve a coleta e análise dos tempos de classificação de cada IDS baseado em ML e DL. Após tratamento dos dados e análise estatística, foi constatado que os dados estavam inconsistentes. Além disso, não houve a coleta do *delay* entre o envio e recebimento dos dados na comunicação entre cliente e servidor.

6.3 Trabalhos Futuros

Como trabalhos futuros, pode-se continuar a avaliar os modelos considerando outros tipos de ataques, ou até mesmo outros *datasets* com o objetivo de aumentar a acurácia dos modelos, principalmente do modelo FNN-Binary. Além disso, coletar informações relacionadas ao consumo de energia nos equipamentos, sempre visando realizar o comparativo entre os ambientes e IDSs. Essa métrica trará mais dados, que auxiliarão na tomada de decisão no momento de implementar um IDS usando IA.

Outra possível extensão deste trabalho é testar a metodologia proposta para avaliar o desempenho de diferentes modelos de ML e DL implementando novos IDSs e, assim, analisar os impactos nos ambientes. Diminuir a quantidade de atributos visando obter menor consumo de CPU e memória RAM sem impactar as métricas de aprendizagem poderia ser outro caminho para a extensão deste trabalho.

Outra possível extensão deste trabalho é analisar como mitigar os impactos no dispositivo de borda e na máquina virtual com o objetivo de evitar, ou até mesmo minimizar, o colapso dos recursos disponíveis.

Por fim, outra possível extensão deste trabalho é a criação de uma ferramenta de automação para execução de testes voltados aos IDSs em ambientes de IoT buscando facilitar, agilizar e padronizar os testes realizados nos IDSs. A padronização tornará, inclusive, a comparação mais justa entre os modelos analisados.

Referências

AGATONOVIC-KUSTRIN, S.; BERESFORD, R. Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research. **Journal of pharmaceutical and biomedical analysis**, Elsevier, v. 22, n. 5, p. 717–727, 2000.

AHMAD, W.; RASOOL, A.; JAVED, A. R.; BAKER, T.; JALIL, Z. Cyber security in iot-based cloud computing: A comprehensive survey. **Electronics**, MDPI, v. 11, n. 1, p. 16, 2021.

ARSHAD, J.; AZAD, M. A.; AMAD, R.; SALAH, K.; ALAZAB, M.; IQBAL, R. A review of performance, energy and privacy of intrusion detection systems for iot. **Electronics**, Multidisciplinary Digital Publishing Institute, v. 9, n. 4, p. 629, 2020.

ASHTON, K. et al. That 'internet of things' thing. **RFID journal**, v. 22, n. 7, p. 97–114, 2009.

AVIZIENIS, A.; LAPRIE, J.-C.; RANDELL, B.; LANDWEHR, C. Basic concepts and taxonomy of dependable and secure computing. **IEEE transactions on dependable and secure computing**, IEEE, v. 1, n. 1, p. 11–33, 2004.

AYODELE, T. O. Types of machine learning algorithms. **New advances in machine learning**, University of Portsmouth UK, v. 3, p. 19–48, 2010.

BADGER, M. L.; GRANCE, T.; PATT-CORNER, R.; VOAS, J. M. **Cloud computing synopsis and recommendations**. [S.l.]: National Institute of Standards & Technology, 2012.

BREWER, R. Ransomware attacks: detection, prevention and cure. **Network security**, Elsevier, v. 2016, n. 9, p. 5–9, 2016.

CHIBA, Z.; ABGHOUR, N.; MOUSSAID, K.; RIDA, M. et al. Intelligent approach to build a deep neural network based ids for cloud environment using combination of machine learning algorithms. **Computers & Security**, Elsevier, v. 86, p. 291–317, 2019.

CHINNAMGARI, S. K. **R Machine Learning Projects: Implement supervised, unsupervised, and reinforcement learning techniques using R 3.5**. [S.l.]: Packt Publishing Ltd, 2019.

