PEDRO HENRIQUE HAUY NETTO DE ARAUJO

Impacto de métodos de seleção de variáveis na classificação de ataques DDoS utilizando XGBoost

São Paulo
2023

PEDRO HENRIQUE HAUY NETTO DE ARAUJO

Impacto de métodos de seleção de variáveis na classificação de ataques DDoS utilizando XGBoost

Versão Corrigida

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

Área de Concentração:
Sistemas Eletrônicos

Orientador:
Sergio Takeo Kofuji

São Paulo
2023

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 11 de Setembro de 2023

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Araujo, Pedro Henrique Hauy Netto de
Impacto de métodos de seleção de variáveis na classificação de ataques DDoS utilizando XGBoost / P. H. H. N. Araujo -- versão corr. -- São Paulo, 2023.
94 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1.Inteligência artificial 2.Aprendizado computacional 3.Seleção de variáveis 4.Segurança de computadores 5.Ataques DDoS I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II.t.

Dedicatória

À minha mãe e ao meu pai.

AGRADECIMENTOS

Agradeço à todos os professores e funcionários do Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) da Escola Politécnica da Universidade de São Paulo pela oportunidade de ter realizado este projeto de pesquisa. Todas as disciplinas cursadas, artigos lidos, artigos escritos, programas desenvolvidos e trocas de conhecimento foram importantes nessa empreitada tão fascinante e desafiadora que é fazer ciência.

Em particular, agradeço ao professor Sergio Takeo Kofuji, pelo voto de confiança, pelas contribuições na redação desta dissertação e na forma de apresentá-la. Também agradeço muito ao professor Anderson Aparecido Alves da Silva pelo inestimável apoio em vários momentos desta pesquisa: desde a sua concepção, passando pelas inúmeras conversas para lapidar a sua execução. Estas trocas de opiniões e experiências foram fundamentais!

Agradeço à minha mãe, Márcia (*in memoriam*), meu maior exemplo em todos os aspectos da vida. Seu profundo amor, senso de justiça e determinação continuam sendo os maiores nortes que eu poderia ter. Se eu sempre gostei e sempre fui incentivado a progredir nos meus estudos, muito se deve à minha mãe. Também agradeço ao meu pai, Edgard (*in memoriam*), por seu carinho, apoio e exemplo de ser humano íntegro e honesto. Eu sou o resultado de vocês.

Agradeço à minha tia, Cláudia, pelos diversos tipos de suporte durante a realização deste projeto de pesquisa, além dos conselhos em várias áreas. Agradeço à minha irmã, Isabella, por todas as conversas, confidências e trocas de aprendizados desde sempre.

Agradeço ao meu amigo Vicente, irmão que a vida me deu, por todos os momentos de descontração e incentivo a toda forma de criatividade.

Agradeço à minha namorada, Denise, por ter me aberto um mundo de possibilidades e, ao mesmo tempo, ter me resgatado valores fundamentais. Por todo amor, pelas conversas que fluem e se entrelaçam por todos os assuntos do universo. Meu chameguim!

"Wir müssen wissen.
Wir werden wissen."

(Nós devemos saber.
Nós iremos saber.)

-- David Hilbert

RESUMO

ARAUJO, P. H. H. N. **Impacto de métodos de seleção de variáveis na classificação de ataques DDoS utilizando XGBoost**. 94 p., 2023. Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo, São Paulo, 2023.

Distributed Denial of Service (DDoS) *Attacks* - Ataques de Negação de Serviço Distribuídos - impõem um grande desafio para os sistemas de segurança atuais, dadas a variedade de suas implementações e as grandes escalas que podem atingir. Uma abordagem para a sua detecção precoce é o uso de técnicas de *Machine Learning* (ML), que criam regras de classificação do tráfego a partir de dados históricos. Contudo, diferentes tipos de dados contribuem de forma desigual para a assertividade do modelo treinado. O uso de métodos de *Feature Selection* (FS) - Seleção de Variáveis - como etapa de pré-processamento permite a identificação dos atributos mais relevantes para o problema em questão. Essa ação diminui o tempo de treinamento e pode melhorar o desempenho de classificadores de ataques quando variáveis ruidosas são eliminadas. Este trabalho toma como base um conjunto de dados público e o algoritmo XGBoost para mensurar o impacto de técnicas de FS no problema de classificação de ataques DDoS. São consideradas tanto técnicas independentes dos rótulos das amostras, como também métodos que se utilizam dessa informação para ranquear as variáveis em ordem de importância. O problema é analisado do ponto de vista da classificação binária e multiclasse. Também é criado um *benchmark* de métricas de classificação e tempos de execução. As comparações envolvem as métricas de Acurácia, Precisão, *Recall* e *F1-score* para diversos métodos de FS, além dos tempos para realização de FS e tempos para treinamento do modelo.

Palavras-Chave – Seleção de Variáveis, DDoS, XGBoost, Classificador Binário, Classificador Multiclasse.

ABSTRACT

ARAUJO, P. H. H. N. **Impact of feature selection methods on the classification of DDoS attacks using XGBoost**. 94 p., 2023. Dissertation (Masters) - Polytechnic School. University of São Paulo, São Paulo, 2023.

Distributed Denial of Service (DDoS) attacks impose a major challenge for today's security systems, given the variety of its implementations and the scale that the attacks can achieve. One approach for their early detection is the use of Machine Learning (ML) techniques, which create rules for classifying traffic from historical data. However, different types of data contribute unequally to the assertiveness of the trained model. The use of Feature Selection (FS) techniques as a pre-processing step allows identification of the most relevant features for the problem in question. This action reduces training time and can improve performance when noisy variables are eliminated. The current work is based on a public dataset and the XGBoost algorithm to measure the impact of FS techniques on the DDoS attack classification problem. Techniques that are independent of sample labels are considered, as well as methods that use this label information to rank the variables in order of importance. The problem is analyzed from the point of view of binary and multiclass classification. A benchmark of classification metrics and execution times is also created. The comparisons involve the metrics of Accuracy, Precision, Recall and F1-score for different FS methods, in addition to the times to perform FS and model training.

Keywords – Feature Selection (FS), DDoS, XGBoost, Binary Classifier, Multiclass Classifier.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Distributed Denial of Service (DDoS) *Attacks* - Ataques de Negação de Serviço Distribuídos - são cada vez mais frequentes e volumosos na Internet. Diariamente, milhares de ataques são disparados contra os mais diversos alvos: governos, empresas de *e-commerce*, operadoras de telecomunicações, distribuidores de conteúdo multimídia, entre outros (NETSCOUT, 2022a). As motivações para estes ataques são as mais distintas, tais como interesses econômicos, ativismo político ou mesmo curiosidade intelectual (MARVI; ARFEEN; UDDIN, 2021).

Atualmente é possível a contratação de ataques DDoS em direção a um alvo específico (NEWMAN, 2020). Os atacantes, que contam com inúmeros dispositivos infectados sob seu comando, cobram pela duração e volume dos disparos. Além dos prejuízos econômicos e sociais diretos pela interrupção dos serviços, as vítimas dos ataques também têm sua reputação e credibilidade severamente manchadas (LOWE, 2019).

