Framework de Detecção de Ataques DDoS na Camada de Aplicação com Uso de *Machine Learning* e *Big Data*

Digenaldo de Brito Rangel Neto, Paulo Ditarso Maciel Jr.

[1] Unidade Acadêmica de Informação e Comunicação
Programa de Pós-Graduação em Tecnologia da Informação (PPGTI)
Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB)
João Pessoa – PB – Brasil

digenaldo.neto@academico.ifpb.edu.br, paulo.maciel@ifpb.edu.br

Abstract. *The growing threat of application-layer DDoS attacks and the continuous advancement of attacker techniques highlight the urgency of developing effective detection methods to protect networked systems. This work proposes a machine learning-based detection framework, integrated with a correlation-based Feature Importance technique and big data tool support. Four algorithms — Naive Bayes, Decision Tree, Logistic Regression, and Random Forest — were evaluated for accuracy and runtime. The results highlight the effectiveness of the approach, demonstrating significant gains in computational efficiency and predictive accuracy, particularly after the application of the variable selection technique. This robust and adaptable solution presents itself as a relevant contribution to strengthening the security of critical infrastructures in the face of an increasingly dynamic and challenging cyber landscape.*

Resumo. *A crescente ameaça de ataques DDoS na camada de aplicação e o avanço contínuo das técnicas utilizadas por atacantes ressaltam a urgência de desenvolver métodos eficazes de detecção para proteger sistemas em rede. Este trabalho propõe um framework de detecção baseado em aprendizado de máquina, integrado com técnica de Feature Importance baseada em correlação e suporte de ferramenta de big data. Quatro algoritmos — Naive Bayes, Decision Tree, Logistic Regression e Random Forest — foram avaliados em relação à acurácia e ao tempo de execução. Os resultados destacam a eficácia da abordagem, demonstrando ganhos significativos de eficiência computacional e precisão preditiva, particularmente após a aplicação da técnica de seleção de variáveis. Esta solução robusta e adaptável apresenta-se como uma contribuição relevante para fortalecer a segurança de infraestruturas críticas diante de um panorama cibernético cada vez mais dinâmico e desafiador.*

1. Introdução

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 à 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 and 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.

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, Algoritmos de Aprendizado de Máquina e Algoritmos Genéticos têm sido extensivamente explorados para uma detecção precisa desses ataques. Por exemplo, pesquisas como em [Alkasassbeh 2018] evidenciam a eficácia das redes neurais na identificação de anomalias no tráfego, ao passo que estudos conduzidos por [Kebede et al. 2022] investigam o uso de algoritmos de aprendizado de máquina para uma detecção proativa de ataques DDoS. Ainda, as pesquisas de [Chaudhary and Shrimal 2019] e [Gong et al. 2019] ressaltam o potencial dos algoritmos genéticos na adaptação dinâmica e na resposta eficaz a esses ataques em constante evolução. Portanto, a IA proporciona uma grande vantagem na identificação dos ataques, possibilitando uma resposta ágil e contribuindo para a segurança de sistemas na web.

Este trabalho propõe um *framework* para a detecção de ataques DDoS na camada de aplicação, utilizando técnicas de aprendizado de máquina, como *Naive Bayes*, *Decision Tree*, *Logistic Regression* e *Random Forest*. A metodologia inclui uma análise detalhada da importância das características utilizadas nos modelos, baseada em correlação, para seleção de variáveis relevantes que melhoram o desempenho alcançado. Essa técnica é conhecida como *Feature Importance* (FI). Adicionalmente, analisa-se uma solução de *big data* para lidar com a alta complexidade e volume dos dados. Os resultados obtidos indicam que o uso de características correlacionadas melhora significativamente tanto o tempo de execução quanto a acurácia dos modelos de aprendizado de máquina, especialmente em cenários de alta complexidade. Dessa forma, o trabalho avança na detecção de ataques DDoS e fortalece a segurança cibernética de infraestruturas críticas, oferecendo uma base sólida para sua mitigação.

O restante deste artigo está organizado como descrito a seguir. A Seção 2 apresenta os trabalhos relacionados. O *framework* proposto para a identificação dos ataques está descrito na Seção 3. A Seção 4 descreve uma avaliação de desempenho e os resul-

tados alcançados a partir da execução do *framework* em um *dataset* público. Por fim, a Seção 5 apresenta as considerações finais e perspectivas de trabalhos futuros.

2. Trabalhos Relacionados

O trabalho de [Yadav and Selvakumar 2015] propõe um método baseado em características de navegação e Aprendizado por Reforço para distinguir entre usuários legítimos e atacantes, reforçando a necessidade da detecção de ataques DDoS na camada de aplicação. Contudo, o foco está na modelagem do comportamento do usuário, sem explorar algoritmos de aprendizado de máquina específicos para identificar padrões anômalos.

O trabalho de [Kumar et al. 2018] propõe o modelo *Network Security against DDoS Attacks* (NSDA) para detectar ataques na camada de aplicação por meio da análise de padrões, como a diferença temporal entre solicitações consecutivas e discrepâncias no tamanho dos arquivos de log. A abordagem utiliza *Naive Bayes* no Weka 3.8, priorizando a identificação de padrões sem uma estratégia proativa de mitigação. Nosso foco vai além da detecção, buscando antecipar e mitigar riscos antes que os ataques se concretizem, tomando uma abordagem adaptativa para a proteção de infraestruturas críticas na web.