CHONG, H.-Q.; TAN, A.-H.; NG, G.-W. Integrated cognitive architectures: a survey. **Artificial Intelligence Review**, Springer, v. 28, n. 2, p. 103–130, 2007.

CHOUDHURY, S.; BHOWAL, A. Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection. In: IEEE. **2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)**. [S.l.], 2015. p. 89–95.

CICIDS. **Intrusion Detection Evaluation Dataset - CICIDS2017**. 2017. <https://www.unb.ca/cic/datasets/ids-2017.html>. Accessed: 2022-05-18.

CREVIER, D. **AI: the tumultuous history of the search for artificial intelligence**. [S.l.]: Basic Books, Inc., 1993.

DAI, Y.; XU, D.; MAHARJAN, S.; QIAO, G.; ZHANG, Y. Artificial intelligence empowered edge computing and caching for internet of vehicles. **IEEE Wireless Communications**, IEEE, v. 26, n. 3, p. 12–18, 2019.

DALIANIS, H. Evaluation metrics and evaluation. In: **Clinical text mining**. [S.l.]: Springer, 2018. p. 45–53.

DANTAS, J. R. Planejamento de infraestrutura de nuvens computacionais para serviço de vod streaming considerando desempenho, disponibilidade e custo. Universidade Federal de Pernambuco, 2018.

DHANABAL, L.; SHANTHARAJAH, S. A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. **International journal of advanced research in computer and communication engineering**, v. 4, n. 6, p. 446–452, 2015.

EREMIA, M.; TOMA, L.; SANDULEAC, M. The smart city concept in the 21st century. **Procedia Engineering**, Elsevier, v. 181, p. 12–19, 2017.

ESKANDARI, M.; JANJUA, Z. H.; VECCHIO, M.; ANTONELLI, F. Passban ids: An intelligent anomaly-based intrusion detection system for iot edge devices. **IEEE Internet of Things Journal**, IEEE, v. 7, n. 8, p. 6882–6897, 2020.

FERNÁNDEZ, C. M.; RODRÍGUEZ, M. D.; MUÑOZ, B. R. An edge computing architecture in the internet of things. In: IEEE. **2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC)**. [S.l.], 2018. p. 99–102.

FERRAG, M. A.; MAGLARAS, L.; MOSCHOYIANNIS, S.; JANICKE, H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. **Journal of Information Security and Applications**, Elsevier, v. 50, p. 102419, 2020.

GE, M.; FU, X.; SYED, N.; BAIG, Z.; TEO, G.; ROBLES-KELLY, A. Deep learning-based intrusion detection for iot networks. In: IEEE. **2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)**. [S.l.], 2019. p. 256–25609.

HARA, K.; SAITO, D.; SHOUNO, H. Analysis of function of rectified linear unit used in deep learning. In: IEEE. **2015 international joint conference on neural networks (IJCNN)**. [S.l.], 2015. p. 1–8.

POSTEL, J. (Ed.). **RFC 791 Internet Protocol - DARPA Inernet Programm, Protocol Specification**. [S.l.], 1981. Disponível em: <http://tools.ietf.org/html/rfc791>.

IVAKHNENKO, A.; LAPA, V. Cybernetic predicting devices. ccm information corporation. **First working Deep Learners with many layers, learning internal representations**, 1965.

IVAKHNENKO, A. G.; IVAKHNENKO, A. G.; LAPA, V. G.; LAPA, V. G. **Cybernetics and forecasting techniques**. [S.l.]: American Elsevier Publishing Company, 1967. v. 8.

JAIN, A. K.; GUPTA, B. A survey of phishing attack techniques, defence mechanisms and open research challenges. **Enterprise Information Systems**, Taylor & Francis, v. 16, n. 4, p. 527–565, 2022.

JAIN, R. **The art of computer systems performance analysis**. [S.l.]: john wiley & sons, 2008.