A detecção de ataques DDoS pode ser realizada por Sistemas de Detecção/Prevenção de Intrusões (SDPI), que tradicionalmente se dividem em duas categorias: (1) detecção por assinatura; e (2) detecção por anomalia (BINDRA; SOOD, 2019). Ainda que assinaturas de ataques desenvolvidas por especialistas sejam capazes de identificar ameaças com grande assertividade, estas se tornam pouco eficazes contra ataques novos. Em contrapartida, o uso da detecção de anomalias é capaz de oferecer alguma proteção mesmo contra ataques *zero-day*. Uma das maiores dificuldades na implementação de sistemas de detecção de DDoS por anomalias está na minimização da ocorrência de alertas falso positivos ou falso negativos.

Há diversas técnicas de detecção de anomalias. Algumas são baseadas na comparação das correlações e ganhos (SILVA et al., 2017)(SILVA et al., 2019), outras têm como foco métodos de agrupamento (JUNIOR et al., 2019). Contudo, modelos de *Machine Learning* (ML) têm ganhado mais relevância graças a avanços na disponibilidade do poder computacional, software especializados e conjuntos de dados (*datasets*) públicos. A aplicação de modelos de ML para a detecção de ataques DDoS vem sendo discutida na literatura há

Figura 1: Mapa global de ataques DDoS

Fonte: ⟨https://horizon.netscout.com/⟩, acesso em 10.06.2022

algumas décadas (POLAT; POLAT; CETIN, 2020)(ALJAWARNEH; ALDWAIRI; YASSEIN, 2018)(GUPTA, 2018). O problema de detecção de anomalias tem sido atacado tanto pelo uso de aprendizado supervisionado (por exemplo, com classificadores), quanto não supervisionado.

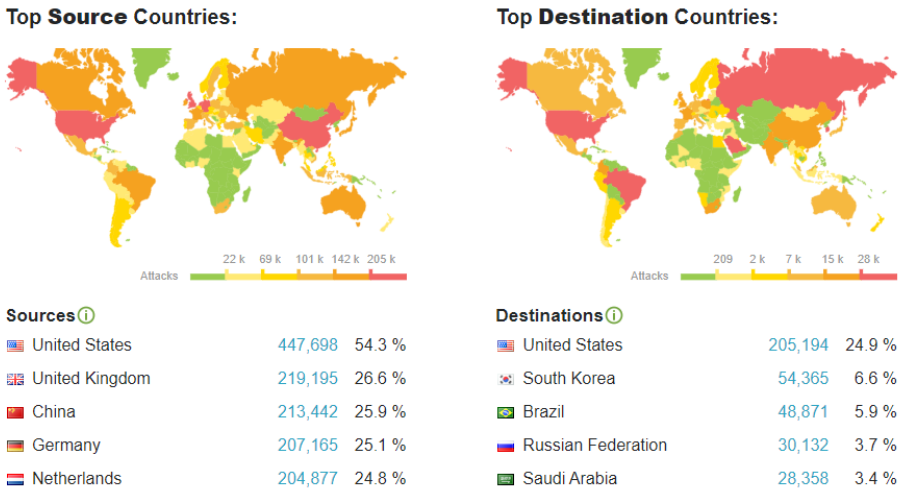
Contudo, o uso de *Feature Selection* (FS), que se trata do processo de seleção de variáveis (atributos, *features*) que compõem o modelo de ML, é um assunto explorado tangencialmente no que se refere à classificação de ataques DDoS. A utilização de FS pode trazer diversos benefícios, entre os quais: (1) detecção de ataques mais ágil; (2) menor necessidade de armazenamento e memória na implementação do classificador; e (3) crescimento na capacidade de interpretação do modelo gerado (POST; PUTTEN; RIJN, 2016). Além disso, o uso de FS pode contribuir para a economia de energia de dispositivos que treinam ou consomem modelos de ML (WANG; YAO; LIU, 2019).

1.1 Motivação

A Figura 1, presente na página web da empresa NETSCOUT (NETSCOUT, 2022b), ilustra uma série de ataques DDoS em andamento ao redor do planeta. Qualitativamente, pode-se observar na Figura 1 o volume de disparos em curso a todo momento (sobretudo em direção aos Estados Unidos).

Quantitativamente, é mostrada, na Figura 2, a distribuição das principais origens e

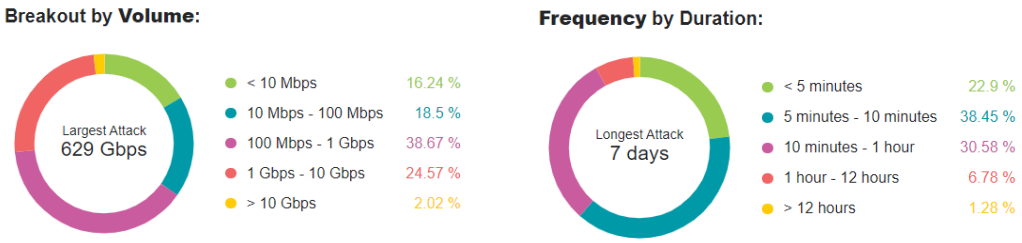
Figura 2: Distribuição de países de origem e destino de ataques DDoS



Fonte: ⟨https://horizon.netscout.com/?atlas=summary⟩, acesso em 10.06.2022

Figura 3: Estatísticas de ataques DDoS

(a) Volume (b) Duração

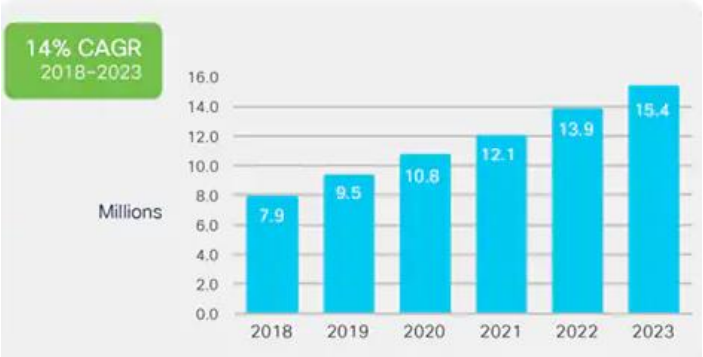


Fonte: ⟨https://horizon.netscout.com/?atlas=summary⟩, acesso em 10.06.2022

destinos de ataques DDoS, retirada da página de estatísticas da NETSCOUT (NETSCOUT, 2022a). Os Estados Unidos são o principal alvo dos ataques, mas também estão envolvidos no disparo de mais da metade deles. Note que o Brasil é o terceiro país que mais sofre ataques DDoS no mundo. Na Figura 3, à esquerda, é possível observar que o volume da maior parte dos ataques é inferior à 1 Gbps e corresponde a velocidades de conexão normalmente disponíveis para usuários residenciais em muitas regiões. Além disso, à direita na Figura 3, pode-se observar que $61,35\%$ destes tem duração curta, inferior a 10 minutos, podendo indicar que muitos destes ataques são resultados de experimentações por parte dos atacantes.

Os volumes mais comuns de ataques, descritos anteriormente, podem ser mitigados por provedores de Internet, que contam com produtos de Anti-DDoS em seu catálogo.

