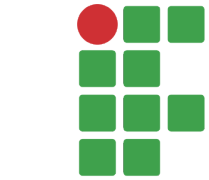


Instituto Federal de Educação, Ciência e Tecnologia da Paraíba

Campus João Pessoa

Programa de Pós-Graduação em Tecnologia da Informação

Nível Mestrado Profissional

DIGENALDO DE BRITO RANGEL NETO

DETECÇÃO DE ATAQUES DDOS NA CAMADA DE APLICAÇÃO: UM ESQUEMA COM APRENDIZADO DE MÁQUINA E BIG DATA

DISSERTAÇÃO DE MESTRADO

JOÃO PESSOA

2024

Digenaldo de Brito Rangel Neto

Detecção de Ataques DDoS na Camada de Aplicação: Um Esquema com Aprendizado de Máquina e Big Data

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Tecnologia da Informação, pelo Programa de Pós-Graduação em Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB.

Orientador: Prof. Dr. Paulo Ditarso Maciel Jr.

João Pessoa

2024

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Nilo Peçanha - *Campus* João Pessoa, PB.

R196d Rangel Neto, Digenaldo de Brito.

Detecção de ataques *DDos* na camada de aplicação : um esquema com aprendizado de máquina e *Big Data* / Digenaldo de Brito Rangel Neto. - 2024.
79 f. : il.

Dissertação (Mestrado em Tecnologia da Informação) – Instituto Federal de Educação da Paraíba / Programa de Pós-Graduação em Tecnologia da Informação (PPGTI), 2024.
Orientação: Prof. Dr. Paulo Ditarso Maciel Jr.

1.Redes de computadores. 2. Cibersegurança. 3. *DDos*. 4. Camada de aplicação. 5. Inteligência artificial. I. Título.

CDU 004.056(043)

Bibliotecária responsável Lucrecia Camilo de Lima – CRB 15/132

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

PROGRAMA DE PÓS-GRADUAÇÃO *STRICTO SENSU*

MESTRADO PROFISSIONAL EM TECNOLOGIA DA INFORMAÇÃO

DIGENALDO DE BRITO RANGEL NETO

DETECÇÃO DE ATAQUES DDOS NA CAMADA DE APLICAÇÃO: UM ESQUEMA COM APRENDIZADO DE MÁQUINA E BIG DATA

Dissertação apresentada como requisito para obtenção do título de Mestre em Tecnologia da Informação, pelo Programa de Pós- Graduação em Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB - Campus João Pessoa.

Aprovado em 19 de dezembro de 2024

Membros da Banca Examinadora:

Dr. Paulo Ditarso Maciel Júnior

IFPB - PPGTI

Dr. Leandro Cavalcanti de Almeida

IFPB - PPGTI

Dr. Anderson Fabiano Batista Ferreira da Costa

IFPB

João Pessoa/2024

Dedico este trabalho aos meus familiares e amigos, com profunda gratidão. Em especial, gostaria de expressar minha eterna gratidão aos meus pais Ronaldo e Josineide, cujo apoio inabalável sempre me proporcionou as condições necessárias para prosseguir com meus estudos. Seu amor, incentivo e sacrifício são a base sólida sobre a qual construí minha jornada acadêmica. À minha esposa, Laísa Diniz, meu porto seguro e fonte de inspiração, cujo apoio incansável e motivação constante foram fundamentais em todas as decisões. Suas palavras de encorajamento e seu apoio foram a luz que guiou meu caminho nos momentos mais desafiadores. Agradeço a Deus e por seus ensinamentos, pois sem Ele, a humanidade é apenas isso, humanidade. Que este trabalho possa ser uma pequena expressão da minha gratidão e do meu compromisso com o conhecimento e o desenvolvimento pessoal e profissional.

AGRADECIMENTOS

Gostaria de expressar minha sincera gratidão a Deus por ter me concedido a fé, cujo poder torna todas as coisas possíveis. Também desejo agradecer a todos que estiveram ao meu lado nesta jornada, fortalecendo-me e motivando-me constantemente. Agradeço imensamente aos meus professores do IFPB pelo conhecimento inestimável que compartilharam comigo. Em especial, gostaria de expressar minha profunda gratidão ao meu orientador, Prof. Dr. Paulo Ditarso, por sua orientação e conhecimento generosos, e por me mostrar os caminhos para me tornar um bom pesquisador. Suas orientações foram essenciais para moldar minhas aspirações acadêmicas e promover meu crescimento.

Além disso, gostaria de agradecer à empresa Jusbrasil por proporcionar conhecimento e disponibilizar tempo para os estudos. Sua contribuição foi fundamental para o meu desenvolvimento profissional e acadêmico. Também expresso minha gratidão aos companheiros de trabalho que, de forma indireta, contribuíram para o meu aprendizado e crescimento ao longo deste período.

RESUMO

A ameaça de ataques DDoS à camada de aplicação está aumentando rapidamente, ressaltando a necessidade de métodos de detecção eficazes para proteger sistemas em rede. Com invasores evoluindo constantemente suas técnicas, os esforços de segurança cibernética estão se tornando mais desafiadores. Em resposta, a utilização de tecnologias avançadas de inteligência artificial surge como um caminho promissor para reforçar as defesas. Este estudo apresenta um esquema de detecção utilizando aprendizado de máquina e avalia seu desempenho utilizando ferramentas de *big data*, como processamento distribuído e armazenamento escalável. Quatro algoritmos de classificação são avaliados quanto à sua precisão e tempo de execução na detecção de ataques DDoS: *Naive Bayes*, *Decision Tree*, *Logistic Regression* e *Random Forest*. As descobertas preliminares sugerem a eficácia do esquema, apresentando altos níveis de precisão e mantendo tempos de resposta razoáveis. Esta abordagem integrada, que combina análise de dados com o uso de plataformas de *big data* para lidar com grandes volumes de tráfego de rede e acelerar o processamento dos modelos de aprendizado de máquina, mostra potencial para fortalecer as defesas contra ameaças cibernéticas num cenário cada vez mais complexo e dinâmico.

Palavras-chaves: Redes de computadores, cibersegurança, DDoS, camada de aplicação.

ABSTRACT

The threat of application-layer DDoS attacks is rapidly increasing, highlighting the need for effective detection methods to protect networked systems. As attackers continuously evolve their techniques, cybersecurity efforts are becoming more challenging. In response, the use of advanced artificial intelligence technologies emerges as a promising approach to strengthen defenses. This study presents a detection scheme using machine learning and evaluates its performance utilizing big data tools, such as distributed processing and scalable storage. Four classification algorithms are assessed in terms of their accuracy and execution time in detecting DDoS attacks: Naive Bayes, Decision Tree, Logistic Regression, and Random Forest. Preliminary findings suggest the effectiveness of the scheme, achieving high accuracy levels while maintaining reasonable response times. This integrated approach, which combines data analysis with the use of big data platforms to handle large volumes of network traffic and accelerate the processing of machine learning models, shows potential to strengthen defenses against cyber threats in an increasingly complex and dynamic landscape.

Key-words: Computer networks, cybersecurity, DDoS, application layer.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação e Definição do Problema

Ataques distribuídos de negação de serviço (DDoS, do inglês *Distributed Denial of Service*) representam um desafio crítico devido a sua capacidade de sobrecarregar servidores em rede com um grande volume de pacotes oriundos de dispositivos comprometidos. De acordo com o (Imperva Blog, 2022), houve um aumento expressivo de 81% nos incidentes de ataques na camada de aplicação entre os segundos semestres de 2021 e 2022. Outro exemplo alarmante, segundo o (Google Cloud Blog, 2023), relata que a empresa enfrentou o maior ataque DDoS registrado em 2023, alcançando um pico acima de 398 milhões de requisições por segundo. Colocando em perspectiva, em apenas dois minutos, o ataque superou o número de visualizações ao Wikipedia em setembro do mesmo ano.

Identificar um ataque DDoS, no entanto, é certamente desafiador. Tais eventos têm a flexibilidade de serem lançados em qualquer nível da pilha de protocolos TCP/IP (PRASEED; THILAGAM, 2018) e diversos tipos de ataques podem ser aplicados em protocolos como ARP, ICMP, TCP, UDP e HTTP. Em especial, quando se trata do nível de aplicação, para executar um ataque DDoS e acessar os serviços dessa camada, é necessário que o usuário estabeleça uma conexão legítima com o servidor web. Para se ter uma ideia, relatos da Cloudfare para o primeiro (Cloudflare Blog, 2022a) e segundo (Cloudflare Blog, 2022b) trimestres de 2022, indicam um aumento anual de 135% e 72% respectivamente, considerando os ataques DDoS na camada de aplicação. Esses dados refletem uma crescente atividade e variação dos ataques nesta camada, destacando a necessidade contínua de vigilância e medidas protetivas contra essas ameaças.

Os atacantes não se restringem mais a equipamentos tecnologicamente avançados, como servidores e computadores pessoais. Ocorreram ataques via dispositivos como refrigeradores e casos de varreduras extensivas realizadas por meio de dispositivos embutidos, como câmeras de vigilância e portas de segurança (TOUTSOP; DAS; KORNEGAY, 2021). Os ataques de DDoS são atualmente disseminados devido à disponibilidade conveniente de um grande número de estações de trabalho comprometidas, agrupadas sob o termo coletivo de "***botnets***" (HACHEM et al., 2011). Há uma tendência crescente em relação à duração, intensidade e diversidade dos ataques envolvendo ***botnets***. Consequentemente, várias pesquisas acadêmicas têm focado em entender as ameaças desse tipo de fonte.

A título de exemplo, a literatura dispõe de trabalhos com os mais variados métodos para a detecção de DDoS. Os autores em (SANMORINO; YAZID, 2013) combinam padrões de fluxo de entrada com um mecanismo de *firewall* em camadas, desenvolvido através de simulações para análise de pacotes. Os trabalhos (QIN; XU; WANG, 2015) e (GIRMA et al., 2015) apresentam uma abordagem baseada em entropia para detecção de DDoS. Por sua vez,

as pesquisas de (CHAUDHARY; SHRIMAL, 2019) e (GONG et al., 2019) destacam o uso de algoritmos genéticos para adaptação dinâmica e resposta eficaz a ataques.

No entanto, a maioria dos trabalhos existentes na literatura não abordam o problema na perspectiva da camada de aplicação ou, quando o fazem, não utilizam informações específicas da rede para ajudar na identificação. Esse padrão pode ser explicado pelo fato de que as soluções tradicionais de detecção de DDoS historicamente se concentram na camada de rede, onde o tráfego volumétrico é mais facilmente identificável e tratável por mecanismos como *firewalls*, *IDS/IPS* e roteamento de mitigação. Além disso, a coleta e processamento de informações na camada de aplicação podem ser mais complexos e computacionalmente custosos, exigindo soluções mais sofisticadas para análise de tráfego sem impacto na disponibilidade dos serviços.

Outro fator relevante é que muitos ataques DDoS em camadas superiores buscam imitar o comportamento legítimo dos usuários, tornando sua detecção mais desafiadora. Como resultado, há uma tendência na literatura em priorizar abordagens baseadas em estatísticas de tráfego e anomalias volumétricas, em vez de uma análise mais detalhada da camada de aplicação. No entanto, com a crescente sofisticação dos ataques, pesquisas recentes vêm explorando novas metodologias para análise de tráfego na camada de aplicação, buscando superar essas limitações.

A identificação eficiente de DDoS na camada de aplicação via ***botnets*** é o primeiro passo para mitigar os efeitos desse ataque. Nesse cenário, a aplicação de técnicas de Inteligência Artificial (IA) desempenha um papel importante, proporcionando abordagens avançadas para detectar e bloquear as ameaças. A análise em tempo real de volumes significativos de dados de tráfego de rede é um exemplo de uso com IA, identificando padrões anômalos característicos em DDoS. Métodos como Redes Neurais Artificiais e Algoritmos de Aprendizado de Máquina têm sido extensivamente explorados para uma detecção precisa desses ataques. Por exemplo, pesquisas como em (AL-JANABI; SAEED, 2011) e (ALKASASSBEH, 2018) evidenciam a eficácia das redes neurais na identificação de anomalias no tráfego, ao passo que estudos conduzidos por (KEBEDE et al., 2022) e (NGUYEN; CHOI, 2010) investigam o uso de algoritmos de aprendizado de máquina para uma detecção proativa de ataques DDoS. Portanto, a IA proporciona uma grande vantagem na identificação dos ataques, possibilitando uma resposta ágil e contribuindo para a segurança de sistemas na web.

Embora a identificação de ataques DDoS, especialmente na camada de aplicação, seja amplamente abordada na literatura, o tratamento eficaz destes ataques representa um desafio devido à crescente sofisticação das ***botnets*** e à similaridade entre o tráfego legítimo e malicioso. Além disso, o rápido crescimento da diversidade dos ataques exige soluções adaptativas para lidar com a escala e a intensidade dessas ameaças.

Este trabalho propõe um esquema para a detecção de ataques DDoS na camada de aplicação utilizando as seguintes técnicas de aprendizado de máquina: *Naive Bayes*, *Decision Tree*, *Logistic Regression* e *Random Forest*. Além disso, apresenta uma análise de desempenho incluindo uma solução de *big data*. Busca-se analisar padrões de comportamento dos ataques na

camada de aplicação, com o objetivo de desenvolver um esquema de detecção eficiente. Nesse contexto, entende-se que esta identificação fornece informações significativas para a mitigação desses ataques, contribuindo para a segurança cibernética de infraestruturas críticas na web. A premissa do esquema proposto consiste em utilizar características do tráfego de redes que outros trabalhos ainda não exploraram, visando identificar padrões e comportamentos que possam melhorar a precisão e a eficácia da detecção de ataques DDoS na camada de aplicação. Alguns dos resultados obtidos nos diversos experimentos realizados mostram que o *Naive Bayes* teve os menores tempos médios de execução, porém com os menores índices de acurácia. Diferentemente, o *Random Forest* apresentou os maiores tempos médios de execução, mas sempre com alta taxa de precisão. Já os algoritmos *Decision Tree* e *Logistic Regression* não apresentaram variações significativas entre os ambientes, com boa acurácia e tempos intermediários de execução.

1.2 Objetivos

1.2.1 Objetivo geral

Este trabalho tem como objetivo propor um esquema para a detecção de ataques DDoS na camada de aplicação, com a aplicação de técnicas de aprendizado de máquina. Para avaliar a eficácia da abordagem proposta, conduzimos uma análise comparativa do desempenho dos algoritmos, utilizando uma solução de *big data* para processamento e validação dos resultados.

1.2.2 Objetivos específicos

- Analisar os padrões de comportamento dos ataques DDoS na camada de aplicação, identificando características distintivas dessas ameaças.
- Fornecer *insights* para a eficiente mitigação de ataques DDoS, visando reforçar a segurança de redes e proteger infraestruturas críticas na web. Os insights incluem a identificação de padrões anômalos na camada de aplicação e a avaliação de impacto de variáveis no desempenho dos classificadores. Estes insights são explorados detalhadamente no Capítulo 5, juntamente com análises comparativas que destacam a eficácia das abordagens propostas.
- Comparar o desempenho de diferentes técnicas de aprendizado de máquina na detecção de ataques DDoS, investigando também o impacto da utilização de uma ferramenta de *big data*.

1.3 Estrutura do Documento

O Capítulo 2 aborda os fundamentos teóricos dos ataques DDoS e das técnicas de detecção; o Capítulo 3 apresenta uma revisão da literatura, identificando lacunas e contextualizando a pesquisa; o Capítulo 4 descreve a metodologia adotada; o Capítulo 5 discute os resultados obtidos; e o Capítulo 6 apresenta as considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma análise detalhada do conceito de ataques distribuídos de negação de serviço (DDoS), fornecendo uma descrição dos princípios fundamentais, teorias relacionadas e aplicações práticas. Serão exploradas as bases teóricas necessárias para uma compreensão sólida do tema em questão, abordando aspectos como a diversidade e flexibilidade dos ataques DDoS, a evolução e complexidade das *botnets*, e as técnicas de inteligência artificial aplicadas na detecção desses ataques.

2.1 Definição e Impacto dos Ataques DDoS

Os ataques distribuídos de negação de serviço (DDoS) representam uma forma de ciberataque que visa sobrecarregar sistemas de rede ao inundá-los com um volume excessivo de tráfego malicioso. Esse tipo de ataque compromete a disponibilidade dos serviços *online* ao impedir que usuários legítimos acessem recursos essenciais, como *websites* e aplicativos (CHAUDHARY; MISHRA, 2023).

O conceito fundamental dos ataques DDoS reside na sua capacidade de coordenar uma grande quantidade de dispositivos, muitas vezes geograficamente distribuídos, para simultaneamente enviar solicitações maliciosas a um alvo específico. Isso pode ser feito através de diversos mecanismos, incluindo *botnets*, redes de dispositivos comprometidos que são controlados remotamente por um atacante.

Como mencionado anteriormente, houve um aumento alarmante na incidência de ataques DDoS nos últimos anos, conforme evidenciado por relatórios recentes. De acordo com (Imperva Blog, 2022), houve um aumento significativo de 81% nos incidentes de ataques na camada de aplicação entre o segundo semestre de 2021 e 2022. Além disso, (Google Cloud Blog, 2023) relatou que em 2023 a empresa enfrentou o maior ataque DDoS registrado, atingindo um pico acima de 398 milhões de requisições por segundo. Relatórios trimestrais da (Cloudflare Blog, 2022a) e (Cloudflare Blog, 2022b) também indicam aumentos anuais de 135% e 72% nos ataques DDoS na camada de aplicação, respectivamente, destacando a crescente atividade e sofisticação dessas ameaças.

Esses dados sublinham a importância crítica de estratégias robustas de defesa cibernética para mitigar os impactos adversos dos ataques DDoS. A compreensão aprofundada desses ataques e o desenvolvimento de medidas proativas são essenciais para proteger infraestruturas críticas e serviços *online* contra interrupções indesejadas.

Nesse capítulo são apresentados os principais conceitos teóricos essenciais para o desenvolvimento do esquema de detecção de ataques DDoS. Utilizamos quatro algoritmos de

aprendizagem de máquina para medir os resultados de desempenho e acurácia: *Naive Bayes*, *Decision Tree*, *Logistic Regression* e *Random Forest*. Primeiramente, é importante compreender a estrutura de um ataque DDoS e, em seguida, estudar as características das bases de dados que serão utilizadas para testar esses algoritmos. Por fim, serão apresentados detalhes de funcionamento dos algoritmos utilizados.

2.2 Diversidade e Flexibilidade dos Ataques DDoS

Os ataques distribuídos de negação de serviço (DDoS) representam uma ameaça persistente e evolutiva no cenário da segurança cibernética, devido à sua capacidade de sobrecarregar sistemas de rede através de diversos protocolos. Esta flexibilidade é evidenciada pela sua capacidade de serem lançados em diferentes camadas da pilha de protocolos TCP/IP (PRASEED; THILAGAM, 2018).

Os protocolos ARP (*Address Resolution Protocol*) e ICMP (*Internet Control Message Protocol*) são frequentemente alvos de ataques DDoS devido à sua importância na resolução de endereços de rede e na comunicação de mensagens de controle, respectivamente. No caso do TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*), os ataques podem explorar vulnerabilidades na forma como esses protocolos gerenciam a comunicação e o transporte de dados. Já os ataques a protocolos na camada de aplicação, como o HTTP (*Hypertext Transfer Protocol*), visam sobrecarregar servidores web através de requisições maliciosas, impactando diretamente a disponibilidade dos serviços *online*.

A adaptação dos ataques DDoS para diferentes níveis da pilha TCP/IP reflete a sua capacidade de se ajustar às condições e defesas específicas de cada ambiente de rede. Por exemplo, ataques na camada de aplicação podem ser mais difíceis de detectar e mitigar devido à sua similaridade com o tráfego legítimo de usuários. A compreensão dessas técnicas de ataque é crucial para o desenvolvimento de estratégias eficazes de defesa cibernética, que visam não apenas detectar e mitigar ataques existentes, mas também antecipar e prevenir futuras ameaças.

Portanto, a diversidade e flexibilidade dos ataques DDoS exigem abordagens de segurança robustas e adaptativas, capazes de monitorar e proteger redes contra uma ampla gama de vetores de ataque em constante evolução.

2.3 Etapas e Motivações de Ataques DDoS

A compreensão detalhada da taxonomia de um ataque de negação de serviço distribuído (DDoS) é fundamental para este estudo. Um ataque DDoS ocorre quando um atacante coordena uma *botnet*, que é uma rede de dispositivos comprometidos (por exemplo, *laptops*, *desktops*, *smartphones* ou servidores). Esses dispositivos são infectados por *malwares* e controlados remotamente pelo atacante, sem o conhecimento dos proprietários legítimos (WANG; MOHAISEN;

CHEN, 2017; SHAFI et al., 2024). A estrutura de um ataque DDoS pode ser dividida em várias etapas, que são descritas a seguir.

2.3.1 Comprometimento de Dispositivos

- **Invasão Inicial** – O atacante compromete um conjunto inicial de dispositivos através de várias técnicas, como *phishing*, exploração de vulnerabilidades ou propagação de *malwares*.
- **Propagação do *Malware*** – Os dispositivos inicialmente comprometidos são usados para disseminar o *malware* a outros dispositivos, expandindo a *botnet*.

2.3.2 Controle da *Botnet*

- **Comunicação** – O atacante usa servidores específicos para enviar instruções aos dispositivos da *botnet*. Esta comunicação pode ser feita através de canais ocultos, como redes sociais ou serviços de mensagens, para evitar detecção.
- **Coordenação** – O atacante coordena a *botnet* para iniciar o ataque de forma sincronizada, maximizando o impacto.

2.3.3 Execução do Ataque

- **Geração de Tráfego Malicioso** – Os dispositivos da *botnet* enviam um grande volume de pacotes maliciosos para o servidor alvo, que podem incluir diversas formas de tráfego, como requisições HTTP, pacotes ICMP ou pacotes SYN, dependendo do tipo de ataque.
- **Sobrecarregamento do Servidor** – O objetivo é inundar o servidor com mais solicitações do que ele pode processar, causando lentidão, interrupção dos serviços ou até a queda completa do servidor.