JENITHA, G.; VENNILA, V. Comparing the partitional and density based clustering algorithms by using weka tool. In: IEEE. **Second International Conference on Current Trends In Engineering and Technology-ICCTET 2014**. [S.l.], 2014. p. 328–331.

JOHN, L. K.; EECKHOUT, L. **Performance evaluation and benchmarking**. [S.l.]: CRC Press, 2018.

KALMAN, B. L.; KWASNY, S. C. Why tanh: choosing a sigmoidal function. In: IEEE. **[Proceedings 1992] IJCNN International Joint Conference on Neural Networks**. [S.l.], 1992. v. 4, p. 578–581.

KASS, G. V. An exploratory technique for investigating large quantities of categorical data. **Journal of the Royal Statistical Society: Series C (Applied Statistics)**, Wiley Online Library, v. 29, n. 2, p. 119–127, 1980.

KHAN, F. A.; GUMAEI, A. A comparative study of machine learning classifiers for network intrusion detection. In: SPRINGER. **International conference on artificial intelligence and security**. [S.l.], 2019. p. 75–86.

KHAN, W. Z.; AHMED, E.; HAKAK, S.; YAQOOB, I.; AHMED, A. Edge computing: A survey. **Future Generation Computer Systems**, Elsevier, v. 97, p. 219–235, 2019.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KOCHER, G.; KUMAR, G. Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges. **Soft Computing**, Springer, v. 25, n. 15, p. 9731–9763, 2021.

KORONIOTIS, N.; MOUSTAFA, N.; SITNIKOVA, E.; TURNBULL, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. **Future Generation Computer Systems**, Elsevier, v. 100, p. 779–796, 2019.

KUMAR, G. P.; VENKATARAM, P. Artificial intelligence approaches to network management: recent advances and a survey. **Computer Communications**, Elsevier, v. 20, n. 15, p. 1313–1322, 1997.

LAU, F.; RUBIN, S. H.; SMITH, M. H.; TRAJKOVIC, L. Distributed denial of service attacks. In: IEEE. **Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0**. [S.l.], 2000. v. 3, p. 2275–2280.

LI, B.; FRIEDMAN, J.; OLSHEN, R.; STONE, C. Classification and regression trees (cart). **Biometrics**, v. 40, n. 3, p. 358–361, 1984.

LI, S.; LI, Y.; HAN, W.; DU, X.; GUIZANI, M.; TIAN, Z. Malicious mining code detection based on ensemble learning in cloud computing environment. **Simulation Modelling Practice and Theory**, Elsevier, v. 113, p. 102391, 2021.

LIAO, H.-J.; LIN, C.-H. R.; LIN, Y.-C.; TUNG, K.-Y. Intrusion detection system: A comprehensive review. **Journal of Network and Computer Applications**, Elsevier, v. 36, n. 1, p. 16–24, 2013.

LOH, W.-Y. Classification and regression trees. **Wiley interdisciplinary reviews: data mining and knowledge discovery**, Wiley Online Library, v. 1, n. 1, p. 14–23, 2011.

MACH, P.; BECVAR, Z. Mobile edge computing: A survey on architecture and computation offloading. **IEEE communications surveys & tutorials**, IEEE, v. 19, n. 3, p. 1628–1656, 2017.

MCCORDUCK, P.; CFE, C. **Machines who think: A personal inquiry into the history and prospects of artificial intelligence**. [S.l.]: CRC Press, 2004.

MELLIT, A.; KALOGIROU, S. A. Artificial intelligence techniques for photovoltaic applications: A review. **Progress in energy and combustion science**, Elsevier, v. 34, n. 5, p. 574–632, 2008.

MERIAH, I.; RABAI, L. B. A. Comparative study of ontologies based iso 27000 series security standards. **Procedia Computer Science**, Elsevier, v. 160, p. 85–92, 2019.

MOTII, M.; SEMMA, A. Towards a new approach to pooling cobit 5 and itil v3 with iso/iec 27002 for better use of itg in the moroccan parliament. **IJCSI International Journal of Computer Science Issues**, v. 14, n. 3, p. 49–58, 2017.

