Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Sistemas de Informação

Classificação de Fluxos de Rede Associados a Ataques DoS com Aprendizado de Máquina e Avaliação em Cenário Experimental

Alfredo Lucas da Silva Neto

JUIZ DE FORA
JULHO, 2026

Classificação de Fluxos de Rede Associados a Ataques DoS com Aprendizado de Máquina e Avaliação em Cenário Experimental

Alfredo Lucas da Silva Neto

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciências da Computação

Bacharelado em Sistemas de Informação

Orientador: Edelberto Franco Silva

JUIZ DE FORA

JULHO, 2026

Classificação de Fluxos de Rede Associados a Ataques DoS com Aprendizado de Máquina e Avaliação em Cenário Experimental

Alfredo Lucas da Silva Neto

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Aprovada por:

Edelberto Franco Silva
Doutor em Computação

Luciano Jerez Chaves
Doutor em Ciência da Computação

Francisco Henrique Cerdeira Ferreira
Doutor em Informática

JUIZ DE FORA

14 DE JULHO, 2026

Aos meus amigos e irmãos.

Aos pais, pelo apoio e sustento.

Resumo

Este trabalho investiga o uso de modelos de aprendizado de máquina para a classificação de fluxos de rede associados a ataques de negação de serviço (DoS). Para isso, utiliza-se a base CICIDS2017, considerando apenas o tráfego benigno e as variações de DoS presentes no conjunto de dados. A metodologia inclui etapas de pré-processamento, remoção e seleção de atributos, divisão estratificada dos dados, avaliação do impacto do balanceamento por *undersampling* e comparação entre os modelos *Random Forest* e *XGBoost*. Além da avaliação *offline*, foi desenvolvida uma ferramenta de monitoramento e classificação contínua, responsável por capturar pacotes, extrair fluxos com o CICFlowMeter, aplicar o modelo treinado e emitir alertas para fluxos classificados como maliciosos. Os resultados na base de teste indicaram alto desempenho, com valores, por exemplo, de $F1 > 0,9921$, levando à seleção do *Random Forest* treinado com a distribuição original dos dados e ponderação automática das classes. Entretanto, a avaliação em ambiente controlado evidenciou uma queda significativa na detecção de ataques gerados fora da base original, associada a diferenças nas distribuições dos atributos mais relevantes. Assim, o trabalho demonstra o potencial da abordagem, mas também ressalta limitações quanto à generalização em cenários práticos.

Palavras-chave: Detecção de intrusão, aprendizado de máquina, DoS, CICIDS2017.

Abstract

This work investigates the use of machine learning models for the classification of network flows associated with denial-of-service (DoS) attacks. To this end, the CICIDS2017 dataset is used, considering only the benign traffic and the DoS variations it contains. The methodology includes preprocessing steps, attribute removal and selection, stratified data splitting, evaluation of the impact of *undersampling*-based balancing, and a comparison between the *Random Forest* and *XGBoost* models. In addition to the *offline* evaluation, a continuous monitoring and classification tool was developed to capture packets, extract flows with CICFlowMeter, apply the trained model, and issue alerts for flows classified as malicious. The results on the test set indicated high performance, with, for example, $F1 > 0.9921$, leading to the selection of *Random Forest* trained on the original class distribution with automatic class weighting. However, the evaluation in a controlled environment revealed a significant drop in the detection of attacks generated outside the original dataset, associated with differences in the distributions of the most relevant attributes. Thus, this work demonstrates the approach's potential but also highlights limitations in generalization to practical scenarios.

Keywords: Intrusion detection, machine learning, DoS, CICIDS2017.

Agradecimentos

Agradeço aos meus pais, pela oportunidade de ter acesso a uma boa educação, pelo apoio constante e por todo o incentivo ao longo da minha trajetória.

Agradeço ao Laboratório de Aplicações e Inovação em Computação (LAPIC) e aos seus membros, que fizeram parte de grande parte dos meus dias durante a faculdade e contribuíram para minha formação acadêmica e pessoal.

Agradeço ao professor Edelberto e ao professor Jairo, que foram importantes para minha evolução ao longo da graduação, tanto pelo conhecimento compartilhado quanto pelo apoio durante esse período.

Agradeço também aos meus amigos com os quais mantive contato de forma online, pelos momentos de descontração, diversão e companhia, que foram importantes para tornar essa caminhada mais leve.

"Audentes fortuna iuvat."

Virgílio, Eneida, X, 284

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
IDS	Sistema de Detecção de Intrusão
HIDS	Sistema de Detecção de Intrusão baseado em Host
NIDS	Sistema de Detecção de Intrusão baseado em Rede
DoS	Negação de Serviço
DDoS	Negação de Serviço Distribuída
ML	Aprendizado de Máquina
RF	Random Forest
XGBoost	Extreme Gradient Boosting
PSI	Population Stability Index
PCAP	Packet Capture
CSV	Comma-Separated Values
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
FTP	File Transfer Protocol
SSH	Secure Shell
MLP	Multilayer Perceptron
DNN	Deep Neural Network
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
SMOTE	Synthetic Minority Over-sampling Technique

1 Introdução

Nos últimos anos, o crescimento da conectividade e da dependência de sistemas computacionais tornou a segurança de redes um tema cada vez mais relevante. Serviços digitais, aplicações corporativas, sistemas em nuvem e dispositivos conectados dependem continuamente da troca de dados por meio de redes de computadores. Nesse contexto, falhas de segurança podem comprometer a disponibilidade, a integridade e a confidencialidade das informações, causando impactos técnicos, financeiros e operacionais. Assim, mecanismos capazes de monitorar o tráfego de rede e identificar comportamentos maliciosos tornam-se fundamentais para a proteção de ambientes computacionais.

Tradicionalmente, muitos mecanismos de detecção de intrusão utilizam regras ou assinaturas previamente definidas para identificar atividades maliciosas (LIAO et al., 2013). Embora essa abordagem seja eficiente para ameaças conhecidas, ela pode apresentar limitações quando o tráfego malicioso não corresponde exatamente a padrões previamente cadastrados (KALITA; BHUYAN; BHATTACHARYYA, 2011; LIAO et al., 2013). Como alternativa, técnicas baseadas em aprendizado de máquina podem ser utilizadas para analisar características do tráfego e identificar padrões de comportamento associados a atividades suspeitas (BUCZAK; GUVEN, 2016). Neste trabalho, essa abordagem é investigada no contexto de ataques de negação de serviço, conhecidos como DoS (*Denial of Service*), escolhidos como recorte experimental por afetarem diretamente a disponibilidade dos serviços e por possuírem diferentes variações representadas na base de dados utilizada.

Este trabalho investiga o uso de modelos de aprendizado de máquina para classificar fluxos de rede associados a ataques DoS a partir de características extraídas do tráfego. Para isso, utiliza-se a base CICIDS2017, restringindo o escopo da análise ao tráfego benigno e às variações de DoS presentes no conjunto de dados (SHARAFALDIN; Habibi Lashkari; GHORBANI, 2018). A abordagem proposta combina três frentes complementares: a avaliação experimental de modelos de classificação sobre a base de dados, o desenvolvimento de uma ferramenta de monitoramento e classificação contínua

e a execução de testes em um ambiente controlado com tráfego gerado fora da base original. Com isso, busca-se analisar não apenas o desempenho dos modelos em uma avaliação *offline*, mas também seu comportamento quando integrados a uma ferramenta aplicada ao tráfego observado em rede.

Embora muitos trabalhos na área de detecção de intrusão com aprendizado de máquina concentrem-se no treinamento e na validação de modelos sobre bases de dados previamente estruturadas (NETO; GOMES, 2019; AKGUN; HIZAL; CAVUSOGLU, 2022), este trabalho busca aproximar essa avaliação de um cenário prático de uso. O diferencial da proposta está em integrar o modelo selecionado a uma ferramenta de monitoramento e, posteriormente, avaliá-lo em um ambiente controlado composto por máquinas virtuais, no qual ataques DoS são executados contra um serviço monitorado. Essa etapa permite observar possíveis diferenças entre o desempenho obtido na base de teste e o comportamento do classificador diante de tráfego gerado em um ambiente distinto daquele utilizado no treinamento.

Diante desse contexto, este trabalho busca responder à seguinte questão de pesquisa: em que medida modelos de aprendizado de máquina treinados com a base CICIDS2017 são capazes de generalizar a classificação de fluxos associados a ataques DoS quando aplicados a tráfego coletado em um ambiente distinto daquele utilizado no treinamento?

Assim, o objetivo geral deste trabalho é investigar o uso de modelos de aprendizado de máquina para a classificação de fluxos de rede associados a ataques de negação de serviço, considerando tanto o desempenho obtido em uma base pública quanto o comportamento do modelo quando integrado a uma ferramenta de monitoramento em ambiente controlado.

Para alcançar esse objetivo geral, são definidos os seguintes objetivos específicos:

- realizar o pré-processamento da base CICIDS2017, restringindo o escopo ao tráfego benigno e às variações de ataques DoS;
- treinar e avaliar modelos de aprendizado de máquina para classificação multiclasse de fluxos de rede;

- comparar o desempenho dos modelos Random Forest e XGBoost, considerando a distribuição original dos dados e a aplicação de undersampling na proporção 10:1;
- desenvolver uma ferramenta de monitoramento e classificação contínua de fluxos de rede;
- avaliar a ferramenta em um ambiente controlado, com tráfego benigno e ataques DoS gerados fora da base original;
- analisar as diferenças entre o desempenho *offline* do modelo e seu comportamento em um cenário prático de uso.

O restante deste trabalho está organizado da seguinte forma. O Capítulo 2 apresenta a fundamentação teórica relacionada a sistemas de detecção de intrusão, ataques de negação de serviço, fluxos de rede e aprendizado de máquina aplicado à segurança de redes. O Capítulo 3 discute trabalhos relacionados à detecção de intrusão com aprendizado de máquina. O Capítulo 4 descreve a metodologia adotada, incluindo a base de dados utilizada, as etapas de pré-processamento, o treinamento e a avaliação dos modelos, a arquitetura da ferramenta de monitoramento e o ambiente controlado utilizado para testá-la. O Capítulo 5 apresenta os resultados obtidos, incluindo a avaliação *offline* dos modelos e a análise do comportamento da ferramenta no cenário com tráfego observado. Por fim, o Capítulo 6 apresenta as considerações finais, destacando as contribuições, limitações e possibilidades de trabalhos futuros.

2 Fundamentação Teórica

2.1 Sistemas de Detecção de Intrusão

Sistemas de Detecção de Intrusão, ou IDS (*Intrusion Detection Systems*), são mecanismos utilizados para monitorar eventos em sistemas computacionais ou tráfego em redes com o objetivo de identificar atividades suspeitas, indevidas ou potencialmente maliciosas (SCARFONE; MELL, 2007; LIAO et al., 2013). Esses sistemas analisam informações coletadas do ambiente monitorado e buscam indícios de comportamentos que possam representar violações de políticas de segurança, tentativas de exploração ou uso não autorizado de recursos computacionais.

Os IDS podem ser classificados de acordo com o tipo de ambiente ou fonte de dados monitorada. Sistemas baseados em host, conhecidos como HIDS (*Host-based Intrusion Detection Systems*), analisam informações relacionadas a uma máquina específica, como registros de sistema, chamadas de sistema, arquivos e eventos locais. Já os sistemas baseados em rede, conhecidos como NIDS (*Network-based Intrusion Detection Systems*), monitoram o tráfego que circula em uma rede ou segmento de rede, analisando pacotes, conexões ou fluxos de comunicação (SCARFONE; MELL, 2007; AXELSSON, 2000). Como este trabalho está relacionado à captura e análise de tráfego de rede, sua proposta se aproxima da abordagem adotada por sistemas de detecção de intrusão baseados em rede.