2.3.4 Motivações dos Ataques

- **Competição Empresarial** – Empresas podem ser alvo de DDoS para prejudicar suas operações e ganhar vantagem competitiva.
- **Motivações Políticas** – Ativistas podem usar um ataque DDoS para protestar contra políticas ou decisões de governos e organizações.
- **Extorsão** – Ataques podem ser realizados para extorquir dinheiro das vítimas, ameaçando prolongar o ataque caso não seja pago um resgate.
- **Guerras Cibernéticas** – Estados-nação podem usar DDoS como parte de estratégias de guerra cibernética para desestabilizar infraestrutura crítica de outros países.

2.4 Evolução e Complexidade das *Botnets*

A evolução das *botnets* representa um desafio contínuo no panorama da segurança cibernética, à medida que dispositivos comuns, anteriormente considerados inofensivos, são explorados para lançar ataques distribuídos de negação de serviço. Um exemplo emblemático é o uso de câmeras de vigilância e eletrodomésticos conectados, que agora são frequentemente incorporados a *botnets*, como destacado por (TOUTSOP; DAS; KORNEGAY, 2021). Esses dispositivos, originalmente projetados para fins domésticos e de monitoramento, são cooptados por invasores para executar operações maliciosas, aproveitando sua vasta distribuição e conectividade à Internet.

A crescente sofisticação e diversificação das *botnets* é um fenômeno documentado há mais de uma década, conforme discutido por (HACHEM et al., 2011). Essas redes de dispositivos comprometidos não apenas aumentaram em escala, mas também se tornaram mais complexas em termos de estratégias de infiltração, comunicação e operação. A sofisticação das *botnets* permite que os atacantes coordenem ataques persistentes, explorando múltiplos vetores de ataque e adaptando-se rapidamente às defesas cibernéticas em evolução.

O papel das *botnets* nos ataques DDoS é multifacetado, contribuindo significativamente para a amplificação e intensidade dessas ameaças. Ao explorar a vasta gama de dispositivos conectados à internet, os atacantes podem ampliar sua capacidade de lançar ataques volumétricos e de aplicação, visando infraestruturas críticas e serviços online. Essa capacidade de escala e diversificação das *botnets* destaca a necessidade urgente de estratégias avançadas de detecção e mitigação de ameaças, capazes de identificar e neutralizar *botnets* antes que causem danos substanciais.

Em resumo, a evolução das *botnets* representa um desafio persistente para a segurança cibernética global, exigindo uma abordagem coordenada e colaborativa entre pesquisadores, profissionais de segurança e legisladores, para mitigar eficazmente os riscos associados a essas redes maliciosas.

2.5 Explorando os Tipos *Slowloris* e *Hulk*

Dois tipos de ataques DDoS destacam-se por suas abordagens singulares, *Slowloris* e *Hulk*. O primeiro utiliza a ocupação estratégica de conexões, explorando vulnerabilidades na camada de aplicação para manter requisições abertas e consumir recursos do servidor, tornando-o inacessível para usuários legítimos. Em contraste, o segundo visa sobrecarregar a infraestrutura com tráfego volumoso e intenso, visando esgotar a largura de banda ou os recursos do servidor. A taxonomia desses ataques identifica técnicas específicas e serve como base para estratégias defensivas. Compreender tais padrões comportamentais permite o desenvolvimento de sistemas de detecção e mitigação mais precisos, capazes de distinguir e responder a variações sutis nos padrões de tráfego. Essa análise contribui para a construção de sistemas de segurança robustos e

resilientes contra ameaças DDoS.

O *Slowloris* aplica uma ocupação estratégica de conexões, onde utiliza pacotes menores para manter várias conexões ativas simultaneamente em um servidor alvo. A peculiaridade desse método está na exploração de uma falha no protocolo HTTP, pois, ao estabelecer conexões parciais e mantê-las abertas, mesmo sem completar a requisição, ele consome os recursos no servidor. Esse processo continua até o limite de conexões do servidor, tornando-o inacessível para novos usuários legítimos (UDDIN; KUMAR; CHAMOLA, 2024). O que torna esse ataque ainda mais desafiador é a sua evolução para variantes mais sofisticadas (SANCHEZ et al., 2021). Tais variantes não apenas mantêm conexões abertas, mas também exploram diferentes camadas do protocolo HTTP, criando conexões mais lentas e consumindo recursos de forma mais sutil. Esse refinamento dificulta sua detecção, tornando a identificação desses ataques um desafio complexo para sistemas de defesa.

O protocolo HTTP pode operar em dois modos principais: conexões persistentes e não persistentes. No modo persistente, várias requisições podem ser enviadas por uma única conexão TCP, reduzindo a latência. No entanto, ataques como Slowloris exploram conexões persistentes para manter as conexões ativas, ocupando recursos do servidor de forma estratégica. Por outro lado, conexões não persistentes encerram a comunicação após cada requisição, tornando esses ataques menos eficazes.

Por outro lado, o ataque *Hulk* opera de maneira distinta, envolvendo a geração de um tráfego massivo por meio de pacotes de tamanho grande. Nesse contexto, pacotes grandes referem-se a requisições HTTP com cargas úteis (*payloads*) significativamente superiores às requisições típicas de navegação web, podendo atingir até 1500 bytes por pacote, que é o limite padrão da MTU (*Maximum Transmission Unit*) na maioria das redes TCP/IP (ASAD et al., 2020). Além disso, o ataque *Hulk* pode gerar uma alta taxa de requisições simultâneas, resultando em uma sobrecarga expressiva de largura de banda e consumo de recursos computacionais do servidor alvo.

A finalidade é sobrecarregar a largura de banda ou os recursos do servidor, congestionando sua capacidade de resposta para lidar com esse tráfego intenso. Este ataque também evoluiu ao longo do tempo, aumentando sua capacidade de gerar tráfego volumoso e intenso. Isso dificulta ainda mais a mitigação do ataque, tornando essencial a implementação de estratégias defensivas mais avançadas e adaptativas para lidar com essa sobrecarga de tráfego (SANCHEZ et al., 2021).

2.6 Base de Dados

O principal foco deste trabalho é desenvolver um esquema de detecção de ataques DDoS utilizando algoritmos de aprendizagem de máquina. Para que esses algoritmos funcionem de maneira eficaz, é essencial que os dados fornecidos sejam de alta qualidade e contenham

informações suficientes e relevantes para a detecção dos ataques.

A seleção e preparação dos dados são etapas cruciais no desenvolvimento de qualquer modelo de aprendizagem de máquina. Dois métodos principais podem ser adotados para isso: a geração de novos dados especificamente para o desenvolvimento do algoritmo, ou a utilização de bases de dados existentes, direcionando o foco para as etapas subsequentes de processamento e análise dos dados.

Neste trabalho, utilizamos bases de dados pré-existentes (AWAN et al., 2021), permitindo concentrar esforços no desenvolvimento, treinamento e avaliação dos algoritmos de detecção. Como mencionado anteriormente, os algoritmos considerados são o *Naive Bayes*, o *Decision Tree*, o *Logistic Regression* e o *Random Forest*. A escolha dos dados apropriados e a sua correta utilização são fundamentais para a criação de um sistema robusto e eficiente de detecção de ataques DDoS.

2.7 Técnicas de Inteligência Artificial na Detecção de DDoS

A utilização de técnicas de Inteligência Artificial (IA) tem se destacado como uma abordagem eficaz para detectar e mitigar ataques DDoS. Entre as técnicas mais promissoras estão as Redes Neurais Artificiais (RNAs), Algoritmos de Aprendizado de Máquina (ML, do inglês *Machine Learning*) e Algoritmos Genéticos, que proporcionam métodos avançados para identificar anomalias no tráfego de rede e responder de forma proativa a esses ataques.

As RNAs, por exemplo, têm sido amplamente exploradas devido à sua capacidade de aprender padrões complexos no tráfego de rede. Estudos como os de (AL-JANABI; SAEED, 2011) e (ALKASASSBEH, 2018) demonstram que as RNAs podem detectar anomalias significativas no comportamento do tráfego, identificando rapidamente padrões característicos de ataques DDoS. Essas redes neurais são treinadas com grandes conjuntos de dados para reconhecer desvios do comportamento normal, permitindo uma resposta ágil e automatizada aos incidentes de segurança.

Além das RNAs, os algoritmos de ML têm sido aplicados com sucesso na detecção de DDoS. Pesquisas como as de (KEBEDE et al., 2022) e (NGUYEN; CHOI, 2010) exploram técnicas de aprendizado supervisionado e não supervisionado para identificar comportamentos anômalos que indicam a presença de um ataque iminente. Esses algoritmos são treinados com base em características específicas do tráfego de rede e podem adaptar seus modelos conforme novos padrões de ataque emergem.

Adicionalmente, os Algoritmos Genéticos têm se destacado pela sua capacidade de adaptação dinâmica e otimização em ambientes adversos. Estudos como os de (CHAUDHARY; SHRIMAL, 2019) e (GONG et al., 2019) exploram como os Algoritmos Genéticos podem ser empregados para ajustar parâmetros de detecção de forma eficiente, maximizando a precisão na identificação de ataques DDoS e minimizando falsos positivos.

Em síntese, as técnicas de IA representam uma abordagem promissora e em constante evolução para mitigar os riscos associados aos ataques DDoS. A integração dessas técnicas em sistemas de segurança cibernética não apenas fortalece a capacidade de detecção precoce e resposta rápida a incidentes, mas também contribui significativamente para a proteção de infraestruturas críticas e serviços *online* contra ameaças cibernéticas cada vez mais sofisticadas. Dentre as várias técnicas possíveis, esta pesquisa tem foco na utilização de algoritmos de ML, que serão descritos em detalhes a seguir.

2.7.1 *Naive Bayes*

O algoritmo *Naive Bayes* é um método de classificação probabilístico baseado no Teorema de Bayes, assumindo independência condicional entre os recursos (preditores), dado o valor da classe. Este método é amplamente utilizado em problemas de classificação de texto, detecção de *spam*, diagnósticos médicos, entre outros, devido à sua simplicidade e eficiência (BERRAR, 2019).

Fundamentos do Naive Bayes

Para compreender o funcionamento do algoritmo, é essencial entender o Teorema de Bayes e a suposição de independência condicional. O teorema é expresso pela seguinte fórmula:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)},$$

onde, $P(A|B)$ é a probabilidade de A ocorrer dado B, $P(B|A)$ é a probabilidade de B ocorrer dado A, $P(A)$ é a probabilidade de A e $P(B)$ é a probabilidade de B. No contexto do *Naive Bayes*, A representa a classe e B representa os atributos.

Aplicações do Naive Bayes

O algoritmo é aplicado em uma variedade de cenários devido à sua simplicidade e eficiência computacional. Algumas das principais aplicações incluem:

- **Classificação de textos** – Utilizado em filtragem de spam, categorização de notícias e análise de sentimentos;
- **Detecção de *spam*** – Efetivo na identificação de emails indesejados com base em padrões de palavras;
- **Diagnósticos médicos** – Empregado na previsão de doenças com base em sintomas e históricos médicos;
- **Reconhecimento de padrões** – Utilizado em reconhecimento de fala, imagens e outros tipos de dados.

Vantagens e Limitações

O *Naive Bayes* apresenta várias vantagens, incluindo sua simplicidade, rapidez no treinamento e eficiência, mesmo com grandes conjuntos de dados. No entanto, suas suposições simplificadoras podem levar a um desempenho inferior em conjuntos de dados complexos, onde existem dependências significativas entre os atributos. Além disso, este algoritmo pode ser sensível a dados irrelevantes ou correlacionados, afetando sua precisão.

2.7.2 *Decision Tree*

A árvore de decisão é um algoritmo de aprendizado de máquina supervisionado amplamente utilizado em problemas de classificação e regressão. Ela representa um modelo de decisão na forma de uma estrutura hierárquica de árvore, onde cada nó interno representa um atributo, cada ramo representa uma decisão baseada nesse atributo, e cada folha representa o resultado final da decisão (LAKSHMINARASIMMAN; RUSWIN; SUNDARAKANTHAM, 2017).

Fundamentos do Decision Tree

O princípio subjacente à árvore de decisão é a divisão recursiva do espaço de atributos em regiões homogêneas, utilizando critérios baseados em testes de atributos. O algoritmo seleciona a divisão que maximiza a pureza dos grupos resultantes em cada nó, utilizando métricas como ganho de informação e índice Gini.

O ganho de informação é baseado na entropia, que mede a incerteza ou impureza de um conjunto de dados. A entropia $H(D)$ de um conjunto de dados D é dada por:

$$H(D) = -\sum_{i=1}^{c} p_i \log_2 p_i,$$

onde c é o número de classes e p_i é a proporção de elementos da classe i em D. O ganho de informação $IG(D,A)$ ao particionar o conjunto de dados D com base no atributo A é dado por:

$$IG(D,A) = H(D) - \sum_{v \in values(A)} \frac{|D_v|}{|D|} H(D_v),$$

onde $values(A)$ é o conjunto de valores distintos do atributo A, D_v é o subconjunto de D para o qual o atributo A tem valor v, e $|D|$ representa o número de elementos em D.

O índice Gini mede a impureza de um nó e é definido como:

$$Gini(D) = 1 - \sum_{i=1}^{c} p_i^2.$$

Construção do Decision Tree

A construção de uma árvore de decisão envolve quatro passos críticos:

1. **Seleção de Atributos** – Escolha do atributo que melhor separa os dados com base em uma métrica de divisão, como ganho de informação ou índice Gini;
2. **Divisão dos Dados** – Particionamento dos dados em subconjuntos de acordo com os valores dos atributos selecionados;
3. **Critérios de Parada** – Definição de condições para interromper a divisão recursiva, como a profundidade máxima da árvore, o número mínimo de amostras em um nó ou a pureza dos nós;
4. **Poda** – Remoção de ramos desnecessários ou redundantes para prevenir *overfitting* e melhorar a generalização do modelo.

Aplicações do Decision Tree

As árvores de decisão são aplicadas em diversos domínios, incluindo:

- **Classificação de Dados Estruturados** – Utilizada em sistemas de recomendação, análise de crédito e segmentação de mercado;
- **Diagnóstico Médico** – Empregada na previsão de doenças com base em sintomas e históricos médicos;
- **Modelagem Preditiva** – Utilizada para prever tendências futuras em séries temporais e análise de riscos.

Vantagens e Limitações

As principais vantagens da árvore de decisão incluem:

- **Interpretabilidade** – O modelo resultante é fácil de entender e visualizar, facilitando a interpretação e explicação das decisões;
- **Capacidade de Lidar com Dados Não Lineares** – Eficiente na modelagem de relações complexas entre atributos;
- **Robustez a *Outliers*** – Menos sensível a valores extremos comparado a outros modelos.

No entanto, árvores de decisão também apresentam limitações:

- **Propensão ao *Overfitting*** – Podem ajustar-se excessivamente aos dados de treinamento se não forem adequadamente podadas ou se o conjunto de dados não for representativo;
- **Instabilidade** – Pequenas variações nos dados podem resultar em árvores muito diferentes, afetando a consistência do modelo.

Para mitigar essas limitações, técnicas como poda pós-processamento, *bagging* (p.ex. *Random Forest*) e *boosting* (p.ex. *Gradient Boosting*) são frequentemente utilizadas.

2.7.3 *Logistic Regression*

Algoritmo de aprendizado de máquina amplamente utilizado para problemas de classificação binária. Diferente da regressão linear, que prevê valores contínuos, a regressão logística estima a probabilidade de um evento ocorrer, utilizando uma função logística para modelar a relação entre os atributos (variáveis independentes) e a probabilidade do evento de interesse (variável dependente) (NAGARAJA; BOREGOWDA; VANGIPURAM, 2021).

Fundamentos do Logistic Regression

O princípio fundamental da regressão logística é a transformação da regressão linear através de uma função logística (*sigmoid*), que mapeia a saída para o intervalo [0, 1]. A função logística é definida como:

$$P(y=1|\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}-b}},$$

onde $\mathbf{w}$ representa os pesos dos atributos, $\mathbf{x}$ é o vetor de atributos e b é o viés (*bias*). A saída da função logística representa a probabilidade do evento de interesse ocorrer, permitindo a classificação binária com um limiar predefinido.

O modelo é ajustado através da maximização da função de verossimilhança, que para uma amostra de dados é dada por:

$$L(\mathbf{w},b) = \prod_{i=1}^{n} P(y_i|\mathbf{x}_i)^{y_i} \cdot (1-P(y_i|\mathbf{x}_i))^{1-y_i}.$$

A função de custo associada, que é minimizada para encontrar os parâmetros ótimos, é a entropia cruzada:

$$J(\mathbf{w},b) = -\frac{1}{n}\sum_{i=1}^{n} \left[y_i \log P(y_i|\mathbf{x}_i) + (1-y_i)\log(1-P(y_i|\mathbf{x}_i)) \right].$$

Aplicações do Logistic Regression

A regressão logística é aplicada em diversos domínios devido à sua simplicidade e interpretabilidade, incluindo:

- **Classificação de *Spam*** – Determinação de emails como *spam* ou não *spam*;
- **Diagnósticos Médicos** – Previsão da presença de doenças com base em sintomas e históricos médicos;
- **Detecção de Fraudes** – Identificação de transações fraudulentas em sistemas financeiros;
- **Modelagem de Probabilidades em Classificação Multiclasse** – Extensão para classificação multiclasse usando técnicas como regressão logística multinomial.

Vantagens e Limitações

Entre as vantagens da regressão logística estão as seguintes características:

- **Interpretação Fácil** – Os coeficientes do modelo são interpretáveis e indicam a importância relativa dos atributos;
- **Eficiência Computacional** – Relativamente rápida de treinar e implementar;
- **Robustez a *Overfitting*** – Utilizando regularização, pode-se controlar o *overfitting* do modelo.

No entanto, a regressão logística apresenta algumas limitações:

- **Limitação a Relações Lineares** – Supõe uma relação linear entre os atributos e o logit da variável dependente;
- **Desempenho em Problemas Não Lineares** – Pode não capturar relações complexas em dados não lineares, onde outros modelos, como Redes Neurais, podem ser mais adequados.

2.7.4 *Random Forest*

Algoritmo de aprendizado de máquina robusto e amplamente utilizado, que combina múltiplas árvores de decisão para realizar tarefas de classificação e regressão. Esse método de *ensemble* constrói um conjunto de árvores de decisão durante o treinamento e agrega suas previsões para melhorar a precisão e a generalização do modelo (CHEN et al., 2020).

Fundamentos do Random Forest

O princípio fundamental do algoritmo é a construção de múltiplas árvores de decisão independentes durante o treinamento. Cada árvore é treinada em uma amostra aleatória do conjunto de dados, conhecida como *bootstrap sample*. Além disso, em cada nó de decisão, uma amostra aleatória de atributos é considerada para divisão. Essa introdução de aleatoriedade nas amostras e na seleção de atributos promove diversidade entre as árvores, reduzindo o risco de *overfitting* e aumentando a robustez do modelo.

A predição final é obtida através de um processo de agregação, onde cada árvore individual contribui com uma votação para a classe ou valor previsto, e a classe com mais votos é selecionada como a previsão final. No caso de problemas de regressão, a média das predições das árvores é utilizada.

Construção do Random Forest

A construção de um *Random Forest* envolve a definição de dois parâmetros principais.

- **Número de Árvores** (n_trees): Determina quantas árvores de decisão serão construídas no modelo. Um maior número de árvores geralmente leva a um melhor desempenho, mas aumenta o custo computacional.
- **Número de Atributos Considerados em Cada Divisão de Nó** (m_try): Em cada nó, um subconjunto aleatório de atributos é considerado para encontrar a melhor divisão. Isso contribui para a diversidade entre as árvores e melhora a capacidade de generalização do modelo.

Aplicações do Random Forest

Esse algoritmo é aplicado em vários domínios devido à sua versatilidade e eficácia. Algumas das aplicações incluem:

- **Classificação de Dados Estruturados** – Utilizado em problemas onde os dados são organizados em tabelas, como em bases de dados relacionais;
- **Previsão de Séries Temporais** – Aplicado na modelagem e previsão de dados sequenciais ao longo do tempo;
- **Análise e Mineração de Texto** – Utilizado para categorizar documentos e extrair informações relevantes de grandes volumes de texto;
- **Detecção de Anomalias** – Identificação de padrões anômalos ou eventos raros em conjuntos de dados;

- **Seleção de Características** – Avaliação da importância das variáveis preditoras para refinar modelos e melhorar a interpretabilidade.

Vantagens e Limitações

O *Random Forest* apresenta diversas vantagens, tais como:

- **Capacidade de Lidar com Grandes Conjuntos de Dados** – Eficiente em cenários com grandes volumes de dados e muitas variáveis;
- **Robustez Contra *Overfitting*** – A aleatoriedade introduzida no treinamento ajuda a evitar o ajuste excessivo aos dados de treino;
- **Interpretabilidade Relativa** – Permite a avaliação da importância das características, facilitando a compreensão do modelo.

Uma fórmula importante relacionada ao *Random Forest* é a da importância das características, que pode ser calculada com base na redução da impureza em cada nó da árvore. A importância de uma característica j é calculada somando a redução ponderada da impureza para todos os nós t onde j é usada para a divisão, em todas as árvores do conjunto:

$$\text{Importance}(j) = \sum_{t:\text{node } t \text{ uses } feature\ j} \frac{N_t}{N} \cdot \Delta i(t),$$

onde:

- N_t é o número de amostras no nó t,
- N é o número total de amostras,
- $\Delta i(t)$ é a redução de impureza no nó t.

No entanto, o *Random Forest* também tem algumas limitações:

- **Desempenho em Conjuntos de Dados Desbalanceados** – Pode ser menos eficaz quando as classes estão desproporcionalmente representadas;
- **Muitas Características Correlacionadas** – A presença de muitos atributos correlacionados pode reduzir a diversidade entre as árvores e afetar o desempenho do modelo.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta uma revisão da literatura com foco nas principais abordagens, técnicas e ferramentas utilizadas na detecção, mitigação e prevenção de ataques DDoS. Esse levantamento permite identificar lacunas na literatura e posicionar a pesquisa no estado da arte, fundamentando a proposta desenvolvida no trabalho.