Figura 4: Têndencia de crescimento do número de ataques DDoS



Fonte: ⟨https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html⟩, acesso em 14.06.2022

Contudo, ao longo da história, diversos atacantes se utilizaram de técnicas sofisticadas, tirando do ar serviços importantes (NICHOLSON, 2022). Em 2020, o Google divulgou que sua infraestrutura havia sofrido ataques de amplificação UDP (*User Datagram Protocol*) que, no pico, chegaram a 2,5 Tbps. Também em 2020, se aproveitando de vulnerabilidades em servidores CLDAP (*Connectionless Lightweight Directory Access Protocol*), houve um ataque à AWS (*Amazon Web Services*, serviço de nuvem da Amazon) que atingiu a marca de 2,3 Tbps. Em 2016, a plataforma de DNS (*Domain Name System*) Dyn foi atacada por uma *botnet* com mais de 10 milhões de endereços IP (*Internet Protocol*) de origem distintos, provenientes de dispositivos IoT (*Internet of Things*) infectados com o *malware* Mirai, como câmeras IP e roteadores domésticos. Este ataque, em seu pico, chegou ao volume de 1 Tbps. Recentemente, a Microsoft anunciou que havia sofrido um ataque de 3,47 Tbps (TUNG, 2022), o maior registrado até o momento.

Em seu relatório anual de tendências, a Cisco estima um crescimento composto anual de 14% no número total de ataques DDoS no mundo entre 2018 e 2023 (CISCO, 2018). Ao longo de 2023, a expectativa é que sejam realizados 15,4 milhões de ataques, como ilustrado na Figura 4. Entre 2018 e 2019, o volume médio dos ataques já era de 1 Gbps. No mesmo período, a ocorrência de ataques entre 100 Gbps e 400 Gbps aumentou 776%. Também neste período o máximo volume registrado cresceu 63%.

Qualquer iniciativa no sentido de detectar precocemente ataques DDoS e mitigá-los é, portanto, de grande valia para as partes envolvidas. Nesse sentido, o uso de métodos de FS pode auxiliar a encurtar os tempos de treinamento de modelos de ML e torná-los mais assertivos (POST; PUTTEN; RIJN, 2016).

1.2 Objetivo

O objetivo deste trabalho é, portanto, mensurar o impacto de técnicas de FS no problema da classificação de ataques DDoS utilizando ML. A proposta é fixar um algoritmo classificador e avaliar a influência de métodos de FS em métricas de desempenho. O uso de classificadores pressupõe um conjunto de dados rotulado, que é prospectado no decorrer do trabalho de pesquisa. Em particular, neste contexto, são analisados:

- O impacto de métodos de FS que **independem** do rótulo das linhas do *dataset*;
- O impacto de métodos de FS que **dependem** do rótulo das linhas do *dataset*;
- O tempo de execução do treinamento do modelo (incluindo FS e o classificador propriamente dito).

Como principais contribuições, pretende-se disponibilizar gráficos e tabelas comparativas em que são mensurados os impactos de técnicas de FS no problema de classificação de ataques DDoS. O trabalho foca em um classificador e um *dataset* específico, mas a metodologia é generalista o suficiente para que possa ser aplicada em outros cenários. Para auxiliar nesta aplicação da metodologia em trabalhos correlatos futuros, o código fonte desenvolvido para as simulações ficará disponível para todos os interessados.

1.3 Metodologia

Com o intuito do cumprimento dos objetivos levantados, é aplicada a metodologia descrita a seguir: uma revisão bibliográfica ampla é realizada sobre técnicas de detecção de ataques DDoS publicadas na literatura recente. Tornando a análise um pouco mais específica, são pesquisadas técnicas de detecção de ataques DDoS **que utilizam ML** para esse intuito. Os principais algoritmos, conjuntos de dados e métodos de pré-processamento (tais como o uso de FS) são analisados nesta etapa. Com este panorama geral levantado, é escolhido um *dataset* em particular e são pesquisados trabalhos correlatos que se utilizam deste mesmo conjunto de dados.

Feito este levantamento inicial, é proposta uma arquitetura de experimentos computacionais que tenham como resultado a quantificação da influência de FS no problema da classificação de ataques DDoS. O problema pode ser atacado do ponto de vista binário (determinar se uma nova amostra é um ataque ou tráfego normal) ou multiclasse (determinar **qual ataque** estamos enfrentando ou se trata de tráfego normal). Este trabalho

explora as duas abordagens. Também são escolhidas, com base na literatura, as principais métricas de qualidade de uma classificação, tais como Acurácia, Precisão, *Recall* e *F1-Score*. Estas métricas podem ser comparadas com trabalhos de outros pesquisadores que usaram o mesmo conjunto de dados, para que, assim, sejam identificados pontos fortes e fracos nas diversas implementações dos sistemas de detecção. De uma maneira mais específica ao hardware disponível, mas também quantificável, são registrados os tempos de execução de todas as simulações computacionais realizadas.

São estudadas e aplicadas uma série de técnicas de FS. Seu impacto é analisado, tanto do ponto de vista das métricas de desempenho, quanto da velocidade de treinamento. Como resultados esperados ao final do desenvolvimento e execução das simulações, pretende-se observar até que ponto é vantajosa a remoção de *features* dos modelos obtidos. Há uma escolha a ser feita entre a qualidade das métricas e tempo de ajuste dos modelos, sendo que este trabalho se propõe a mensurar este custo-benefício.

1.4 Organização do Trabalho

O Capítulo 2 apresenta o Referencial Teórico do trabalho, contextualizando conceitos de Segurança da Informação (como o que são ataques DDoS e formas de executá-los), ML, FS e, mais especificamente, uma breve descrição das técnicas de FS que são utilizadas no decorrer do trabalho.

O Capítulo 3 traz Trabalhos Relacionados ao problema de detecção de ataques DDoS na literatura. Em particular, é introduzido o *dataset* CICDDoS2019 (SHARAFALDIN et al., 2019), que serve como base para os experimentos realizados.

O Capítulo 4 detalha a Proposta de Simulação Computacional, incluindo o ambiente utilizado, a arquitetura de dados e as métricas de qualidade da classificação.

O Capítulo 5 apresenta os Resultados dos experimentos através de tabelas e gráficos. Aqui o balanceamento do conjunto de dados escolhido é discutido, assim como a aplicação de todas as técnicas de FS apresentadas ao longo do trabalho: tanto independentes, como dependentes do rótulo das amostras.

Por fim, o capítulo 6, Conclusão, retoma as principais contribuições do projeto e sugere trabalhos futuros.