O trabalho de [Rahal et al. 2020] propõe uma abordagem hierárquica para a predição de ataques DDoS causados por *botnets*, operando em dois níveis: local e internet. No nível local, sinais precoces são identificados por meio de indicadores estatísticos, enquanto no nível da internet, um método híbrido baseado em *clustering* e processamento de sinais em grafos é utilizado para detectar *bots* antecipadamente. Essa abordagem se destaca pelo uso de indicadores avançados, como taxa de retorno, autocorrelação e coeficiente de variação, além da aplicação de aprendizado de máquina não supervisionado para agrupar dispositivos conforme seu comportamento. Enquanto este trabalho foca na predição de *botnets* antes que os ataques causem danos, nosso estudo prioriza a adaptação dinâmica e a exploração de características do tráfego de rede ainda pouco investigadas para melhorar a eficiência na detecção de ataques DDoS na camada de aplicação.

O trabalho de [Praseed and Thilagam 2021] propõe um modelo baseado em Autômato Probabilístico Temporal para a detecção rápida de ataques DDoS na camada de aplicação, reduzindo falsos positivos e permitindo aprendizado incremental. No entanto, ele se concentra na diferenciação entre usuários legítimos e maliciosos sem explorar profundamente a camada de aplicação. Diferentemente, nossa pesquisa não apenas investiga características ainda pouco exploradas para aprimorar a detecção adaptativa de ataques DDoS, mas também realiza uma comparação detalhada do desempenho dos algoritmos em termos de tempo de execução, acurácia e relevância das características.

O trabalho de [Raj and Kang 2022] investiga a detecção de ataques DDoS em redes definidas por software (SDN), comparando algoritmos como *Logistic Regression*, SVM, XGBoost, *Decision Tree* e KNN com base em métricas de desempenho, incluindo acurácia e *F1-score*. O foco desta análise é comparar os algoritmos para identificar a melhor abordagem na mitigação de ataques. Diferentemente, nosso estudo se concentra na detecção de ataques DDoS na camada de aplicação, enquanto este trabalho avalia principalmente o desempenho dos classificadores em uma Rede Definida por Software.

Diferentemente dos estudos elencados, realizamos uma análise comparativa mais abrangente entre quatro algoritmos distintos de aprendizado de máquina para detectar

ataques DDoS na camada de aplicação, comparando acurácia, tempo de execução e importância de características de tráfego. Adicionalmente, utilizamos uma solução de *big data* para detecção de ataques, algo não explorado pelos trabalhos citados. A Tabela 1 compara três aspectos principais dos trabalhos: o foco da pesquisa, os algoritmos utilizados e os principais resultados alcançados.

Tabela 1. Comparação dos Trabalhos sobre Detecção de Ataques DDoS

Referência	Foco do Trabalho	Algoritmos Utilizados	Resultados Principais
[Praseed and 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.
[Kumar et al. 2018]	Detecção de ataques DDoS na camada de aplicação com variáveis novas e aprendizado de máquina.	*Naive Bayes*, Weka 3.8	Foco na criação de variáveis e pré-processamento. Identificação de padrões, mas sem abordagem preventiva.
[Yadav and Selvakumar 2015]	Identificação de comportamentos de usuários e ataques com características de navegação e Aprendizado por Reforço.	Aprendizado por Reforço	Modelagem do comportamento do usuário para distinguir entre usuários regulares e atacantes.
[Rahal et al. 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. Alta precisão na identificação de bots.
[Raj and 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. Análise comparativa com foco em redes definidas por software.
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 de aprendizado de máquina em termos de tempo de execução, acurácia e importância de características baseadas em correlação.

3. *Framework* de Detecção

Esta seção apresenta o *framework* proposto, cuja metodologia consiste essencialmente em: (i) identificar e classificar os tipos de ataques; (ii) analisar e selecionar as variáveis (características do tráfego); (iii) treinar e aplicar os modelos de ML. Durante a coleta do tráfego (i), são identificados e classificados os diferentes tipos de ataques DDoS, com ênfase na camada de aplicação, como ataques de inundação HTTP, exaustão de recursos e métodos específicos, como *Slowloris* e *Hulk*. Essa atividade fornece uma base para treinar e avaliar os modelos de detecção. Em um momento seguinte (ii), são aplicadas técnicas de análise de dados para identificar as variáveis mais relevantes, utilizando a matriz de correlação e métodos de *Feature Importance*. Tais atividades eliminam redundâncias e aprimoram a eficiência computacional e a precisão dos modelos de aprendizado de máquina. Os dados são então usados para treinar os algoritmos (iii), onde cada modelo é avaliado com base em métricas de tempo de execução e acurácia.

3.1. Fases do *Framework*

O *framework* é composto por quatro fases distintas de execução conforme diagrama ilustrado na Figura 1. Na primeira fase, denominada **"Obtenção dos Dados"**, são realizadas atividades para garantir a qualidade e confiabilidade dos dados utilizados na análise. Mais

especificamente, verifica-se 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"**, uma organização e padronização dos dados é realizada como parte do processo. O objetivo é validar a consistência dos dados para uma classificação mais precisa das informações. Na fase seguinte, denominada **"Modelagem ML"**, os algoritmos de aprendizado de máquina são aplicados nos 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"**, interpreta-se os resultados obtidos pelo algoritmo, analisando o desempenho e a eficácia do *framework* na identificação de ataques DDoS.