3.1 Definição do Escopo

Para uma compreensão da literatura relacionada, é importante entender a taxonomia e os efeitos dos ataques DDoS, suas características distintas e estratégias de mitigação (UDDIN; KUMAR; CHAMOLA, 2024). Além disso, é importante explorar o uso da Inteligência Artificial (IA) na identificação e prevenção de ameaças à segurança, permitindo a detecção de padrões sutis e anômalos no tráfego de rede, resultando em uma detecção mais rápida e precisa de atividades maliciosas (CHALE et al., 2023; KUMAR et al., 2023).

Os trabalhos existentes fornecem uma base sólida para essas investigações. Por exemplo, (ASAD et al., 2020) explora o uso de técnicas de aprendizado profundo para a detecção de ataques DDoS, destacando como essas técnicas podem aprimorar a identificação de ameaças complexas. Em (SANCHEZ et al., 2021), os autores propõem técnicas de extração de características para melhorar a precisão da detecção de ataques, oferecendo percepções sobre como otimizar modelos de detecção. Além disso, (HADI et al., 2022) discute o desenvolvimento de sistemas de defesa cibernética baseados em IA para proteger contra ameaças emergentes, como os ataques DDoS na camada de aplicação, enfatizando a importância de estratégias de defesa adaptáveis e robustas.

Diante desse desafio, é essencial permanecer à frente das ameaças, adaptando-se aos novos padrões de ataque e implementando estratégias de segurança ágeis e adaptáveis, aproveitando as capacidades preditivas e analíticas oferecidas pela IA. O estudo de (CHAVAN et al., 2022) destaca a persistência dos ataques DDoS como uma das ameaças mais significativas no contexto da segurança cibernética e das redes. Este trabalho utiliza aprendizado de máquina para a identificação de DDoS e a prevenção de ***botnets***, adotando um conjunto específico de dados (ZAIB, 2019) como base para o treinamento e teste dos modelos. As técnicas aplicadas incluem Aprendizado por Reforço, Máquinas de Vetor de Suporte, K Vizinhos Mais Próximos e Árvores de Decisão, resultando em taxas de precisão de aproximadamente 90%, 90%, 89% e 82%, respectivamente. No entanto, o estudo não avaliou o desempenho dos modelos em ambientes de produção ou cenários reais, uma lacuna importante que merece ser abordada em análises futuras.

Ao sintetizar essas informações, o capítulo busca fornecer uma visão geral das abordagens atuais e identificar áreas onde melhorias podem ser implementadas, especialmente no contexto de detecção e mitigação de ataques DDoS utilizando técnicas avançadas de IA.

3.2 Revisão de Literatura

O trabalho em (PRASEED; THILAGAM, 2021) aborda uma ameaça crítica enfrentada por servidores web, que são os ataques assimétricos de DDoS na camada de aplicação, capazes de derrubar sistemas com poucas conexões. Tais ataques, semelhantes ao tráfego legítimo, são notoriamente difíceis de detectar devido à sua similaridade, consumindo a banda limitada de rede. Os métodos atuais, que se baseiam em representações indiretas e técnicas complexas, apresentam altas taxas de falsos positivos e um tempo prolongado de detecção, inviabilizando seu uso em tempo real. Diante desse cenário, há uma clara necessidade de mecanismos de detecção simples, eficientes e adaptáveis. Os autores propõem um modelo baseado em Autômato Probabilístico Temporal e um mecanismo de pontuação para diferenciar comportamentos legítimos e maliciosos. Essa abordagem permite uma detecção extremamente rápida, com baixa taxa de falsos positivos, e capacidade de aprendizado incremental em tempo de execução, tornando-a adaptável e reduzindo ainda mais os falsos positivos. Contudo, o trabalho foca na identificação de comportamentos de usuários legítimos e maliciosos, sem considerar a camada de aplicação.

Nosso trabalho difere significativamente ao propor um esquema para a detecção de ataques DDoS na camada de aplicação utilizando técnicas de aprendizado de máquina: *Naive Bayes*, *Decision Tree*, *Logistic Regression* e *Random Forest*. Além disso, apresentamos uma análise de desempenho, incluindo uma solução de *big data*, para examinar padrões de comportamento dos ataques na camada de aplicação.

A abordagem visa desenvolver um esquema de detecção eficiente, explorando características do tráfego de rede ainda não suficientemente investigadas por outros trabalhos, com foco na adaptação dinâmica e eficiente na detecção de ataques DDoS. Trabalhos anteriores, como (PRASEED; THILAGAM, 2021), concentram-se na detecção baseada em aprendizado de máquina, mas sem realizar uma análise comparativa detalhada do impacto dos diferentes classificadores em termos de tempo de execução e acurácia. Embora algumas pesquisas na literatura já tenham explorado a comparação de algoritmos de classificação, nosso estudo se diferencia ao aplicar essa avaliação no contexto de serviços na camada de aplicação, considerando cenários de tráfego realista e incorporando técnicas de *big data* para escalabilidade.

Em resumo, enquanto (PRASEED; THILAGAM, 2021) foca na detecção de comportamentos legítimos e maliciosos utilizando Autômatos Probabilísticos, nosso trabalho se concentra na aplicação de aprendizado de máquina para detecção de ataques DDoS na camada de aplicação, comparando diferentes algoritmos para identificar o mais adequado com base em precisão e eficiência temporal.

Uma vez que a detecção na camada de aplicação se torna uma tarefa desafiadora devido à sua natureza mutável, os autores em (KUMAR; SHARMA et al., 2018) buscam compreender as particularidades desses ataques através do modelo denominado *Network Security against DDoS Attacks* (NSDA). Tal modelo visa criar novas variáveis, como a diferença temporal entre

solicitações consecutivas por endereço IP, indicando similaridades e discrepâncias no tamanho (em bytes) dos arquivos de log, oferecendo uma detecção mais eficaz desses ataques na camada de aplicação. O emprego de pré-processamento em Java e a ferramenta de aprendizado de máquina Weka 3.8 possibilitam essa abordagem, enquanto métodos de reamostragem convertem conjuntos de dados principais em treinamento, validação cruzada e testes. A utilização do classificador *Naive Bayes* no Weka 3.8 permite a análise e detecção dos ataques DDoS. O trabalho prioriza a identificação de padrões, sem, no entanto, explorar uma abordagem de prevenção proativa. A abordagem proposta nesta pesquisa busca antecipar e mitigar potenciais riscos, antes que eles se concretizem.

Em comparação com o trabalho mencionado, o presente estudo amplia a análise ao avaliar diferentes algoritmos de aprendizado de máquina em termos de tempo de execução e acurácia. Essa avaliação é essencial para selecionar o algoritmo mais adequado para diferentes ambientes, considerando tanto a precisão quanto a eficiência temporal.

Além disso, este trabalho investiga o impacto da utilização de uma solução de *big data* na análise de desempenho, examinando padrões de comportamento dos ataques na camada de aplicação, algo não explorado em (KUMAR; SHARMA et al., 2018). Para validar essa abordagem, os testes foram realizados tanto com quanto sem a solução de *big data*, permitindo avaliar os ganhos em escalabilidade e eficiência no processamento do tráfego.

Ao utilizar características do tráfego de rede que ainda não foram amplamente exploradas, este estudo visa desenvolver um esquema de detecção mais preciso e eficaz. Especificamente, analisamos métricas como *Flow_IAT_Max*, que captura o maior intervalo entre pacotes consecutivos dentro de um fluxo, *SYN_Flag_Count*, que indica o número de pacotes com sinalizador SYN, e *Flow_Bytes_Sec*, que mede a taxa de transferência de dados por segundo. A combinação dessas características possibilita a detecção de ataques que exploram tanto padrões de rajadas de tráfego quanto persistência em conexões, fornecendo informações significativas para a mitigação desses ataques e contribuindo para a segurança cibernética de infraestruturas críticas na web.

Em (YADAV; SELVAKUMAR, 2015), os autores destacam o impacto disruptivo nos serviços dos servidores web e, consequentemente, nos usuários legítimos, devido aos crescentes ataques DDoS. Propõe-se um novo método fundamentado em características de navegação e Aprendizado por Reforço, com o intuito de modelar o comportamento do usuário para distinguir entre usuários regulares e possíveis atacantes. Segundo os autores, a urgência de abordagens mais eficazes na detecção de ataques DDoS na camada de aplicação enfatiza a importância de métodos mais precisos e adaptáveis para salvaguardar a integridade dos serviços web. Entretanto, o trabalho concentra-se na identificação dos comportamentos de usuários legítimos, sem explorar a aplicação de algoritmos de aprendizado de máquina específicos na detecção de comportamentos anômalos na camada de aplicação.

Ao contrário do trabalho de (YADAV; SELVAKUMAR, 2015), que se concentra na distinção entre usuários legítimos e atacantes com base em características de navegação e Apren-

dizado por Reforço, este estudo amplia a abordagem ao avaliar detalhadamente o desempenho de diferentes algoritmos de aprendizado de máquina na detecção de ataques DDoS na camada de aplicação. A análise comparativa desses algoritmos considera tanto o tempo de execução quanto a precisão, permitindo a escolha do modelo mais eficaz para diferentes cenários.

Adicionalmente, a aplicação de técnicas de *big data* para examinar padrões de comportamento de ataques na camada de aplicação representa uma novidade que não foi abordada em (YADAV; SELVAKUMAR, 2015).

No trabalho de (RAHAL; SANTOS; NOGUEIRA, 2020), é apresentada uma abordagem inovadora para a previsão e detecção de ataques DDoS causados por *botnets*. O estudo propõe uma arquitetura hierárquica que opera em dois níveis: local e internet. No nível local, sinais precoces de um ataque DDoS são identificados utilizando indicadores estatísticos, enquanto no nível da internet, um método híbrido de detecção de bots é aplicado. Este método combina técnicas de *clustering* e processamento de sinais em grafos para analisar o tráfego de rede e identificar dispositivos que se comportam como bots.

O trabalho se destaca pela utilização de indicadores avançados, como taxa de retorno, autocorrelação, coeficiente de variação e assimetria, para prever tendências de ataques DDoS antes que eles alcancem estágios avançados. Além disso, a arquitetura proposta realiza uma análise detalhada do tráfego de rede, selecionando características que melhor descrevem o comportamento dos dispositivos, e emprega uma técnica de aprendizado de máquina não supervisionado para agrupar dispositivos com comportamentos semelhantes, separando os bots de nós benignos.

Os resultados do estudo, baseados em avaliações com os conjuntos de dados CTU-13, CAIDA e Botnet 2014, demonstram a eficácia da abordagem em prever ataques DDoS e identificar bots com alta precisão. O sistema proposto não só melhora a detecção precoce de ataques, mas também proporciona uma análise robusta e detalhada do tráfego de rede, facilitando a mitigação de ataques antes que causem danos significativos.

Enquanto o trabalho de (RAHAL; SANTOS; NOGUEIRA, 2020) utiliza uma abordagem hierárquica e indicadores estatísticos avançados para a previsão e detecção de ataques DDoS, o trabalho proposto neste estudo adota um enfoque diferente. Este estudo propõe um esquema para a detecção de ataques DDoS na camada de aplicação utilizando técnicas de aprendizado de máquina.

A principal diferença reside na abordagem metodológica e nas técnicas empregadas. O estudo citado foca na utilização de indicadores estatísticos e técnicas de *clustering*, combinadas com processamento de sinais em grafos. Em contraste, o presente trabalho enfatiza a avaliação do desempenho de algoritmos de aprendizado de máquina, considerando métricas como tempo de execução e acurácia.

Dessa forma, enquanto o estudo citado explora padrões estatísticos e relações estruturais no tráfego de rede, este trabalho busca compreender a eficácia de diferentes modelos preditivos

para detecção de ataques DDoS, oferecendo uma análise quantitativa comparativa que pode complementar abordagens baseadas em *clustering*.

O sistema proposto por (RAHAL; SANTOS; NOGUEIRA, 2020) busca uma detecção precoce e a mitigação de ataques antes que causem danos significativos. Por outro lado, o estudo em questão destaca a importância da adaptação dinâmica e da utilização de características do tráfego de redes para identificar padrões e comportamentos específicos de ataques DDoS, visando uma detecção mais precisa e eficiente.

O trabalho de (RAJ; KANG, 2022) aborda o crescimento exponencial dos ataques de *phishing*, que têm afetado diversos setores da sociedade. Dentre esses ataques, destaca-se o ataque DDoS, em que um servidor é sobrecarregado com um grande volume de solicitações simultâneas, tornando-o indisponível. Os motivos por trás desses ataques podem variar, incluindo exigências de resgate, vingança, competição ou outras motivações.

Neste trabalho, os autores discutem detalhadamente o ataque DDoS e as técnicas de detecção associadas. É proposto um modelo de aprendizado de máquina para a detecção de ataques DDoS em redes definidas por software (SDN). O modelo proposto compara a acurácia, precisão, *recall* e *F1-score* de vários algoritmos de aprendizado de máquina, incluindo Regressão Logística, SVM, XGBoost, Árvore de Decisão e KNN. Com base na acurácia prevista para esses algoritmos, é elaborado um relatório de análise comparativa detalhado.

A pesquisa destaca a eficácia das diferentes técnicas de detecção e oferece uma análise crítica sobre como cada algoritmo se comporta em termos de desempenho. O estudo não só contribui para uma melhor compreensão das técnicas de detecção de DDoS, mas também oferece insights valiosos para aprimorar as estratégias de mitigação de ataques em ambientes de SDNs.

Enquanto o trabalho mencionado acima se concentra na avaliação de diferentes algoritmos de classificação para detectar ataques DDoS e DoS, utilizando métricas de acurácia, o trabalho proposto neste estudo adota uma abordagem distinta.

O presente trabalho aborda a detecção de ataques DDoS na camada de aplicação, explorando o uso de aprendizado de máquina para identificar padrões maliciosos no tráfego de rede. A fim de avaliar a eficácia das técnicas empregadas, realizamos uma análise comparativa dos algoritmos, considerando métricas como tempo de execução e acurácia. Além disso, incorporamos uma solução de *big data* para aprimorar a escalabilidade e a eficiência na detecção de padrões de ataque.

O trabalho proposto também explora diferentes características do tráfego de rede para identificar padrões e comportamentos específicos que podem melhorar a precisão e a eficácia da detecção de ataques DDoS. Enquanto o estudo mencionado avalia principalmente o desempenho do XGBoost em comparação com outros algoritmos, o trabalho proposto oferece uma análise mais abrangente dos algoritmos de aprendizado de máquina e como eles se comportam em ambientes diferentes.

3.3 Comparação de Resultados

A Tabela 1 a seguir oferece uma comparação dos resultados obtidos em diversos estudos sobre a detecção de ataques DDoS. Cada trabalho é avaliado com base em três aspectos principais: o foco da pesquisa, os algoritmos utilizados e os principais resultados alcançados. A análise abrange desde abordagens tradicionais, como o uso de autômatos probabilísticos, até técnicas avançadas que incorporam algoritmos de aprendizado de máquina, *clustering* e processamento de sinais.

Com relação ao foco do trabalho, busca-se o objetivo principal e a abordagem adotada para a detecção de ataques DDoS, incluindo o contexto e a metodologia. Já em relação aos algoritmos utilizados, analisa-se os algoritmos e ferramentas específicas aplicadas para a detecção e análise dos ataques, fornecendo uma visão sobre a diversidade e inovação nas técnicas empregadas. Por fim, os principais achados de cada estudo são destacados, como a precisão dos algoritmos, a eficácia na redução de falsos positivos e a capacidade de adaptação a novas ameaças. Esta comparação visa proporcionar uma visão das estratégias e técnicas atuais para enfrentar o problema dos ataques DDoS, facilitando a identificação de tendências, lacunas e oportunidades para futuros desenvolvimentos na área.

Tabela 1 – Comparação dos Trabalhos sobre Detecção de Ataques DDoS.

Referência	**Foco do Trabalho**	**Algoritmos Utilizados**	**Resultados Principais**	***Features* Utilizadas**
(PRASEED; THILAGAM, 2021)	Detecção de ataques DDoS na camada de aplicação com Autômatos Probabilísticos e pontuação.	Autômato Probabilístico Temporal	Alta rapidez na detecção, baixa taxa de falsos positivos, capacidade de aprendizado incremental.	Probabilidade de transição, tempo de reflexão, carga de trabalho da requisição, sequência de requisições, pontuação de suspeita.
(KUMAR; SHARMA et al., 2018)	Detecção de ataques DDoS na camada de aplicação com variáveis novas e aprendizado de máquina.	*Naive Bayes*, Weka 3.8 (não especificado o algoritmo)	Foco na criação de variáveis e pré-processamento. Identificação de padrões, mas sem abordagem preventiva.	Diferença entre tempos consecutivos de requisições (DT), variação no tamanho dos bytes das requisições (Bpt), número de requisições por IP, tamanho da resposta em bytes.
(YADAV; SELVAKUMAR, 2015)	Modelagem do comportamento do usuário para detectar ataques DDoS na camada de aplicação usando regressão logística.	*Logistic Regression*, PCA para seleção de features	Alta taxa de detecção (98,6%) e baixa taxa de falso positivo (1,4%). Método mais eficiente que abordagens anteriores.	Frequência de requisições HTTP, número de URLs únicas acessadas, quantidade de respostas HTTP 200, 302 e 404, duração da sessão, taxa média de requisições por tempo, percentual de requisições únicas e percentual de respostas bem-sucedidas.
(RAHAL; SANTOS; NOGUEIRA, 2020)	Previsão e detecção de ataques DDoS causados por *botnets* com arquitetura hierárquica.	Técnicas de *clustering*, processamento de sinais em grafos	Utiliza indicadores avançados para previsão de ataques. Análise detalhada do tráfego com alta precisão na identificação de bots.	Retorno de pacotes, autocorrelação, coeficiente de variação, skewness, número de portas de origem e destino, percentual de protocolos usados, centralidade dos nós na rede.
(RAJ; KANG, 2022)	Detecção de ataques DDoS em redes definidas por software com comparação de algoritmos.	*Logistic Regression*, SVM, XGBoost, *Decision Tree*, KNN	Avaliação de precisão, *recall*, *F1-score* e acurácia dos algoritmos. Detalhada análise comparativa com foco em redes definidas por software.	Número de bytes transmitidos e recebidos por porta de switch, tempo entre fluxos, taxa de pacotes, tipo de protocolo, encoding de features categóricas.
Este Trabalho	Detecção de ataques DDoS na camada de aplicação, através de um esquema com aprendizado de máquina e *big data*.	*Naive Bayes*, *Decision Tree*, *Logistic Regression*, *Random Forest*	Comparação detalhada de algoritmos em termos de tempo de execução e acurácia. Uso de solução de *big data* para análise de padrões.	As características distintas dos demais trabalhos são abordadas na Seção 4.2.2.

4 DESCRIÇÃO DA PROPOSTA

Este capítulo oferece uma visão detalhada da metodologia adotada neste estudo. Inicialmente apresenta os procedimentos metodológicos utilizados na condução da pesquisa, incluindo detalhes sobre a coleta de dados, instrumentos de pesquisa, técnicas de análise e abordagens metodológicas específicas. Além disso, serão discutidas as justificativas por trás das escolhas metodológicas e como elas contribuem para alcançar os objetivos estabelecidos neste trabalho.

4.1 Aplicabilidade

A proposta apresentada estabelece um esquema para um sistema de detecção de ataques distribuídos de negação de serviço na camada de aplicação, empregando técnicas de aprendizado de máquina: *Naive Bayes*, *Decision Tree*, *Logistic Regression* e *Random Forest*. A abrangência dessa proposta é significativa, abordando vários aspectos da segurança cibernética e da proteção de infraestruturas críticas na web. A seguir, destacamos algumas das áreas de aplicação desse arcabouço.

1. **Proteção de Serviços Web:** com o aumento da frequência de ataques DDoS direcionados à camada de aplicação, é fundamental contar com mecanismos eficazes de detecção e resposta. A utilização de algoritmos de ML para identificação de padrões de comportamento anômalos pode fortalecer a segurança dos serviços web, assegurando uma disponibilidade mais consistente e confiável.

2. **Defesa contra *Botnets*:** os ataques DDoS frequentemente se valem de dispositivos comprometidos controlados remotamente por um invasor. A proposta de utilizar ML pode auxiliar na identificação e mitigação das atividades maliciosas dessas *botnets*, pois os algoritmos de aprendizado de máquina são capazes de identificar padrões anômalos no tráfego de rede que indicam atividades coordenadas de *botnets*. Além disso, esses modelos podem se adaptar continuamente a novas variantes de ataque, permitindo uma detecção mais precisa e uma mitigação eficaz, fortalecendo a resiliência das redes contra ameaças em constante evolução.

3. **Análise de Tráfego de Rede:** a análise de tráfego de rede é crucial para identificar padrões suspeitos que possam indicar a ocorrência de um ataque DDoS em curso. Ao empregar técnicas de ML, é possível automatizar essa análise e fornecer alertas em tempo real sobre possíveis ataques, permitindo uma resposta ágil e eficaz por parte dos administradores de rede.

4. **Melhoria da Resposta a Incidentes:** uma vez detectado um ataque DDoS, é essencial contar com procedimentos claros e eficientes para responder e mitigar seu impacto. A

utilização de técnicas de ML pode contribuir para priorizar e coordenar as ações de resposta a incidentes, reduzindo o tempo de inatividade e os danos causados pelo ataque.

5. **Pesquisa e Desenvolvimento Contínuos:** a segurança cibernética, enquanto área multidisciplinar, está em constante evolução, sendo continuamente desafiada por novas técnicas de ataque. No contexto deste trabalho, o uso de ML para identificação de ataques DDoS na camada de aplicação insere-se no subcampo de defesa contra ameaças cibernéticas na camada de aplicação e detecção de tráfego anômalo. Essa abordagem proporciona *insights* específicos, como a identificação de padrões anômalos no tráfego de rede, a avaliação de variáveis que mais impactam a precisão dos classificadores e a comparação de algoritmos quanto ao tempo de resposta e eficácia. Esses resultados podem orientar tanto o desenvolvimento de sistemas mais robustos quanto estratégias para mitigar ataques de maneira mais eficiente.