Em uma arquitetura de segurança, um IDS atua como um mecanismo de monitoramento e apoio à identificação de incidentes. Diferentemente de mecanismos de prevenção, sua função principal não é necessariamente bloquear uma ação maliciosa, mas detectar comportamentos suspeitos, registrar eventos relevantes e gerar alertas que possam auxiliar na resposta a incidentes (SCARFONE; MELL, 2007). Dessa forma, IDS são normalmente utilizados como parte de uma estratégia mais ampla de segurança, complementando outros mecanismos, como firewalls, controles de acesso, sistemas de prevenção de intrusão e ferramentas de análise de logs.

Neste trabalho, o conceito de IDS é utilizado como base para o desenvolvimento de uma ferramenta voltada à análise de tráfego de rede. A proposta concentra-se na captura de pacotes, na extração de fluxos e na classificação desses fluxos por meio de um modelo de aprendizado de máquina. Assim, embora a ferramenta desenvolvida não tenha como objetivo substituir soluções completas de IDS, ela explora um dos princípios fundamentais desses sistemas: o monitoramento do tráfego com o objetivo de identificar indícios de atividade maliciosa. No escopo deste estudo, essa análise é direcionada à identificação de fluxos associados a ataques de negação de serviço.

2.1.1 Detecção Baseada em Assinaturas e em Comportamento

Uma das abordagens tradicionais utilizadas por sistemas de detecção de intrusão é a detecção baseada em assinaturas. Nessa abordagem, o tráfego ou os eventos monitorados são comparados com um conjunto de regras, padrões ou assinaturas previamente conhecidas, que representam comportamentos associados a ataques ou atividades indevidas (SCARFONE; MELL, 2007; LIAO et al., 2013). Assim, quando uma sequência de eventos ou uma característica do tráfego corresponde a uma assinatura cadastrada, o sistema pode registrar a ocorrência e gerar um alerta.

A principal vantagem dessa abordagem está em sua capacidade de identificar ameaças conhecidas com boa precisão, desde que as assinaturas estejam corretamente definidas e atualizadas. Como os padrões procurados são explicitamente descritos, sistemas baseados em assinaturas tendem a apresentar bom desempenho na identificação de ataques já catalogados e bem compreendidos (SCARFONE; MELL, 2007; LIAO et al., 2013). Por esse motivo, esse tipo de mecanismo é amplamente utilizado em soluções de segurança, especialmente em cenários nos quais há bases de regras mantidas e atualizadas continuamente.

Apesar dessas vantagens, a detecção baseada em assinaturas apresenta limitações importantes. Como depende de padrões previamente cadastrados, essa abordagem pode ter dificuldade para identificar ataques novos, variações de ataques existentes ou comportamentos maliciosos que não correspondam exatamente às regras definidas (SCARFONE; MELL, 2007; LIAO et al., 2013). Dessa forma, alterações na forma de execução de um

ataque ou a ausência de uma assinatura específica podem reduzir a capacidade de detecção do sistema.

Na literatura, abordagens que não dependem exclusivamente de assinaturas são frequentemente discutidas no contexto da detecção baseada em anomalias, pois buscam identificar desvios ou padrões suspeitos em relação ao comportamento observado no ambiente monitorado (SCARFONE; MELL, 2007; BUCZAK; GUVEN, 2016; AXELSSON, 2000; LIAO et al., 2013). Neste trabalho, utiliza-se o termo detecção baseada em comportamento para enfatizar que a análise é orientada por características observáveis do tráfego, e não apenas pela correspondência com regras previamente definidas.

Assim, em vez de procurar apenas uma assinatura específica, a detecção baseada em comportamento procura analisar padrões presentes nos dados monitorados. Em redes de computadores, essa análise pode considerar informações como volume de tráfego, duração das conexões, quantidade de pacotes, taxas de envio e recebimento, distribuição de tamanhos dos pacotes e outras características extraídas dos fluxos de rede (BUCZAK; GUVEN, 2016). Essas características permitem representar a comunicação de forma estruturada e podem ser utilizadas por modelos computacionais para identificar padrões associados a atividades suspeitas.

2.1.2 Ataques de Negação de Serviço

Ataques de negação de serviço, conhecidos como DoS (*Denial of Service*), têm como objetivo comprometer a disponibilidade de um serviço, sistema ou recurso de rede, degradando ou interrompendo seu funcionamento e dificultando o acesso por usuários legítimos.

De forma geral, ataques DoS podem afetar um ambiente por meio do consumo excessivo de recursos computacionais ou de rede. Isso pode ocorrer, por exemplo, pelo envio de grande quantidade de requisições, pelo consumo de largura de banda, pela ocupação de conexões ou pela sobrecarga de processamento do sistema alvo (CARL et al., 2006). Como consequência, o serviço pode apresentar aumento no tempo de resposta, degradação de desempenho ou interrupção de suas operações.

Uma variação desse tipo de ataque é o DDoS (*Distributed Denial of Service*). Enquanto em um ataque DoS a origem do tráfego malicioso tende a ser concentrada

em uma ou poucas fontes, em ataques DDoS o tráfego é gerado de forma distribuída, geralmente a partir de múltiplas máquinas comprometidas ou coordenadas (MIRKOVIC; REIHER, 2004). Essa característica pode aumentar o volume do ataque e dificultar sua mitigação, uma vez que o tráfego malicioso passa a ter múltiplas origens.

No contexto da detecção de intrusão, ataques DoS são relevantes por afetarem diretamente a disponibilidade dos serviços e por poderem gerar alterações observáveis no tráfego de rede. Mudanças no volume de dados, na frequência de requisições, na duração das conexões ou na quantidade de pacotes podem ser representadas por atributos extraídos dos fluxos, servindo como base para análises automatizadas de tráfego.

2.1.3 Fluxos de Rede

A análise do tráfego de rede pode ser realizada em diferentes níveis de granularidade. Em uma análise baseada em pacotes, cada pacote é considerado individualmente, com seus respectivos cabeçalhos e dados associados. Já em uma análise baseada em fluxos, os pacotes são agrupados de acordo com características comuns de uma comunicação, permitindo representar uma sequência de pacotes como uma única unidade de análise. Dessa forma, um fluxo de rede pode ser entendido como um conjunto de pacotes observados em um ponto da rede durante determinado intervalo de tempo e que compartilham propriedades em comum (AITKEN; CLAISE; TRAMMELL, 2013).

Em geral, essas propriedades estão relacionadas a informações presentes nos cabeçalhos dos pacotes, como endereço IP de origem, endereço IP de destino, porta de origem, porta de destino e protocolo de transporte. Esse conjunto de informações é frequentemente utilizado para caracterizar uma comunicação entre dois pontos da rede. Além disso, ferramentas de extração de fluxos podem representar essas comunicações de forma bidirecional, separando atributos referentes ao sentido de ida e ao sentido de retorno da comunicação (AITKEN; CLAISE; TRAMMELL, 2013; Habibi Lashkari et al., 2017; DRAPER-GIL et al., 2016).

Em abordagens de detecção de intrusão baseadas em aprendizado de máquina, os modelos normalmente utilizam uma representação estruturada dos dados de rede. Nesse contexto, em vez de utilizar diretamente os pacotes brutos, são empregados atri-

butos extraídos dos fluxos, que resumem aspectos relevantes da comunicação observada (BUCZAK; GUVEN, 2016). Essa representação permite transformar o tráfego de rede em instâncias adequadas para tarefas de classificação, nas quais cada fluxo é descrito por um conjunto de características.

Entre os atributos que podem ser extraídos de um fluxo estão a duração da comunicação, a quantidade de pacotes, o volume de bytes trafegados, o tamanho dos pacotes, as taxas de pacotes e bytes por segundo, além de informações relacionadas às flags TCP e estatísticas calculadas separadamente para cada direção do fluxo. Esses atributos permitem descrever o comportamento da comunicação de forma agregada, servindo como entrada para métodos automatizados de análise e classificação de tráfego (Habibi Lashkari et al., 2017; DRAPER-GIL et al., 2016; BUCZAK; GUVEN, 2016).

Neste trabalho, a extração dos atributos dos fluxos é realizada com o CICFlowMeter, ferramenta utilizada para processar arquivos de captura no formato PCAP e gerar arquivos estruturados com características dos fluxos de rede. Sua utilização também contribui para manter compatibilidade entre os atributos utilizados no treinamento, provenientes da base CICIDS2017, e os atributos extraídos durante a execução da ferramenta desenvolvida.

2.2 Aprendizado de Máquina

Aprendizado de máquina é uma área da computação que estuda métodos capazes de aprender padrões a partir de dados e utilizar esses padrões para realizar tarefas como previsão, classificação ou agrupamento. Diferentemente de abordagens baseadas exclusivamente em regras definidas manualmente, modelos de aprendizado de máquina são ajustados a partir de exemplos, buscando generalizar o conhecimento aprendido para novos dados (GÉRON; RAVAGLIA, 2021).

As técnicas de aprendizado de máquina podem ser organizadas em diferentes categorias, entre elas o aprendizado supervisionado e o aprendizado não supervisionado. No aprendizado supervisionado, os modelos são treinados com dados previamente rotulados, nos quais cada instância possui atributos de entrada e uma saída conhecida, como uma classe ou valor alvo. Já no aprendizado não supervisionado, os dados não possuem rótulos

previamente definidos, e o objetivo é identificar estruturas, agrupamentos ou padrões presentes nos próprios dados (GÉRON; RAVAGLIA, 2021).

Na detecção de intrusão, técnicas de aprendizado de máquina podem ser utilizadas para apoiar a identificação de padrões associados a tráfego benigno e malicioso. Em vez de depender apenas de regras previamente definidas, os modelos podem explorar relações presentes nos dados, considerando características extraídas do tráfego de rede, como estatísticas dos fluxos, volume de pacotes, duração das conexões e outras informações utilizadas para representar o comportamento da comunicação (BUCZAK; GUVEN, 2016; LIAO et al., 2013).

2.2.1 Árvores de Decisão

Árvores de decisão são modelos de aprendizado supervisionado utilizados em tarefas de classificação e regressão. Esses modelos organizam o processo de decisão em uma estrutura hierárquica, na qual os dados são divididos sucessivamente com base nos valores dos atributos de entrada (GÉRON; RAVAGLIA, 2021). Em problemas de classificação, o objetivo é conduzir cada instância por uma sequência de decisões até uma classe prevista.

A estrutura de uma árvore de decisão é composta por nós internos, ramos e folhas. Os nós internos representam testes realizados sobre os atributos, os ramos representam os possíveis resultados desses testes e as folhas indicam a saída do modelo, como uma classe ou valor previsto (GÉRON; RAVAGLIA, 2021). Dessa forma, a classificação de uma instância ocorre ao percorrer a árvore a partir da raiz até uma folha, de acordo com os valores observados em seus atributos.

Durante a construção da árvore, o algoritmo busca escolher divisões capazes de separar as instâncias de forma mais adequada em relação à variável alvo. Para isso, são utilizados critérios de divisão que avaliam a qualidade da separação produzida por determinado atributo. Em tarefas de classificação, esses critérios estão associados à redução da impureza dos nós, isto é, à formação de grupos cada vez mais homogêneos em relação às classes presentes nos dados (GÉRON; RAVAGLIA, 2021; SONG; LU, 2015).

Embora árvores de decisão sejam modelos interpretáveis e capazes de representar relações não lineares entre atributos, árvores muito profundas podem se ajustar excessi-

vamente aos dados de treinamento. Esse comportamento, conhecido como sobreajuste, ocorre quando o modelo passa a capturar particularidades específicas da base de treinamento, prejudicando sua capacidade de generalização para novos dados (GÉRON; RAVAGLIA, 2021). Por esse motivo, métodos baseados em múltiplas árvores, como *Random Forest* e *XGBoost*, são frequentemente utilizados para melhorar a robustez e o desempenho em tarefas de classificação.

Neste trabalho, as árvores de decisão são relevantes porque servem como componentes básicos dos modelos utilizados posteriormente. Tanto o *Random Forest* quanto o *XGBoost* são métodos baseados em árvores, mas utilizam estratégias que combinam múltiplas árvores com o objetivo de melhorar o desempenho e a capacidade de generalização em relação a uma árvore individual.