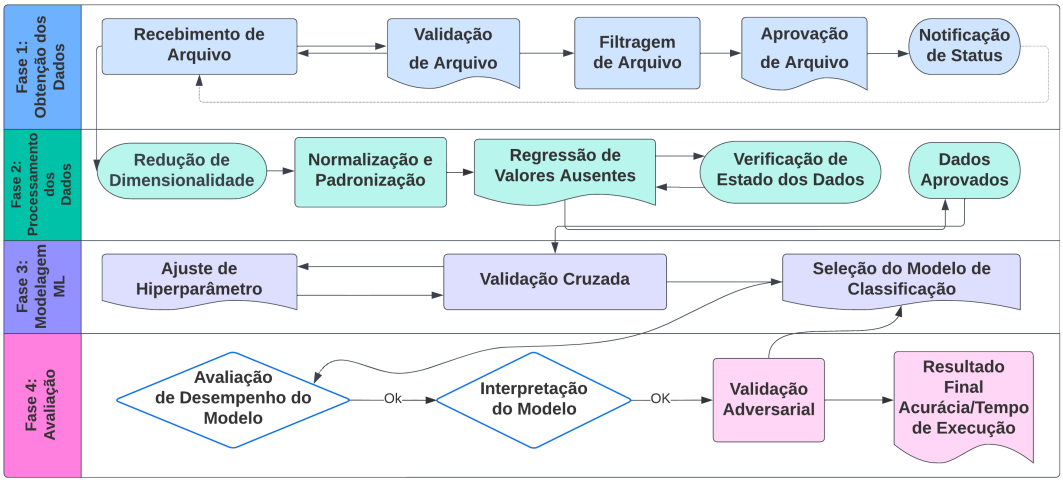


Figura 1. Diagrama do *framework* de Detecção de Ataques DDoS.

3.1.1. Obtenção dos Dados (1ª Fase)

Esta etapa concentra-se em descrever e discutir a importância do processo de análise dos dados. Uma série de atividades é realizada para assegurar que os dados utilizados na análise atendam a padrões de qualidade e confiabilidade, conforme ilustradas na Figura 1. São abordadas questões relacionadas à verificação da integridade dos dados, ao tratamento de erros ou inconsistências, e à confirmação da adequação do formato do arquivo.

Recebimento de Arquivo: A etapa inicial consiste em receber o arquivo de dados de tráfego de rede e verificar sua integridade e completude, utilizando algoritmos como *checksum* para detectar dados corrompidos.

Validação de Arquivo: Após o recebimento, o arquivo é validado para garantir sua integridade, autenticidade e conformidade com os critérios de qualidade. São utilizadas técnicas como verificação de assinatura digital e conferência de dados essenciais para a análise de ataques DDoS, incluindo IPs, *timestamps* e tipos de tráfego.

Filtragem de Arquivo: Na terceira etapa, um algoritmo filtra o arquivo para remover dados irrelevantes ou redundantes, como colunas desnecessárias e registros incompletos. Isso reduz o ruído e melhora a eficácia do aprendizado de máquina, focando nas

informações mais relevantes.

Aprovação de Arquivo: Após a filtragem, o arquivo passa por uma revisão final para verificar se atende aos critérios estabelecidos. Se aprovado, segue para análise; caso contrário, são aplicadas correções, como remoção de dados irrelevantes ou ajuste de inconsistências.

Notificação de Status: Por fim, um serviço de mensageria gera uma notificação assíncrona informando o status do processo, incluindo a aprovação ou rejeição do arquivo e as ações corretivas aplicadas, se necessário.

3.1.2. 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 processados para remover características redundantes ou irrelevantes, reduzindo a complexidade do conjunto. Métricas que não contribuem para a detecção de ataques, como certos tipos de protocolo, são eliminadas.

Normalização e Padronização: A normalização e padronização garantem que as variáveis tenham escalas comparáveis, ajustando os valores para uma escala comum e equilibrando a contribuição de cada métrica na análise.

Regressão de Valores Ausentes: Registros de tráfego podem ter dados ausentes, como pacotes por segundo ou tempo de resposta. Para corrigir isso, aplicamos imputação de dados, substituindo valores ausentes por estimativas baseadas nos dados existentes.

Verificação de Estado dos Dados: Os dados passam por uma verificação completa para garantir seu preparo para análise, checando normalização, padronização e ausência de valores faltantes ou duplicados. *Scripts* de validação identificam erros, como valores fora dos intervalos esperados ou formatos incorretos.

Dados Aprovados: Com os dados aprovados, realiza-se uma validação cruzada para avaliar o desempenho e selecionar o melhor algoritmo de classificação.

3.1.3. Modelagem ML (3ª Fase)

Os dados processados são usados para treinar algoritmos de aprendizado de máquina na detecção de padrões incomuns e classificação de ataques DDoS. Ajustes garantem um treinamento eficaz, enquanto a validação cruzada avalia o desempenho e seleciona o melhor modelo, assegurando precisão e eficiência na identificação dos ataques.

Ajuste de Hiperparâmetros: O desempenho dos algoritmos é otimizado pelo ajuste de hiperparâmetros, como taxa de aprendizado e profundidade da árvore de decisão. Técnicas como busca em grade ou aleatória testam diferentes combinações para identificar os melhores resultados na detecção de ataques DDoS.

Validação Cruzada: A validação cruzada avalia a capacidade de generalização do algoritmo, dividindo os dados em múltiplos conjuntos de treinamento e teste. Esse processo

ajuda a detectar sobreajuste e subajuste, garantindo melhor adaptação a novos dados.

Seleção do Algoritmo de Classificação: Com base na validação cruzada, seleciona-se o algoritmo de classificação mais adequado, avaliando *Naive Bayes*, *Decision Tree*, *Logistic Regression* e *Random Forest*. A escolha considera métricas como Precisão, *Recall* e *F1-score*, buscando o melhor equilíbrio entre desempenho e generalização.