4.2 Metodologia

Esta seção apresenta a metodologia utilizada para a coleta, organização e interpretação do tráfego de rede. Cada etapa do processo é apresentada, desde a definição dos campos a serem registrados até a aplicação das ferramentas para analisar e extrair informações relevantes dos fluxos de dados. Em resumo, o processo segue as seguintes etapas:

1. **Identificação e Classificação de Tipos de Ataques** – Foi realizada uma pesquisa exploratória sobre os tipos de ataques DDoS mais atuais. Com base nessa pesquisa, os diferentes tipos de ataques foram identificados e classificados, incluindo inundações, exaustão de recursos e ataques de aplicação. Dentre os ataques analisados, destacamos o *Slowloris* e o *Hulk*, pois representam abordagens distintas dentro do espectro de ataques DDoS na camada de aplicação. O *Slowloris* ilustra ataques baseados na exaustão de conexões persistentes, enquanto o *Hulk* representa ataques de inundação de requisições HTTP. Essa escolha permite cobrir dois vetores principais de ataque na camada de aplicação, tornando a análise relevante para um cenário abrangente. Embora existam outros tipos de ataques, a seleção desses dois casos se justifica pela sua popularidade e impacto documentado na literatura, além da necessidade de representar diferentes estratégias utilizadas por invasores.

2. **Análise de Dados e Pesquisa Exploratória** – Nesta etapa, foi realizada uma pesquisa exploratória detalhada com base em um *dataset* previamente adquirido e estudado. A seleção dos campos utilizados foi motivada por estudos prévios na literatura, em especial pelo trabalho de (AWAN et al., 2021), que demonstrou a relevância de características da camada de rede na detecção eficaz de ataques DDoS. Com base nessa fundamentação, foram escolhidos campos como endereços IP, portas de destino e *timestamps*, entre outros,

que desempenham um papel essencial na identificação de padrões de tráfego anômalo. A escolha desses atributos garantiu a inclusão de todas as informações relevantes para facilitar a posterior detecção de anomalias e ataques DDoS. Além disso, a análise demonstrou que esses campos foram particularmente eficazes na identificação dos ataques Slowloris e Hulk, permitindo distinguir comportamentos maliciosos de tráfego legítimo e contribuindo significativamente para a robustez do modelo proposto e sua execução bem-sucedida.

3. **Análise Exploratória sobre Detecção de Ataques DDoS com ML** – Nesta etapa, foi realizada uma pesquisa exploratória detalhada sobre métodos de detecção de ataques DDoS utilizando algoritmos de *machine learning*. Foram investigados diferentes algoritmos e técnicas para identificar quais informações dos fluxos de dados são mais relevantes para a detecção eficaz de ataques.

4. **Elaboração do Esquema de Detecção e Análise de Desempenho** – Nesta fase da metodologia, foi desenvolvido um esquema para um sistema de detecção de ataques DDoS. Este esquema foi testado e avaliado em dois ambientes de teste distintos, utilizando a execução de algoritmos de ML. A análise de desempenho considerou a eficácia do sistema em detectar ataques, bem como sua capacidade de operar em diferentes cenários de teste.

5. **Geração e Análise dos Resultados de Classificação** – A última etapa envolve a geração dos resultados de classificação a partir dos algoritmos de aprendizado de máquina previamente treinados e aplicados aos dados analisados. Nesta fase, foram processados os dados do dataset em diferentes algoritmos de machine learning, permitindo a comparação dos tempos de execução e da acurácia em ambientes distintos.

O *dataset* (AWAN et al., 2021) utilizado nas análises contempla um grande volume de dados sobre acessos, requisições e respostas na rede. Tais informações são essenciais para a compreensão do comportamento dos usuários e das operações envolvidas em ataques DDoS.

4.2.1 Conjunto de Dados

Os dados do *dataset* foram coletados na camada de aplicação e oferecem um extenso registro de atividades de tráfego de rede ao longo de um dia específico. Esses dados compreendem aproximadamente 0,9 milhão de registros, contendo 77 características e uma coluna designada como alvo. Para a análise, os rótulos foram atribuídos a três classes: (1a) Benigno - Legítimo, (2a) DDoS - Slowloris e (3a) DDoS - Hulk. Embora os dados sejam oriundos da camada de aplicação, o dataset inclui informações sobre os endereços IP de origem, permitindo a análise de padrões de tráfego por IP. Essa informação foi utilizada para verificar o comportamento dos ataques, como a distribuição dos IPs em cada classe e a presença de IPs reincidentes nos ataques.

Os registros de acesso à rede armazenados em logs contêm informações detalhadas, incluindo endereços IP dos clientes (camada de rede) e dados da camada de aplicação, como

métodos de comunicação, recursos acessados e códigos de resposta. A análise desses dados foi crucial para identificar padrões de tráfego distintos associados aos ataques Slowloris e Hulk. A partir da extração dessas informações, foi possível detectar estratégias específicas, como a ocupação de conexões com pequenos pacotes no ataque Slowloris, enquanto o ataque Hulk sobrecarregou a rede com pacotes de maior tamanho e intensidade.

Essa distinção foi fundamental para o desenvolvimento de um algoritmo de inteligência artificial baseado em aprendizado supervisionado, utilizando técnicas como *Random Forest* e *Logistic Regression* para classificar os ataques DDoS. O modelo foi treinado com um conjunto de dados balanceado, contendo registros de tráfego legítimo e malicioso, sendo avaliadas características como *Flow_IAT_Max*, *SYN_Flag_Count* e *Flow_Bytes_Sec*, que se mostraram relevantes na diferenciação dos ataques. Para otimizar a acurácia do modelo, foram aplicados ajustes de hiperparâmetros e validação cruzada, garantindo a robustez da classificação.

Os resultados indicam que o modelo foi capaz de discernir e diferenciar esses tipos específicos de ataques DDoS com alta precisão, aprimorando significativamente as estratégias de detecção e prevenção.

4.2.2 Extração de Características dos Dados

A extração de características dos dados foi realizada para identificar padrões e informações relevantes na análise do tráfego de rede. Para garantir uma investigação detalhada, foi essencial compreender os parâmetros e métricas envolvidas nesse ambiente dinâmico. A Tabela 2 apresenta os principais campos do conjunto de dados utilizado, destacando aqueles mais relevantes para a detecção de anomalias e ataques DDoS (ZHOU et al., 2022).

Campo(s)	**Significado(s)**
`Destination_Port`	Porta de destino usada na comunicação
`Flow_Duration`	Duração total do fluxo em microssegundos
`Total_Fwd_Packets`	Número total de pacotes encaminhados pelo fluxo
`Total_Backward_Packets`	Número total de pacotes recebidos pelo fluxo
`Total_Length_of_Fwd_Packets`	Comprimento total dos pacotes encaminhados
`Total_Length_of_Bwd_Packets`	Comprimento total dos pacotes recebidos
`Fwd_Packet_Length_*`	Estatísticas do comprimento dos pacotes encaminhados: Max, Min, Mean ou Std
`Bwd_Packet_Length_*`	Estatísticas do comprimento dos pacotes recebidos: Max, Min, Mean ou Std
`Flow_Bytes_Sec`	Taxa de bytes por segundo no fluxo
`Flow_Packets_Sec`	Taxa de pacotes por segundo no fluxo
`Flow_IAT_*`	Estatísticas do tempo interarrival do fluxo: Mean, Std, Max ou Min
`Fwd_IAT_*`	Estatísticas do tempo interarrival dos pacotes encaminhados: Total, Mean, Std, Max ou Min
`Bwd_IAT_*`	Estatísticas do tempo interarrival dos pacotes recebidos: Total, Mean, Std, Max ou Min
`Flag_Counts_*`	Contagem de flags TCP presentes nos pacotes: FIN, SYN, RST, PSH, ACK, URG, CWE ou ECE
`Header_Lengths_*`	Comprimento dos cabeçalhos encaminhados (Fwd) ou recebidos (Bwd)
`Packet_Length_*`	Estatísticas do comprimento dos pacotes no fluxo: Min, Max, Mean, Std ou Variance
`Average_Packet_Size`	Tamanho médio dos pacotes
`Avg_*_Segment_Size`	Tamanho médio dos segmentos encaminhados (Fwd) ou recebidos (Bwd)
`Active_*`	Estatísticas dos períodos ativos do fluxo: Mean, Std, Max ou Min
`Idle_*`	Estatísticas dos períodos ociosos do fluxo: Mean, Std, Max ou Min
`Label`	Rótulo indicando se o fluxo é benigno ou um ataque DDoS específico (e.g., DoS Hulk, DoS Slowloris)

Tabela 2 – Campos no conjunto de dados de tráfego de rede.

O *dataset* analisado contém um total de 77 campos distintos, abrangendo diversas métricas de tráfego. No entanto, para fins de clareza e objetividade, a Tabela 2 exibe um

subconjunto desses campos, selecionados por sua importância na análise exploratória dos dados e na construção dos modelos de detecção. Cada campo listado é cuidadosamente definido para proporcionar uma compreensão clara de seu papel na identificação de padrões de tráfego e possíveis ameaças.

- `Destination_Port`: Este campo indica a porta de destino usada na comunicação. A análise das portas de destino é essencial para identificar possíveis vulnerabilidades e padrões de comunicação incomuns que podem indicar atividades maliciosas, como tentativas de exploração de serviços específicos.
- `Flow_Duration`: Representa a duração total do fluxo em microssegundos. Esta métrica é crucial para entender a persistência das conexões, com ataques DDoS geralmente exibindo fluxos de longa duração em comparação ao tráfego benigno, que tende a ter durações mais variadas.
- `Total_Fwd_Packets` e `Total_Backward_Packets`: Indicam, respectivamente, o número total de pacotes encaminhados e recebidos pelo fluxo. Estas contagens ajudam a avaliar a intensidade do tráfego em ambas as direções, um fator importante na identificação de desequilíbrios típicos de certos tipos de ataques.
- `Total_Length_of_Fwd_Packets` e `Total_Length_of_Bwd_Packets`: Correspondem ao comprimento total dos pacotes encaminhados e recebidos. Essas métricas fornecem uma visão sobre a quantidade de dados transferidos, sendo fundamentais para detectar anomalias no volume de tráfego.
- `Fwd_Packet_Length_*` e `Bwd_Packet_Length_*`: Estes campos incluem estatísticas (Max, Min, Mean, Std) sobre os comprimentos dos pacotes encaminhados e recebidos. A análise dessas estatísticas permite identificar variações e distribuições anômalas nos tamanhos dos pacotes, que podem indicar a presença de tráfego malicioso.
- `Flow_Bytes_Sec` e `Flow_Packets_Sec`: Medem a taxa de bytes e pacotes por segundo no fluxo. Estas métricas são críticas para avaliar a intensidade do tráfego ao longo do tempo, especialmente útil na detecção de ataques DDoS que tentam sobrecarregar os recursos da rede.
- `Flow_IAT_*`: Estatísticas do tempo do tempo entre chegadas (*interarrival time*) do fluxo (Mean, Std, Max, Min), que é o intervalo entre a chegada de pacotes sucessivos. Analisar essas estatísticas ajuda a identificar padrões de temporização irregulares, comuns em ataques DDoS.
- `Fwd_IAT_*` e `Bwd_IAT_*`: Estatísticas do tempo entre chegadas dos pacotes encaminhados e recebidos (Total, Mean, Std, Max, Min). Essas métricas fornecem indícios sobre a

periodicidade e variação nos tempos de chegada dos pacotes em ambas as direções do fluxo.

- `Flag_Counts_*`: Contagem de flags TCP (FIN, SYN, RST, PSH, ACK, URG, CWE, ECE) presentes nos pacotes. Flags como SYN e ACK são especialmente relevantes na identificação de ataques baseados em protocolos de transporte, como o *SYN Flood*.
- `Header_Lengths_*`: Comprimento dos cabeçalhos encaminhados (Fwd) e recebidos (Bwd). O comprimento do cabeçalho pode fornecer informações adicionais sobre a estrutura dos pacotes e a complexidade das sessões de comunicação.
- `Packet_Length_*`: Estatísticas do comprimento dos pacotes no fluxo (Min, Max, Mean, Std, Variance). Esses campos são úteis para entender a variabilidade nos tamanhos dos pacotes e identificar padrões incomuns que podem estar associados a ataques.
- `Average_Packet_Size`: Tamanho médio dos pacotes. Esta métrica oferece uma visão geral sobre o tamanho típico dos pacotes no fluxo, auxiliando na detecção de desvios significativos.
- `Avg_*_Segment_Size`: Tamanho médio dos segmentos encaminhados (Fwd) ou recebidos (Bwd). Analisar o tamanho dos segmentos pode ajudar a identificar anomalias na fragmentação dos pacotes.
- `Active_*` e `Idle_*`: Estatísticas dos períodos ativos e ociosos do fluxo (Mean, Std, Max, Min). Essas métricas fornecem informações sobre os padrões de atividade e inatividade, importantes para diferenciar entre tráfego legítimo e malicioso.
- `Label`: Rótulo indicando se o fluxo é benigno ou um ataque DDoS específico (e.g., DoS Hulk, DoS Slowloris). Este campo é fundamental para a tarefa de classificação supervisionada, fornecendo a base para a construção e avaliação de modelos de detecção de anomalias.

A análise exploratória dos dados envolvendo estes campos foi essencial para compreender a distribuição, a variabilidade e os padrões presentes no tráfego de rede. Esta compreensão permite identificar características-chave que diferenciam tráfego benigno de tráfego malicioso, facilitando o desenvolvimento de modelos de detecção de alta precisão para ataques DDoS.

4.2.3 Determinação da Frequência das Coletas de Dados

A análise da frequência das coletas de dados foi essencial para compreender o comportamento temporal do tráfego de rede e a detecção de anomalias. Como a documentação do dataset não especifica a frequência de coleta, aplicamos um cálculo baseado na métrica *Flow_IAT_Max* para inferir essa informação. Essa métrica representa o maior intervalo de tempo entre pacotes consecutivos dentro de um fluxo, permitindo estimar a periodicidade das amostras registradas.

Frequência de coleta refere-se ao intervalo de tempo entre sucessivas amostras de tráfego de rede registradas no dataset. Como a documentação original do dataset não especifica essa informação, a frequência de coleta foi inferida a partir da análise dos tempos de chegada dos fluxos, utilizando a métrica *Flow_IAT_Max*, que representa o maior intervalo entre pacotes consecutivos dentro de um fluxo.

Os dados incluem diversos atributos que descrevem o fluxo de pacotes na rede, tais como *Flow_Duration*, *Total_Fwd_Packets*, *Total_Backward_Packets*, entre outros. Para determinar a frequência das coletas, focamos especificamente nos atributos *Flow_IAT_Max*, *Flow_IAT_Min* e *Flow_IAT_Mean*, que representam o intervalo de chegada dos pacotes.

A frequência de coleta foi deduzida através da seguinte metodologia:

4.2.3.1 Cálculo dos Intervalos de Tempo entre as Coletas

- ***Flow_IAT_Max***: Representa o maior intervalo de tempo entre a chegada de dois pacotes sucessivos.
- ***Flow_IAT_Min***: Representa o menor intervalo de tempo entre a chegada de dois pacotes sucessivos.
- ***Flow_IAT_Mean***: Representa a média dos intervalos de tempo entre a chegada dos pacotes.

4.2.3.2 Análise Estatística

A análise estatística dos valores de *Flow_IAT* forneceu a variabilidade e a distribuição dos tempos de chegada dos pacotes. Observamos que os valores de *Flow_IAT_Max* e *Flow_IAT_Min* apresentaram grandes variações, indicando uma natureza não uniforme na chegada dos pacotes.

Para verificar a adequação ao modelo de um processo de Poisson, analisamos a distribuição dos tempos de chegada. Caso o tráfego seguisse um processo de Poisson, os tempos entre chegadas de pacotes deveriam seguir uma distribuição exponencial. No entanto, a análise estatística revelou variações significativas e um comportamento não estacionário, sugerindo a presença de padrões de rajadas (*burstiness*) ou mudanças na taxa de chegada ao longo do tempo. Isso indica que o tráfego analisado pode não se ajustar ao modelo de Poisson tradicional, possivelmente devido a variações na origem dos fluxos e na natureza dos ataques detectados.

4.2.3.3 Dedução da Frequência

- Com base nos valores observados, inferimos que as coletas não são realizadas em intervalos de tempo fixos, mas sim baseadas na atividade da rede. Isso é evidenciado pelas grandes variações nos valores de *Flow_IAT*.

- Para fluxos de ataques DoS, os valores de *Flow_IAT* são significativamente maiores quando comparados ao tráfego benigno. No caso do ataque *DoS Hulk*, os valores de *Flow_IAT_Max* chegam a 79.000.000 microsegundos, enquanto *Flow_IAT_Min* é de apenas 2 microsegundos, evidenciando picos de alta intensidade seguidos por períodos de baixa atividade. Já no ataque *Slowloris*, *Flow_IAT_Max* atinge 1.000.000 microsegundos, enquanto *Flow_IAT_Min* é de 50.000 microsegundos, refletindo a estratégia de manter conexões abertas por longos períodos. Em contraste, fluxos benignos apresentam valores mais uniformemente distribuídos, indicando um padrão de tráfego mais constante e regular.

4.2.3.4 Exemplos de Análise

- **DoS Hulk**: Para os fluxos rotulados como *DoS Hulk*, os valores de *Flow_IAT_Max* chegam a 79.000.000 microsegundos, enquanto os valores de *Flow_IAT_Min* são de apenas 2 microsegundos. Isso sugere que os ataques são realizados com picos de alta intensidade seguidos por períodos de baixa atividade.
- **Slowloris**: Para os fluxos rotulados como *Slowloris*, os valores de *Flow_IAT_Max* chegam a 1.000.000 microsegundos, enquanto os valores de *Flow_IAT_Min* são de 50.000 microsegundos. Isso indica uma técnica de ataque que visa manter as conexões abertas o maior tempo possível, utilizando intervalos regulares entre pacotes para evitar a detecção.
- **Tráfego Benigno**: Para fluxos benignos, os valores de *Flow_IAT* são mais uniformemente distribuídos, refletindo uma atividade de rede mais constante e regular.

Em resumo, a frequência das coletas de dados varia significativamente dependendo do tipo de tráfego, com ataques DoS mostrando grandes variações nos intervalos de chegada dos pacotes em comparação ao tráfego benigno.

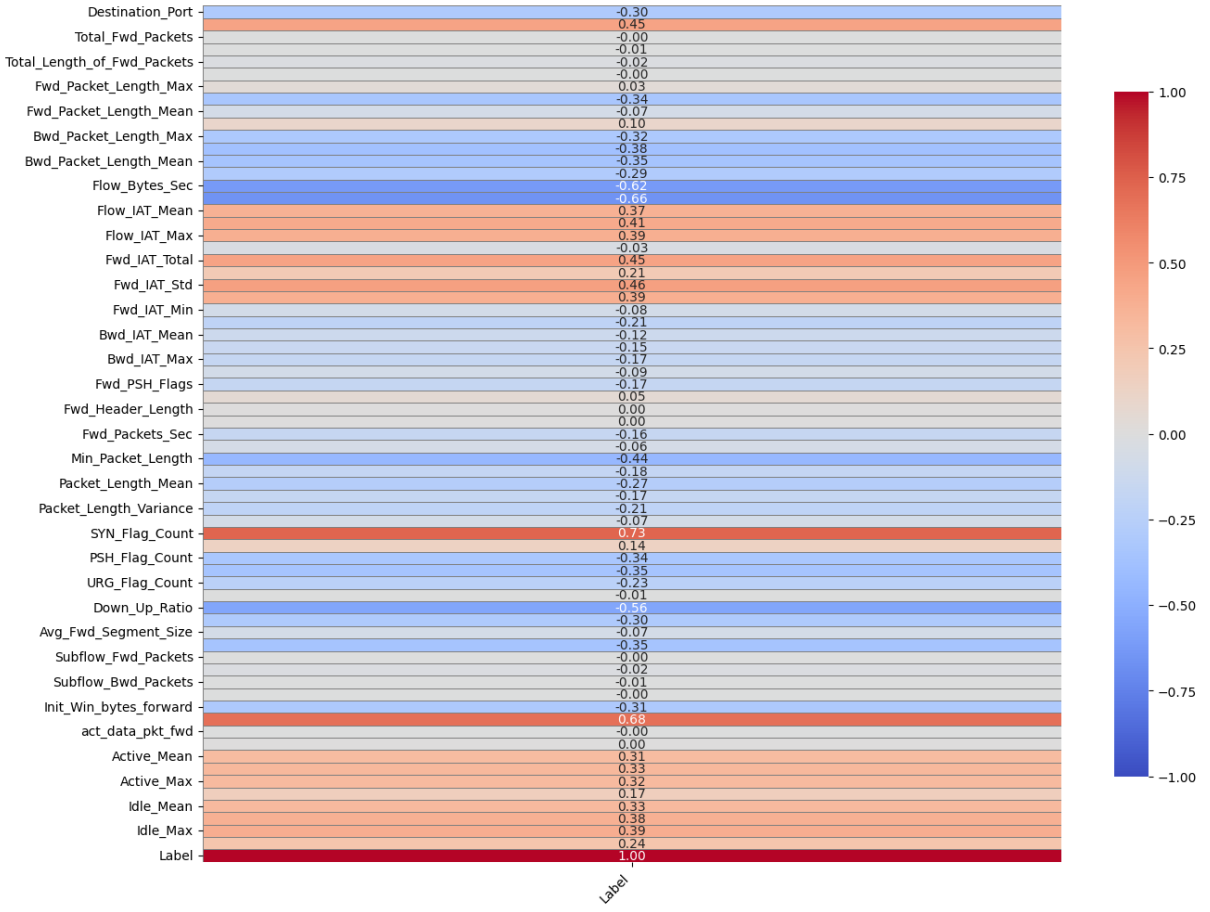
4.2.4 Correlação das Variáveis

A correlação quantifica a relação linear entre duas variáveis, oferecendo *insights* sobre como essas variáveis estão associadas entre si (SARRAF et al., 2020). No contexto deste *dataset*, a interpretação das correlações pode fornecer uma compreensão sobre como as variáveis independentes influenciam a variável de interesse (Label). Na figura 1, apresentamos uma abordagem para interpretar os valores de correlação obtidos.

4.2.4.1 Correlação Positiva e Negativa

- **Correlação Positiva:** Valores próximos de +1 indicam que as variáveis tendem a aumentar ou diminuir juntas. P. ex., uma correlação de 0.733 entre `SYN_Flag_Count` e a `Label` sugere que um aumento no `SYN_Flag_Count` está associado a um aumento na `Label`. Isso indica uma forte associação positiva.