2.2.2 Random Forest

O *Random Forest* é um método de aprendizado de máquina baseado em *ensemble*, isto é, na combinação de múltiplos modelos para produzir uma decisão final. Nesse caso, os modelos combinados são árvores de decisão, de modo que o *Random Forest* utiliza um conjunto de árvores em vez de depender de uma única árvore para realizar a classificação (BREIMAN, 2001; GÉRON; RAVAGLIA, 2021). Essa estratégia busca melhorar a capacidade de generalização do classificador, reduzindo a instabilidade que pode ocorrer em árvores individuais.

Um dos mecanismos utilizados pelo *Random Forest* para introduzir diversidade entre as árvores é a amostragem dos dados de treinamento. Cada árvore é construída a partir de uma amostra do conjunto original, geralmente obtida por meio de *bootstrap*, técnica em que os exemplos são selecionados aleatoriamente com reposição (BREIMAN, 2001). Dessa forma, diferentes árvores são treinadas com diferentes subconjuntos de instâncias, fazendo com que cada uma aprenda padrões parcialmente distintos da base.

Além da variação nos dados, o *Random Forest* também utiliza aleatoriedade na escolha dos atributos. Durante a construção de cada árvore, em cada divisão, apenas um subconjunto dos atributos disponíveis é considerado para selecionar o melhor critério de separação (BREIMAN, 2001). Esse processo reduz a correlação entre as árvores, pois

evita que todas elas utilizem sempre os mesmos atributos mais fortes nas mesmas divisões.

Em problemas de classificação, cada árvore do conjunto produz uma classe como saída. A predição final do *Random Forest* é obtida pela agregação das decisões individuais, geralmente por votação majoritária, em que a classe mais votada pelas árvores é escolhida como resultado final (BREIMAN, 2001). Assim, a decisão do modelo não depende de uma única sequência de divisões, mas do comportamento coletivo das árvores que compõem a floresta.

2.2.3 XGBoost

O *XGBoost*, abreviação de *Extreme Gradient Boosting*, é um método de aprendizado de máquina baseado em *ensemble* de árvores de decisão. Assim como outros métodos de conjunto, ele combina múltiplos modelos para produzir uma predição final. No entanto, diferentemente de abordagens como o *Random Forest*, o *XGBoost* utiliza a técnica de *boosting*, na qual as árvores são construídas de forma sequencial (FRIEDMAN, 2001; CHEN; GUESTRIN, 2016).

No processo de *boosting*, cada nova árvore é adicionada ao modelo com o objetivo de complementar as árvores já existentes. De forma geral, as árvores construídas posteriormente buscam reduzir os erros cometidos pelo modelo nas etapas anteriores, fazendo com que o conjunto seja ajustado progressivamente aos dados de treinamento (FRIEDMAN, 2001). Dessa forma, o modelo final é formado pela combinação das contribuições de várias árvores, adicionadas ao longo do processo de treinamento.

Um aspecto importante do *XGBoost* é sua função objetivo, utilizada para orientar o treinamento do modelo. Essa função considera tanto uma função de perda, associada ao erro entre as predições e os valores reais, quanto termos de regularização, que penalizam a complexidade das árvores (CHEN; GUESTRIN, 2016). A inclusão da regularização tem como objetivo reduzir o risco de sobreajuste, evitando que o modelo se torne excessivamente ajustado aos dados de treinamento.

Outro elemento relevante no treinamento do *XGBoost* é a taxa de aprendizado, responsável por controlar a contribuição de cada nova árvore adicionada ao conjunto. Valores menores fazem com que cada árvore tenha impacto mais limitado sobre o mo-

delo final, tornando o processo de aprendizado mais gradual. Ao final do treinamento, a predição é obtida pela combinação das contribuições das árvores construídas sequencialmente (CHEN; GUESTRIN, 2016).

3 Trabalhos Relacionados

Diversos trabalhos investigam o uso de aprendizado de máquina em sistemas de detecção de intrusão, especialmente com bases públicas como a CICIDS2017. Um estudo relacionado avalia classificadores supervisionados nessa base, considerando tráfego benigno e diferentes categorias de ataque em uma configuração multiclasse (NETO; GOMES, 2019). Os autores comparam diferentes modelos, incluindo *Decision Tree*, *Random Forest* e Perceptron Multicamadas, além de aplicar seleção de atributos baseada em importância. Os resultados indicam desempenho elevado na avaliação *offline*, com métricas próximas de 98% a 99%. Esse trabalho se aproxima desta pesquisa pelo uso da CICIDS2017 e pela comparação de classificadores, mas difere por considerar várias categorias de ataque e por não integrar o modelo a uma ferramenta de monitoramento contínuo.

Outra proposta apresenta um sistema híbrido e on-line para detecção e classificação de tráfego malicioso (ABREU; ABELéM, 2022). A solução combina diferentes técnicas em múltiplos estágios e é avaliada em bases como NSL-KDD, UNSW-NB15 e CICIDS17. Além das métricas de classificação, os autores também analisam o tempo de detecção, aspecto relevante para sistemas aplicados a fluxos contínuos de tráfego. A relação com esta pesquisa está na preocupação com a classificação em um cenário de monitoramento, embora este trabalho adote uma arquitetura mais simples, direcionada às classes DoS da CICIDS2017.

No contexto de ataques de negação de serviço, outro estudo propõe uma abordagem baseada em aprendizado profundo para detecção de diferentes tipos de DDoS na base CIC-DDoS2019 (AKGUN; HIZAL; CAVUSOGLU, 2022). O trabalho avalia arquiteturas como Redes Neurais Profundas (DNN), Redes Neurais Convolucionais (CNN) e redes do tipo Memória de Longo e Curto Prazo (LSTM), além de realizar etapas de pré-processamento e seleção de atributos. Embora também trate de ataques relacionados à negação de serviço, a proposta utiliza uma base voltada a DDoS e modelos de aprendizado profundo, enquanto esta pesquisa utiliza modelos baseados em árvores e concentra-se nas variações de DoS presentes na CICIDS2017.

De modo geral, os trabalhos analisados mostram que modelos de aprendizado de máquina podem alcançar desempenho elevado em bases públicas de detecção de intrusão. Entretanto, a maior parte das avaliações concentra-se no desempenho *offline* dos classificadores. Esta pesquisa se diferencia por combinar a avaliação *offline* dos modelos e a integração do classificador selecionado a uma ferramenta de monitoramento e classificação contínua, posteriormente avaliada em um ambiente controlado.

4 Metodologia

4.1 Base de Dados

Para este trabalho, foi utilizada a base de dados CICIDS2017, proposta por Sharafaldin, Habibi Lashkari e Ghorbani (2018) para apoiar a avaliação de sistemas de detecção de intrusão. A base foi gerada em um ambiente controlado, composto por uma rede de máquinas vítimas e uma rede destinada à execução dos ataques. Essa estrutura contou com diferentes sistemas operacionais e serviços de rede, permitindo a captura de fluxos benignos e maliciosos em cenários previamente definidos.

A coleta ocorreu durante cinco dias. Nesse período, foram geradas atividades benignas associadas ao uso de serviços como HTTP, HTTPS, FTP, SSH e e-mail, juntamente com diferentes cenários de ataque. Os dados são disponibilizados em arquivos de captura de tráfego e em arquivos CSV contendo fluxos rotulados e atributos extraídos com o CICFlowMeter.

Os arquivos CSV disponibilizados com a CICIDS2017 foram produzidos com a versão 3 do CICFlowMeter. No ambiente experimental deste trabalho, foi utilizada a versão 4 da ferramenta, pois a versão originalmente empregada na construção da base não se encontrava mais disponível. Não foi encontrada documentação suficientemente detalhada que permitisse confirmar a equivalência do cálculo de todos os atributos entre essas versões. Essa diferença é considerada posteriormente como uma possível ameaça à compatibilidade entre os dados.

Neste trabalho, foram utilizados os arquivos estruturados em fluxos, pois seus atributos representam as informações utilizadas como entrada pelos modelos de classificação. Antes de qualquer filtragem ou tratamento, a base analisada continha 2.830.743 instâncias, distribuídas entre tráfego benigno e diferentes categorias de ataque, conforme apresentado na Tabela 4.1.

A partir dessa base original, foram realizados os tratamentos necessários para adequar os dados ao escopo e às etapas de treinamento deste trabalho, conforme descrito

Tabela 4.1: Distribuição original das classes na base CICIDS2017

Classe	**Quantidade de instâncias**
BENIGN	2.273.097
DoS Hulk	231.073
PortScan	158.930
DDoS	128.027
DoS GoldenEye	10.293
FTP-Patator	7.938
SSH-Patator	5.897
DoS slowloris	5.796
DoS Slowhttptest	5.499
Bot	1.966
Web Attack – Brute Force	1.507
Web Attack – XSS	652
Infiltration	36
Web Attack – Sql Injection	21
Heartbleed	11
Total	**2.830.743**

na subseção seguinte.

4.2 Pré-Processamento

Como o foco deste trabalho está na detecção de ataques DoS, a primeira etapa do pré-processamento consistiu na filtragem das classes presentes na base. Foram mantidas as instâncias correspondentes à classe BENIGN e às variações de ataques DoS, enquanto as demais categorias de ataque foram removidas.

Após essa filtragem, a base passou a conter as classes BENIGN, DoS Hulk, DoS GoldenEye, DoS slowloris e DoS Slowhttptest. A distribuição resultante é apresentada na Tabela 4.2.

Tabela 4.2: Distribuição das classes após a filtragem da base

Classe	**Quantidade de instâncias**
BENIGN	2.273.097
DoS Hulk	231.073
DoS GoldenEye	10.293
DoS slowloris	5.796
DoS Slowhttptest	5.499
Total	**2.525.758**

Após a filtragem das classes, foi analisada a presença de valores nulos e infinitos nos atributos da base. Esses valores foram identificados apenas nas variáveis *Flow Bytes/s* e *Flow Packets/s*, conforme apresentado na Tabela 4.3.

Tabela 4.3: Quantidade e percentual de valores inválidos por variável e classe

Variável	**Classe**	**Total de registros**	**Valores inválidos**	**Percentual inválido**
Flow Bytes/s	BENIGN	2.273.097	1777	0.07818%
Flow Bytes/s	DoS Hulk	231.073	949	0.41069%
Flow Packets/s	BENIGN	2.273.097	1777	0.07818%
Flow Packets/s	DoS Hulk	231.073	949	0.41069%

Embora a tabela apresente 5.452 ocorrências de valores inválidos, essas ocorrências estavam concentradas nas mesmas instâncias para as variáveis *Flow Bytes/s* e *Flow Packets/s*. Dessa forma, foram identificadas 2.726 instâncias distintas contendo valores inválidos, correspondentes a 1.777 registros da classe BENIGN e 949 registros da classe DoS Hulk. Como essa quantidade representa uma parcela reduzida da base, essas instâncias foram removidas. Após essa etapa, a base passou a conter 2.523.032 instâncias.

Além da remoção de instâncias, também foram analisadas algumas características dos atributos presentes na base. A coluna *Destination Port* foi removida, pois representa diretamente a porta de destino da conexão e poderia fazer com que o modelo aprendesse padrões muito dependentes desse identificador, reduzindo sua capacidade de generalização para outros cenários. Também foi removida a coluna *Fwd Header Length.1*, por apresentar informação semelhante à coluna *Fwd Header Length*.

Em seguida, foram removidas colunas com baixa variabilidade, isto é, atributos em que mais de 99% dos registros possuíam valor igual a zero. Como esses atributos praticamente não apresentam variação na base analisada, sua contribuição para o treinamento tende a ser limitada. As colunas removidas por esse critério são apresentadas na Tabela 4.4.

Para os atributos que também apresentavam alta concentração de valores zero, mas que não ultrapassaram o limite de 99% definido para remoção direta, foi realizada uma análise adicional. Nessa etapa, foram selecionadas as colunas em que mais de 80% dos registros possuíam valor igual a zero. O objetivo foi verificar se esses atributos, apesar da baixa variabilidade geral, ainda poderiam apresentar relevância para alguma classe específica.