2 REFERENCIAL TEÓRICO

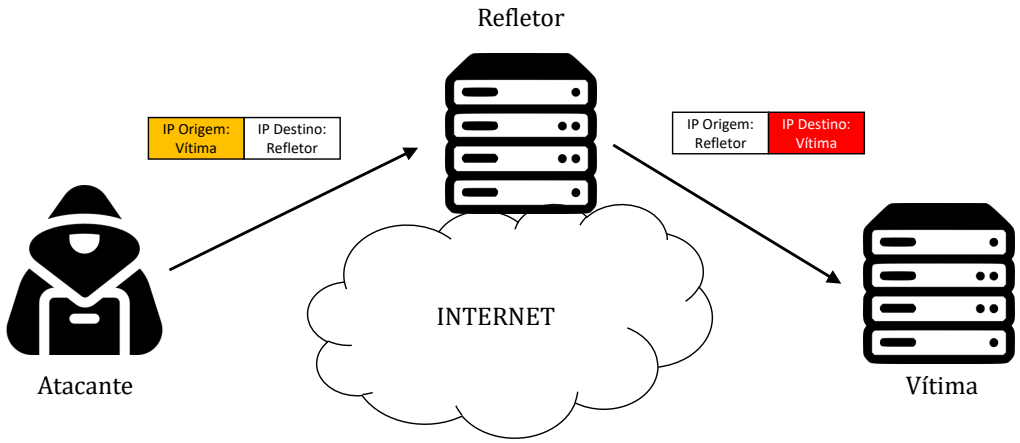
Neste capítulo, do ponto de vista de Segurança da Informação, é apresentada a ideia de ataques DDoS e seus vetores de atuação. Uma vez que técnicas de ML vêm ganhando espaço na detecção de ataques DDoS, é importante definir as principais características deste tipo de abordagem, o que também é feito no decorrer do capítulo. Por fim, para embasar os experimentos realizados neste trabalho, são apresentados o algoritmo XGBoost (usado como classificador) e diversas técnicas de FS.

2.1 Ataques DDoS

Denial of Service (DoS) *Attacks* - Ataques de Negação de Serviço - têm por objetivo impedir que usuários legítimos tenham acesso a um serviço computacional. Procuram exaurir recursos de um servidor, sejam sua memória, processador ou largura de banda de suas interfaces de rede, de modo que este não consiga mais responder a novas requisições (NOORIBAKHSH; MOLLAMOTALEBI, 2020). Atacantes podem usar técnicas de *spoofing* (isto é, modificam o endereço IP) para ocultar sua origem e dificultar sua detecção (BELLOVIN, 2004). Sua intenção **não é o acesso** sem autorização a outras máquinas e serviços pelo atacante, mas negar esta possibilidade de acesso aos usuários legítimos (TANENBAUM; WETHERALL, 2011). Para isso, uma grande quantidade de pacotes é disparada em direção a um servidor alvo ou uma vulnerabilidade de algum protocolo que este esteja executando é explorada (YUSOF; UDZIR; SELAMAT, 2019).

A versão distribuída dos ataques DoS, chamada de *Distributed Denial of Service* (DDoS) *Attacks*, é ainda mais nociva para as vítimas. Neste modo de operação, o ataque provém de múltiplos dispositvos, simultaneamente. Além de aumentar o volume dos disparos, seja em bps (bits por segundo) ou em pps (pacotes por segundo), este tipo de ataque reduz as chances de detecção, justamente por contar com um grande número de origens (TANENBAUM; WETHERALL, 2011). Muitas vezes, os usuários destes dispositivos não se dão conta que estão sendo usados para enviar requisições contra um alvo

Figura 5: Ataque de Reflexão



(DUNHAM; MELNICK, 2008). Suas máquinas, infectadas com algum tipo de *malware*, podem ser controladas remotamente pelo atacante e efetuar disparos em direção ao servidor vítima. Este conjunto de dispositivos a serviço do atacante é chamado de *botnet* (ou, ainda, de zumbis) (GUPTA, 2018).

Os ataques DDoS também podem contar com nós intermediários para realizar o envio de pacotes. Servidores públicos de DNS ou NTP (*Network Time Protocol*), por exemplo, respondem a requisições de qualquer usuário e as retornam ao endereço IP que as solicitou. Um atacante pode forjar um pacote com o **IP de origem da vítima** e o IP de destino sendo um destes servidores públicos. O pacote de retorno é, então, enviado (refletido) para a máquina da vítima, dando nome a este tipo de ataque, Reflexão (NOORIBAKHSH; MOLLAMOTALEBI, 2020). Este processo é representado esquematicamente pela Figura 5. Alguns protocolos têm, por caraterística de seu funcionamento normal, respostas com um volume maior de dados do que as solicitações que as originaram. Ao usar este tipo de servidor (com respostas mais volumosas que as requisições) como nó intermediário, tem-se um ataque por Amplificação (DHINGRA; SACHDEVA, 2018).

Existem diversas maneiras de se categorizar os tipos de ataques DDoS. O *National Institute of Standards and Technology* (NIST), orgão do governo americano responsável pela regulação de padrões e medidas em seu território, sugere a divisão entre ataques de inundação (*flooding*) e ataques de exploração de falhas (KIOURKOULIS, 2020). A principal diferença é que os ataques de inundação têm como alvo a rede que suporta o sistema. Seu objetivo é saturar interfaces de rede com um grande volume de tráfego (principalmente através de protocolos de camada 3 e 4), impedindo que usuários legítimos acessem os recursos desejados (ELSAYED et al., 2020). Já ataques de exploração de falhas

têm como alvo o *software* do sistema, procurando consumir seus recursos computacionais, como processamento e memória. Estes ataques são mais sofisticados e, muitas vezes, não necessitam de muita largura de banda para serem executados.

Uma classificação semelhante é a que divide ataques DDoS em ataques de volumetria, protocolo e aplicação (LOPEZ; MOHAN; NAIR, 2019). Ataques de **volumetria** geram um grande volume de tráfego em direção ao alvo com o objetivo de exaurir suas interfaces de rede. Sua intensidade é medida em bits por segundo (bps). Ataques de **protocolo** se aproveitam de alguma vulnerabilidade em protocolos de comunicação (como o TCP) para consumir recursos de servidores ou equipamentos intermediários de uma rede (como *firewalls* ou balanceadores de carga). Sua intensidade é medida em pacotes por segundo (pps). Por fim, ataques de **aplicação** procuram imitar o comportamento de um usuário comum e exploram vulnerabilidades nesta camada. Atravessam o bloqueio de equipamentos de camada 3 e 4, atingindo a aplicação e fazendo diversas requisições à ela (como chamadas HTTP ou *queries* a bancos de dados). Sua intensidade é medida em requisições por segundo (rps).

Dados os altos prejuízos (tangíveis e intangíveis) que uma organização corre o risco de sofrer ao ser vítima de um ataque DDoS, estratégias de prevenção, detecção e mitigação destes ataques são objetos de estudo recorrentes da indústria e da academia (LIANG; ZNATI, 2019). Métodos de **prevenção** não impedem que ataques DDoS ocorram, mas impõem barreiras que dificultam as investidas dos atacantes. Reduzir a superfície de ataque pode ajudar a concentrar os esforços de mitigação em algum pontos da rede. Além disso, o uso de *firewalls* e listas de controle de acesso são importantes filtros para as requisições que chegam no sistema. Em última análise, aumentar a largura de banda disponível também contribui para que, mesmo em situação de ataque, o sistema consiga absorver parte deste tráfego e não se torne indisponível.

O ponto em que será implementado um sistema de **detecção** também é de grande interesse para as partes envolvidas, podendo estar mais próximo da vítima ou da fonte dos ataques. Sistemas de detecção próximos das redes vítimas são mais comuns, mas acabam consumindo mais recursos de rede destas. Detectar e bloquear o ataque mais próximo da fonte deste é mais eficiente, porém muito mais difícil de se operacionalizar (NOORIBAKHSH; MOLLAMOTALEBI, 2020). Isto se deve tanto à natureza distribuída dos ataques (que contam com diversas fontes), quanto a necessidade de cooperação de todos os roteadores intermediários do tráfego (o que é difícil de se garantir, na prática).