Figura 1 – Matriz de Correlação das *Features* do *Dataset*.



- **Correlação Negativa:** Valores próximos de -1 indicam que, enquanto uma variável aumenta, a outra tende a diminuir. P. ex., uma correlação de -0.621 entre `Flow_Bytes_Sec` e a `Label` sugere que, conforme `Flow_Bytes_Sec` aumenta, a `Label` tende a diminuir, indicando uma forte associação negativa.

4.2.4.2 Correlação Nula

Valores próximos de 0 indicam pouca ou nenhuma relação linear significativa entre as variáveis. Por exemplo, uma correlação de 0.002749 entre `Fwd_Header_Length` e a `Label` sugere que há uma relação linear insignificante entre essas variáveis.

4.2.4.3 Correlação Moderada a Alta

Correlações entre +0.5 e +1, ou -0.5 e -1, são consideradas moderadas a altas. Por exemplo, a correlação de 0.679896 entre `Init_Win_bytes_backward` e a `Label` indica uma associação positiva forte, sugerindo que essa variável tem uma relação significativa com a `Label`.

4.2.4.4 Variáveis Fortemente Correlacionadas com a Label

- `SYN_Flag_Count` (0.733122): Alta correlação positiva, sugerindo que o número de pacotes com a flag SYN é um indicador robusto da `Label`.
- `Init_Win_bytes_backward` (0.679896): Alta correlação positiva, indicando uma relação significativa com a `Label`.

4.2.4.5 Variáveis Negativamente Correlacionadas com a Label

- `Flow_Bytes_Sec` (-0.621945): Correlação negativa alta, sugerindo que o aumento no `Flow_Bytes_Sec` está associado a uma diminuição na `Label`.
- `Flow_Packets_Sec` (-0.662877): Correlação negativa alta, similar ao `Flow_Bytes_Sec`, indicando que o aumento no `Flow_Packets_Sec` está associado a uma diminuição na `Label`.

4.2.4.6 Variáveis Com Correlação Fraca ou Nula

- `Fwd_Header_Length` (0.002749): Correlação muito próxima de zero, indicando uma relação linear insignificante com a `Label`.
- `act_data_pkt_fwd` (-0.002609): Correlação praticamente nula, sugerindo uma relação linear quase inexistente com a `Label`.

A análise das correlações entre as variáveis do *dataset* e a variável de interesse (Label) fornece uma visão detalhada das relações lineares e suas implicações. As correlações positivas e negativas destacam como diferentes variáveis se relacionam com a Label, ajudando a identificar quais características são mais influentes para a modelagem preditiva.

As features *Flow_IAT* foram destacadas anteriormente devido à sua relevância na caracterização dos ataques analisados, como *DoS Hulk* e *Slowloris*. A análise dos tempos entre pacotes permitiu identificar comportamentos específicos desses ataques, como rajadas de tráfego intenso ou conexões prolongadas. Esses padrões temporais foram fundamentais para diferenciar tráfego benigno de tráfego malicioso.

No entanto, ao aprofundar a análise estatística do *dataset*, observamos que outras variáveis, como `SYN_Flag_Count` e `Init_Win_bytes_backward`, também apresentam correlações significativas com a variável alvo (*Label*). Essas features ajudam a compreender o comportamento de ataques que exploram a manipulação de pacotes SYN ou a forma como os pacotes são recebidos e processados pelo servidor.

Enquanto as variáveis de *Flow_IAT* foram cruciais para detectar padrões temporais de tráfego, `SYN_Flag_Count` e `Init_Win_bytes_backward` destacam-se na modelagem preditiva,

contribuindo para uma melhor performance na classificação de ataques. Dessa forma, ambas as abordagens — análise temporal (*IAT*) e análise estrutural dos pacotes — são complementares para a detecção eficaz de ataques DDoS no dataset.

Variáveis com correlações próximas de zero, como por exemplo `Fwd_Header_Length` e `act_data_pkt_fwd`, mostram pouca ou nenhuma relação linear com a Label. Portanto, essas variáveis podem não impactar significativamente a variável de interesse e podem ser consideradas menos relevantes para o modelo preditivo.

4.3 Esquema para Previsão de Detecção de Ataques DDoS

Nesta seção, apresentamos a estrutura do esquema proposto para identificação de ataques por meio de técnicas de aprendizado de máquina. O processo é dividido em quatro fases principais, conforme ilustrado na Figura 2. Cada fase corresponde a uma etapa fundamental para garantir a eficácia do modelo na detecção de ataques DDoS, desde a aquisição dos dados até a avaliação do desempenho dos algoritmos.

A primeira fase, denominada **"Obtenção dos Dados"**, envolve atividades para garantir a qualidade e confiabilidade dos dados utilizados na análise. Mais especificamente, verificamos se o arquivo recebido é adequado para a análise, incluindo a verificação de integridade dos dados, o tratamento de possíveis erros/inconsistências e a confirmação de que o formato do arquivo está correto.

Na segunda fase, denominada **"Processamento dos Dados"**, realizamos a organização e padronização dos dados como parte do processo. O objetivo é validar a consistência dos dados para uma classificação mais precisa das informações.

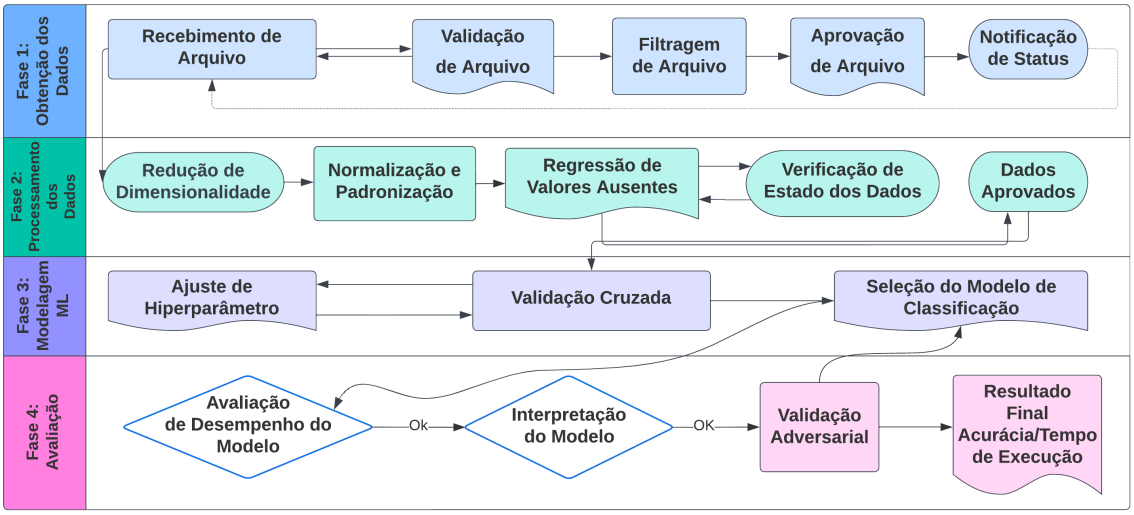
A terceira fase, denominada **"Modelagem ML"**, aplica os algoritmos de aprendizado de máquina aos dados processados, treinando o modelo para identificar padrões de comportamento anômalos e classificar os diferentes tipos de ataques DDoS.

Por fim, na quarta fase, denominada **"Avaliação"**, realizamos a interpretação dos resultados obtidos pelo algoritmo, analisando o desempenho e a eficácia do esquema na identificação de ataques DDoS.

Vale ressaltar que a execução completa de todas as etapas pode variar dependendo do experimento realizado. Em alguns testes, determinadas fases podem ser simplificadas ou omitidas com base em fatores como a disponibilidade dos dados, a necessidade de ajustes metodológicos ou a viabilidade computacional. Por exemplo, em experimentos preliminares, a fase de **Modelagem ML** pode ser limitada a um subconjunto de algoritmos para avaliação inicial, enquanto em testes finais todas as fases são plenamente executadas para validar o desempenho do esquema completo. Essas variações são definidas de acordo com os objetivos específicos de cada experimento, garantindo que a análise seja conduzida de maneira eficiente e alinhada com

as necessidades da pesquisa.

Figura 2 – Fases do esquema proposto para detecção de ataques DDoS.



4.3.1 Obtenção dos Dados (1ª Fase)

Esta primeira etapa concentra-se na obtenção e no pré-processamento inicial dos dados. Durante essa fase, são realizadas atividades para garantir que os dados utilizados atendam a critérios de qualidade e confiabilidade, assegurando sua adequação para a análise posterior.

Especificamente, verificamos a integridade dos dados por meio da consistência dos registros e da ausência de valores corrompidos ou inválidos. Além disso, aplicamos verificações de qualidade baseadas em critérios como completude das informações, conformidade com os formatos esperados e ausência de duplicações que possam comprometer a análise. O formato do arquivo utilizado é `CSV`, estruturado com colunas que representam variáveis extraídas do tráfego de rede, incluindo *timestamps*, endereços IP e indicadores de protocolo.

Por fim, essa fase também inclui um pré-processamento inicial, que envolve a padronização dos dados, remoção de inconsistências e conversão de formatos quando necessário, garantindo que os dados estejam adequados para as próximas etapas do estudo. As tarefas desta fase estão ilustradas na Figura 2 e são descritas a seguir.

Recebimento de Arquivo: A etapa inicial do processo envolve receber o arquivo contendo os dados de tráfego de rede e garantir sua adequação para análise. Isso é feito por meio de verificações para assegurar que o arquivo não esteja corrompido e que todos os dados necessários estejam presentes, utilizando, por exemplo, algoritmos de *checksum* para detectar qualquer corrupção nos dados.

Validação de Arquivo: Após o recebimento, o arquivo passa por um processo de validação para garantir que atenda aos critérios de qualidade estabelecidos. Isso inclui verificações adicionais de integridade dos dados, como a comparação de funções hash (*e.g.*, SHA-256) para assegurar

que não houve alterações indevidas durante a transferência. Além disso, técnicas de assinatura digital podem ser utilizadas para garantir a autenticidade do arquivo, confirmando sua origem e evitando repúdio. Também verificamos se o arquivo contém todas as informações necessárias para a análise de ataques DDoS, como registros de endereços IP fonte e destino, *timestamps* e tipos de tráfego.

Filtragem de Arquivo: Na terceira etapa, o arquivo é filtrado para remover dados irrelevantes ou redundantes, assegurando que apenas informações essenciais sejam incluídas na análise.

Os critérios para definir dados irrelevantes incluem colunas que não contribuem para a detecção de ataques, como identificadores não informativos, dados fora do período de análise e atributos com variabilidade extremamente baixa. Além disso, registros incompletos, nos quais informações essenciais estão ausentes, também são descartados para evitar distorções no treinamento dos modelos de aprendizado de máquina.

Para lidar com essa filtragem, o esquema utiliza um algoritmo que avalia a relevância de cada atributo por meio de técnicas de análise estatística e seleção de *features*. Isso inclui a remoção de colunas com alto percentual de valores ausentes, baixa correlação com a variável alvo e redundância entre atributos. Esse processo reduz o ruído e melhora a eficácia do modelo de aprendizado de máquina, garantindo que a análise se concentre nas informações mais relevantes e precisas.

Aprovação de Arquivo: Após a filtragem, o arquivo é submetido a uma revisão final para garantir que as etapas anteriores tenham sido concluídas com sucesso. Para isso, um algoritmo de validação baseado em regras é utilizado para verificar se todos os critérios estabelecidos foram atendidos.

O algoritmo executa uma série de verificações automatizadas, como a detecção de colunas ausentes, análise de completude dos registros e conferência da conformidade dos dados com o formato esperado. Além disso, ele verifica se o arquivo contém apenas informações relevantes, como logs de tráfego de rede sem redundâncias ou inconsistências.

Se o arquivo atender a todos os critérios, ele é aprovado para análise adicional. Caso contrário, medidas corretivas são tomadas automaticamente ou por meio de intervenção manual, dependendo da gravidade do problema identificado. Essas correções incluem a remoção de dados irrelevantes, a normalização de valores discrepantes ou a sinalização de registros que requerem ajustes manuais.

Notificação de Status: Finalmente, uma notificação de status é gerada para informar o resultado do processo de preparação dos dados. Um serviço de mensageria, como o *Apache Kafka* ou *RabbitMQ*, é responsável por realizar essa notificação, disparando mensagens de forma assíncrona para os componentes interessados no processamento dos dados.

Essas notificações incluem informações sobre a aprovação ou rejeição do arquivo, bem como quaisquer ações corretivas aplicadas. A comunicação ocorre por meio de um sistema de

filas, garantindo a entrega confiável das mensagens e permitindo que os processos subsequentes reajam dinamicamente ao status do arquivo.

4.3.2 Processamento dos Dados (2ª Fase)

A segunda fase se concentra na organização e padronização dos dados. O objetivo é estabelecer consistência nos dados, proporcionando uma classificação mais precisa das informações. As seguintes tarefas são realizadas nesta fase.

Redução de Dimensionalidade: Os dados são identificados para remover características redundantes ou irrelevantes, diminuindo a complexidade do conjunto, que possui várias métricas de tráfego de rede, como pacotes por segundo, tipo de protocolo e endereços IP de origem. Métricas que não contribuem significativamente para a detecção de ataques são removidas, como p.ex. tipos de protocolo específicos.

Normalização e Padronização: Como parte da abordagem proposta, os processos de normalização e padronização dos dados são aplicados para garantir que todas as variáveis tenham uma escala comparável e uma distribuição semelhante.

Métricas de tráfego de rede podem ter escalas muito diferentes, com algumas variando de 0 a 1000 e outras de 0 a 1. Para evitar que discrepâncias nas escalas influenciem a análise, a normalização é utilizada para ajustar os valores para uma escala comum, garantindo que todas as características contribuam igualmente para os modelos de aprendizado de máquina. Além disso, aplicamos a padronização para que os dados tenham uma média de zero e um desvio padrão de um, o que melhora a estabilidade dos algoritmos utilizados.

Vale ressaltar que essas técnicas são parte integrante da metodologia adotada neste trabalho e foram aplicadas tanto na abordagem geral quanto nos experimentos conduzidos para validação do esquema proposto.

Regressão de Valores Ausentes: Os registros de tráfego de rede podem não conter algumas entradas, como o número de pacotes por segundo ou o tempo de resposta. Para lidar com isso, utilizamos técnicas de imputação de dados, substituindo os valores ausentes por estimativas baseadas nos dados existentes. Dependendo da natureza dos dados, aplicamos diferentes métodos de imputação, incluindo a substituição pela média dos valores disponíveis, interpolação linear ou regressão baseada em variáveis correlacionadas. A escolha da técnica depende da distribuição dos dados e do impacto esperado na análise.

Verificação de Estado dos Dados: Os dados são submetidos a uma verificação completa para garantir que estejam corretamente processados e prontos para análise. Verifica-se, por exemplo, se todos os registros de tráfego de rede foram corretamente normalizados e padronizados, sem valores ausentes ou duplicados. *Scripts* de validação detectam erros ou inconsistências, como entradas fora dos intervalos esperados ou formatos incorretos.

Dados Aprovados: Após as etapas anteriores, os dados são considerados aprovados para análise

adicional. Uma validação cruzada é então realizada para verificar o desempenho e escolher o melhor algoritmo de classificação com base nos resultados obtidos.

4.3.3 Modelagem ML (3ª Fase)

Os dados processados servem de entrada para algoritmos de aprendizado de máquina nesta fase, com o intuito de reconhecer padrões de comportamento incomuns e categorizar os vários tipos de ataques DDoS. Nesse processo, os dados são ajustados para garantir que o algoritmo seja treinado de forma eficaz.

Para avaliar o desempenho dos modelos, utilizamos validação cruzada do tipo *k-fold cross-validation* com $k = 10$, garantindo que o conjunto de dados seja dividido em 10 subconjuntos, onde, a cada iteração, um subconjunto é utilizado para teste e os demais para treinamento. Essa abordagem reduz a dependência da divisão inicial dos dados e fornece uma avaliação mais confiável da performance dos algoritmos. Além disso, métricas como acurácia, precisão, *recall* e *F1-score* foram utilizadas para comparar os modelos e escolher a melhor abordagem para a classificação dos ataques.

Tais ajustes garantem a identificação e resposta aos ataques com precisão e eficiência. As três etapas desta fase são descritas a seguir.

Ajuste de Hiperparâmetros: O desempenho dos algoritmos é otimizado a partir do ajuste dos hiperparâmetros, que controlam o processo de aprendizado e podem afetar a eficácia na classificação dos dados. Como exemplo no contexto da detecção de ataque DDoS, a taxa de aprendizado ou a profundidade máxima da árvore de decisão podem ser ajustadas no algoritmo de classificação. Técnicas como busca em grade ou aleatória permitem explorar diferentes combinações de hiperparâmetros para encontrar aquelas que produzem os melhores resultados. Isso significa testar uma variedade de valores para cada hiperparâmetro e avaliar como cada conjunto de valores afeta o desempenho do algoritmo.

Validação Cruzada: Para avaliar o desempenho de um algoritmo e sua capacidade de generalização para novos dados, empregamos a técnica de validação cruzada. Dividimos nossos dados em múltiplos conjuntos de treinamento e teste. Em cada iteração, um conjunto é usado para treinar o algoritmo e o outro para testá-lo. Isso permite avaliar a adaptabilidade do algoritmo a diferentes conjuntos de dados e identificar possíveis problemas de sobreajuste, onde o algoritmo se ajusta demais aos dados de treinamento, ou subajuste, onde o algoritmo não consegue capturar adequadamente os padrões nos dados

Seleção do Algoritmo de Classificação: A partir da validação cruzada, o algoritmo de classificação que melhor se adequa aos nossos dados e ao problema em questão é selecionado. Após realizarmos a validação cruzada com diferentes algoritmos, como *Naive Bayes*, *Decision Tree*, *Logistic Regression* e *Random Forest*, avaliamos múltiplos fatores para determinar qual algoritmo oferece o melhor equilíbrio entre desempenho e capacidade de generalização.

Os fatores analisados incluem métricas de desempenho, como Precisão, *Recall* e *F1-score*, além de aspectos como tempo de execução, robustez a dados desbalanceados e complexidade computacional.

4.3.4 Avaliação (4ª Fase)

O foco desta fase é a interpretação dos resultados obtidos na anterior, bem como a análise da eficácia na identificação de ataques DDoS. A seguir, são apresentadas as etapas para validar a detecção de ataques DDoS à camada de aplicação.

Avaliação de Desempenho do Algoritmo: O objetivo é avaliar a capacidade dos algoritmos em realizar previsões precisas, através da análise de métricas de desempenho como Precisão, *Recall* e *F1-score*. Os algoritmos são treinados e testados em conjuntos distintos de dados e, durante esse processo, comparações de desempenho são realizadas a fim de definir qual deles é mais adequado para um determinado cenário.

Interpretação do Algoritmo: Esta etapa consiste na análise dos resultados gerados pelos modelos de aprendizado de máquina, incluindo a interpretação dos pesos e da importância das características. Técnicas como *Feature Importance*, *SHAP* (Shapley Additive Explanations) e análise de coeficientes são utilizadas para examinar como cada variável influencia a decisão do modelo. Por exemplo, em uma Regressão Logística, analisamos os coeficientes associados a cada variável, enquanto em algoritmos baseados em árvores de decisão, avaliamos a importância atribuída a cada feature na construção dos nós de decisão.

A análise é realizada de forma automatizada por meio de relatórios gerados pelos algoritmos de classificação, permitindo a extração de métricas como tempo de execução, consumo de recursos computacionais e taxas de acerto em intervalos regulares. Além disso, gráficos interpretativos, como matrizes de confusão e curvas ROC, são gerados para facilitar a visualização do desempenho dos modelos. Essa abordagem possibilita um monitoramento contínuo da eficácia do esquema, garantindo ajustes e otimizações conforme necessário.

Validação Adversarial: O algoritmo é testado quanto à sua robustez em face de entradas conflitantes, que são pequenas perturbações nos dados de entrada projetadas para enganar o algoritmo, levando-o a fazer previsões incorretas. Esta etapa verifica as últimas entradas e analisa possíveis anomalias, decidindo a melhor ação a ser realizada.

Resultado Final (Acurácia/Tempo de execução): Por fim, é realizada uma análise em termos da acurácia e do tempo de execução dos algoritmos, no intuito de validar requisitos aceitáveis de desempenho considerando eventuais ambientes de produção. Por exemplo, se um sistema baseado no esquema proposto for implantado em um ambiente de alto tráfego, é essencial que o algoritmo escolhido seja capaz de realizar previsões precisas em um tempo de execução razoável, a fim de tomar decisões rápidas de mitigação. Já para um sistema executado em um ambiente com restrições de recursos computacionais, como dispositivos IoT, é importante que o algoritmo

tenha uma eficiência computacional superior, mesmo que isso reduza de forma aceitável a precisão. Esta etapa considera esses aspectos do ambiente de produção para garantir a adequação às necessidades específicas.

5 RESULTADOS

Este capítulo é reservado para a apresentação dos resultados obtidos, que serão analisados e discutidos em profundidade, destacando as descobertas mais significativas e sua relevância para a pesquisa em questão. Serão abordadas também as limitações encontradas durante o processo e possíveis direções para futuras investigações.

5.1 Avaliação de Desempenho

Neste capítulo daremos início à avaliação dos resultados da aplicação do modelo nos ataques DDoS registrados e compilados no conjunto de dados disponível, concentrando-nos na minuciosa investigação dos padrões, comportamentos e atributos específicos desses ataques. Nosso objetivo principal é aprofundar a compreensão de sua natureza, buscando enriquecer os resultados obtidos até o momento. Vale ressaltar que este processo vai além da mera identificação dos ataques por meio de processos de inteligência artificial, envolvendo também a extração de percepções significativas.

Nesta etapa, também executamos os passos da Fase 2 do esquema, que incluem a normalização e padronização dos dados, bem como a regressão de valores ausentes. Esses passos irão garantir a qualidade e a consistência dos dados, melhorando assim o desempenho dos algoritmos de *machine learning* na detecção de ataques DDoS. A normalização e padronização ajudam a equilibrar as diferentes escalas dos atributos, enquanto a regressão de valores ausentes assegura que nenhuma informação vital seja perdida devido a dados incompletos. Adicionalmente, foi realizada uma etapa da Fase 3 chamada ajuste de hiperparâmetros, que visa otimizar o desempenho dos algoritmos de *machine learning* ajustando os parâmetros que controlam o processo de aprendizado.