3.1.4. 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 precisão das previsões dos algoritmos por meio de métricas como Precisão, *Recall* e *F1-score*. Os modelos são treinados e testados em conjuntos distintos, permitindo comparações para identificar o mais adequado ao cenário.

Interpretação do Algoritmo: A interpretação dos resultados envolve a análise da importância das características, como coeficientes em uma regressão logística ou uma árvore de decisão. Relatórios dos algoritmos são examinados para acompanhar métricas como tempo de execução, consumo de recursos e taxas de acerto ao longo do tempo.

Validação Adversarial: O algoritmo é testado quanto à robustez diante de entradas conflitantes, pequenas perturbações projetadas para induzir erros. Essa etapa analisa anomalias nas últimas entradas e determina a melhor ação a ser tomada.

Resultado Final (Acurácia/Tempo de execução): Por fim, avalia-se a acurácia e o tempo de execução dos algoritmos para validar sua viabilidade em diferentes ambientes de produção. Em sistemas de alto tráfego, por exemplo, prioriza-se a detecção rápida; em dispositivos com restrições, busca-se eficiência, mesmo com leve perda de precisão.

3.2. *Dataset*

Os dados do *dataset* [Awan et al. 2021] 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: (1^a) Benigno - Legítimo, (2^a) DDoS - *Slowloris* e (3^a) DDoS - *Hulk*.

Os registros de acesso à rede armazenados em logs contêm informações detalhadas, desde endereços IP dos clientes, até detalhes sobre 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 de informações da camada de aplicação do sistema. Essa análise permitiu a detecção de estratégias como a ocupação de conexões por meio de pequenos pacotes no *Slowloris*, enquanto o ataque *Hulk* sobrecarregou a rede com pacotes de maior tamanho e intensidade. Essas distinções baseadas nos dados foram fundamentais para o desenvolvimento de um algoritmo de inteligência artificial capaz de discernir e diferenciar esses tipos específicos de ataques DDoS, aprimorando significativamente as estratégias de detecção e prevenção.

3.3. Extração de Características do *Dataset*

A extração de características dos dados foi fundamental para identificar padrões e informações cruciais. Para uma análise detalhada do tráfego de rede, foi essencial compreender os parâmetros e métricas de funcionamento nesse ambiente dinâmico. A Tabela 2 apresenta os campos do *dataset*, oferecendo uma descrição das características consideradas na análise do tráfego. Estes campos se apresentam como uma referência fundamental para a compreensão dos detalhes e comportamento do tráfego na rede, alguns dos quais são cruciais para o esquema de identificação de DDoS proposto.

Campo(s)	Significado(s)
`Destination_Port`	Porta de destino usada na comunicação
`Flow_Duration`	Tempo em que uma sequência de pacotes permaneceu ativa
`Total_*_Packets`	Total de pacotes enviados (Fwd) ou recebidos (Backward)
`Total_Length_of_*_Packets`	Comprimento total dos pacotes enviados (Fwd) ou recebidos (Bwd)
`Fwd_Packet_Length_*`	Tamanhos do pacote enviado: Max, Min, Mean ou Std
`Bwd_Packet_Length_*`	Tamanhos do pacote recebido: Max, Min, Mean ou Std
`Flow_*_Sec`	Contagem de bytes (Bytes) ou pacotes (Packets) por segundo do fluxo
`Flow_IAT_*`	Tempo entre chegadas para o fluxo de pacotes enviados: Mean, Std, Max ou Min
`Bwd_IAT_*`	Tempo entre chegadas para o fluxo de pacotes recebidos: Total, Mean, Std, Max ou Min
`*_PSH_Flags`	Contagem de *flags* de Push para frente (Fwd) ou para trás (Bwd)
`*_URG_Flags`	Contagem de *flags* URG para frente (Fwd) ou para trás (Bwd)
`*_Header_Length`	Comprimento do cabeçalho para frente (Fwd) ou para trás (Bwd)
`*_Packets_Sec`	Pacotes por segundo para frente (Fwd) ou para trás (Bwd)
`*_Packet_Length`	Comprimento mínimo (Min) ou máximo (Max) do pacote
`Packet_Length_*`	Comprimento do pacote: Mean, Std ou Variance
`*_Flag_Count`	Contagem de *flags*: FIN, SYN, RST, PSH, ACK, URG, CWE ou ECE
`Down_Up_Ratio`	Proporção de downlink-uplink
`Average_Packet_Size`	Tamanho médio do pacote
`Avg_*_Segment_Size`	Tamanho médio do segmento encaminhado (Fwd) ou recebido (Bwd)
`Fwd_Avg_*_Bulk`	Média de bytes (Bytes) ou pacotes (Packets) bulk para frente
`*_Avg_Bulk_Rate`	Taxa média de bulk para frente (Fwd) ou para trás (Bwd)
`Bwd_Avg_Packets_Bulk`	Média de pacotes bulk para trás
`Bwd_Avg_Bulk_Rate`	Taxa média de bulk para trás
`Subflow_Fwd_*`	Número de subfluxos para frente em pacotes (Packets) ou bytes (Bytes)
`Subflow_Bwd_*`	Número de subfluxos para trás em pacotes (Packets) ou bytes (Bytes)
`Init_Win_bytes_*`	Janela inicial em bytes para frente (forward) e para trás (backward)
`act_data_pkt_fwd`	Número de pacotes de dados ativos para frente
`min_seg_size_forward`	Tamanho mínimo do segmento para frente
`Active_*`	Tempo de atividade: Mean, Std, Max ou Min
`Idle_*`	Tempo de inatividade: Mean, Std, Max ou Min
`Label`	Rótulo do tipo de tráfego: BENIGN, DoS *Slowloris* ou DoS *Hulk*

Tabela 2. Campos no *dataset* de tráfego de rede ([Awan et al. 2021]).