Uma estratégia de **mitigação** eficiente deve ser capaz de diferenciar o tráfego normal

do tráfego malicioso (TUAN et al., 2020a). Desta forma, usuários legítimos não têm seus acessos ao serviço negados. Após o fluxo atacante ser detectado como tal, sua mitigação pode consistir no seu descarte, roteando-o para um *blackhole*, por exemplo (ELSAYED et al., 2020). Outra alternativa é seu encaminhamento para um servidor *honeypot*, para que mais investigações sejam feitas.

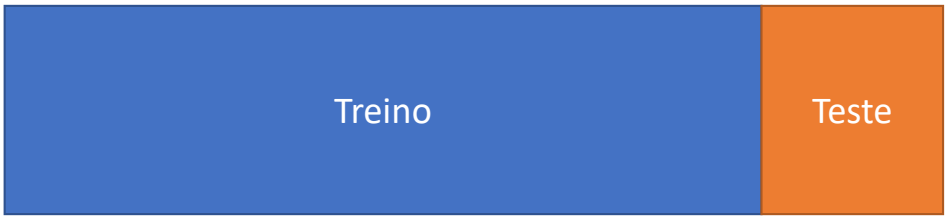
Outra opção para absover o tráfego de ataque é o uso de *Content Delivery Networks* (CDNs) - Redes de Distribuição de Conteúdo. Trata-se de uma arquitetura em que o conteúdo estático de uma aplicação é servido o mais próximo possível do usuário. Para isso, esses componentes da aplicação são replicados em servidores distribuídos geograficamente. Desta maneira, a carga no servidor de origem do conteúdo diminui. Quando ocorre um ataque em direção à aplicação, este afetará os servidores mais próximos do atacante (IMTHIYAS et al., 2020). Neste cenário, o servidor central se mantém disponível mesmo com o serviço sob ataque.

2.2 *Machine Learning*

O uso de técnicas de *Machine Learning* (ML) consiste em fornecer exemplos para um agente computacional para que este crie uma representação interna do conjunto de dados em questão. O agente pode, desta maneira, aprender com os dados. Difere-se, portanto, da programação tradicional, em que o desenvolvedor cria explicitamente regras para a tomada de decisão e, ao invés disso, se baseia em algoritmos de reconhecimento de padrões nos dados disponíveis. Há diversos motivos para se atacar um problema pela abordagem de ML (NORVIG; RUSSELL, 2014): (1) o projetista do sistema pode não conseguir antecipar todas as situações possíveis em que o agente estará inserido; (2) a distribuição dos dados pode mudar com o tempo (*data-drift*) e a programação inicial ter seu desempenho prejudicado; (3) o programador humano pode não ter ideia de como desenvolver o sistema, ou seja, quais regras são relevantes para alcançar seu objetivo. Todos estes pontos são cruciais no desenvolvimento de um SDPI, em particular no contexto de detecção de ataques DDoS: novos ataques surgem constantemente na Internet; e é importante que sistemas de Segurança possam reagir a este tipo de anomalia assim que seja detectada.

Algoritmos de ML se dividem em três grandes tipos (MURPHY, 2012). O objetivo do Aprendizado Supervisionado está em estimar uma função que transforme um conjunto de dados de entrada em dados de saída através de exemplos **rotulados** que lhe são fornecidos. Recebe um conjunto $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, denominado **conjunto de treino**,

Figura 6: Divisão do *dataset* entre treino e teste (*hold-out*)



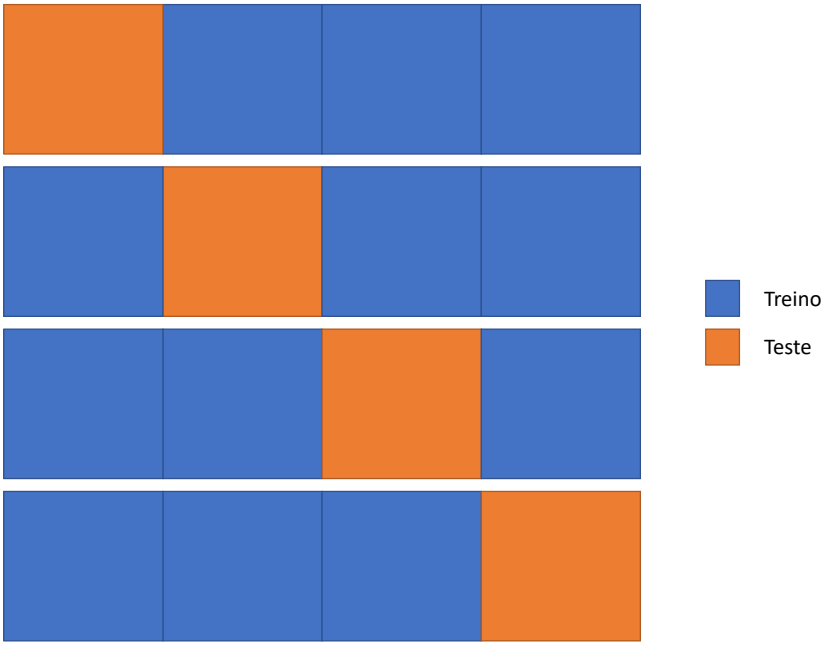
com N amostras (exemplos, elementos), em que cada $\mathbf{x}_i$ é um vetor multidimensional, por vezes podendo contar com um número muito elevado de *features*, e está acompanhado do rótulo y_i no espaço de saída. Algoritmos de Aprendizado Supervisionado procuram criar um mapeamento $\hat{f}(\mathbf{x}_i) = \hat{y}_i$ de maneira que $\hat{y}_i$ (a estimativa) e y_i (o dado fornecido) estejam "próximos" segundo uma função custo, ou seja, minimizar o erro da estimação.

O segundo tipo de algoritmo é o que executa o Aprendizado Não-Supervisionado. Nele, os dados de entrada não são rotulados e o objetivo é encontrar padrões interessantes, como agrupamentos e novas formas de visualizar os dados. Recebe, portanto, apenas um conjunto $D = \{\mathbf{x}_i\}_{i=1}^{N}$ de amostras. O terceiro tipo é o Aprendizado por Reforço, em que o agente aprende uma tarefa através de recompensas e punições, interagindo com o ambiente. Trata-se de uma forma de aprendizado significativamente diferente das anteriores (Supervisionado e Não-Supervisionado) (MURPHY, 2012).

Espera-se que modelos de Aprendizado Supervisionado sejam capazes de generalizar para além dos exemplos do conjunto de treino. Para se medir o desempenho do processo de treinamento, é separada uma parcela do *dataset*, chamada de **conjunto de teste** (NORVIG; RUSSELL, 2014). Este processo também é chamado de *hold-out* e é ilustrado na Figura 6: nesta figura, é mostrado que uma parte do conjunto de dados de entrada é usada para treinar o modelo (Treino) e outra para avaliá-lo (Teste). As amostras do conjunto de teste "nunca foram vistas" pelo modelo até o momento da avaliação e o desempenho neste conjunto serve de estimativa para a capacidade de generalização do modelo.