Uma análise de desempenho é realizada como parte integrante da terceira e quarta fases do esquema proposto, a partir da comparação dos algoritmos *Naive Bayes* (NB), *Decision Tree* (DT), *Logistic Regression* (LR) e *Random Forest* (RF). Os resultados apresentados a seguir referem-se ao desempenho dos classificadores com base nos algoritmos utilizados, tanto em um ambiente de programação geral, quanto em outro com solução de *Big Data*. Assim como em (AWAN et al., 2021), o objetivo é quantificar a diferença do desempenho em ambos os ambientes. As métricas utilizadas são: **Precisão**, proporção de verdadeiros positivos entre todas as previsões positivas feitas pelo algoritmo; ***Recall***, proporção de verdadeiros positivos que foram corretamente identificados pelo algoritmo em relação ao total de verdadeiros positivos existentes; ***F1-score***, harmonização do equilíbrio entre a Precisão e o *Recall*; e **Acurácia**, proporção de todas as previsões corretas (verdadeiros positivos e verdadeiros negativos) em relação ao total de previsões feitas pelo algoritmo.

No relatório dos algoritmos de aprendizado de máquina, são geradas diferentes classes que representam as categorias de saída do algoritmo. Essas classes são utilizadas para avaliar o desempenho do algoritmo em categorizar corretamente os dados de entrada em suas respectivas classes durante o processo de treinamento e teste. O código do esquema proposto está disponível em repositório público[1].

5.1.1 Ambiente de Teste para as Análises de Desempenho

Os testes de análise de desempenho do esquema proposto foram conduzidos primeiramente em um ambiente de nuvem, selecionado para assegurar um desempenho mínimo durante a execução dos experimentos. As especificações detalhadas da máquina utilizada como ambiente de teste são: CPU de 2,6 GHz 6-Core Intel Core i7; 16GB de memória RAM 2667 MHz DDR4; e SSD com capacidade de 499,96 GB. Os testes foram divididos em dois ambientes distintos para comparar diferentes aspectos do desempenho:

- **Ambiente de Programação Geral (APG)** – para testar a eficácia dos algoritmos em um cenário convencional de programação, através da biblioteca *scikit-learn*, onde os recursos são otimizados para um desenvolvimento mais geral e flexível.
- **Ambiente de *Big Data* (ABD)** – para simular condições de grandes volumes de dados e processos intensivos, utilizando o *Apache Spark*, permitindo comparar a escalabilidade e robustez dos algoritmos em relação ao ambiente anterior.

Em um segundo momento, com o intuito de destacar o potencial do ABD, novos experimentos foram executados em um servidor local com maior poder computacional. As especificações da máquina utilizada como servidor local são: CPU de 3.5 GHz 8-Core Intel Xeon; 80GB de memória RAM 2133 MHz DDR4; e HDD com capacidade de 1 TB. Acredita-se que uma maior quantidade de memória RAM e um maior poder de processamento proporcionarão um ambiente mais eficiente para a ferramenta de *big data*.

5.1.2 Resultados das Métricas de Desempenho

A etapa foi dividida em dois momentos distintos para avaliar o impacto das transformações nos dados e dos ajustes de hiperparâmetros sobre o desempenho dos algoritmos.

No primeiro momento, denominado **Execução Base**, os testes foram realizados sem a aplicação de normalização, padronização ou ajustes de hiperparâmetros. Essa abordagem permitiu estabelecer uma linha de base (*baseline*) de desempenho dos algoritmos, identificando quais modelos eram naturalmente mais robustos aos dados brutos.

No segundo momento, denominado **Execução Otimizada**, aplicamos as transformações nos dados, incluindo normalização e padronização, além da otimização dos hiperparâmetros

[1] https://github.com/digenaldo/malicious-traffic-detection-ml.

dos algoritmos. Esse ajuste possibilitou avaliar a melhoria no desempenho e identificar quais modelos se beneficiaram mais das transformações realizadas.

Ao nomear esses momentos como **Execução Base** e **Execução Otimizada**, facilita-se a associação direta com os resultados apresentados, permitindo uma análise mais clara da influência de cada etapa sobre a classificação dos ataques DDoS.

No primeiro momento, os testes foram executados sem a normalização e padronização dos dados da Fase 2 e sem os ajustes de hiperparâmetros da Fase 3. Essa abordagem inicial permitiu estabelecer uma linha de base (*baseline*) de desempenho dos algoritmos, identificando quais modelos eram naturalmente mais robustos aos dados brutos.

No segundo momento, rodamos os algoritmos de classificação nos dois ambientes, APG e ABD, aplicando os ajustes necessários das Fases 2 e 3. Essa etapa possibilitou medir a melhoria no desempenho dos modelos e analisar quais algoritmos foram mais sensíveis à normalização e à otimização de hiperparâmetros. Dessa forma, a comparação entre os dois momentos foi fundamental para entender a influência dessas transformações na eficácia da detecção de ataques DDoS.

A normalização e padronização são técnicas utilizadas para transformar os dados de entrada, garantindo que diferentes escalas de atributos não prejudiquem o desempenho dos algoritmos de aprendizado de máquina. A normalização redimensiona os valores das características para uma faixa padrão, normalmente entre 0 e 1, enquanto a padronização transforma os dados para que tenham média zero e desvio padrão igual a um. Essas técnicas são essenciais para garantir que todos os atributos contribuam igualmente para o modelo, evitando vieses que podem surgir devido a escalas diferentes (PATIL et al., 2019) (PEDREGOSA et al., 2011) (FOUNDATION, 2024).

A hiperparametrização, por sua vez, refere-se ao processo de ajuste dos hiperparâmetros de um modelo de aprendizado de máquina, que são parâmetros cujo valor é configurado antes do treinamento do modelo. Ao contrário dos parâmetros aprendidos pelo modelo durante o treinamento, os hiperparâmetros são definidos para otimizar o desempenho do modelo. Isso pode incluir ajustes como a profundidade de uma árvore de decisão, a taxa de aprendizado em um modelo de regressão logística ou o número de estimadores em um modelo de floresta aleatória (SILVA et al., 2022).

Os resultados da Tabela 3 indicam uma boa Precisão dos classificadores nos dois ambientes, observadas algumas diferenças. No APG, os classificadores utilizando NB, DT e RF apresentaram uma precisão máxima (1.00) para todas as classes, enquanto o LR obteve uma precisão ligeiramente menor, variando entre 0.92 e 0.95. Já no ABD, embora alguns resultados sejam semelhantes, algumas diferenças significativas foram notadas. Por exemplo, o NB apresentou redução na precisão para todas as classes, com valores entre 0.40 e 0.91. O LR também obteve resultados diferentes, mas, no geral, com uma precisão maior, quando comparada aos

resultados no APG. No entanto, o DT e o RF mantiveram uma precisão máxima para todas as classes.

Portanto, considerando os resultados em ambos os ambientes, podemos concluir que o DT e o RF se destacaram como os classificadores mais consistentes, mantendo uma precisão máxima para todas as classes.

Esse desempenho pode ser justificado pelo fato de que ambos os algoritmos são baseados em árvores de decisão, que possuem uma estrutura robusta para lidar com relações não lineares entre as variáveis do dataset. Além disso, o Random Forest, por ser um conjunto de múltiplas árvores treinadas com amostras aleatórias dos dados, reduz significativamente o risco de *overfitting* e melhora a generalização do modelo.

Outro fator relevante é que os dados analisados apresentam padrões distintos entre as classes de tráfego, o que favorece algoritmos baseados em regras decisórias claras, como DT e RF. Além disso, métricas como *Flow_IAT* e *SYN_Flag_Count*, que se mostraram relevantes na diferenciação dos ataques, são bem aproveitadas por esses modelos, permitindo uma separação eficiente entre tráfego legítimo e tráfego malicioso.

Por outro lado, modelos como *Logistic Regression* e *Naive Bayes* podem ter encontrado dificuldades devido à não linearidade dos dados e à presença de outliers, o que reforça a adequação do DT e RF ao problema estudado.

Tabela 3 – Comparativo do desempenho da Precisão dos classificadores.

Classificador (APG)	Precisão (Classe 0)	Precisão (Classe 1)	Precisão (Classe 2)
Naive Bayes (NB)	1.00	0.92	0.88
Decision Tree (DT)	1.00	1.00	1.00
Logistic Regression (LR)	0.92	0.95	0.93
Random Forest (RF)	1.00	1.00	1.00
Classificador (ABD)	**Precisão (Classe 0)**	**Precisão (Classe 1)**	**Precisão (Classe 2)**
Naive Bayes (NB)	0.91	0.91	0.40
Decision Tree (DT)	1.00	1.00	1.00
Logistic Regression (LR)	1.00	0.99	0.98
Random Forest (RF)	1.00	1.00	0.99

A Tabela 4 apresenta os desempenhos em relação à métrica *Recall*. Observa-se no APG que os classificadores com DT e RF obtiveram valores máximos (1.00) para todas as classes; ou seja, foram capazes de identificar corretamente todos os exemplos positivos de cada classe. O NB também apresentou um *Recall* alto (0.94 a 0.95), enquanto o desempenho do LR foi um pouco inferior para a Classe 1 (0.87), embora elevado para as outras classes (0.99 e 0.90). Os resultados foram ligeiramente diferentes no ABD. O NB teve um desempenho pior para a Classe 1 (0.36), mas um desempenho equivalente para as outras classes (0.96 e 0.92). O RF e o DT apresentaram um *Recall* máximo em todas as classes, enquanto o LR apresentou resultados

próximos. Portanto, conclui-se que os classificadores com o DT e o RF obtiveram os melhores resultados, com um *Recall* máximo independente da classe e ambiente de execução.

Tabela 4 – Comparativo do desempenho de *Recall* dos classificadores.

Classificador (APG)	*Recall* (Classe 0)	*Recall* (Classe 1)	*Recall* (Classe 2)
Naive Bayes (NB)	0.95	0.94	0.94
Decision Tree (DT)	1.00	1.00	1.00
Logistic Regression (LR)	0.99	0.87	0.90
Random Forest (RF)	1.00	1.00	1.00
Classificador (ABD)	***Recall* (Classe 0)**	***Recall* (Classe 1)**	***Recall* (Classe 2)**
Naive Bayes (NB)	0.96	0.36	0.92
Decision Tree (DT)	1.00	1.00	1.00
Logistic Regression (LR)	1.00	0.99	0.99
Random Forest (RF)	1.00	1.00	1.00

A Tabela 5 apresenta o desempenho dos classificadores em relação ao *F1-score*, que fornece uma medida de equilíbrio entre as métricas de Precisão e *Recall*. No APG, os classificadores DT e RF apresentaram valores máximos (1.00), ratificando o ótimo desempenho na Precisão e *Recall* em todas as classes. Ou seja, foram capazes de alcançar um equilíbrio entre as duas métricas, resultando em um alto desempenho na classificação dos exemplos. O NB e o LR obtiveram um desempenho semelhante, com um *F1-score* elevado para a Classe 0 e ligeiramente menores para as outras duas classes.

Os resultados no ABD foram novamente diferentes. Os classificadores DT, LR e RF mantiveram um *F1-score* virtualmente máximo para todas as classes, mas o NB apresentou um desempenho bem inferior, principalmente para as classes 1 e 2.

Essa melhoria de desempenho dos classificadores DT e RF pode ser atribuída à sua capacidade de modelar relações não lineares e capturar interações complexas entre as variáveis do conjunto de dados. O RF, em particular, melhora a generalização ao combinar múltiplas árvores de decisão treinadas com subconjuntos aleatórios dos dados, reduzindo o risco de *overfitting* e aumentando a robustez do modelo.

Por outro lado, o menor desempenho do NB pode ser explicado pela suposição de independência condicional entre as variáveis, o que pode não ser válido para os padrões de tráfego de rede analisados. Como ataques DDoS frequentemente apresentam correlações entre diferentes atributos, essa limitação do NB impacta sua capacidade de separar eficientemente as classes 1 e 2.

Dessa forma, pelos resultados obtidos, conclui-se novamente que os classificadores DT e RF obtiveram os melhores desempenhos para o *F1-score* em todas as classes, devido à sua flexibilidade na modelagem dos dados e à capacidade de lidar com interações mais complexas entre as variáveis analisadas.

Tabela 5 – Comparativo do desempenho do *F1-score* dos classificadores.

Classificador (APG)	*F1-score* (Classe 0)	*F1-score* (Classe 1)	*F1-score* (Classe 2)
Naive Bayes (NB)	0.98	0.93	0.91
Decision Tree (DT)	1.00	1.00	1.00
Logistic Regression (LR)	0.96	0.91	0.91
Random Forest (RF)	1.00	1.00	1.00
Classificador (ABD)	***F1-score* (Classe 0)**	***F1-score* (Classe 1)**	***F1-score* (Classe 2)**
Naive Bayes (NB)	0.93	0.52	0.56
Decision Tree (DT)	1.00	1.00	1.00
Logistic Regression (LR)	1.00	0.99	0.98
Random Forest (RF)	1.00	1.00	1.00

Os valores para o desempenho da Acurácia dos classificadores estão presentes na Tabela 6, que mede a proporção de exemplos corretamente classificados pelo algoritmo em relação ao total. Observamos que o RF e o DT obtiveram a maior acurácia (1.00) considerando os dois ambientes, classificando corretamente 100% dos exemplos no conjunto de dados de teste. O LR também apresentou um bom desempenho, com uma acurácia um pouco menor no APG. Esse desempenho bom dos três classificadores indica uma alta capacidade de generalização e precisão na classificação. O NB obteve uma boa acurácia no APG (0.95), mas apresentou uma queda no ABD (0.73).

O destaque do RF e DT pode ser explicado pelo fato de que ambos os algoritmos baseiam-se em árvores de decisão, que são eficientes na modelagem de dados com padrões bem definidos e interações não lineares entre as features. Além disso, o RF, por utilizar múltiplas árvores treinadas de forma aleatória, melhora a capacidade de generalização e reduz a susceptibilidade ao *overfitting*.

Já o LR, apesar de apresentar bom desempenho, pode ter sido impactado pela presença de relações não lineares nos dados, que dificultam a separação das classes com um modelo linear. O Naive Bayes, por sua vez, assume independência condicional entre as features, o que pode ter sido um fator limitante no ambiente ABD, onde as variáveis apresentam correlações mais fortes, reduzindo a precisão da classificação.

Portanto, com base nos resultados apresentados na Tabela 6, percebe-se que o RF e o DT se destacam com a melhor acurácia nos dois ambientes devido à sua flexibilidade na modelagem dos dados e capacidade de lidar com relações complexas entre as features. O LR teve um desempenho ligeiramente inferior no APG devido à sua natureza linear, enquanto o NB mostrou queda significativa no ABD, possivelmente devido à violação da suposição de independência entre as variáveis.

Tabela 6 – Comparativo do desempenho da Acurácia dos classificadores.

Classificador (APG)	Acurácia	Classificador (ABD)	Acurácia
Naive Bayes (NB)	0.95	Naive Bayes (NB)	0.73
Decision Tree (DT)	1.00	Decision Tree (DT)	1.00
Logistic Regression (LR)	0.93	Logistic Regression (LR)	0.99
Random Forest (RF)	1.00	Random Forest (RF)	1.00

5.1.3 Resultados de Tempos de Execução

Cada gráfico da Figura 3 compara os resultados do tempo médio de execução (eixo Y à esquerda) e da média das acurácias (eixo Y à direita) obtidos pelos algoritmos de classificação, nos dois ambientes de experimentação e nas duas máquinas utilizadas (uma na nuvem e a outra um servidor local). Levando em consideração a plataforma na nuvem primeiro, nas Figuras 3(a) e (b), observa-se que o tempo médio de execução segue o mesmo padrão em ambos os ambientes (APG e ABD), com o NB apresentando os menores valores (2.94s e 11.18s, respectivamente) e o RF os maiores (174.64s e 233.23s, respectivamente). Para este último, destaca-se a grande variabilidade nos tempos de execução, indicada pelas barras de erro em 30 repetições de cada cenário, obtidas com um nível de confiança de 95%.

O DT também apresenta um maior tempo médio de execução no ABD (26.63s) comparado ao APG (8.90s). A exceção foi o LR, onde não é possível indicar estatisticamente que os resultados são diferentes. Com relação à acurácia, como indicado anteriormente, todos os algoritmos apresentaram um bom desempenho no APG, com médias acima de 0.90 e com destaque para os valores máximos obtidos pelo DT e RF. Os resultados da acurácia no ABD são praticamente os máximos para o DT, LR e RF, embora o NB tenha apresentado o pior valor médio.

Os resultados indicam que, embora haja variação nos tempos de execução nos dois ambientes, a acurácia permanece virtualmente estável. Isso demonstra uma relativa robustez dos algoritmos em lidar com conjuntos grandes de dados, mantendo um desempenho consistente independentemente das diferenças no tempo de execução.

Aproveitando um ambiente computacional mais robusto, as Figuras 3(c) e (d) apresentam uma maior vantagem em relação aos tempos de execução obtidos. Como esperado, houve uma redução nos tempos médios de execução em ambos os ambientes (APG e ABD), com um ganho ainda mais significativo para o RF, que reduziu de aproximadamente 3.8 minutos para pouco mais de 20 segundos. Esse resultado demonstra o impacto da capacidade computacional na eficiência dos algoritmos de aprendizado de máquina.

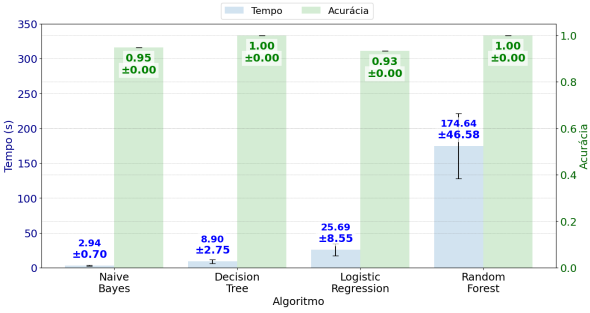
No contexto deste estudo, um ambiente menos robusto refere-se a uma infraestrutura com menor poder de processamento, memória RAM limitada e menor paralelismo na execução das tarefas. No experimento, esse ambiente foi representado por um servidor local com especificações inferiores em relação à plataforma em nuvem utilizada, que possui maior capacidade de

processamento distribuído e otimização para cargas de trabalho intensivas.

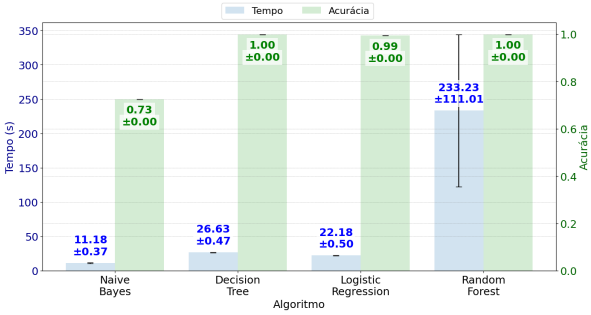
Considerando o potencial do modelo de detecção proposto para ser utilizado em um cenário real, quanto mais rápido for o tempo de execução, melhor será sua aplicabilidade. Nesse sentido, o uso de uma ferramenta de *big data*, aliado a uma plataforma computacional mais robusta, mostrou-se eficiente para reduzir o tempo de execução dos algoritmos sem comprometer sua acurácia.

Figura 3 – Comparativo dos tempos de execução e acurácia.

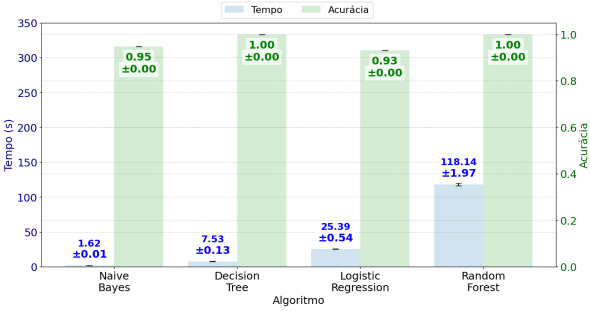
(a) Nuvem – APG



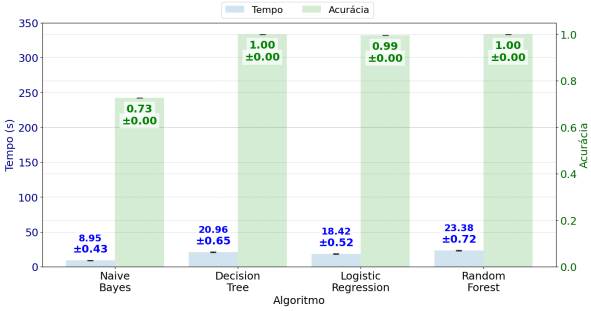
(b) Nuvem – ABD



(c) Servidor – APG



(d) Servidor – ABD



5.1.4 Análise de Impacto das *Features* no Desempenho

Buscando uma nova abordagem para aprimorar o desempenho do modelo, realizamos uma série de testes com foco na análise da importância das variáveis (FI, do termo em inglês *feature importance*). A análise de FI é essencial para entender como cada variável contribui para as previsões, proporcionando *insights* sobre as características que mais impactam a precisão e a robustez do modelo. Nosso objetivo foi explorar a FI para avaliar se priorizar variáveis mais relevantes poderia efetivamente melhorar tanto o tempo de processamento quanto a acurácia geral do modelo.

Estudos na literatura destacam que a avaliação de FI não apenas facilita a interpretação de modelos complexos, mas também aprimora o desempenho ao permitir a eliminação de variáveis redundantes ou irrelevantes, contribuindo para uma melhor generalização e eficiência computacional (ALDUAILIJ et al., 2022) (SANMORINO; GUSTRIANSYAH; ALIE, 2022) (KIM; KIM; KIM, 2022) (ZHOU et al., 2022).