Cada campo do conjunto representa uma métrica específica que descreve o tráfego de rede. Por exemplo, `Destination_Port` indica a porta de destino utilizada, que é crucial para identificar potenciais vulnerabilidades ou padrões incomuns de comunicação. Campos como `Total_Fwd_Packets` e `Total_Backward_Packets` indicam a intensidade da atividade na rede pelo número total de pacotes enviados e recebidos. Já os campos relacionados ao tamanho dos pacotes (`Total_Length_of_Fwd_Packets` e `Total_Length_of_Bwd_Packets`), como também as métricas de duração de fluxo (`Flow_Duration`), fornecem informações essenciais para identificar comportamentos anômalos, tais como a quantidade de dados transferidos e a duração das comunicações.

Métricas estatísticas, como a média (`Fwd_Packet_Length_Mean`) e o desvio padrão (`Packet_Length_Std`) dos comprimentos dos pacotes, representam a distribuição e variação desses tamanhos, auxiliando na identificação de anomalias. Cam-

pos relacionados a *flags* de controle (como **PSH_Flag_Count** e **SYN_Flag_Count**) também indicam atividades associadas a diferentes tipos de ataques, como **SYN Flood**.

A extração dessas características permite elaborar soluções de detecção de ataques mais eficazes, identificando padrões específicos e comportamentos incomuns que seriam difíceis de detectar apenas observando os dados brutos. Essas características são fundamentais para a identificação, prevenção e mitigação de ataques DDoS. Adicionalmente, a correlação das características quantifica a relação linear entre duas variáveis, oferecendo entendimento sobre como essas estão associadas entre si [Sarraf et al. 2020]. No contexto deste *dataset*, a interpretação das correlações fornece uma compreensão sobre como as variáveis independentes influenciam a variável de interesse (*Label*). Conjectura-se que o uso de um número menor de características, desde que sejam fortemente correlacionadas, melhora a eficiência dos modelos em termos de execução, garantindo uma eficácia aceitável em relação à acurácia.

4. Avaliação de Desempenho

Esta seção apresenta os resultados de uma avaliação de desempenho do *framework* proposto. O objetivo é validar a eficácia e a eficiência do *framework* em detectar ataques, considerando métricas como tempo de execução e acurácia dos modelos. As subseções a seguir detalham a metodologia de avaliação e os resultados obtidos.

4.1. Metodologia

A metodologia de experimentação é detalhada, descrevendo o ambiente de testes com duas suítes de software distintas e a seleção de *Features* a partir de uma análise de correlação das características do *dataset*.

4.1.1. Ambiente de Testes

Os testes de análise de desempenho foram conduzidos em um servidor específico, selecionado para assegurar um ambiente controlado e a reprodutibilidade na execução dos experimentos. As especificações detalhadas da máquina utilizada como ambiente de teste são: CPU de 3.5 GHz 8-Core Intel Xeon, 80GB de memória RAM 2133 MHz DDR4 e HDD com capacidade de 1 TB. Os testes foram divididos em dois ambientes distintos para comparar diferentes aspectos do desempenho por duas suítes de software distintas:

- **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.

A utilização de um ambiente controlado é fundamental para garantir a validade dos resultados, minimizando a influência de variáveis externas e permitindo uma análise precisa do impacto das intervenções propostas. Além disso, assegurar a reprodutibilidade dos testes é essencial para a verificação por outros pesquisadores, contribuindo para

a confiabilidade do estudo e possibilitando investigações futuras com diferentes abordagens. Detalhes da implementação do *framework* está disponível em repositório público[1].

4.1.2. Seleção de *Features* Correlacionadas

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, ajudando a identificar quais características são mais influentes para a modelagem preditiva. Para a obtenção dos resultados, foram utilizadas duas abordagens distintas: uma com o conjunto inteiro de características do *dataset*; e outra priorizando as variáveis com as correlações absolutas mais altas, tanto positivas quanto negativas. Essa última abordagem é comumente conhecida como uma técnica de *Feature Importance*, daqui em diante referenciada como FI nos resultados. Os valores dessa correlação podem ser vistos na Figura 2.