As variáveis usadas por um algoritmo de ML podem ser quantitativas ou qualitativas. Dentro da Aprendizagem Supervisionada, aos problemas cujo objetivo é estimar o valor de uma variável quantitativa, dá-se o nome de **regressão**. A estimação de uma variável qualitativa (uma classe, categoria) recebe o nome de **classificação** (JAMES et al., 2013). Caso a classificação se dê entre apenas duas categorias, esta recebe o nome de **classificação binária**. Quando a tarefa é estimar uma classe dentro de um conjunto

Figura 7: Exemplo de *K-Fold Cross-Validation* com $K = 4$



$C = \{C_i\}_{i=1}^{S}$ (com $S > 2$), esta é chamada de **classificação multiclasse**. O problema da classificação de ataques DDoS pode ser abordado tanto do ponto de vista binário, em que é determinado se uma amostra é um ataque ou tráfego normal, ou ainda do ponto de vista multiclasse, em que é predito **qual ataque** (dentro de um conjunto finito) que uma amostra de tráfego representa.

O modo como o processo de *hold-out* é realizado pode influenciar fortemente nas métricas de desempenho. Dependendo do conjunto particular de amostras que é separado para teste, os resultados podem ser melhores ou piores, incorrendo em grande variância das métricas (JAMES et al., 2013). Como alternativa, o *dataset* pode ser dividido em K grupos, para que o processo de treino e teste seja repetido K vezes, de modo a não privilegiar nenhum subgrupo do conjunto de dados. As métricas são, então, tomadas como sendo a média dos resultados de cada repetição. Esta técnica é chamada de *K-Fold Cross-Validation* (CV) e é ilustrada na Figura 7 para o caso particular em que $K = 4$.

Modelos de ML aprendem parâmetros numéricos a partir dos dados a que são expostos no processo de treinamento. Contudo, existem parâmetros que são incluídos no modelo a partir de decisões de projeto que ocorreram antes do treinamento e impactam diretamente no desempenho do sistema final. Estes recebem o nome de **hiperparâmetros** (NORVIG; RUSSELL, 2014). Podem ser escolhidos por conhecimento de domínio ou através de técnicas sistemáticas.

A saída de um classificador multiclasse é um conjunto de probabilidades de pertencimento a cada uma das categorias consideradas. Ou seja, se trata do conjunto $\hat{y}_i = \{\hat{y}_{i1}, \hat{y}_{i2}, \ldots, \hat{y}_{iS}\}$ para cada amostra i e para as S classes. Para se treinar os classificadores, algoritmos são desenvolvidos de forma a minimizar uma função custo. Uma escolha popular de função custo é a *Categorical Cross-Entropy*, dada por (HASTIE; TIBSHIRANI; FRIEDMAN, 2009):

$$L(y_i, \hat{y}_i) = y_{i1} \ln \hat{y}_{i1} + y_{i2} \ln \hat{y}_{i2} + \ldots + y_{iS} \ln \hat{y}_{iS} = \sum_{j=1}^{S} y_{ij} \ln \hat{y}_{ij} \tag{2.1}$$

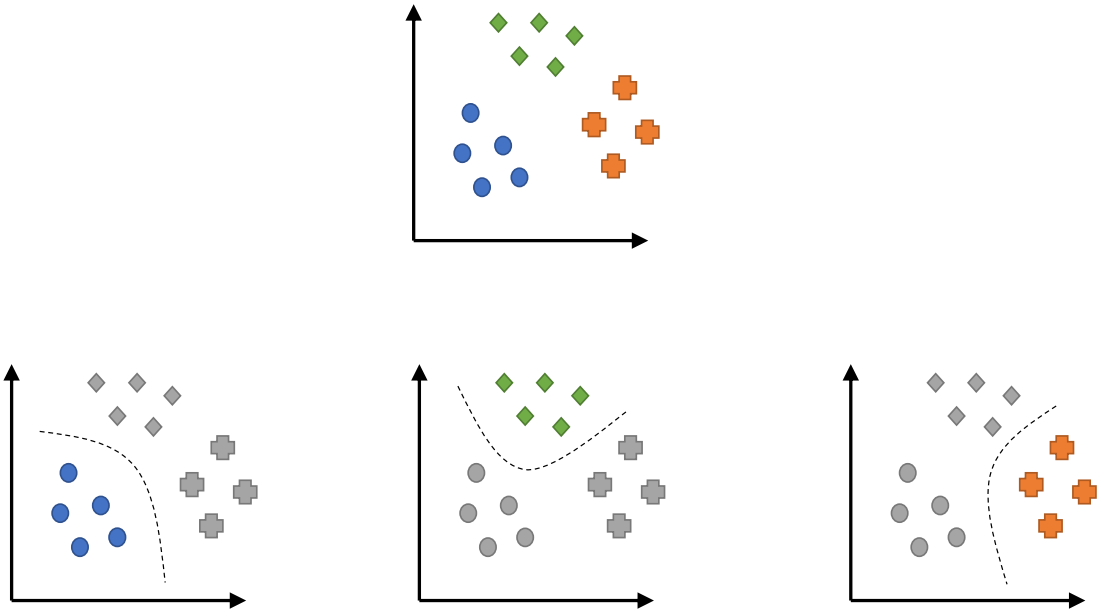
em que cada y_i é um vetor *one-hot-encoded* (isto é, seus elementos só podem assumir os valores 0 ou 1) dos dados da variável alvo original. A Equação 2.1 também é válida para o caso binário (com $S = 2$), passando a ser chamada de *Binary Cross-Entropy*.

Ainda que, tradicionalmente, a *Categorical Cross-Entropy* seja a função custo de classificadores multiclasse, também é possível treinar este tipo de modelo a partir de diversos classificadores binários. Para isso, são usadas estratégias de meta-aprendizado, como *One-Vs-Rest* (OvR) (MURPHY, 2012) e *One-Vs-One* (OvO) (BISHOP; NASRABADI, 2006).

A estratégia OvR consiste em transformar o problema de classificação entre S classes em S problemas de classificação binária. Cada classificador determina se a amostra pertence a uma certa classe C_i ou não (e, neste último caso, pertence ao "resto") (MURPHY, 2012). A Figura 8 ilustra esta estratégia: nela, cada classe é representada por uma cor e cada ponto é uma amostra plotada em um espaço bidimensional. As amostras em cinza representam a classe "resto". Cada classificador binário retorna uma probabilidade e o classificador com maior probabilidade indica a classe predita para a amostra.