Tabela 4.4: Colunas removidas por baixa variabilidade

Coluna removida
Bwd PSH Flags
Fwd URG Flags
Bwd URG Flags
RST Flag Count
CWE Flag Count
ECE Flag Count
Fwd Avg Bytes/Bulk
Fwd Avg Packets/Bulk
Fwd Avg Bulk Rate
Bwd Avg Bytes/Bulk
Bwd Avg Packets/Bulk
Bwd Avg Bulk Rate

Para isso, foi realizada uma análise baseada no *lift*, medida utilizada em mineração de dados para avaliar o grau de associação entre eventos (HAN; PEI; TONG, 2022). Essa análise foi necessária porque uma variável com muitos valores iguais a zero pode, ainda assim, apresentar relevância para a classificação caso seus valores diferentes de zero estejam concentrados em uma classe específica.

Dessa forma, o *lift* foi utilizado para comparar a frequência relativa de valores diferentes de zero em cada classe com a frequência relativa desses valores na base como um todo. Valores de *lift* superiores a 1 indicam que a ocorrência de valores diferentes de zero é proporcionalmente maior em determinada classe do que no conjunto completo de dados, sugerindo uma possível associação entre o atributo e a classe analisada.

A Tabela 4.5 apresenta os valores de *lift* para os atributos analisados em relação a cada classe. Observa-se que algumas variáveis, mesmo possuindo alta concentração de valores zero, apresentam associação relevante com classes específicas. A variável *FIN Flag Count*, por exemplo, apresentou maior associação com a classe DoS Hulk, com *lift* de 9.628608. As variáveis *SYN Flag Count* e *Fwd PSH Flags* apresentaram maior associação com a classe DoS slowloris, enquanto a variável *Idle Std* apresentou maior associação com a classe DoS Slowhttptest.

Com base nessa análise, as variáveis *FIN Flag Count*, *SYN Flag Count*, *Fwd PSH Flags*, *Active Std* e *Idle Std* foram mantidas na base, pois apresentaram associação relevante com pelo menos uma das classes analisadas. Por outro lado, a variável *URG Flag Count* não apresentou valores de *lift* expressivos, ficando próxima de 1 mesmo em

Tabela 4.5: Valores de lift para atributos com alta concentração de zeros

Atributo	BENIGN	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS slowloris
FIN Flag Count	0.298565	0.000000	9.628608	0.000000	0.000000
SYN Flag Count	1.073305	0.000000	0.000000	2.488110	5.024525
Fwd PSH Flags	1.073305	0.000000	0.000000	2.488110	5.024525
Active Std	1.081099	0.067172	0.002966	0.496662	3.700173
Idle Std	1.051559	0.084054	0.195014	6.090613	3.573278
URG Flag Count	1.088051	0.063796	0.007697	1.109026	0.253732

seu maior valor. Por esse motivo, essa coluna também foi removida da base.

Por fim, foi realizada a análise de registros duplicados na base resultante. Essa etapa foi executada após a remoção dos atributos que não seriam utilizados pelo modelo, pois registros originalmente distintos poderiam tornar-se idênticos no espaço final de atributos após a exclusão dessas colunas. Instâncias com os mesmos atributos e o mesmo rótulo não acrescentam novos padrões ao treinamento e podem aumentar artificialmente a frequência de determinadas combinações. Nessa etapa, foram identificadas e removidas 434.840 instâncias duplicadas, fazendo com que a base final passasse a conter 2.088.192 instâncias.

4.3 Treinamento do Modelo

Embora o foco principal deste trabalho não esteja em realizar uma comparação ampla entre diferentes modelos de aprendizado de máquina, foi considerada importante a avaliação de mais de uma alternativa para a classificação dos fluxos de rede. Dessa forma, foram selecionados os modelos *Random Forest* e *XGBoost*, uma vez que ambos apresentaram bons resultados em trabalhos relacionados que utilizaram a base CICIDS2017 (MASEER et al., 2021; FARHAT et al., 2023; TALUKDER et al., 2024).

Os dois modelos foram treinados e avaliados com o objetivo de selecionar aquele que apresentasse o desempenho mais adequado para sua utilização como componente do sistema de detecção de intrusão proposto neste trabalho.

4.3.1 Divisão Treino e Teste

Antes de prosseguir para as próximas etapas do treinamento, foi realizada a divisão da base de dados em conjuntos de treino e teste. Essa divisão foi feita antes do balanceamento

e do treinamento dos modelos, de modo que essas etapas fossem realizadas apenas sobre os dados de treinamento. O conjunto de teste foi mantido separado e utilizado apenas na avaliação final dos modelos.

Para essa divisão, foram destinados 80% dos dados para treinamento e 20% para teste. A separação foi realizada de forma estratificada, preservando, aproximadamente, a proporção das classes em ambos os subconjuntos. Dessa forma, uma classe com determinada representatividade na base original mantém uma proporção semelhante tanto na base de treinamento quanto na base de teste.

O conjunto consolidado utilizado neste trabalho não preservava a coluna temporal nem um identificador da captura de origem de cada fluxo. Por esse motivo, não foi possível realizar a separação dos dados por execução de ataque, janela temporal ou arquivo de captura. Assim, foi adotada a divisão aleatória e estratificada descrita anteriormente. Embora as duplicatas exatas tenham sido removidas antes dessa divisão, não se pode descartar completamente a presença, nos dois subconjuntos, de fluxos semelhantes provenientes de um mesmo cenário de tráfego.

A distribuição resultante da divisão é apresentada na Tabela 4.6. Observa-se que os percentuais das classes foram preservados nos dois conjuntos, mantendo a mesma proporção entre treino e teste.

Tabela 4.6: Distribuição das classes nos conjuntos de treino e teste

Classe	**Treino**	**Teste**	**Treino (%)**	**Teste (%)**
BENIGN	1.515.557	378.890	90.72	90.72
DoS Hulk	138.277	34.569	8.28	8.28
DoS GoldenEye	8.229	2.057	0.49	0.49
DoS slowloris	4.308	1.077	0.26	0.26
DoS Slowhttptest	4.182	1.046	0.25	0.25

4.3.2 Tratamento do Desbalanceamento por Undersampling

A base de dados de treinamento possui um grande desbalanceamento entre as classes, como pode ser observado na Figura 4.1.

Como a classe BENIGN possui uma quantidade muito maior de instâncias do que as classes relacionadas a ataques DoS, o modelo poderia ser influenciado pela classe majoritária, apresentando melhor desempenho nessa classe e maior dificuldade na classificação

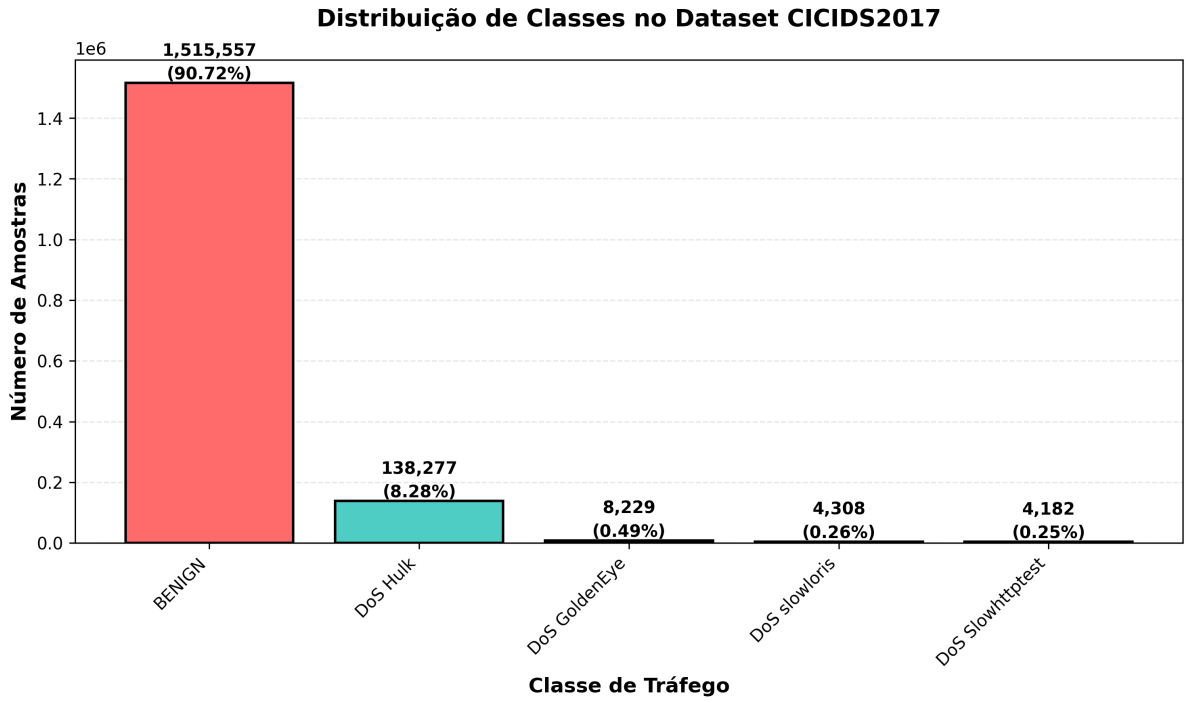

Figura 4.1: Distribuição das classes na base de treinamento antes do balanceamento

das classes minoritárias, problema frequentemente observado em tarefas de aprendizado com dados desbalanceados (HE; GARCIA, 2009). Por esse motivo, foi avaliado o uso de uma técnica de balanceamento, com o objetivo de verificar seu impacto nos resultados do modelo.

A técnica avaliada foi o *undersampling*, aplicada somente sobre a base de treinamento. Essa escolha foi feita porque técnicas de aumento das classes minoritárias, como o SMOTE (*Synthetic Minority Over-sampling Technique*), podem introduzir exemplos sintéticos na base (CHAWLA et al., 2002), o que não era o objetivo desta etapa. Dessa forma, optou-se por testar a redução da quantidade de instâncias das classes majoritárias, mantendo as classes minoritárias com seus registros originais.

Foi utilizado o *undersampling* randômico, no qual uma parte das instâncias das classes majoritárias é selecionada aleatoriamente, enquanto o restante é descartado. Neste teste, foi adotada uma estratégia baseada na proporção de 10:1 em relação à menor classe da base de treinamento. Dessa forma, as classes majoritárias foram reduzidas para que não possuíssem mais do que dez vezes a quantidade de instâncias da menor classe, correspondente a 4.182 registros da classe DoS Slowhttptest. A Figura 4.2 apresenta o resultado dessa estratégia. Os resultados obtidos com a distribuição original e com a

aplicação do undersampling 10:1 são comparados posteriormente na etapa de avaliação.