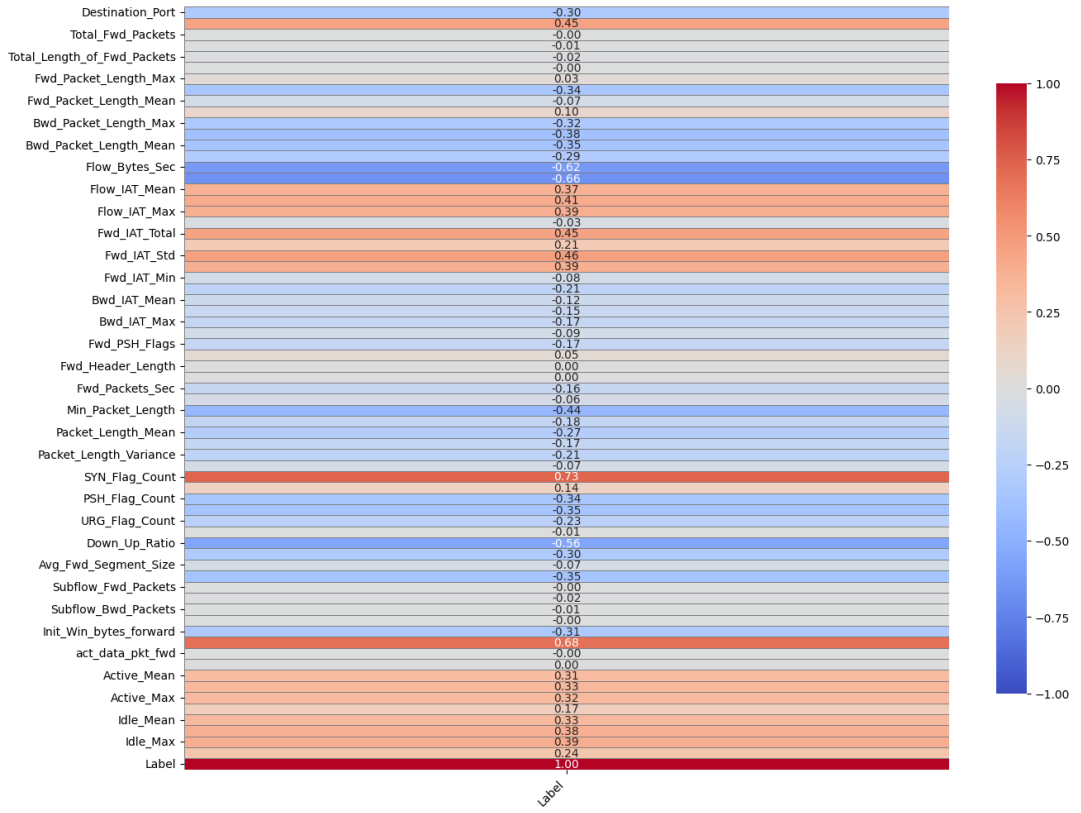


Figura 2. Matriz de correlação das *Features* do *dataset*.

Os resultados indicam que variáveis como **`SYN_Flag_Count`** e **`Init_Win_bytes_backward`** possuem correlações positivas elevadas, sugerindo forte impacto na *Label*, enquanto **`Flow_Bytes_Sec`** e **`Flow_Packets_Sec`** apresentam correlações negativas altas, demonstrando uma relação inversa. Por outro lado, variáveis com correlações próximas de zero, como **`Fwd_Header_Length`** e **`act_data_pkt_fwd`**, foram consideradas pouco relevantes para a modelagem.

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

A análise das correlações orienta a seleção de características para modelos preditivos, permitindo focar nos fatores mais influentes. Isso contribui para a melhoria da identificação e classificação de padrões, especialmente no contexto de detecção de ataques, aprimorando a eficácia do modelo. Para efeito de comparação, os experimentos foram executados ora com as 70 características disponíveis no *dataset*, ora com as 6 características que apresentaram as maiores correlações positivas ou negativas.

4.1.3. Métricas

Assim como a abordagem utilizada em [Awan et al. 2021], o objetivo desta pesquisa é quantificar a diferença do desempenho dos modelos escolhidos nas duas suítes de software consideradas. Uma delas representando um ambiente geral de programação e outra um ambiente de *big data*. As métricas de eficácia dos modelos utilizadas nesta avaliação 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 de cada algoritmo e categorizar corretamente os dados de entrada em suas respectivas classes durante o treinamento e teste.

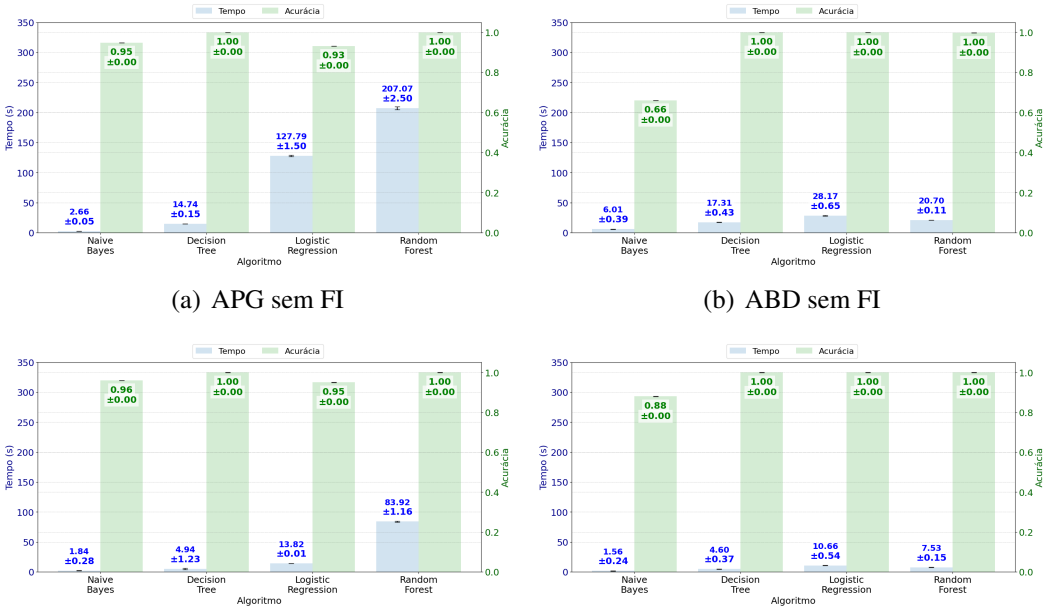
Mais relacionada com a eficiência dos modelos, a outra métrica é o *tempo de execução* (tempo para classificar os fluxos do *dataset*). Esse tempo é um fator crítico a ser considerado, especialmente quando há a intenção de utilizar um determinado modelo de ML em um ambiente de produção no futuro. Desempenho inadequado pode impactar diretamente a escalabilidade, a experiência do usuário e os custos operacionais, tornando essencial a avaliação criteriosa desse aspecto desde as fases iniciais do projeto.