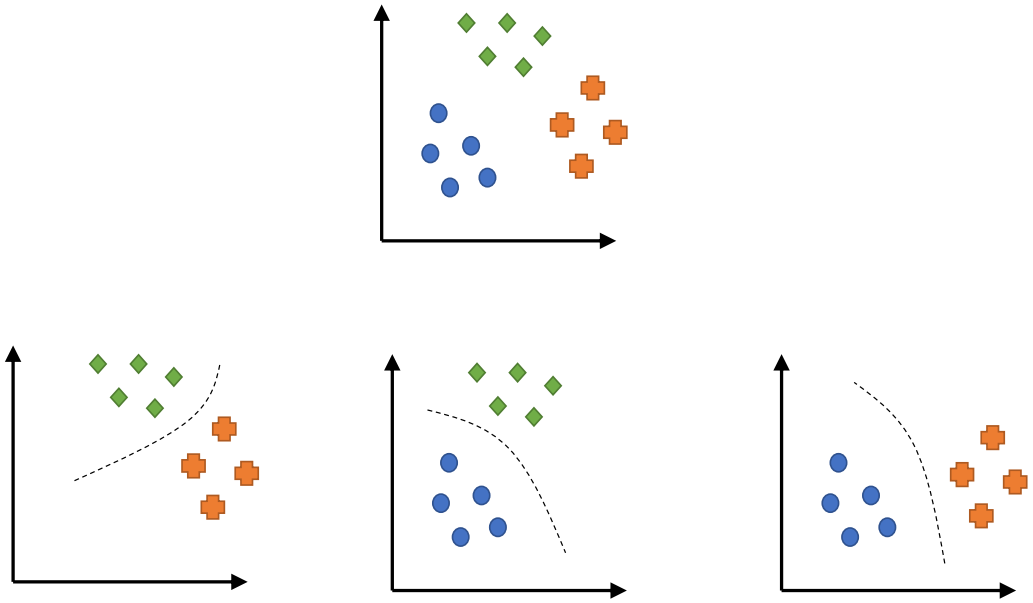
A estratégia OvO demanda um maior poder computacional, uma vez que esta compara todas as combinações de 2 elementos possíveis entre as S classes (BISHOP; NASRABADI, 2006). Isto dá um total de $S(S-1)/2$ classificadores binários. Cada classificador vota em uma classe, entre as duas que está comparando. A classe com maior número de votos é a escolhida pelo modelo. A Figura 9 ilustra a estratégia OvO. É possível observar que para o treinamento do modelo, apenas um par de classes é considerado por vez e todos os elementos das demais classes são ignorados.

Figura 8: Representação esquemática da estratégia One-vs-Rest



Fonte: Adpatado de (ZHANG et al., 2016)

Figura 9: Representação esquemática da estratégia One-vs-One



Fonte: Adpatado de (ZHANG et al., 2016)

2.3 XGBoost

O algoritmo *eXtreme Gradient Boosting* (XGBoost) (CHEN; GUESTRIN, 2016) realiza previsões através de conjuntos (arranjos, *ensembles*) de Árvores de Decisão. As Árvores de Decisão são treinadas sequencialmente, de forma que a árvore atual a ser inserida no modelo procure minimizar o erro que árvores anteriores cometeram ao aproximar a função que descreve os dados (CHEN et al., 2018), sendo esta técnica conhecida como *boosting*. Desta maneira, produz um modelo de previsão forte através da combinação de modelos fracos. Seja $\hat{f}_m(\mathbf{x}_i)$ a m-ésima Árvore de Decisão inserida no arranjo, $\hat{y}_i^{(m)}$ o resultado total da estimação após m iterações e F_0 uma constante de inicialização do algoritmo (escolhida para minimizar uma função custo). Depois de M rodadas de treinamento, é obtido (CHERIF; KORTEBI, 2019):

$$\begin{aligned}
\hat{y}_i^{(0)} &= \hat{f}_0(\mathbf{x}_i) = F_0 \\
\hat{y}_i^{(1)} &= \hat{y}_i^{(0)} + \hat{f}_1(\mathbf{x}_i) = F_0 + \hat{f}_1(\mathbf{x}_i) \\
\hat{y}_i^{(2)} &= \hat{y}_i^{(1)} + \hat{f}_2(\mathbf{x}_i) = F_0 + \hat{f}_1(\mathbf{x}_i) + \hat{f}_2(\mathbf{x}_i) \\
\hat{y}_i^{(3)} &= \hat{y}_i^{(2)} + \hat{f}_3(\mathbf{x}_i) = F_0 + \hat{f}_1(\mathbf{x}_i) + \hat{f}_2(\mathbf{x}_i) + \hat{f}_3(\mathbf{x}_i) \\
&\vdots \\
\hat{y}_i^{(M)} &= \hat{y}_i^{(M-1)} + \hat{f}_M(\mathbf{x}_i) = \sum_{m=0}^{M} \hat{f}_m(\mathbf{x}_i)
\end{aligned} \tag{2.2}$$

As novas árvores $\hat{f}_m(\mathbf{x}_i)$ são criadas de maneira que a atualização da estimação vá contra o gradiente de uma função custo. A função custo escolhida contém termos de regularização para que, assim, modelos muito complexos sejam penalizados. Esta é dada por (CHEN; GUESTRIN, 2016):

$$\begin{aligned}
\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) &= \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \sum_{m=0}^{M} \Omega(\hat{f}_m) \\
\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) &= \underbrace{\sum_{i=1}^{N} L(y_i, \hat{y}_i)}_{\text{Perda}} + \underbrace{\sum_{m=0}^{M} \alpha F_m + \frac{1}{2}\lambda \|w_m\|^2}_{\text{Regularização}}
\end{aligned} \tag{2.3}$$

Onde $\mathbf{y}$ e $\hat{\mathbf{y}}$ são, respectivamente, os vetores dos rótulos de todas as amostras e todas as estimações, $L(.)$ é uma função diferenciável (como a da Equação 2.1, por exemplo), F_m

é o número de folhas de cada árvore e w_m a saída de cada folha. É possível interpretar o parâmetro α como controle de uma regularização LASSO (*Least Absolute Shrinkage and Selection Operator*) e λ de uma regularização Ridge (OMAR, 2018).

2.4 *Feature Selection*

Os algoritmos de FS estão inseridos no contexto de redução de dimensionalidade. Seu objetivo é encontrar um subconjunto de variáveis de entrada, de modo que estas estejam mais próximas da variável rótulo das observações e mais distantes entre si (CHANDRASHEKAR; SAHIN, 2014). Neste caso, há várias formas de se definir distância, cada uma mais adequada para situações epecíficas (GUYON; ELISSEEFF, 2003). Uma técnica de FS se caracteriza pela escolha de um subconjunto de variáveis, dentre as variáveis originais, sem a realização de qualquer transformação ou criação de novas variáveis. Difere, portanto, de técnicas de *Feature Extraction* (FE) - Extração de Variáveis - como *Principal Component Analysis* (PCA) ou *Autoencoders*, que levam os atributos de entrada para um espaço diferente do original (KWON et al., 2019). Uma consequência indesejada de métodos que transformam as variáveis originais é a falta de interpretabilidade das novas *features*. No caso de métodos de FS, a interpretação semântica das variáveis é preservada.

Em dados tabulares, FS pode ser vista como uma forma de redução horizontal do conjunto de dados (isto é, executa a remoção de colunas). Seu contraponto seria a redução vertical do *dataset*, dada por processos de amostragem, que removem linhas do conjunto de dados (SILVA et al., 2021). Um ponto a ser observado é a chamada estabilidade do método de FS (CHANDRASHEKAR; SAHIN, 2014). Isto é: caso seja alimentado com amostras diferentes de um mesmo conjunto de dados, as variáveis selecionadas pela técnica serão as mesmas? Existem procedimentos quantitativos para se mensurar a estabilidade de um método de FS (AWADA et al., 2012).

Os métodos de FS podem ser divididos em: (1) filtros; (2) *wrappers*; (3) *embedded*; e (4) híbridos (WANG et al., 2019). O uso de filtros envolve a comparação de uma única variável com o alvo, sem levar em conta a relação dela com as demais variáveis. Por conta disso, estes métodos são computacionalmente eficientes. Os *wrappers* são problemas de busca exaustiva no espaço de todas as variáveis. Envolvem diretamente o uso de algum método de classificação. Os métodos *embedded* se caracterizam como modelos de aprendizado que já contam com alguma maneira de ordenar a importância das variáveis em seu treinamento.