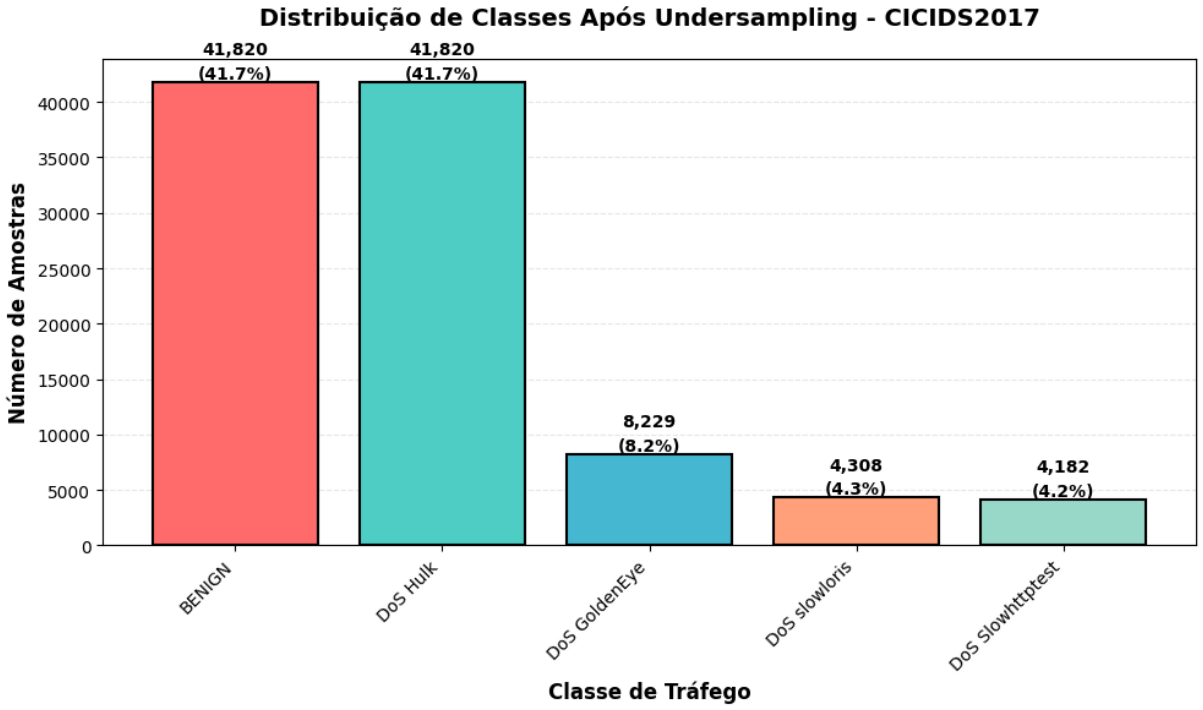

Figura 4.2: Distribuição das classes na base de treinamento após o *undersampling* com proporção 10:1

4.3.3 Treinamento

Após a divisão da base e a definição dos cenários com a distribuição original e com aplicação de undersampling, foi realizada a etapa de treinamento dos modelos. Os modelos receberam como entrada os atributos extraídos dos fluxos de rede, enquanto a classe de cada instância foi utilizada como variável alvo. Dessa forma, o objetivo do treinamento foi permitir que os modelos aprendessem padrões capazes de diferenciar o tráfego benigno das variações de ataques DoS consideradas neste trabalho.

Foram treinadas quatro configurações: *Random Forest* com a base de treinamento original, *Random Forest* com a base submetida ao *undersampling*, *XGBoost* com a base de treinamento original e *XGBoost* com a base submetida ao *undersampling*. Essa organização permitiu analisar tanto o efeito do balanceamento quanto o desempenho dos dois modelos selecionados.

As configurações adotadas para cada modelo foram:

- *Random Forest*:

  - implementação: `RandomForestClassifier(sklearn)`;
  - `random_state`: 42;
  - `n_estimators`: 100;
  - `class_weight`: `balanced`.

- *XGBoost*:
  - implementação: `XGBClassifier(xgboost)`;
  - `random_state`: 42;
  - `n_estimators`: 100;
  - `objective`: `multi:softprob`;
  - `eval_metric`: `mlogloss`;
  - `tree_method`: `hist`.

Nas duas configurações do Random Forest foi utilizado o parâmetro class_weight, configurado com o valor "balanced". Esse parâmetro não modifica a quantidade de instâncias de cada classe, mas atribui pesos inversamente proporcionais às suas frequências durante o treinamento. Portanto, as configurações do Random Forest diferem pela aplicação ou não do undersampling, mas ambas utilizam ponderação automática das classes.

A definição de um valor fixo para o parâmetro `random_state` teve como objetivo tornar os experimentos reprodutíveis.

Como os modelos utilizados são baseados em árvores de decisão, não foi aplicada normalização aos atributos, pois esse tipo de modelo não depende diretamente da escala dos valores para realizar as divisões durante o treinamento.

Após a avaliação das configurações treinadas, o modelo selecionado foi exportado para ser utilizado posteriormente pelo módulo de classificação contínua do sistema.

4.3.4 Métricas de Avaliação

Após o treinamento, as configurações obtidas foram avaliadas utilizando o conjunto de teste separado anteriormente. Esse conjunto não foi submetido ao balanceamento, man-

tendo a distribuição original das classes. Dessa forma, foi possível avaliar os modelos em um cenário mais próximo daquele em que seriam utilizados pelo sistema.

Para avaliar o desempenho dos modelos, foram consideradas as métricas de acurácia, precisão, *recall* e F1-score. Entretanto, devido ao desbalanceamento presente na base, a acurácia foi utilizada apenas como uma medida geral de desempenho, não sendo considerada isoladamente para a escolha do modelo final. A análise priorizou as métricas macro e os resultados obtidos individualmente para cada classe, evitando que o desempenho da classe majoritária predominasse sobre a avaliação das classes de ataque (SOKOLOVA; LAPALME, 2009).

A acurácia representa a proporção de instâncias classificadas corretamente em relação ao total de instâncias avaliadas. Considerando C classes e uma matriz de confusão M, em que M_{ii} representa a quantidade de instâncias corretamente classificadas na classe i, a acurácia é calculada conforme a Equação 4.1.

$$\text{Acurácia} = \frac{\sum_{i=1}^{C} M_{ii}}{N} \tag{4.1}$$

em que N representa o número total de instâncias avaliadas.

As métricas de precisão, *recall* e F1-score foram calculadas individualmente para cada classe. Para uma determinada classe, verdadeiro positivo (VP) representa as instâncias corretamente classificadas como pertencentes à classe; falso positivo (FP) representa as instâncias classificadas incorretamente como pertencentes à classe; e falso negativo (FN) representa as instâncias pertencentes à classe que foram classificadas como outra categoria.

A precisão indica, entre as instâncias classificadas como pertencentes a uma determinada classe, quantas realmente pertencem a ela, sendo definida pela Equação 4.2.

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{4.2}$$

O *recall* representa a proporção de instâncias de uma determinada classe que foram corretamente identificadas pelo modelo, conforme a Equação 4.3.

$$\textit{Recall} = \frac{VP}{VP + FN} \tag{4.3}$$

O F1-score corresponde à média harmônica entre precisão e *recall*, permitindo considerar conjuntamente a ocorrência de falsos positivos e falsos negativos.

$$\text{F1-score} = 2 \cdot \frac{\text{Precisão} \cdot \textit{Recall}}{\text{Precisão} + \textit{Recall}} \tag{4.4}$$

Como a tarefa de classificação deste trabalho envolve múltiplas classes, também foi utilizada a média macro das métricas. Essa média calcula o valor da métrica individualmente para cada classe e, posteriormente, realiza a média simples entre elas.

$$\text{Métrica}_{macro} = \frac{1}{C} \sum_{i=1}^{C} \text{Métrica}_i \tag{4.5}$$

em que C representa o número total de classes e Métrica_i representa o valor da métrica calculado para a classe i.

Os resultados obtidos a partir dessas métricas são apresentados e discutidos no capítulo de resultados.

4.4 Monitoramento e Classificação Contínua

Após a seleção do modelo, foi desenvolvida uma ferramenta para aplicá-lo sobre o tráfego observado em uma interface de rede. O objetivo é permitir a análise contínua do tráfego capturado, classificando os fluxos identificados como tráfego benigno ou como uma das variações de ataques DoS consideradas neste trabalho. Nas subseções seguintes, são apresentados a arquitetura da ferramenta e o funcionamento de seus componentes.

4.4.1 Visão Geral da Arquitetura

A arquitetura da ferramenta foi organizada em dois blocos principais: o módulo de monitoramento e extração e o módulo classificador. Essa organização tem como objetivo separar as etapas responsáveis pela obtenção e transformação do tráfego de rede da etapa responsável pela classificação dos fluxos identificados.

A Figura 4.3 apresenta uma visão geral da arquitetura proposta. Inicialmente, o tráfego observado na placa de rede é capturado pelo módulo de monitoramento, que regis-

tra os pacotes trafegados e os encaminha para a etapa de extração de fluxos. Nessa etapa, as capturas geradas são transformadas em uma representação estruturada, adequada para ser utilizada como entrada pelo modelo de aprendizado de máquina.

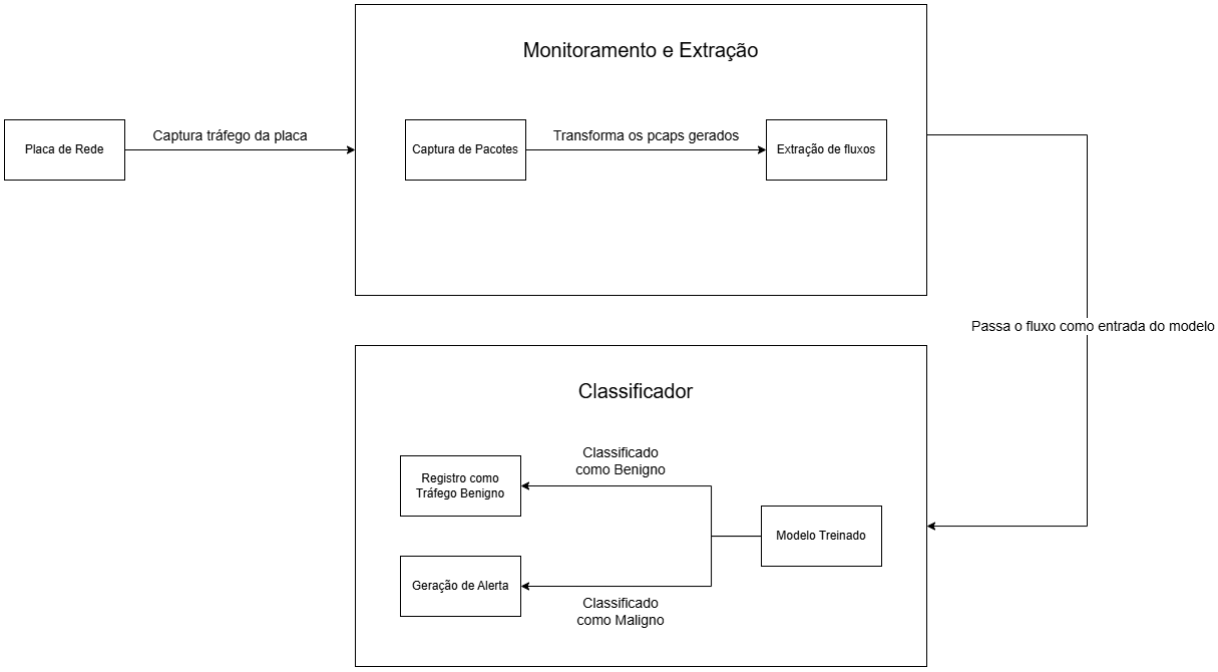

Figura 4.3: Visão geral da arquitetura da ferramenta de monitoramento, extração e classificação de fluxos de rede

Após a extração, os fluxos são enviados ao módulo classificador. Esse módulo utiliza o modelo previamente treinado para analisar os atributos de cada fluxo e atribuir uma classe ao tráfego observado. Quando o fluxo é classificado como benigno, o resultado é apenas registrado pelo sistema. Por outro lado, quando o fluxo é classificado como malicioso, é gerado um alerta indicando a possível ocorrência de um ataque.

4.4.2 Monitoramento e Extração dos Fluxos

O módulo de monitoramento e extração é responsável por capturar o tráfego observado na interface de rede e transformá-lo em fluxos estruturados, que posteriormente serão utilizados como entrada pelo módulo classificador. Esse processo envolve a geração periódica de arquivos de captura, o gerenciamento desses arquivos em uma fila de processamento e a extração dos atributos dos fluxos de rede.