4.2. Resultados

Uma análise de desempenho é realizada como parte integrante da terceira e quarta fases do *framework* proposto, a partir da comparação dos algoritmos *Naive Bayes* (NB), *Decision Tree* (DT), *Logistic Regression* (LR) e *Random Forest* (RF). A Figura 3 apresenta o desempenho dos classificadores com base nos algoritmos utilizados, tanto em um ambiente de programação geral, quanto em outro com solução de *Big Data*, avaliados em cenários com e sem a aplicação de *Feature Importance*. Cada gráfico da figura compara os resultados do tempo médio de execução (eixo Y à esquerda) e da média das acurácias (eixo Y à direita), com 30 repetições de cada cenário e nível de confiança de 95%.

No geral, considerando os resultados nos dois ambientes (APG e ABD), observa-se que a acurácia dos modelos *Decision Tree*, *Logistic Regression* e *Random Forest* atingiu 1.00 (100%) na maioria dos casos, independentemente do uso de FI. Por sua vez, o *Naive Bayes* apresentou acurácia inferior no ambiente ABD sem FI (0.66) e ABD com FI (0.88), enquanto no ambiente APG teve acurácia superior (0.95–0.96). O tempo de execução foi menor no ambiente ABD em comparação ao ambiente APG, indicando que o processamento neste último é mais eficiente.

Também conclui-se pelos resultados que a técnica de FI reduziu significativamente



(a) APG sem FI (b) ABD sem FI

(c) APG com FI (d) ABD com FI

Figura 3. Avaliação de desempenho dos modelos de classificação.

o tempo de execução em todos os modelos e ambientes. No APG sem FI, o modelo *Random Forest* apresentou o maior tempo de execução ($207.07s$), enquanto no APG com FI, esse tempo caiu para $83.92s$. De forma similar, no ABD sem FI, o *Logistic Regression* teve $28.17s$, enquanto com FI caiu para $10.66s$. O *Naive Bayes* e o *Decision Tree* foram os modelos menos afetados pelo FI, pois já apresentavam tempos relativamente baixos.

Em relação ao tempo de execução dos modelos, o *Naive Bayes* foi consistentemente o mais rápido, tanto no APG quanto no ABD. O *Decision Tree* apresentou tempos intermediários, significativamente menores do que os da *Logistic Regression* e do *Random Forest*. Este último foi o modelo mais custoso computacionalmente em todas as condições analisadas, embora tenha mantido acurácia perfeita (1.00) na maioria dos casos.

Em linhas gerais, o ambiente ABD é mais eficiente que o APG, pois reduz o tempo de execução dos modelos sem comprometer a acurácia. Assim como o uso de *Feature Importance* (FI) melhora a eficiência computacional sem impacto negativo na acurácia. O *Naive Bayes* teve a menor acurácia no ABD, o que pode limitar sua aplicabilidade. O *Random Forest* foi o mais preciso, mas com um custo computacional alto, sendo mais viável quando otimizado com FI. Portanto, a escolha do ambiente e do modelo deve levar em consideração o equilíbrio entre acurácia e eficiência computacional.

As Tabelas 3 e 4 apresentam os resultados das métricas de desempenho (Precisão, *Recall* e *F1-score*) no ambiente APG[2], considerando cenários com e sem a aplicação de Feature Importance (FI). Estas métricas foram calculadas individualmente para cada classe (0, 1 e 2), permitindo uma análise detalhada do desempenho em diferentes condições. No cenário sem a aplicação de FI, o DT e o RF apresentaram desempenho

[2]Por questões de espaço, as tabelas com os resultados do ambiente ABD foram omitidas, embora sigam o mesmo padrão evidenciado na Figura 3.

Tabela 3. APG sem FI.

Ambiente	Classificador	Precisão			Recall			F1-score		
		Classe 0	Classe 1	Classe 2	Classe 0	Classe 1	Classe 2	Classe 0	Classe 1	Classe 2
APG	Naive Bayes (NB)	0.94	0.95	0.94	0.94	0.94	0.95	0.94	0.95	0.94
APG	Decision Tree (DT)	0.99	1.00	0.99	0.99	1.00	0.99	0.99	1.00	0.99
APG	Logistic Regression (LR)	0.92	0.93	0.92	0.92	0.92	0.93	0.92	0.93	0.92
APG	Random Forest (RF)	0.99	1.00	0.99	0.99	1.00	0.99	0.99	1.00	0.99

Tabela 4. APG com FI.

Ambiente	Classificador	Precisão			Recall			F1-score		
		Classe 0	Classe 1	Classe 2	Classe 0	Classe 1	Classe 2	Classe 0	Classe 1	Classe 2
APG	Naive Bayes (NB)	0.95	0.96	0.95	0.95	0.95	0.96	0.95	0.96	0.95
APG	Decision Tree (DT)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
APG	Logistic Regression (LR)	0.95	0.95	0.94	0.95	0.94	0.94	0.95	0.95	0.94
APG	Random Forest (RF)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

semelhante, alcançando valores de Precisão, *Recall* e *F1-score* iguais ou próximos a 1.00. Este desempenho consistente reflete a capacidade desses métodos de criar divisões precisas no espaço de dados, mesmo sem otimizações específicas de características. O NB obteve métricas mais modestas, com Precisão, *Recall* e *F1-score* na faixa entre 0.94 e 0.95 para todas as classes, indicando sua limitação ao lidar com dados que não atendem completamente suas suposições de independência. O LR apresentou métricas intermediárias, com valores entre 0.92 e 0.93, evidenciando sua eficácia em identificar padrões nos dados.