MOUSTAFA, N.; TURNBULL, B.; CHOO, K.-K. R. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. **IEEE Internet of Things Journal**, IEEE, v. 6, n. 3, p. 4815–4830, 2018.

NORMAN, D. A. Approaches to the study of intelligence. **Artificial Intelligence**, Elsevier, v. 47, n. 1-3, p. 327–346, 1991.

NSL-KDD. **NSL-KDD dataset**. 2009. <https://www.unb.ca/cic/datasets/nsl.html>. Accessed: 2022-05-18.

ONGSULEE, P. Artificial intelligence, machine learning and deep learning. In: IEEE. **2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE)**. [S.l.], 2017. p. 1–6.

OTOUM, Y.; LIU, D.; NAYAK, A. Dl-ids: a deep learning–based intrusion detection framework for securing iot. **Transactions on Emerging Telecommunications Technologies**, Wiley Online Library, v. 33, n. 3, p. e3803, 2022.

POSTEL, J. **RFC 793: Transmission Control Protocol**. 1981. See also STD0007 (**??**). Status: STANDARD. Disponível em: <ftp://ftp.internic.net/rfc/rfc793.txt,ftp://ftp.math.utah.edu/pub/rfc/rfc793.txt>.

POUYANFAR, S.; SADIQ, S.; YAN, Y.; TIAN, H.; TAO, Y.; REYES, M. P.; SHYU, M.-L.; CHEN, S.-C.; IYENGAR, S. S. A survey on deep learning: Algorithms, techniques, and applications. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 51, n. 5, p. 1–36, 2018.

QIAN, L.; LUO, Z.; DU, Y.; GUO, L. Cloud computing: An overview. In: SPRINGER. **IEEE International Conference on Cloud Computing**. [S.l.], 2009. p. 626–631.

QURESHI, K. N.; JEON, G.; PICCIALLI, F. Anomaly detection and trust authority in artificial intelligence and cloud computing. **Computer Networks**, Elsevier, v. 184, p. 107647, 2021.

REVATHI, S.; MALATHI, A. A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection. **International Journal of Engineering Research & Technology (IJERT)**, Citeseer, v. 2, n. 12, p. 1848–1853, 2013.

SABAHI, F.; MOVAGHAR, A. Intrusion detection: A survey. In: IEEE. **2008 Third International Conference on Systems and Networks Communications**. [S.l.], 2008. p. 23–26.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural networks**, Elsevier, v. 61, p. 85–117, 2015.

SHAIKH, F. B.; HAIDER, S. Security threats in cloud computing. In: IEEE. **2011 International conference for Internet technology and secured transactions**. [S.l.], 2011. p. 214–219.

SHARMA, S.; SHARMA, S.; ATHAIYA, A. Activation functions in neural networks. **towards data science**, v. 6, n. 12, p. 310–316, 2017.

SHI, W.; CAO, J.; ZHANG, Q.; LI, Y.; XU, L. Edge computing: Vision and challenges. **IEEE internet of things journal**, IEEE, v. 3, n. 5, p. 637–646, 2016.

SOE, Y. N.; FENG, Y.; SANTOSA, P. I.; HARTANTO, R.; SAKURAI, K. Implementing lightweight iot-ids on raspberry pi using correlation-based feature selection and its performance evaluation. In: SPRINGER. **International Conference on Advanced Information Networking and Applications**. [S.l.], 2019. p. 458–469.

SOLUTIONS, C. F. C. Unleash the power of the internet of things. **Cisco Systems Inc**, 2015.

SURYATEJA, P. Threats and vulnerabilities of cloud computing: a review. **International Journal of Computer Sciences and Engineering**, v. 6, n. 3, p. 297–302, 2018.

SVOZIL, D.; KVASNICKA, V.; POSPICHAL, J. Introduction to multi-layer feed-forward neural networks. **Chemometrics and intelligent laboratory systems**, Elsevier, v. 39, n. 1, p. 43–62, 1997.