A Figura 4.4 apresenta a organização interna desse módulo. Inicialmente, a cap-

tura dos pacotes é realizada por meio da ferramenta *tcpdump*[1], executada por um script em Python responsável por iniciar e controlar o processo de monitoramento. O tráfego observado é registrado em arquivos no formato PCAP, gerados em intervalos de tempo configuráveis. Para isso, foi utilizado o parâmetro *-G* do *tcpdump*, que permite rotacionar o arquivo de saída após um intervalo definido em segundos. Na implementação utilizada neste trabalho, esse intervalo foi definido em 60 segundos, fazendo com que um novo arquivo de captura fosse gerado a cada minuto. Esse valor foi adotado como uma configuração operacional da ferramenta, buscando limitar o tamanho dos arquivos de captura e permitir que os blocos concluídos fossem encaminhados periodicamente para processamento. O intervalo não foi obtido por meio de uma otimização sobre a CICIDS2017, podendo ser ajustado de acordo com os requisitos e os recursos do ambiente de execução.

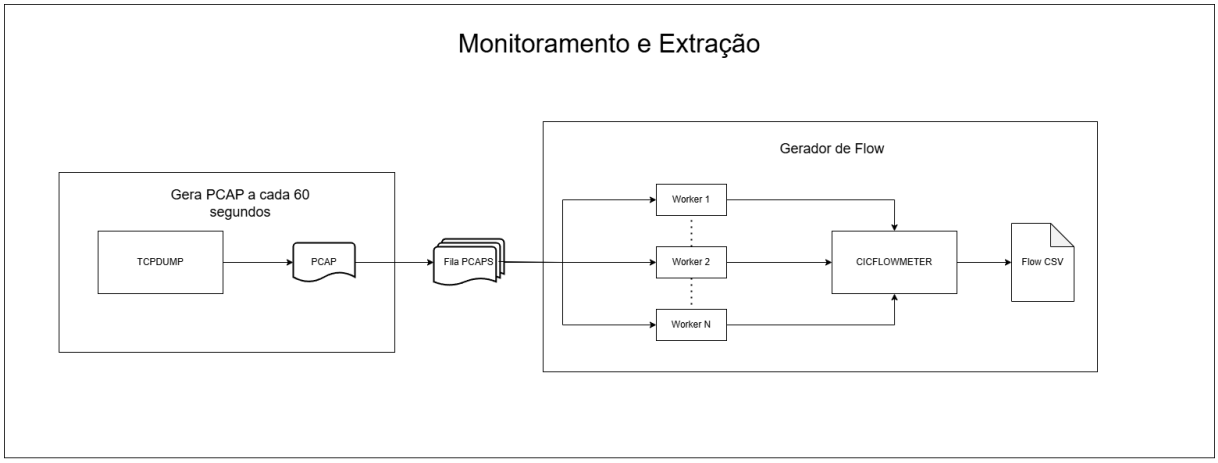

Figura 4.4: Organização do módulo de monitoramento e extração dos fluxos

A divisão da captura em blocos temporais evita a geração de um único arquivo PCAP extenso e organiza o tráfego em janelas menores de análise. Ao final de cada intervalo, o arquivo atual é encerrado, disponibilizado para processamento e uma nova captura é iniciada.

Os arquivos PCAP finalizados são inseridos em uma fila de processamento, consumida por um ou mais *workers*. Essa organização desacopla a captura da extração dos fluxos, permitindo que arquivos já concluídos sejam processados sem interromper o monitoramento da interface de rede.

A quantidade de *workers* pode ser configurada de acordo com os recursos dis-

[1]https://www.tcpdump.org/manpages/tcpdump.1.html

poníveis no ambiente de execução. A utilização de apenas um *worker* faz com que os arquivos sejam processados sequencialmente, reduzindo a concorrência e o consumo simultâneo de recursos. Por outro lado, o uso de múltiplos *workers* permite que diferentes arquivos PCAP sejam processados em paralelo, aumentando a vazão da etapa de extração. Entretanto, essa configuração também pode elevar o uso de CPU, memória e operações de entrada e saída, sendo necessário ajustá-la conforme a capacidade da máquina utilizada.

A extração dos fluxos é realizada com o CICFlowMeter[2], ferramenta destinada à geração e análise de fluxos de tráfego de rede. O CICFlowMeter agrupa os pacotes em fluxos bidirecionais, nos quais o primeiro pacote observado define os sentidos de encaminhamento e retorno. A partir desses fluxos, são calculados atributos estatísticos relacionados, por exemplo, à duração da comunicação, à quantidade de pacotes, ao volume de bytes trafegados e ao tamanho dos pacotes.

A utilização do CICFlowMeter permitiu gerar atributos correspondentes à estrutura esperada pelo modelo, uma vez que a CICIDS2017 também foi construída com essa ferramenta (SHARAFALDIN; Habibi Lashkari; GHORBANI, 2018). Entretanto, foram utilizadas versões diferentes do CICFlowMeter na base original e no ambiente experimental. Por esse motivo, embora os nomes e a organização dos atributos tenham sido compatibilizados, não se assume que todos os valores sejam calculados de maneira estritamente equivalente entre as versões.

Ao final do processo, os arquivos CSV gerados são disponibilizados para o módulo de classificação.

4.4.3 Classificação dos Fluxos

O módulo de classificação é responsável por aplicar o modelo previamente treinado sobre os fluxos gerados na etapa de extração. Para isso, recebe como entrada os arquivos CSV produzidos a partir das capturas processadas pelo CICFlowMeter. Diferentemente da etapa de treinamento, essa fase corresponde à inferência, na qual o modelo já selecionado é utilizado para atribuir uma classe aos novos fluxos observados pela ferramenta.

Antes da predição, os dados extraídos passam por uma etapa de preparação.

[2]https://www.unb.ca/cic/research/applications.html#CICFlowMeter

Como o modelo foi treinado com um conjunto específico de atributos, é necessário garantir que os fluxos recebidos estejam organizados no mesmo formato utilizado durante o treinamento. Essa preparação envolve a seleção das colunas esperadas pelo modelo e a organização dos atributos na mesma ordem adotada na base de treinamento, evitando incompatibilidades entre os dados extraídos e o classificador exportado.

Após essa preparação, o modelo selecionado na etapa de avaliação é carregado pelo sistema e aplicado sobre cada fluxo presente no arquivo de entrada. Para cada instância analisada, o classificador produz uma predição indicando se o fluxo corresponde a tráfego benigno ou a uma das variações de ataques DoS consideradas neste trabalho.

Os resultados da classificação são então persistidos em um novo arquivo estruturado. Esse arquivo mantém os atributos extraídos dos fluxos e acrescenta as informações produzidas pelo classificador, como a classe prevista e uma medida de confiança da decisão. Esses registros permitem consultar posteriormente como cada fluxo foi classificado pelo modelo, servindo como histórico da análise realizada pela ferramenta.

Além do registro dos resultados, a ferramenta também pode atuar como mecanismo de alerta. Quando um ou mais fluxos são classificados como maliciosos, o sistema gera uma notificação indicando a possível ocorrência de atividade suspeita. Na implementação desenvolvida, essa notificação é enviada por meio do Telegram, contendo informações resumidas sobre os fluxos suspeitos identificados. Esse alerta possui caráter informativo, pois a ferramenta opera em modo IDS, realizando detecção e notificação, mas sem executar bloqueio automático de tráfego, alteração de regras de firewall ou contenção direta da comunicação.

4.5 Avaliação em Ambiente Controlado

Após a descrição da arquitetura da ferramenta, foi definido um procedimento de avaliação em ambiente controlado com o objetivo de observar o comportamento do classificador em fluxos gerados fora da base CICIDS2017. Essa etapa foi realizada após o treinamento e a seleção do modelo, utilizando a ferramenta desenvolvida para capturar, extrair e classificar tráfego observado em uma rede de laboratório.

Inicialmente, a avaliação foi planejada considerando um cenário benigno e quatro

cenários associados às classes DoS presentes no recorte utilizado neste trabalho: DoS Hulk, DoS GoldenEye, DoS slowloris e DoS Slowhttptest. Entretanto, durante a execução dos testes, não foi possível gerar o cenário associado à classe DoS GoldenEye de forma adequada no ambiente controlado. Por esse motivo, a avaliação prática foi conduzida com quatro cenários: um cenário benigno e três cenários de ataque, correspondentes às classes DoS Hulk, DoS slowloris e DoS Slowhttptest.

4.5.1 Ambiente de Teste

A máquina vítima foi configurada para executar um serviço de rede e a ferramenta proposta. Assim, o tráfego destinado a esse serviço era capturado na interface da própria vítima, convertido em fluxos e encaminhado ao módulo de classificação. A máquina atacante foi utilizada para gerar tráfego direcionado à vítima, incluindo execuções de ataques DoS. A configuração geral desse ambiente é apresentada na Tabela 4.7.

Tabela 4.7: Configuração geral do ambiente de teste da ferramenta

Máquina	**Sistema operacional**	**Função no ambiente**
VM atacante	Kali Linux	Geração de tráfego direcionado à VM vítima, incluindo ataques DoS
VM vítima	Ubuntu	Execução do serviço alvo e da ferramenta de classificação contínua

Durante os testes, a ferramenta seguiu o mesmo fluxo descrito nas subseções anteriores: captura dos pacotes, geração de arquivos PCAP, extração dos fluxos com o CICFlowMeter, organização dos atributos e classificação com o modelo treinado. As execuções foram separadas por tipo de tráfego, permitindo comparar os períodos de tráfego benigno e de ataque com as classes previstas pelo classificador. Os resultados dessa avaliação são apresentados no capítulo de resultados.

4.5.2 Cenários de Tráfego

Para avaliar a ferramenta no ambiente controlado, foram definidos quatro cenários de tráfego, sendo um cenário benigno e três cenários associados às classes DoS consideradas neste trabalho. Cada cenário foi executado uma única vez, com duração de 30 segundos,

e foi adotado um intervalo de 30 segundos entre execuções consecutivas. Essa pausa teve como objetivo facilitar a filtragem posterior dos ataques com base no timestamp do tráfego.

O cenário C1 correspondeu ao tráfego benigno. Nesse caso, foram geradas requisições HTTP da máquina atacante para o serviço executado na máquina vítima, utilizando a ferramenta *curl*. As requisições foram configuradas com variações de caminhos, métodos, cabeçalhos e intervalos de envio, sem concorrência elevada ou tentativa de manter conexões abertas artificialmente. Esse cenário foi utilizado como referência para observar o comportamento da ferramenta em um período sem ataque.

Para representar a classe DoS Hulk, no cenário C2 foi utilizada a ferramenta *ApacheBench* (*ab*), com o objetivo de gerar um volume elevado de requisições HTTP concorrentes contra o serviço da vítima. A execução foi configurada com 2000 requisições por rodada e 200 conexões concorrentes, repetindo-se durante o tempo definido para o cenário.

A classe DoS slowloris foi avaliada no cenário C3 por meio do script *slowloris.py*, executado a partir da máquina atacante contra o serviço HTTP da máquina vítima. A execução foi limitada ao tempo definido para o cenário, permitindo gerar conexões HTTP mantidas abertas de forma lenta, comportamento característico desse tipo de ataque.

Por fim, o cenário C4 foi associado à classe DoS Slowhttptest. Para esse caso, foi utilizada a ferramenta *slowhttptest* no modo de envio lento do corpo da requisição, por meio da opção `-B`. A execução foi configurada com 200 conexões, taxa de criação de 50 conexões, método POST, corpo de requisição com 4096 bytes e intervalo de persistência de 3 segundos. Esse cenário buscou representar um comportamento no qual o servidor mantém conexões ocupadas durante o envio gradual do corpo da requisição.

A Tabela 4.8 resume os cenários utilizados na avaliação da ferramenta.

É importante destacar que os cenários de ataque foram definidos como aproximações controladas dos comportamentos associados às classes DoS presentes na CICIDS2017. A documentação da base não especifica, de forma suficientemente detalhada, todas as ferramentas, versões e parâmetros utilizados na geração de cada ataque. Assim, o objetivo não foi reproduzir de forma equivalente os fluxos da base original, mas gerar

Tabela 4.8: Cenários de tráfego utilizados na avaliação da ferramenta

Cenário	**Ferramenta**	**Parâmetros principais**	**Classe esperada**
C1	*curl*	Requisições HTTP variadas	BENIGN
C2	*ab*	2000 requisições; 200 conexões concorrentes	DoS Hulk
C3	*slowloris.py*	Execução limitada a 30 s	DoS slowloris
C4	*slowhttptest -B*	200 conexões; POST; 4096 bytes; duração de 30 s	DoS Slowhttptest

comportamentos funcionalmente relacionados a cada tipo de ataque e observar o classificador em um ambiente distinto daquele utilizado no treinamento. Portanto, a classe esperada atribuída a cada cenário representa a associação experimental pretendida, e não uma garantia de equivalência estatística com os fluxos da CICIDS2017.