No cenário com a aplicação de FI, observou-se uma melhoria geral nas métricas para todos os classificadores. Em especial, os classificadores DT e RF alcançaram valores máximos (1.00 em Precisão, *Recall* e *F1-score* para todas as classes), evidenciando que a utilização de FI contribuiu significativamente para a modelagem mais precisa dos dados. O LR também apresentou melhorias, com Precisão e *F1-score* alcançando valores de até 0.95 para a Classe 0 e ligeiras variações nas demais classes. Já o NB teve um aumento discreto nos resultados, com métricas chegando a 0.96 em algumas classes, refletindo uma leve melhora na sua capacidade preditiva com a aplicação de FI.

5. Conclusão

Os resultados obtidos indicam a eficácia e eficiência do *framework* para detecção de ataques DDoS na camada de aplicação, destacando o impacto da seleção de variáveis no desempenho dos modelos. O *Naive Bayes* se mostrou eficiente em tempo de execução, mas com desempenho preditivo limitado. O *Decision Tree* e o *Random Forest* apresentaram alta acurácia, com o último exigindo maior custo computacional. O *Logistic Regression* teve desempenho intermediário, beneficiando-se da redução de dimensionalidade. Dessa forma, este trabalho apresenta contribuições ao validar o *framework* proposto, enfatizando a importância da técnica de FI na otimização do tempo de execução e desempenho preditivo, além de direcionar a escolha de algoritmos e estratégias de pré-processamento, visando aprimorar a eficiência computacional sem comprometer a acurácia. Como contribuição, este trabalho oferece diretrizes para escolha de algoritmos e estratégias de pré-processamento em sistemas de detecção de ataques DDoS. Trabalhos futuros poderão explorar novas técnicas de seleção de características, aprendizado profundo e validação do *framework* em ambientes de produção.

Referências

[Alkasassbeh 2018] Alkasassbeh, M. (2018). A novel hybrid method for network anomaly detection based on traffic prediction and change point detection. *arXiv preprint arXiv:1801.05309*.

[Awan et al. 2021] Awan, M. et al. (2021). Real-Time DDoS Attack Detection System using Big Data Approach. Sustainability 2021, 131, 10743.

[Chaudhary and Shrimal 2019] Chaudhary, A. and Shrimal, G. (2019). 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), Jaipur-India*.

[Cloudflare Blog 2022a] Cloudflare Blog (2022a). DDoS Attack Trends for 2022 Q1. https://blog.cloudflare.com/ddos-attack-trends-for-2022-q1/. Acessado em 14/11/24.

[Cloudflare Blog 2022b] Cloudflare Blog (2022b). DDoS Attack Trends for 2022 Q2. https://blog.cloudflare.com/ddos-attack-trends-for-2022-q2/. Acessado em 14/11/24.

[Gong et al. 2019] Gong, C. et al. (2019). An improved quantum genetic algorithms and application for ddos attack detection. In *2019 IEEE Int. Conf. on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking*, pages 427–434. IEEE.

[Google Cloud Blog 2023] Google Cloud Blog (2023). Google cloud mitigated largest ddos attack peaking above 398 million rps. https://cloud.google.com/blog/products/identity-security/google-cloud-mitigated-largest-ddos-attack-peaking-above-398-million-rps. Acessado em 14/11/24.

[Imperva Blog 2022] Imperva Blog (2022). 81% Increase in Large Volume DDoS Attacks. https://www.imperva.com/blog/81-increase-in-large-volume-ddos-attacks/. Acessado em 14/11/24.

[Kebede et al. 2022] Kebede, S. D., Tiwari, B., Tiwari, V., and Chandravanshi, K. (2022). Predictive machine learning-based integrated approach for ddos detection and prevention. *Multimedia Tools and Applications*, 81(3):4185–4211.

[Kumar et al. 2018] Kumar, V., Sharma, H., et al. (2018). Detection and analysis of ddos attack at application layer using naive bayes classifier. *Journal of Computer Engineering & Technology*, 9(3):208–217.

[Praseed and Thilagam 2018] Praseed, A. and Thilagam, P. S. (2018). Ddos attacks at the application layer: Challenges and research perspectives for safeguarding web applications. *IEEE Communications Surveys & Tutorials*, 21(1):661–685.

[Praseed and Thilagam 2021] Praseed, A. and Thilagam, P. S. (2021). Modelling behavioural dynamics for asymmetric application layer ddos detection. *IEEE Transactions on Information Forensics and Security*, 16:617–626.

[Rahal et al. 2020] Rahal, B. M., Santos, A., and Nogueira, M. (2020). A distributed architecture for ddos prediction and bot detection. *IEEE Access*, 8:159756–159772.

[Raj and Kang 2022] Raj, R. and Kang, S. S. (2022). Mitigating ddos attack using machine learning approach in sdn. In *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 462–467. IEEE.

[Sarraf et al. 2020] Sarraf, S. et al. (2020). Analysis and detection of ddos attacks using machine learning techniques. *Am. Sci. Res. J. Eng. Technol. Sci*, 66(1):95–104.

[Yadav and Selvakumar 2015] Yadav, S. and Selvakumar, S. (2015). Detection of application layer ddos attack by modeling user behavior using logistic regression. In *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions)*, pages 1–6. IEEE.