TENSORFLOW. **Instalar o TensorFlow com PIP - Requisitos de Sistema**. 2022. <https://www.tensorflow.org/install/pip#system-requirements>. Accessed: 2021-12-06.

TSESMELIS, M.; DAVID, D. P.; MAILLART, T.; DOLAMIC, L.; TRESOLDI, G.; LACUBE, W.; BARSCHEL, C.; LADETTO, Q.; SCHÄRER, C.; LENDERS, V. et al. Cybersecurity technologies: An overview of trends & activities in switzerland and abroad. **Thomas and Dolamic, Ljiljana and Tresoldi, Giorgio and Lacube, William and Barschel, Colin and Ladetto, Quentin and Schärer, Claudia and Lenders, Vincent and Cuche, Kilian and Mermoud, Alain, Cybersecurity Technologies: An Overview of Trends & Activities in Switzerland and Abroad (January 20, 2022)**, 2022.

TSHARK. **Ferramenta de análise de rede TShark**. 2022. [Online]. <https://www.wireshark.org/>. Accessed: 2022-08-08.

VIVO, M. D.; CARRASCO, E.; ISERN, G.; VIVO, G. O. D. A review of port scanning techniques. **ACM SIGCOMM Computer Communication Review**, ACM New York, NY, USA, v. 29, n. 2, p. 41–48, 1999.

VMWARE. **Cliente VSphere**. 2022. [Online]. <https://www.vmware.com/au/products/vsphere.html>. Accessed: 2022-09-03.

_____. **Hipervisor ESXi**. 2022. [Online]. <https://www.vmware.com/au/products/esxi-and-esx.html>. Accessed: 2022-09-03.

WAN, J.; YANG, J.; WANG, Z.; HUA, Q. Artificial intelligence for cloud-assisted smart factory. **IEEE Access**, IEEE, v. 6, p. 55419–55430, 2018.

WANG, L.; LASZEWSKI, G. V.; YOUNGE, A.; HE, X.; KUNZE, M.; TAO, J.; FU, C. Cloud computing: a perspective study. **New generation computing**, Springer, v. 28, n. 2, p. 137–146, 2010.

WEISER, M. Hot topics-ubiquitous computing. **Computer**, IEEE, v. 26, n. 10, p. 71–72, 1993.

XIAOLIANG, Z.; HONGCAN, Y.; JIAN, W.; SHANGZHUO, W. Research and application of the improved algorithm c4. 5 on decision tree. In: IEEE. **2009 International Conference on Test and Measurement**. [S.l.], 2009. v. 2, p. 184–187.

XU, Z.; LIU, W.; HUANG, J.; YANG, C.; LU, J.; TAN, H. Artificial intelligence for securing iot services in edge computing: a survey. **Security and Communication Networks**, Hindawi, v. 2020, 2020.

YAN, F.; JIAN-WEN, Y.; LIN, C. Computer network security and technology research. In: IEEE. **2015 Seventh International Conference on Measuring Technology and Mechatronics Automation**. [S.l.], 2015. p. 293–296.

YILMAZ, M. H.; ARSLAN, H. A survey: Spoofing attacks in physical layer security. In: IEEE. **2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops)**. [S.l.], 2015. p. 812–817.

YU, W.; LIANG, F.; HE, X.; HATCHER, W. G.; LU, C.; LIN, J.; YANG, X. A survey on the edge computing for the internet of things. **IEEE access**, IEEE, v. 6, p. 6900–6919, 2017.

ZEYU, H.; GEMING, X.; ZHAOHANG, W.; SEN, Y. Survey on edge computing security. In: IEEE. **2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)**. [S.l.], 2020. p. 96–105.

ZHANG, H. Exploring conditions for the optimality of naive bayes. **International Journal of Pattern Recognition and Artificial Intelligence**, World Scientific, v. 19, n. 02, p. 183–198, 2005.