4.5.3 Procedimento de Execução

A execução dos testes foi automatizada por meio de um script executado na máquina atacante. Esse script acionou os cenários C1, C2, C3 e C4 em sequência, respeitando os intervalos definidos na etapa anterior.

Além de controlar a ordem de execução, o script registrou o horário de início, o horário de término e a classe esperada de cada cenário em um arquivo CSV. Esse arquivo foi utilizado posteriormente para associar os fluxos classificados pela ferramenta aos respectivos períodos de tráfego analisados.

5 Resultados

5.1 Resultados da Avaliação dos Modelos

Nesta seção, são apresentados os resultados obtidos na avaliação dos modelos treinados para a classificação de fluxos de rede. A análise considera as quatro configurações definidas na metodologia: *Random Forest* com a distribuição original e ponderação automática das classes, *Random Forest* com *undersampling* 10:1 e ponderação automática das classes, *XGBoost* com a distribuição original e *XGBoost* com *undersampling* 10:1. Os modelos foram avaliados no conjunto de teste, que manteve a distribuição original das classes.

5.1.1 Comparação Geral das Configurações

A Tabela 5.1 apresenta os resultados agregados obtidos pelas quatro configurações avaliadas. A comparação considera a acurácia, a precisão macro, o *recall* macro e o F1-score macro.

Tabela 5.1: Comparação das configurações avaliadas no conjunto de teste

Configuração	**Acurácia**	**Precisão Macro**	**Recall Macro**	**F1 Macro**
Random Forest original + ponderação	0.9992	0.9894	0.9950	0.9921
Random Forest *undersampling* 10:1 + ponderação	0.9981	0.9637	0.9970	0.9798
XGBoost distribuição original	0.9995	0.9862	0.9964	0.9912
XGBoost *undersampling* 10:1	0.9993	0.9761	0.9977	0.9864

A comparação das métricas macro indica que as configurações *Random Forest* com distribuição original e ponderação automática das classes e *XGBoost* com distribuição original apresentaram os resultados agregados mais elevados. Ambas obtiveram F1-score macro próximo de 0,99, indicando desempenho geral semelhante quando consideradas as médias entre as classes.

Apesar disso, a análise agregada não é suficiente para a escolha final do modelo, pois a base apresenta desbalanceamento entre as classes. Por esse motivo, também foi realizada uma análise das métricas obtidas individualmente para cada classe, principalmente nas classes DoS com menor quantidade de instâncias.

5.1.2 Análise por Classe

A Tabela 5.2 apresenta as métricas obtidas por classe para as duas configurações com melhor desempenho agregado: *Random Forest* sem balanceamento e *XGBoost* sem balanceamento. Essa análise permite verificar se o desempenho elevado se mantém também nas classes de ataque com menor representatividade na base.

Tabela 5.2: Comparação das métricas por classe entre as configurações com melhor desempenho agregado

	Random Forest original + ponderação			**XGBoost distribuição original**		
Classe	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
BENIGN	0.9995	0.9996	0.9996	0.9999	0.9996	0.9997
DoS GoldenEye	0.9985	0.9917	0.9951	0.9956	0.9956	0.9956
DoS Hulk	0.9974	0.9956	0.9965	0.9972	0.9992	0.9982
DoS Slowhttptest	0.9533	0.9943	0.9733	0.9403	0.9943	0.9665
DoS Slowloris	0.9981	0.9935	0.9958	0.9981	0.9935	0.9958

Observa-se que ambas as configurações apresentaram desempenho elevado em todas as classes avaliadas. Entretanto, na classe DoS Slowhttptest, os dois modelos obtiveram resultados inferiores em comparação às demais classes. Nesse caso, o *Random Forest* treinado com a distribuição original e ponderação automática das classes apresentou F1-score superior ao obtido pelo *XGBoost* treinado com a distribuição original, indicando melhor equilíbrio entre precisão e *recall* para essa classe.

Considerando as métricas macro e o desempenho por classe, o modelo *Random Forest* treinado com a distribuição original dos dados, sem aplicação de undersampling e com ponderação automática das classes, foi selecionado para integração à ferramenta de monitoramento e classificação contínua. Essa escolha foi baseada no desempenho mais equilibrado entre precisão e *recall*, especialmente na classe DoS Slowhttptest, que apresentou maior dificuldade relativa de classificação. Dessa forma, os resultados da

avaliação *offline* foram utilizados como base para definir o classificador incorporado à ferramenta.

5.2 Resultados em Ambiente Controlado

5.2.1 Classificação dos Fluxos Observados

Após a execução dos cenários no ambiente controlado, os fluxos classificados pela ferramenta foram agrupados de acordo com a classe esperada em cada intervalo de teste. A Tabela 5.3 apresenta a matriz de confusão obtida, considerando as classes reais associadas aos cenários executados e as classes previstas pelo modelo.

Embora o cenário DoS GoldenEye não tenha sido executado no ambiente controlado, essa classe permanece presente nas predições possíveis do modelo, pois fez parte do conjunto de classes utilizado no treinamento.

Tabela 5.3: Matriz de confusão dos fluxos classificados no ambiente controlado

Classe real	**BENIGN**	**DoS GoldenEye**	**DoS Hulk**	**DoS Slowhttptest**	**DoS slowloris**
BENIGN	64 (100,0000%)	0 (0,0000%)	0 (0,0000%)	0 (0,0000%)	0 (0,0000%)
DoS Hulk	106528 (99,9962%)	0 (0,0000%)	4 (0,0038%)	0 (0,0000%)	0 (0,0000%)
DoS Slowhttptest	1628 (100,0000%)	0 (0,0000%)	0 (0,0000%)	0 (0,0000%)	0 (0,0000%)
DoS slowloris	1144 (95,0166%)	60 (4,9834%)	0 (0,0000%)	0 (0,0000%)	0 (0,0000%)

A matriz apresenta as quatro classes reais correspondentes aos cenários executados e as cinco classes que poderiam ser previstas pelo modelo. A ausência de uma linha real para DoS GoldenEye ocorre porque esse cenário não foi executado, enquanto a coluna foi preservada por permanecer como uma saída possível do classificador.

A matriz de confusão mostra que o modelo classificou corretamente os fluxos do cenário benigno, mas apresentou desempenho limitado nos cenários de ataque executados no ambiente controlado. Em especial, observa-se que a maior parte dos fluxos associados aos ataques foi classificada como BENIGN, indicando baixa capacidade de detecção nesse cenário prático.

Esse comportamento contrasta com os resultados obtidos na avaliação *offline* com a base CICIDS2017, na qual o modelo apresentou métricas elevadas. Assim, os resultados sugerem uma limitação de generalização do classificador quando aplicado a fluxos gerados fora da base utilizada no treinamento. Essa diferença reforça a importância de avaliar

modelos de detecção não apenas em bases públicas previamente estruturadas, mas também em cenários controlados de execução.

5.2.2 Análise dos Atributos mais Relevantes

Após a análise da matriz de confusão obtida no ambiente controlado, foi realizada uma comparação entre os valores observados na CICIDS2017 e os valores obtidos nos fluxos gerados nas máquinas virtuais. O objetivo dessa análise foi verificar se os atributos mais relevantes para o modelo apresentavam mudanças de distribuição nas classes em que ocorreram problemas de classificação.

A análise foi concentrada nas classes DoS Hulk, DoS Slowhttptest e DoS slowloris, pois foram os cenários de ataque efetivamente executados no ambiente controlado e que apresentaram baixa taxa de detecção pelo classificador. A classe BENIGN não foi incluída nessa análise por ter sido corretamente classificada nos fluxos observados. A classe DoS GoldenEye também não foi considerada, pois não foi possível executar esse ataque no ambiente simulado utilizado nos testes.

Para essa comparação, foram selecionados os três atributos com maior importância no modelo *Random Forest* utilizado pela ferramenta: *Flow IAT Std*, *Bwd Packets/s* e *Flow IAT Mean*. A sigla IAT, do inglês *Inter-Arrival Time*, representa o intervalo de tempo entre chegadas consecutivas de pacotes de um fluxo. Nesse contexto, *Flow IAT Mean* corresponde à média desses intervalos, enquanto *Flow IAT Std* representa seu desvio-padrão. O atributo *Bwd Packets/s*, por sua vez, representa a taxa de pacotes por segundo observada no sentido de retorno do fluxo. Como medida principal de comparação, foi utilizado o PSI (*Population Stability Index*), métrica empregada para comparar distribuições entre duas populações e identificar deslocamentos entre uma distribuição de referência e uma distribuição observada posteriormente (YURDAKUL; NARANJO, 2020). Valores mais elevados de PSI indicam maior diferença entre a distribuição de referência e a distribuição comparada.

A Tabela 5.4 apresenta os resultados obtidos para os três atributos analisados nas classes de ataque avaliadas no ambiente controlado.

Observa-se que todos os atributos analisados apresentaram valores elevados de

Tabela 5.4: Comparação por PSI dos atributos mais relevantes nas classes de ataque

Classe	**Atributo**	**Média CICIDS2017**	**Média ambiente**	**Diferença (%)**	**PSI**
DoS Hulk	Flow IAT Std	$2{,}19 \times 10^{7}$	$5{,}40 \times 10^{5}$	-97,53	10,58
DoS Hulk	Bwd Packets/s	$9{,}49 \times 10^{1}$	$1{,}24 \times 10^{3}$	1210,69	9,75
DoS Hulk	Flow IAT Mean	$6{,}39 \times 10^{6}$	$1{,}71 \times 10^{5}$	-97,33	9,34
DoS Slowhttptest	Bwd Packets/s	$9{,}21 \times 10^{2}$	$5{,}47 \times 10^{3}$	493,56	13,79
DoS Slowhttptest	Flow IAT Mean	$1{,}00 \times 10^{7}$	$6{,}51 \times 10^{5}$	-93,51	11,60
DoS Slowhttptest	Flow IAT Std	$1{,}55 \times 10^{7}$	$1{,}40 \times 10^{6}$	-91,01	11,57
DoS slowloris	Flow IAT Std	$1{,}90 \times 10^{7}$	$5{,}08 \times 10^{6}$	-73,25	11,16
DoS slowloris	Flow IAT Mean	$1{,}09 \times 10^{7}$	$3{,}33 \times 10^{6}$	-69,31	8,52
DoS slowloris	Bwd Packets/s	$6{,}55 \times 10^{2}$	$1{,}69 \times 10^{-1}$	-99,97	6,43

PSI nas classes de ataque. Esse resultado indica mudanças significativas entre as distribuições observadas na CICIDS2017 e aquelas produzidas no ambiente controlado. As diferenças são especialmente relevantes porque os atributos analisados correspondem aos três atributos de maior importância para o modelo *Random Forest* selecionado.

No cenário DoS Hulk, os atributos temporais *Flow IAT Std* e *Flow IAT Mean* apresentaram reduções superiores a 97% em suas médias no ambiente controlado, enquanto o atributo *Bwd Packets/s* apresentou aumento expressivo. Nos cenários DoS Slowhttptest e DoS slowloris, também foram observadas mudanças elevadas nos atributos temporais e no comportamento de pacotes de retorno por segundo. Esses deslocamentos indicam que, embora os cenários tenham sido associados às mesmas classes DoS utilizadas no treinamento, os fluxos gerados nas máquinas virtuais não reproduziram a mesma distribuição dos atributos mais relevantes observada na CICIDS2017.