Para essa análise, utilizamos a função `find_common_features` do Código 5.1, que foi projetada para extrair as 10 *features* mais relevantes de cada modelo de aprendizado supervisionado. Essa seleção das 10 maiores características de cada modelo permitiu uma comparação direta das variáveis mais influentes entre diferentes algoritmos, destacando as diferenças de impacto que cada característica exerce em modelos distintos. Para essa avaliação, empregamos técnicas como a importância de permutação, adequada para algoritmos de árvore de decisão, e coeficientes de importância adaptados a modelos lineares.

```
# Funcao para encontrar top N features de cada modelo
def find_common_features(feature_importances_all, feature_names,
    output_path="common_and_top_features.csv", top_n=10):
    # Lista para armazenar DataFrames com o Top N de cada modelo
    top_features_list = []

    # Itera sobre cada modelo para obter as top N features
    for model_name, importance_df in feature_importances_all.items():
        # Seleciona o Top N
        top_features = importance_df.head(top_n).copy()
        # Converte indices para nomes
        top_features['feature'] = top_features['feature'].apply(lambda x:
    feature_names[x])
        top_features['model'] = model_name
        top_features_list.append(top_features)

    # Concatenar as top N features de todos os modelos em um unico DataFrame
    all_top_features_df = pd.concat(top_features_list, ignore_index=True)

    # Encontrar as features comuns entre os Top N de todos os modelos
    common_features = set(all_top_features_df['feature'])
    for model_name, importance_df in feature_importances_all.items():
        model_top_features =
    set(importance_df.head(top_n)['feature'].apply(lambda x: feature_names[x]))
        common_features &= model_top_features

    # Preparar DataFrame para salvar
    common_features_df = pd.DataFrame({'feature': list(common_features),
    'type': 'Common Feature'})

    # Salvar em CSV
    all_features_df = pd.concat([common_features_df, all_top_features_df],
    ignore_index=True)
    all_features_df.to_csv(output_path, index=False)
```

```
    logging.info(f"Features comuns e Top {top_n} features de cada modelo
    salvas com sucesso em '{output_path}'")

    return list(common_features)
```

Código 5.1 – Função para encontrar as top-N características de cada modelo.

A importância de permutação é uma técnica útil para medir a relevância de cada característica em relação ao desempenho do modelo. Em resumo, essa abordagem embaralha os valores de cada característica individualmente e calcula a diferença de desempenho (tipicamente medida por acurácia ou outra métrica) entre o modelo original e o modelo com a característica permutada. Quanto maior a queda de desempenho, mais importante é a determinada característica para o modelo (ABDELAZIZ et al., 2025) (SINGH; KUMARI; KAUR, 2024) (ZHAO et al., 2020).

No Código 5.2, a função `permutation_importance` implementa essa técnica. Ela começa calculando a acurácia inicial do modelo com base nos dados de teste (`X_test` e `y_test`). Em seguida, para cada característica, os valores dessa coluna são permutados e a acurácia do modelo é recalculada com os dados modificados. A FI é então definida como a diferença entre a acurácia inicial e a acurácia permutada. Esse processo permite capturar o impacto de cada característica no desempenho do modelo de forma independente, auxiliando na sua interpretação e otimização.

```
def permutation_importance(model, X_test, y_test):
    baseline_accuracy = accuracy_score(y_test, model.predict(X_test))
    importances = []

    for i in range(X_test.shape[1]):
        X_permuted = X_test.copy()
        np.random.shuffle(X_permuted[:, i])
        permuted_accuracy = accuracy_score(y_test, model.predict(X_permuted))
        importance = baseline_accuracy - permuted_accuracy
        importances.append(importance)

    return np.array(importances)
```

Código 5.2 – Função para calcular a importância das características baseada em permutação.

A importância das variáveis indica o peso ou a relevância de cada característica nas previsões de um modelo. No caso de algoritmos de árvore, por exemplo, essa importância é calculada pela redução média da impureza em cada divisão que utiliza uma característica específica. Para modelos lineares, tal importância pode ser associada aos coeficientes atribuídos a cada variável, refletindo o impacto direto que cada característica tem sobre o resultado previsto

pelo modelo.

A Tabela 7 exibe as variáveis mais relevantes para cada modelo, com seus respectivos valores de importância arredondados para duas casas decimais. Esses valores indicam o impacto individual de cada característica no desempenho do modelo, auxiliando na interpretação daquelas que mais influenciam as previsões. Uma maior importância sugere uma influência mais significativa no desempenho do modelo, enquanto valores mais baixos indicam variáveis com menor impacto.

Modelo	***Feature***	**Importância**
Naive Bayes	Destination_Port	0.20
	Flow_Bytes_Sec	0.17
	Subflow_Bwd_Packets	0.15
	Total_Backward_Packets	0.15
	Init_Win_bytes_forward	0.13
	Bwd_Packet_Length_Max	0.07
	Flow_Packets_Sec	0.07
	Bwd_IAT_Total	0.07
	Bwd_IAT_Max	0.07
	Bwd_IAT_Mean	0.06
Decision Tree	min_seg_size_forward	0.64
	act_data_pkt_fwd	0.27
	Max_Packet_Length	0.05
	Flow_Bytes_Sec	0.03
	Fwd_IAT_Min	0.01
	Flow_IAT_Max	0.00
	ACK_Flag_Count	0.00
	Bwd_Header_Length	0.00
	SYN_Flag_Count	0.00
	Flow_IAT_Mean	0.00
Logistic Regression	Packet_Length_Variance	12.61
	Bwd_PSH_Flags	8.10
	RST_Flag_Count	8.08
	Init_Win_bytes_forward	7.91
	Min_Packet_Length	7.64
	SYN_Flag_Count	5.99
	Bwd_Packet_Length_Min	3.82
	Flow_Packets_Sec	3.21
	Flow_Bytes_Sec	3.07
	Fwd_PSH_Flags	2.66
Random Forest	min_seg_size_forward	0.10
	Flow_Packets_Sec	0.08
	Flow_Bytes_Sec	0.07
	Fwd_Packet_Length_Std	0.07
	Max_Packet_Length	0.06
	Total_Length_of_Bwd_Packets	0.05
	Fwd_Packet_Length_Max	0.05
	Avg_Bwd_Segment_Size	0.04
	Init_Win_bytes_backward	0.03
	Fwd_IAT_Std	0.03

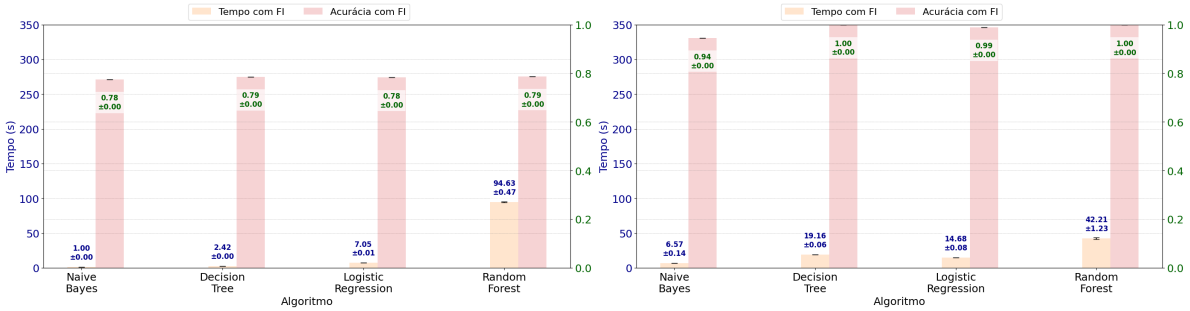
Tabela 7 – Tabela com as top-10 características mais importantes dos modelos.

Realizamos uma avaliação no ambiente controlado de servidor local com o objetivo de analisar a FI utilizando um conjunto de dados específico. Tal análise foi conduzida para investigar o impacto da priorização de variáveis mais relevantes, identificadas pela técnica de FI, nas métricas de desempenho dos modelos, especificamente no tempo de execução e na acurácia. Para isso, foram testados nos dois ambientes distintos, APG e ABD. Essa abordagem possibilitou avaliar a eficiência computacional e a robustez dos modelos em condições variadas,

oferecendo detalhes sobre a viabilidade do uso de FI para otimização de desempenho em sistemas de aprendizado supervisionado. A Figura 4 compara os resultados obtidos, detalhados a seguir.

Figura 4 – Comparativo dos tempos de execução e acurácia com FI.

(a) Servidor – APG (b) Servidor – ABD



Na Figura 4(a), observamos o impacto da FI nos modelos em um ambiente APG. De maneira geral, percebe-se um tempo de execução reduzido, mas uma acurácia piorada em todos os modelos, quando comparados com os resultados da Figura 3(c).

O uso de FI com o *Naive Bayes* teve um ligeiro impacto no tempo de execução (redução de 1.62s para 1s), mas uma acurácia piorada de 0.78 em comparação com o valor anterior de 0.95 para o mesmo ambiente e mesma plataforma. Isso indica que, para o *Naive Bayes*, a seleção de variáveis baseada em FI não trouxe benefícios substanciais neste cenário.

A aplicação de FI com *Decision Tree* resultou em uma redução significativa do tempo de execução, de 7.53s para 2.42s; mas com uma redução também na acurácia, de 1.00 para 0.79. Essa economia de tempo destaca a eficácia da FI em simplificar a complexidade do modelo *Decision Tree*, possivelmente eliminando variáveis redundantes e reduzindo a quantidade de cálculos necessários. Porém, a alteração das variáveis de entrada (*features*) neste cenário indica a perda de informações críticas, impactando negativamente a acurácia.

Com o *Logistic Regression*, observamos um impacto similar ao caso anterior, com uma redução no tempo de execução de 25.39s para 7.05s após o uso de FI; reduzindo também a acurácia de 0.93 para 0.78. Novamente, a redução das *features* de entrada impacta de maneira negativa na acurácia neste cenário, embora tenha experimentado uma redução significativa no tempo de execução.

Por fim, o *Random Forest* também apresentou ganho de desempenho em relação ao tempo de execução, caindo de 118.14s para 94.63s com o uso de FI. Essa diminuição indica que a priorização de variáveis mais relevantes ajuda a reduzir o número de árvores necessárias para a previsão, economizando tempo de processamento. Contudo, de maneira similar aos demais modelos, a perda de acurácia (de 1.00 para 0.79) reflete o impacto da redução de *features* na precisão do *Random Forest*.

Na Figura 4(b), que representa os resultados no ambiente ABD, notamos algumas

diferenças nos impactos da técnica de FI em comparação com os resultados obtidos no ambiente APG, bem como com os resultados obtidos anteriormente e ilustrados na Figura 3(d).

O tempo de execução com o *Naive Bayes* e com FI no ambiente ABD foi maior do que o observado no APG (6.57s em comparação a 1s), mas com uma acurácia maior de 0.94 (em comparação a 0.78). Entretanto, para o *Naive Bayes*, os resultados com FI se mostraram melhores do que os resultados obtidos com todo o conjunto de *features* (Figura 3(d)). Além de um tempo de execução menor, a acurácia também melhorou, o que significa que o modelo se beneficiou da remoção de *features* irrelevantes ou redundantes, permitindo que ele se concentre nas informações mais úteis para a classificação.

O impacto do FI no tempo de execução com o *Decision Tree* foi significativamente maior em comparação ao ambiente APG, mas similar ao tempo obtido com todas as *features*, aproximadamente 20s. Por sua vez, a acurácia permaneceu elevada em 1.00, assim como nos resultados com todas as *features*, significativamente maior quando comparada ao ambiente APG.

Com o *Logistic Regression*, por outro lado, a aplicação de FI elevou o tempo de processamento em relação ao ambiente APG, mas reduziu em relação aos resultados com todas as *features* (de 18.42s para 14.68s). Já para a acurácia, os resultados no ambiente ABD (com e sem FI) são de aproximadamente 100% de precisão.

Por fim, observamos uma pequena melhoria na acurácia com o *Random Forest* em comparação com o ambiente APG, mantendo o valor 1.00, assim como nos resultados sem FI. Entretanto, o tempo de execução reduziu de 94.63s para 42.21s nos resultados com FI, mas aumentou em relação aos resultados sem FI (23.38s).

Em resumo, a Figura 4(a) demonstra que, embora o uso de FI tenha sido eficaz em reduzir o tempo de processamento no ambiente APG, essa melhoria veio acompanhada de uma perda significativa de acurácia em todos os modelos analisados. Já no ambiente ABD, a Figura 4(b) mostra que o uso de FI foi eficaz em reduzir os tempos de processamento de todos os modelos, exceto para o *Random Forest*. Adicionalmente, o *Naive Bayes* obteve um ganho significativo na acurácia do modelo, ao passo que os demais modelos permaneceram com, virtualmente, 100% de acurácia.

5.1.5 Técnicas de Engenharia de Características e Hiperparametrização

Nesta etapa, realizamos uma análise detalhada para explorar e avaliar o impacto de diferentes técnicas de engenharia de características e ajustes de hiperparâmetros no desempenho de modelos de aprendizado de máquina nos dois ambientes, ABD e APG. Essas técnicas foram aplicadas com o objetivo de otimizar a capacidade dos modelos em identificar padrões relevantes nos dados, contribuindo para uma melhoria significativa na acurácia e na robustez das previsões.

A análise foi inspirada por abordagens avançadas descritas em estudos sobre *frameworks* de engenharia de características (MALIK; DUTTA et al., 2023) (AAMIR; ZAIDI, 2019) (LIU

et al., 2023). Adotamos essas práticas como base para adaptar métodos eficazes aos contextos analisados, buscando soluções que maximizem a eficiência dos modelos.

Os quatro modelos avaliados foram aplicados a dados submetidos a diversas técnicas de pré-processamento (ZHENG; CASARI, 2018), incluindo *Min-Max Scaling*, PCA, *Polynomial Features*, SelectKBest, *Standard Scaling* e um cenário base sem a aplicação de técnicas específicas de engenharia de características. Os resultados da análise destacam o impacto das técnicas de seleção e transformação de características sobre o desempenho dos modelos, medido por meio de validação cruzada. Observamos que a aplicação criteriosa de métodos como normalização e redução de dimensionalidade não apenas melhora a eficácia preditiva, mas também reduz a complexidade computacional, especialmente em cenários onde a qualidade das características desempenha um papel crítico.

Na Tabela 8, são apresentados os resultados de uma análise comparativa entre diferentes técnicas de engenharia de características e ajustes de hiperparâmetros no desempenho de modelos de aprendizado de máquina para o cenário APG. Esta análise destaca como diferentes combinações de pré-processamento de dados e ajuste fino dos modelos podem impactar diretamente a acurácia das previsões. Para cada combinação, os modelos foram avaliados tanto com quanto sem ajuste de hiperparâmetros, permitindo compreender a sua influência no desempenho.

Os resultados revelam que o *Random Forest* consistentemente alcançou alta acurácia em todos os cenários, com destaque para combinações com *Standard Scaling* e *Min-Max Scaling*, onde atingiu até 99,67%. Por outro lado, o *Naive Bayes* apresentou limitações em cenários mais complexos, como o uso de PCA e *Polynomial Features*, onde a acurácia ficou significativamente reduzida (p.ex.: 36,17% com *Polynomial Features*). Já o *Decision Tree* e o *Logistic Regression* apresentaram desempenho intermediário, com melhorias marginais após o ajuste de hiperparâmetros em determinados cenários.

Entre as técnicas de engenharia de características, *Standard Scaling* e *Min-Max Scaling* se destacaram por melhorar a acurácia da maioria dos modelos, enquanto *Polynomial Features* e PCA mostraram desempenho mais variável, dependendo do modelo utilizado. SelectKBest também demonstrou bom potencial, especialmente quando combinado com ajustes de hiperparâmetros.

Tabela 8 – Engenharia de características e ajuste de hiperparâmetros - APG.

Eng. de Características	**Modelo**	**Hiperparâmetros**	**Acurácia**
-	Naive Bayes	Sem Ajuste	0.945
-	Decision Tree	Sem Ajuste	0.9933
-	Logistic Regression	Sem Ajuste	0.9067
-	Random Forest	Sem Ajuste	0.9967
-	Naive Bayes	Com Ajuste	0.945
-	Decision Tree	Com Ajuste	0.9933
-	Logistic Regression	Com Ajuste	0.9067
-	Random Forest	Com Ajuste	0.9967
Standard Scaling	Naive Bayes	Sem Ajuste	0.975
Standard Scaling	Decision Tree	Sem Ajuste	0.9933
Standard Scaling	Logistic Regression	Sem Ajuste	0.9833
Standard Scaling	Random Forest	Sem Ajuste	0.9967
Standard Scaling	Naive Bayes	Com Ajuste	0.98
Standard Scaling	Decision Tree	Com Ajuste	0.9933
Standard Scaling	Logistic Regression	Com Ajuste	0.9833
Standard Scaling	Random Forest	Com Ajuste	0.9967
Min-Max Scaling	Naive Bayes	Sem Ajuste	0.9817
Min-Max Scaling	Decision Tree	Sem Ajuste	0.9933
Min-Max Scaling	Logistic Regression	Sem Ajuste	0.97
Min-Max Scaling	Random Forest	Sem Ajuste	0.9967
Min-Max Scaling	Naive Bayes	Com Ajuste	0.975
Min-Max Scaling	Decision Tree	Com Ajuste	0.9933
Min-Max Scaling	Logistic Regression	Com Ajuste	0.9867
Min-Max Scaling	Random Forest	Com Ajuste	0.9967
PCA	Naive Bayes	Sem Ajuste	0.7367
PCA	Decision Tree	Sem Ajuste	0.94
PCA	Logistic Regression	Sem Ajuste	0.7517
PCA	Random Forest	Sem Ajuste	0.9467
PCA	Naive Bayes	Com Ajuste	0.7367
PCA	Decision Tree	Com Ajuste	0.95
PCA	Logistic Regression	Com Ajuste	0.7667
PCA	Random Forest	Com Ajuste	0.9467
Polynomial Features	Naive Bayes	Sem Ajuste	0.3617
Polynomial Features	Decision Tree	Sem Ajuste	0.9917
Polynomial Features	Logistic Regression	Sem Ajuste	0.6933
Polynomial Features	Random Forest	Sem Ajuste	0.9967
Polynomial Features	Naive Bayes	Com Ajuste	0.3617
Polynomial Features	Decision Tree	Com Ajuste	0.9917
Polynomial Features	Logistic Regression	Com Ajuste	0.6933
Polynomial Features	Random Forest	Com Ajuste	0.9967
SelectKBest	Naive Bayes	Sem Ajuste	0.885
SelectKBest	Decision Tree	Sem Ajuste	0.96
SelectKBest	Logistic Regression	Sem Ajuste	0.7267
SelectKBest	Random Forest	Sem Ajuste	0.955
SelectKBest	Naive Bayes	Com Ajuste	0.885
SelectKBest	Decision Tree	Com Ajuste	0.9883
SelectKBest	Logistic Regression	Com Ajuste	0.73
SelectKBest	Random Forest	Com Ajuste	0.96

A Tabela 9 detalha os resultados obtidos no cenário ABD, permitindo uma visão abrangente das variações no desempenho dos modelos em diferentes contextos. Foram avaliados os mesmos modelos de aprendizado de máquina e as mesmas técnicas de engenharia de caracterísi-

cas, além de um cenário base sem transformações. Cada modelo também foi analisado com e sem ajustes de hiperparâmetros.

Tabela 9 – Engenharia de características e ajuste de hiperparâmetros - ABD.

Engen. de Características	**Modelo**	**Hiperparâmetros**	**Acurácia**
-	Decision Tree	Sem Ajuste	1
-	Logistic Regression	Sem Ajuste	0.9983
-	Random Forest	Sem Ajuste	0.993
-	Naive Bayes	Com Ajuste	0.715
-	Decision Tree	Com Ajuste	1
-	Logistic Regression	Com Ajuste	0.9983
-	Random Forest	Com Ajuste	0.993
Standard Scaling	Naive Bayes	Sem Ajuste	0.715
Standard Scaling	Decision Tree	Sem Ajuste	1
Standard Scaling	Logistic Regression	Sem Ajuste	0.9983
Standard Scaling	Random Forest	Sem Ajuste	0.993
Standard Scaling	Naive Bayes	Com Ajuste	0.715
Standard Scaling	Decision Tree	Com Ajuste	1
Standard Scaling	Logistic Regression	Com Ajuste	0.9983
Standard Scaling	Random Forest	Com Ajuste	0.993
Min-Max Scaling	Naive Bayes	Sem Ajuste	0.715
Min-Max Scaling	Decision Tree	Sem Ajuste	1
Min-Max Scaling	Logistic Regression	Sem Ajuste	0.9983
Min-Max Scaling	Random Forest	Sem Ajuste	0.993
Min-Max Scaling	Naive Bayes	Com Ajuste	0.715
Min-Max Scaling	Decision Tree	Com Ajuste	1
Min-Max Scaling	Logistic Regression	Com Ajuste	0.9983
Min-Max Scaling	Random Forest	Com Ajuste	0.993
PCA	Naive Bayes	Sem Ajuste	0.715
PCA	Decision Tree	Sem Ajuste	1
PCA	Logistic Regression	Sem Ajuste	0.9983
PCA	Random Forest	Sem Ajuste	0.993
PCA	Naive Bayes	Com Ajuste	0.715
PCA	Decision Tree	Com Ajuste	1
PCA	Logistic Regression	Com Ajuste	0.9983
PCA	Random Forest	Com Ajuste	0.993
Polynomial Features	Naive Bayes	Sem Ajuste	0.715
Polynomial Features	Decision Tree	Sem Ajuste	1
Polynomial Features	Logistic Regression	Sem Ajuste	0.9983
Polynomial Features	Random Forest	Sem Ajuste	0.993
Polynomial Features	Naive Bayes	Com Ajuste	0.715
Polynomial Features	Decision Tree	Com Ajuste	1
Polynomial Features	Logistic Regression	Com Ajuste	0.9983
Polynomial Features	Random Forest	Com Ajuste	0.993
SelectKBest	Naive Bayes	Sem Ajuste	0.715
SelectKBest	Decision Tree	Sem Ajuste	1
SelectKBest	Logistic Regression	Sem Ajuste	0.9983
SelectKBest	Random Forest	Sem Ajuste	0.993
SelectKBest	Naive Bayes	Com Ajuste	0.715
SelectKBest	Decision Tree	Com Ajuste	1
SelectKBest	Logistic Regression	Com Ajuste	0.9983
SelectKBest	Random Forest	Com Ajuste	0.993

Os resultados revelam que o *Decision Tree* manteve um desempenho excepcional em

todos os cenários, alcançando a acurácia máxima, independentemente das combinações de técnicas de engenharia de características ou do ajuste de hiperparâmetros. O *Random Forest*, por sua vez, demonstrou um desempenho igualmente consistente, alcançando acurácia elevada (99,3%) em todos os cenários avaliados. Este comportamento reflete sua capacidade de generalização e estabilidade, independentemente das condições impostas. Embora não tenha alcançado a acurácia máxima, o *Logistic Regression* apresentou resultados consistentemente altos, registrando acurácia de 99,83% em todos os cenários. Isso sugere que a escolha das técnicas de pré-processamento tem um impacto menos significativo para este modelo em particular. Por outro lado, o *Naive Bayes* apresentou limitações claras devido à simplicidade de seu algoritmo, com acurácia limitada a 71,5% em todos os cenários avaliados. Esses resultados indicam dificuldades do modelo em lidar com padrões complexos presentes nos dados.