Dessa forma, os valores de PSI ajudam a contextualizar os resultados obtidos na matriz de confusão. A baixa detecção dos ataques no ambiente controlado pode estar relacionada à diferença entre os padrões aprendidos pelo modelo na CICIDS2017 e os padrões produzidos pelas ferramentas utilizadas no cenário experimental.

5.2.3 Discussão Sobre a Generalização do Modelo

A avaliação em ambiente controlado evidenciou uma diferença relevante entre o desempenho obtido na base CICIDS2017 e o comportamento do classificador em fluxos gerados fora dela. Embora a avaliação *offline* tenha apresentado métricas elevadas, os resultados práticos indicaram que o modelo não manteve a mesma capacidade de identificação nos cenários de ataque executados.

A análise dos atributos mais relevantes contribui para explicar esse comportamento. Os elevados valores de PSI indicam que as distribuições dos principais atributos utilizados pelo *Random Forest* se deslocaram de forma significativa entre a CICIDS2017 e o ambiente controlado. Assim, os fluxos gerados nas máquinas virtuais, embora associados às mesmas categorias de ataque, apresentaram características diferentes daquelas aprendidas pelo modelo durante o treinamento.

Esse comportamento é compatível com estudos sobre generalização em sistemas de detecção de intrusão baseados em aprendizado de máquina. Em avaliações *cross-dataset*, modelos que apresentam desempenho elevado quando treinados e testados em uma mesma base podem sofrer queda significativa quando aplicados a dados provenientes de outros ambientes, especialmente quando há heterogeneidade entre as distribuições dos dados (CANTONE; MARROCCO; BRIA, 2024).

Dessa forma, os resultados sugerem uma limitação de generalização do classificador. A ferramenta conseguiu executar o fluxo proposto de captura, extração, classificação e registro dos fluxos, mas o desempenho do modelo mostrou-se dependente da proximidade entre os dados de treinamento e o tráfego observado no ambiente de aplicação. Esse resultado reforça a importância de complementar avaliações *offline* com testes em cenários fora da base de dados.

6 Considerações Finais

Este trabalho investigou a aplicação de aprendizado de máquina na classificação de fluxos de rede associados a ataques de negação de serviço, buscando responder em que medida modelos treinados com a base CICIDS2017 são capazes de generalizar para fluxos coletados em um ambiente distinto daquele utilizado no treinamento. Para isso, foram avaliados modelos sobre a base CICIDS2017 e o classificador selecionado foi integrado a uma ferramenta de monitoramento e classificação contínua de tráfego.

Os resultados obtidos mostraram que o desempenho elevado em uma avaliação *offline* não implica, necessariamente, desempenho equivalente em um cenário de aplicação distinto. Embora o modelo *Random Forest* treinado com a distribuição original e ponderação automática das classes tenha apresentado bons resultados na base de teste da CICIDS2017, sua aplicação no ambiente controlado revelou baixa capacidade de identificação dos ataques gerados nas máquinas virtuais. Esse comportamento indicou uma limitação de generalização do classificador diante de fluxos produzidos fora da base utilizada no treinamento.

A análise dos atributos mais relevantes reforçou essa interpretação ao indicar diferenças significativas entre as distribuições observadas na CICIDS2017 e no ambiente controlado. Dessa forma, o trabalho evidenciou que a qualidade dos resultados obtidos em bases públicas deve ser analisada com cautela, principalmente quando modelos de detecção de intrusão são avaliados apenas em dados provenientes da mesma origem. A ferramenta desenvolvida permitiu observar esse comportamento em um fluxo de uso mais próximo de uma aplicação prática, ainda que em um ambiente experimental reduzido.

Entre as limitações deste trabalho, destacam-se o recorte restrito às classes DoS da CICIDS2017 e o uso de um ambiente controlado simplificado com poucas máquinas virtuais e baixa variação nos ataques. Também deve ser considerada a diferença entre as versões do CICFlowMeter utilizadas: enquanto a base CICIDS2017 foi gerada com a versão 3 da ferramenta, a implementação desenvolvida neste trabalho utilizou o CICFlowMeter v4, uma vez que a versão originalmente utilizada na construção da base não

se encontra mais disponível. Não foi encontrada documentação suficientemente detalhada para determinar se todos os atributos são calculados de maneira equivalente entre as versões. Portanto, essa diferença não pode ser apontada como causa confirmada da queda de desempenho observada. Ainda assim, ela constitui uma possível ameaça à compatibilidade entre os dados de treinamento e os fluxos extraídos no ambiente controlado. Outra limitação está relacionada à estratégia de divisão dos dados. Como o conjunto consolidado não preservava informações temporais ou identificadores da captura de origem, não foi possível realizar uma separação agrupada por execução ou período de coleta. Dessa forma, embora as duplicatas exatas tenham sido removidas, a divisão aleatória pode ter distribuído fluxos semelhantes entre os conjuntos de treino e teste.

Como trabalhos futuros, considera-se a avaliação de novos cenários de tráfego, de diferentes serviços de rede e de outras bases de dados. Além disso, a ferramenta desenvolvida pode ser aprimorada por meio da criação de uma interface gráfica, uma vez que, na versão atual, sua operação depende da execução de scripts. Essa evolução poderia tornar seu uso mais acessível, permitindo configurar parâmetros de captura, acompanhar classificações, visualizar alertas e consultar resultados de forma integrada. Outro ponto relevante consiste na utilização e na análise de dados coletados em ambientes reais de produção, a fim de avaliar o desempenho e o impacto da solução de IDS.

Declaração do Uso de IA

Declaro que utilizei ferramentas de Inteligência Artificial Generativa como apoio durante a elaboração deste trabalho. A ferramenta foi utilizada para auxiliar na revisão textual, organização de ideias, melhoria da clareza da escrita acadêmica e apoio à codificação, incluindo sugestões de implementação, revisão de trechos de código e auxílio na depuração de scripts.

O uso dessas ferramentas ocorreu de forma auxiliar, não substituindo a análise, a interpretação dos resultados, a condução dos experimentos, a validação das implementações nem a responsabilidade autoral sobre o conteúdo apresentado. Todas as informações, decisões metodológicas, códigos, resultados, discussões e conclusões foram revisados e validados pelo autor, que assume integral responsabilidade pelo conteúdo final deste trabalho.

Bibliografia

ABREU, D.; ABELéM, A. Sistema híbrido e on-line de detecção e classificação de tráfego malicioso. In: *Anais do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2022. p. 322–335. ISSN 2177-9384. Disponível em: ⟨https://sol.sbc.org.br/index.php/sbrc/article/view/21180⟩.

AITKEN, P.; CLAISE, B.; TRAMMELL, B. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*. RFC Editor, 2013. RFC 7011. (Request for Comments, 7011). Disponível em: ⟨https://www.rfc-editor.org/info/rfc7011⟩.

AKGUN, D.; HIZAL, S.; CAVUSOGLU, U. A new ddos attacks intrusion detection model based on deep learning for cybersecurity. *Computers Security*, v. 118, p. 102748, 2022. ISSN 0167-4048. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0167404822001432⟩.

AXELSSON, S. *Intrusion Detection Systems: A Survey and Taxonomy*. Göteborg, Sweden, 2000. Disponível em: ⟨https://www.cse.msu.edu/~cse960/Papers/security/axelsson00intrusion.pdf⟩.

BREIMAN, L. Random forests. *Machine Learning*, v. 45, n. 1, p. 5–32, Oct 2001. ISSN 1573-0565. Disponível em: ⟨https://doi.org/10.1023/A:1010933404324⟩.

BUCZAK, A. L.; GUVEN, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, v. 18, n. 2, p. 1153–1176, 2016.

CANTONE, M.; MARROCCO, C.; BRIA, A. Machine learning in network intrusion detection: A cross-dataset generalization study. *IEEE Access*, v. 12, p. 144489–144508, 2024.

CARL, G. et al. Denial-of-service attack-detection techniques. *IEEE Internet Computing*, v. 10, n. 1, p. 82–89, 2006.

CHAWLA, N. et al. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, v. 16, p. 321–357, 06 2002.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 785–794. ISBN 9781450342322. Disponível em: ⟨https://doi.org/10.1145/2939672.2939785⟩.

DRAPER-GIL, G. et al. Characterization of encrypted and vpn traffic using time-related features. In: INSTICC. *Proceedings of the 2nd International Conference on Information Systems Security and Privacy - ICISSP*. [S.l.]: SciTePress, 2016. p. 407–414. ISBN 978-989-758-167-0. ISSN 2184-4356.

FARHAT, S. et al. Evaluation of dos/ddos attack detection with ml techniques on cicids2017 dataset. In: INSTICC. *Proceedings of the 9th International Conference on Information Systems Security and Privacy - ICISSP*. [S.l.]: SciTePress, 2023. p. 287–295. ISBN 978-989-758-624-8. ISSN 2184-4356.

FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, Institute of Mathematical Statistics, v. 29, n. 5, p. 1189 – 1232, 2001. Disponível em: ⟨https://doi.org/10.1214/aos/1013203451⟩.

GÉRON, A.; RAVAGLIA, C. *Mãos à obra aprendizado de máquina com Scikit-Learn, Keras & TensorFlow: conceitos, ferramentas e técnicas para a construção de sistemas inteligentes*. Alta Books, 2021. ISBN 9788550815480. Disponível em: ⟨https://books.google.com.br/books?id=FcP0zwEACAAJ⟩.

Habibi Lashkari, A. et al. Characterization of tor traffic using time based features. In: INSTICC. *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP*. [S.l.]: SciTePress, 2017. p. 253–262. ISBN 978-989-758-209-7. ISSN 2184-4356.

HAN, J.; PEI, J.; TONG, H. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2022. (The Morgan Kaufmann Series in Data Management Systems). ISBN 9780128117613. Disponível em: ⟨https://books.google.com.br/books?id=NR1oEAAAQBAJ⟩.

HE, H.; GARCIA, E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, v. 21, n. 9, p. 1263–1284, 2009.

KALITA, J. K.; BHUYAN, M. H.; BHATTACHARYYA, D. K. Survey on incremental approaches for network anomaly detection. *International Journal of Communication Networks and Information Security (IJCNIS)*, v. 3, n. 3, dez. 2011. Disponível em: ⟨https://www.ijcnis.org/index.php/ijcnis/article/view/104⟩.

LIAO, H.-J. et al. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, v. 36, n. 1, p. 16–24, 2013. ISSN 1084-8045. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S1084804512001944⟩.

MASEER, Z. K. et al. Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset. *IEEE Access*, v. 9, p. 22351–22370, 2021.

MIRKOVIC, J.; REIHER, P. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, Association for Computing Machinery, New York, NY, USA, v. 34, n. 2, p. 39–53, abr. 2004. ISSN 0146-4833. Disponível em: ⟨https://doi.org/10.1145/997150.997156⟩.

NETO, M. S.; GOMES, D. G. Network intrusion detection systems design: A machine learning approach. In: *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2019. p. 932–945. ISSN 2177-9384. Disponível em: ⟨https://sol.sbc.org.br/index.php/sbrc/article/view/7413⟩.

SCARFONE, K. A.; MELL, P. M. *Guide to Intrusion Detection and Prevention Systems (IDPS)*. Gaithersburg, MD, 2007. Disponível em: ⟨https://doi.org/10.6028/NIST.SP.800-94⟩.

SHARAFALDIN, I.; Habibi Lashkari, A.; GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: INSTICC. *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP*. [S.l.]: SciTePress, 2018. p. 108–116. ISBN 978-989-758-282-0. ISSN 2184-4356.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. *Information Processing Management*, v. 45, n. 4, p. 427–437, 2009. ISSN 0306-4573. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0306457309000259⟩.

SONG, Y.-Y.; LU, Y. Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, v. 27, n. 2, p. 130–5, 2015.

TALUKDER, M. A. et al. Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. *Journal of Big Data*, v. 11, n. 1, p. 33, 2024. ISSN 2196-1115. Disponível em: ⟨https://doi.org/10.1186/s40537-024-00886-w⟩.

YURDAKUL, B.; NARANJO, J. Statistical properties of the population stability index. *Journal of Risk Model Validation*, 12 2020.