Entre as técnicas de engenharia de características, *Standard Scaling* e *Min-Max Scaling* se destacaram por manter a estabilidade dos resultados e melhorar a acurácia em vários casos. No entanto, técnicas mais avançadas, como PCA e *Polynomial Features*, demonstraram um impacto limitado neste cenário, com resultados mais variáveis dependendo do modelo utilizado.

O ajuste de hiperparâmetros não resultou em melhorias significativas para nenhum dos modelos avaliados, reforçando a ideia de que a configuração padrão de parâmetros é suficiente para o conjunto de dados analisado. Isso sugere que o impacto das técnicas de pré-processamento foi dominante em relação ao ajuste fino dos modelos.

Os resultados da análise comparativa entre os ambientes APG e ABD evidenciam a importância de técnicas de engenharia de características e ajustes de hiperparâmetros no desempenho de modelos de aprendizado de máquina. Embora ambos os ambientes apresentem diferenças nos padrões de complexidade dos dados, os achados apontam para conclusões consistentes sobre a eficácia dos modelos e técnicas avaliados.

O *Decision Tree* demonstrou uma estabilidade significativa em ambos os ambientes, atingindo acurácia próxima ao máximo independentemente das técnicas de pré-processamento aplicadas ou do ajuste de hiperparâmetros. Isso evidencia sua capacidade de generalização e resistência a variações nos dados de entrada, tornando-o uma escolha confiável para diferentes cenários de classificação.

Essa adaptação foi avaliada sob diferentes condições, incluindo normalização e padronização dos dados, ajustes na profundidade máxima da árvore, critérios de divisão (*gini* e *entropy*) e variações no tamanho do conjunto de treinamento. Os resultados indicaram que, mesmo com essas modificações, o *Decision Tree* manteve um desempenho consistente, o que reforça sua eficácia em lidar com conjuntos de dados diversos sem necessidade de otimizações complexas.

O *Random Forest* também manteve desempenho consistente e elevado, destacando-se por sua capacidade de generalização e estabilidade, com acurácias acima de 94% em todos os ambientes. Este comportamento o posiciona como uma alternativa versátil e eficaz, especialmente

em situações onde a robustez do modelo é crítica.

O *Logistic Regression* apresentou resultados satisfatórios na maioria das combinações testadas, com resultados piorados em alguns casos do ambiente APG. Isso sugere que seu desempenho é menos dependente das técnicas de pré-processamento aplicadas, tornando-o uma opção eficiente em contextos de menor variabilidade.

Por outro lado, o *Naive Bayes* revelou limitações significativas em ambos os ambientes, com acurácias restritas a 71,5% no cenário ABD e resultados ainda mais baixos em ambientes complexos do APG. Este desempenho reflete a simplicidade do modelo, que não é ideal para lidar com padrões mais sofisticados nos dados.

O ajuste de hiperparâmetros, embora essencial em muitos estudos, não trouxe ganhos significativos nos ambientes analisados. Este resultado sugere que, para os dados avaliados, os parâmetros padrão dos modelos foram suficientes, reforçando o impacto dominante das técnicas de pré-processamento no desempenho geral.

Em síntese, esta análise demonstra que a escolha do modelo e das técnicas de engenharia de características deve ser orientada pela natureza dos dados e pelo objetivo da aplicação. Modelos como *Decision Tree* e *Random Forest* se destacaram como soluções robustas e consistentes, enquanto técnicas como *Standard Scaling* e *Min-Max Scaling* mostraram-se eficazes em diferentes cenários.

6 CONSIDERAÇÕES FINAIS

Neste capítulo são apresentadas as considerações finais deste trabalho, consolidando os principais destaques obtidos e realçando suas implicações práticas e teóricas. Além disso, serão apresentadas propostas para a continuação do trabalho, identificando áreas para pesquisa adicional, potenciais melhorias metodológicas e possíveis aplicações práticas dos resultados alcançados.

6.1 Discussão Sobre os Resultados

Este trabalho propõe um esquema para a detecção de ataques DDoS na camada de aplicação utilizando as seguintes técnicas de aprendizado de máquina: *Naive Bayes*, *Decision Tree*, *Logistic Regression* e *Random Forest*. Além disso, apresenta uma análise de desempenho incluindo uma solução de *big data*. Busca-se analisar padrões de comportamento dos ataques na camada de aplicação, com o objetivo de desenvolver um esquema de detecção eficiente.

Os resultados indicam que o *Naive Bayes* apresentou os menores tempos médios de execução, porém com pior acurácia no geral; enquanto o *Random Forest* teve os maiores tempos, mas com grande precisão. O *Decision Tree* e o *Random Forest* destacaram-se pela alta acurácia em ambos os ambientes testados, sendo consistentes na classificação dos dados. No entanto, em um ambiente de programação geral (APG), observou-se um impacto significativo na acurácia com o uso da técnica de *Feature Importance (FI)*, especialmente para os modelos *Decision Tree* e *Logistic Regression*. No ambiente ABD, a técnica de FI apresentou ganhos mais modestos, com uma leve melhora no tempo de execução e manutenção da acurácia.

Adicionalmente, verificou-se que técnicas de pré-processamento, como *Standard Scaling* e *Min-Max Scaling*, contribuíram para estabilizar os resultados e maximizar a acurácia em diversos cenários. No ambiente APG, o *Random Forest* alcançou até 99,67% de acurácia nessas condições, enquanto o *Decision Tree* demonstrou robustez ao atingir a acurácia máxima de 100% no ambiente ABD, independentemente das combinações avaliadas. Por outro lado, técnicas como *PCA* e *Polynomial Features* apresentaram resultados mais variáveis, refletindo maior dependência do modelo em uso.

Conclui-se que, no geral, os algoritmos de aprendizado de máquina são eficientes em detectar ataques DDoS na camada de aplicação a partir do registro de utilização da rede. O *Decision Tree* e o *Random Forest* destacaram-se como os classificadores mais consistentes e precisos, apresentando melhorias significativas em termos de tempos de execução e acurácia quando avaliados em um ambiente mais robusto. O *Naive Bayes*, embora rápido, apresentou limitações em ambientes com grandes volumes de dados. Esses resultados reforçam que um ambiente computacional mais robusto pode otimizar significativamente a performance dos algoritmos

de *machine learning*, melhorando tanto a eficiência quanto a precisão na detecção de ataques DDoS. Nesse contexto, entende-se que esta identificação fornece informações significativas para a mitigação desses ataques, contribuindo para a segurança cibernética de infraestruturas críticas na web. A premissa do esquema proposto consiste em utilizar características do tráfego de redes que ainda não foram amplamente exploradas por outros estudos, conforme discutido na Seção 3.

Em contraste, o esquema proposto explora características como, por exemplo, *Flow_IAT _Max*, que captura intervalos máximos entre pacotes dentro de um fluxo, *SYN_Flag_Count*, que indica o número de pacotes SYN enviados, e *Flow_Bytes_Sec*, que mede a taxa de transferência de dados por segundo. Essas métricas permitem diferenciar ataques que utilizam padrões temporais específicos, como o *Slowloris*, e ataques que sobrecarregam o servidor com rajadas de tráfego, como o *Hulk*. Dessa forma, o modelo proposto preenche lacunas deixadas por abordagens anteriores, contribuindo para uma detecção mais eficaz de ataques na camada de aplicação.

6.2 Propostas para Continuação da Pesquisa

Para dar continuidade à pesquisa e aprimorar ainda mais as estratégias de detecção de ataques DDoS na camada de aplicação, sugerimos as propostas descritas a seguir.

1. **Aplicação em Outros *Datasets*:** Analisar a utilização do esquema proposto a partir de outros *datasets*, como forma de garantir a validade e a confiabilidade dos resultados. Essa abordagem ajuda a garantir que o modelo testado seja não apenas eficaz em dados conhecidos, mas também confiável para dados futuros.

2. **Simulação de Execução em Tempo Real:** Desenvolver um cenário de simulação a partir de uma carga sintética baseada no *dataset* utilizado, com o objetivo de validar a capacidade do esquema proposto em detectar e barrar ataques DDoS em tempo de execução.

Essas propostas visam aprimorar a eficácia e a adaptabilidade dos sistemas de detecção de ataques DDoS, garantindo uma resposta rápida e eficaz diante de ameaças emergentes. Além disso, entende-se que a aplicação dos modelos em diferentes *datasets* e a validação em tempo real poderão contribuir para o avanço da pesquisa na área de segurança cibernética.

REFERÊNCIAS BIBLIOGRÁFICAS

AAMIR, M.; ZAIDI, S. M. A. Ddos attack detection with feature engineering and machine learning: the framework and performance evaluation. *International Journal of Information Security*, Springer, v. 18, p. 761–785, 2019. Citado na página 65.

ABDELAZIZ, M. T. et al. Enhancing network threat detection with random forest-based nids and permutation feature importance. *Journal of Network and Systems Management*, Springer, v. 33, n. 1, p. 2, 2025. Citado na página 61.

AL-JANABI, S. T. F.; SAEED, H. A. A neural network based anomaly intrusion detection system. In: IEEE. *2011 Developments in E-systems Engineering*. [S.l.], 2011. p. 221–226. Citado 2 vezes nas páginas 13 e 20.

ALDUAILIJ, M. et al. Machine-learning-based ddos attack detection using mutual information and random forest feature importance method. *Symmetry*, MDPI, v. 14, n. 6, p. 1095, 2022. Citado na página 59.

ALKASASSBEH, M. A novel hybrid method for network anomaly detection based on traffic prediction and change point detection. *arXiv preprint arXiv:1801.05309*, 2018. Citado 2 vezes nas páginas 13 e 20.

ASAD, M. et al. Deepdetect: detection of distributed denial of service attacks using deep learning. *The Computer Journal*, Oxford University Press, v. 63, n. 7, p. 983–994, 2020. Citado 2 vezes nas páginas 19 e 28.

AWAN, M. et al. *Real-Time DDoS Attack Detection System using Big Data Approach. Sustainability 2021, 131, 10743*. [S.l.]: s Note: MDPI stays neu-tral with regard to jurisdictional claims in . . . , 2021. Citado 4 vezes nas páginas 20, 36, 37 e 52.

BERRAR, D. Bayes' theorem and naive bayes classifier. Elsevier, 2019. Citado na página 21.

CHALE, M. et al. Constrained optimization based adversarial example generation for transfer attacks in network intrusion detection systems. *Optimization Letters*, Springer, p. 1–20, 2023. Citado na página 28.

CHAUDHARY, A.; SHRIMAL, G. Intrusion detection system based on genetic algorithm for detection of distribution denial of service attacks in manets. In: *Proc. of Int. Conf. on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India*. [S.l.: s.n.], 2019. Citado 2 vezes nas páginas 13 e 20.

CHAUDHARY, S.; MISHRA, P. K. Ddos attacks in industrial iot: A survey. *Computer Networks*, Elsevier, p. 110015, 2023. Citado na página 15.

CHAVAN, N. et al. DDoS attack detection and botnet prevention using machine learning. In: IEEE. *2022 8th Int. Conf. on Advanced Computing and Communication Systems (ICACCS)*. [S.l.], 2022. v. 1, p. 1159–1163. Citado na página 28.

CHEN, Y. et al. Ddos attack detection based on random forest. In: IEEE. *2020 IEEE international conference on progress in informatics and computing (PIC)*. [S.l.], 2020. p. 328–334. Citado na página 25.

Cloudflare Blog. *DDoS Attack Trends for 2022 Q1*. 2022. https://blog.cloudflare.com/ddos-attack-trends-for-2022-q1/. Acessado em 14/11/23. Citado 2 vezes nas páginas 12 e 15.

Cloudflare Blog. *DDoS Attack Trends for 2022 Q2*. 2022. https://blog.cloudflare.com/ddos-attack-trends-for-2022-q2/. Acessado em 14/11/23. Citado 2 vezes nas páginas 12 e 15.

FOUNDATION, T. A. S. *StandardScaler*. 2024. <https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.StandardScaler.html>. Accessed: 2024-08-03. Citado na página 54.

GIRMA, A. et al. Analysis of ddos attacks and an introduction of a hybrid statistical model to detect ddos attacks on cloud computing environment. In: IEEE. *2015 12th International Conference on Information Technology-New Generations*. [S.l.], 2015. p. 212–217. Citado na página 12.

GONG, C. et al. An improved quantum genetic algorithms and application for ddos attack detection. In: IEEE. *2019 IEEE Int. Conf. on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking*. [S.l.], 2019. p. 427–434. Citado 2 vezes nas páginas 13 e 20.

Google Cloud Blog. *Google Cloud Mitigated Largest DDoS Attack Peaking Above 398 Million RPS*. 2023. https://cloud.google.com/blog/products/identity-security/google-cloud-mitigated-largest-ddos-attack-peaking-above-398-million-rps. Acessado em 14/11/23. Citado 2 vezes nas páginas 12 e 15.

HACHEM, N. et al. Botnets: lifecycle and taxonomy. In: IEEE. *2011 Conference on Network and Information Systems Security*. [S.l.], 2011. p. 1–8. Citado 2 vezes nas páginas 12 e 18.

HADI, H. J. et al. Developing Realistic Distributed Denial of Service (DDoS) Dataset for Machine Learning-based Intrusion Detection System. In: IEEE. *9th Int. Conf. on Internet of Things: Systems, Management and Security (IOTSMS)*. [S.l.], 2022. p. 1–6. Citado na página 28.

Imperva Blog. *81% Increase in Large Volume DDoS Attacks*. 2022. https://www.imperva.com/blog/81-increase-in-large-volume-ddos-attacks/. Acessado em 14/11/23. Citado 2 vezes nas páginas 12 e 15.

KEBEDE, S. D. et al. Predictive machine learning-based integrated approach for ddos detection and prevention. *Multimedia Tools and Applications*, Springer, v. 81, n. 3, p. 4185–4211, 2022. Citado 2 vezes nas páginas 13 e 20.

KIM, Y.-E.; KIM, Y.-S.; KIM, H. Effective feature selection methods to detect iot ddos attack in 5g core network. *Sensors*, MDPI, v. 22, n. 10, p. 3819, 2022. Citado na página 59.

KUMAR, R. et al. Blockchain-based authentication and explainable ai for securing consumer iot applications. *IEEE Transactions on Consumer Electronics*, IEEE, 2023. Citado na página 28.

KUMAR, V.; SHARMA, H. et al. Detection and analysis of ddos attack at application layer using naive bayes classifier. *Journal of Computer Engineering & Technology*, v. 9, n. 3, p. 208–217, 2018. Citado 3 vezes nas páginas 29, 30 e 34.

LAKSHMINARASIMMAN, S.; RUSWIN, S.; SUNDARAKANTHAM, K. Detecting ddos attacks using decision tree algorithm. In: IEEE. *2017 fourth international conference on signal processing, communication and networking (ICSCN)*. [S.l.], 2017. p. 1–6. Citado na página 22.

LIU, Z. et al. A ddos detection method based on feature engineering and machine learning in software-defined networks. *Sensors*, MDPI, v. 23, n. 13, p. 6176, 2023. Citado na página 66.

MALIK, M.; DUTTA, M. et al. Feature engineering and machine learning framework for ddos attack detection in the standardized internet of things. *IEEE Internet of Things Journal*, IEEE, v. 10, n. 10, p. 8658–8669, 2023. Citado na página 65.

NAGARAJA, A.; BOREGOWDA, U.; VANGIPURAM, R. Study of detection of ddos attacks in cloud environment using regression analysis. In: *International Conference on Data Science, E-learning and Information Systems 2021*. [S.l.: s.n.], 2021. p. 166–172. Citado na página 24.

NGUYEN, H.-V.; CHOI, Y. Proactive detection of ddos attacks utilizing k-nn classifier in an anti-ddos framework. *International Journal of Computer and Information Engineering*, Citeseer, v. 4, n. 3, p. 537–542, 2010. Citado 2 vezes nas páginas 13 e 20.

PATIL, A. et al. Software defined network: Ddos attack detection. *International Research Journal of Engineering and Technology*, v. 6, p. 7302–7307, 2019. Citado na página 54.

PEDREGOSA, F. et al. *StandardScaler*. 2011. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>. Accessed: 2024-08-03. Citado na página 54.

PRASEED, A.; THILAGAM, P. S. Ddos attacks at the application layer: Challenges and research perspectives for safeguarding web applications. *IEEE Communications Surveys & Tutorials*, IEEE, v. 21, n. 1, p. 661–685, 2018. Citado 2 vezes nas páginas 12 e 16.

PRASEED, A.; THILAGAM, P. S. Modelling behavioural dynamics for asymmetric application layer ddos detection. *IEEE Transactions on Information Forensics and Security*, v. 16, p. 617–626, 2021. Citado 2 vezes nas páginas 29 e 34.

QIN, X.; XU, T.; WANG, C. Ddos attack detection using flow entropy and clustering technique. In: IEEE. *2015 11th International Conference on Computational Intelligence and Security (CIS)*. [S.l.], 2015. p. 412–415. Citado na página 12.

RAHAL, B. M.; SANTOS, A.; NOGUEIRA, M. A distributed architecture for ddos prediction and bot detection. *IEEE Access*, v. 8, p. 159756–159772, 2020. Citado 3 vezes nas páginas 31, 32 e 34.

RAJ, R.; KANG, S. S. Mitigating ddos attack using machine learning approach in sdn. In: IEEE. *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*. [S.l.], 2022. p. 462–467. Citado 2 vezes nas páginas 32 e 34.

SANCHEZ, O. R. et al. Feature selection evaluation towards a lightweight deep learning DDoS detector. In: IEEE. *ICC 2021-IEEE International Conference on Communications*. [S.l.], 2021. p. 1–6. Citado 2 vezes nas páginas 19 e 28.

SANMORINO, A.; GUSTRIANSYAH, R.; ALIE, J. Ddos attacks detection method using feature importance and support vector machine. *JUITA: Jurnal Informatika*, v. 10, n. 2, p. 167–171, 2022. Citado na página 59.

SANMORINO, A.; YAZID, S. Ddos attack detection method and mitigation using pattern of the flow. In: IEEE. *2013 International conference of Information and communication technology (ICoICT)*. [S.l.], 2013. p. 12–16. Citado na página 12.

SARRAF, S. et al. Analysis and detection of ddos attacks using machine learning techniques. *Am. Sci. Res. J. Eng. Technol. Sci*, v. 66, n. 1, p. 95–104, 2020. Citado na página 42.

SHAFI, M. et al. Toward generating a new cloud-based distributed denial of service (ddos) dataset and cloud intrusion traffic characterization. *Information*, MDPI, v. 15, n. 4, p. 195, 2024. Citado na página 17.

SILVA, D. H. et al. Big data analytics for critical information classification in online social networks using classifier chains. *Peer-to-Peer Networking and Applications*, Springer, v. 15, n. 1, p. 626–641, 2022. Citado na página 54.

SINGH, V. P.; KUMARI, R.; KAUR, M. Machine learning for intrusion detection system in iot environment with permutation importance. 2024. Citado na página 61.

TOUTSOP, O.; DAS, S.; KORNEGAY, K. Exploring The Security Issues in Home-Based IoT Devices Through Denial of Service Attacks. In: IEEE. *2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation*. [S.l.], 2021. p. 407–415. Citado 2 vezes nas páginas 12 e 18.

UDDIN, R.; KUMAR, S. A.; CHAMOLA, V. Denial of service attacks in edge computing layers: Taxonomy, vulnerabilities, threats and solutions. *Ad Hoc Networks*, Elsevier, v. 152, p. 103322, 2024. Citado 2 vezes nas páginas 19 e 28.

WANG, A.; MOHAISEN, A.; CHEN, S. An adversary-centric behavior modeling of ddos attacks. In: IEEE. *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. [S.l.], 2017. p. 1126–1136. Citado na página 17.

YADAV, S.; SELVAKUMAR, S. Detection of application layer ddos attack by modeling user behavior using logistic regression. In: IEEE. *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions)*. [S.l.], 2015. p. 1–6. Citado 3 vezes nas páginas 30, 31 e 34.

ZAIB, H. *NSL-KDD - Network Security, Information Security, Cyber Security*. 2019. https://www.kaggle.com/datasets/hassan06/nslkdd. Acessado em 27/04/24. Citado na página 28.

ZHAO, Q. et al. Machine-learning based tcp security action prediction. In: IEEE. *2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*. [S.l.], 2020. p. 1329–1333. Citado na página 61.

ZHENG, A.; CASARI, A. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2018. ISBN 1491953241. Citado na página 66.

ZHOU, L. et al. A feature selection-based method for DDoS attack flow classification. *Future Generation Computer Systems*, Elsevier, v. 132, p. 67–79, 2022. Citado 2 vezes nas páginas 38 e 59.

ENTREGA DA VERSÃO FINAL DE DISSERTAÇÃO

Eu, PROF. DR. PAULO DITARSO MACIEL JR., autorizo o aluno(a) DIGENALDO DE BRITO RANGEL NETO a entregar a versão final da dissertação de mestrado, à secretaria do PPGTI, que foi por mim analisada e está de acordo com os apontamentos feitos pelos membros da banca de apresentação do referido aluno.

gov.br
Documento assinado digitalmente
PAULO DITARSO MACIEL JUNIOR
Data: 25/02/2025 09:31:07-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Paulo Ditarso Maciel Jr.
Orientador

João Pessoa, 25 de Fevereiro de 2025.