O custo computacional em se realizar os diferentes métodos de FS pode ser estimado levando-se em conta seu grau de dependência em algoritmos de ML. Filtros, por exemplo, não realizam treinamento de modelos para serem executados. Métodos *embedded* se aproveitam do resultado de um treinamento de um algoritmo de ML. Portanto, no caso de um sistema composto por FS *embedded* e classificador, ocorrem 2 treinamentos de modelos. Para técnicas com *wrappers*, o custo computacional é o mais alto de todos: é possível que sejam necessárias diversas rodadas de treinamento para que n variáveis sejam selecionadas.

Outra forma de categorizar as técnicas de FS está no uso que fazem da variável rótulo (*label, target*, alvo). Caso estas técnicas sejam realizadas de maneira independente do rótulo, são denominidads de FS não supervisionado. Caso utilizem a informação do rótulo, são denominadas de FS supervisionada (CAI et al., 2018).

2.4.1 Métodos de Seleção de Variáveis Independentes do Rótulo

Uma forma trivial de se eliminar variáveis está na remoção de colunas duplicadas no *dataset*. Colunas duplicadas podem ser artefatos da integração de conjuntos de dados, por exemplo. Devem ser removidas antes do treinamento de modelos de ML, para que não dividam a real importância que possuem entre si (GUYON; ELISSEEFF, 2003).

Atributos com baixa variância não contribuem com modelos de classificação e também podem ser removidas. Ao eliminar atributos de baixa variância, também são eliminados os constantes. Seja X uma variável aleatória constante, que sempre assume o valor c ($x_i = c, \forall i$). Sua média μ_X é igual a essa própria constante, portanto, $\mu_X = c$ (MONTGOMERY; RUNGER, 2003). Para este caso, pela definição de variância de uma população com N elementos:

$$\sigma_X^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)^2 = \frac{1}{N}\sum_{i=1}^{N}(c - c)^2 = \frac{1}{N}\sum_{i=1}^{N}(0)^2 = 0 \tag{2.4}$$

Pela Equação 2.4 e sabendo-se que a variância é um número estritamente positivo, pode-se observar, portanto, que a ação de remover *features* com $\sigma^2 < L$ (para algum limiar L) também remove *features* constantes.

Variáveis altamente correlacionadas também dividem entre si a importância em modelos de ML (GUYON; ELISSEEFF, 2003). Estas podem ser agrupadas e serem representadas por um único elemento do grupo (descartando os demais). Sob esta perspectiva,

variáveis duplicadas podem ser vistas como um caso particular de variáveis correlacionadas (no caso, são **perfeitamente** correlacionadas). O método consiste em calcular o Coeficiente de Correlação de Pearson (CCP) entre todos os pares de atributos, dado por (MONTGOMERY; RUNGER, 2003):

$$\rho = \frac{\sum_{i=1}^{N}(x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{X})^2 \sum_{i=1}^{N}(y_i - \bar{Y})^2}} \tag{2.5}$$

onde X e Y são dois atributos que são comparados, x_i e y_i são seus elementos, $\bar{X}$ e $\bar{Y}$ são suas médias. Escolhe-se um limiar L e são agrupadas todas as *features* com $|\rho| > L$. Destes grupos, apenas um elemento é mantido, eliminando os demais, que são redundantes.

2.4.2 Métodos de Seleção de Variáveis Dependentes do Rótulo

Existem também métodos de FS que dependem do rótulo das amostras. Nestes tipos de técnica, escolhe-se previamente quantas variáveis são selecionadas. É possível utilizar a Análise de Variância (ANOVA), por exemplo, para verificar se um alvo categórico (como é o caso na classificação de ataques DDoS) influencia no comportamento de variáveis numéricas (como são diversas estatísticas de parâmetros de rede). ANOVA usa o Teste F para determinar se duas ou mais médias vem da mesma distribuição ou não (KUHN; JOHNSON, 2019). Seja X uma variável aleatória numérica, em que cada amostra pode pertencer a uma classe $C_i \in \{1, 2, \ldots, S\}$, cada classe possui n_i elementos e o total de elementos da variável é denominado N. Seja $\bar{X}_i$ a média dentro da classe C_i, $\bar{X}$ a média geral e x_{ij} o j-ésimo elemento da classe C_i. Define-se a estatística F pela equação:

$$F = \frac{\sum_{i=1}^{S} n_i(\bar{X}_i - \bar{X})^2/(S-1)}{\sum_{i=1}^{S}\sum_{j=1}^{n_i}(x_{ij} - \bar{X})^2/(N-S)} \tag{2.6}$$

Um valor alto da estatística F implica que a classe do rótulo influencia a distribuição da variável numérica e esta última deve ser adicionada no conjunto das *features*, sendo, assim, selecionada.

O conceito de Informação Mútua (IM) vem da Teoria da Informação (SHANNON, 1948) e pode ser usado para selecionar atributos. É uma forma de mensurar o quanto conhecer uma variável aleatória reduz a incerteza que se tem sobre outra variável aleatória. Sejam, então, X e Y duas variáveis aleatórias, $p(x, y)$ sua distribuição de probabilidades conjunta, $p(x)$ e $p(y)$ suas distribuições marginais. Sejam $\mathcal{X}$ e $\mathcal{Y}$ os conjuntos dos grupos

de valores que X e Y podem assumir. A Informação Mútua é dada por:

$$I(X;Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \ln \frac{p(x,y)}{p(x)p(y)} \tag{2.7}$$

Normalmente, a IM é calculada entre dois atributos categóricos, mas pode ser adaptada para o caso em que existe uma variável numérica e outra categórica (ROSS, 2014). Um valor alto de IM entre uma *feature* e o rótulo sugere que a primeira deva ser incluída no modelo.

A família de algoritmos Relief executa FS atribuindo um peso para cada variável (URBANOWICZ et al., 2018). Algumas amostras são tomadas do *dataset* e comparadas com seus vizinhos mais próximos: aqueles da mesma categoria que a amostra (acertos mais próximos) e aqueles de categorias distintas (erros mais próximos). O peso é computado de maneira a favorecer os acertos e penalizar os erros. *Features* com um alto valor de peso são selecionadas. ReliefF é uma extensão do algoritmo Relief orginal, capaz de lidar com rótulo multiclasse.

Como um dos resultados do treinamento de um classificador XGBoost, *rankings* de importância das variáveis são criados. Uma destas listagens é a de Ganho de cada atributo (CHEN; GUESTRIN, 2016), uma quantidade que mede a contribuição relativa de cada variável em cada árvore treinada pelo modelo. É uma forma de medir o poder discriminador de uma variável. Pode-se, então, selecionar apenas as *features* consideradas mais importantes por esta métrica, que se torna um método de FS embutido (*embedded*) no treinamento do modelo.

Recursive Feature Elimination (RFE) é uma técnica de FS do tipo *wrapper*. Começa com o conjunto completo de variáveis do *dataset* e usa um modelo de ML para listar as variáveis mais importantes. Deste ponto em diante, a pior variável é descartada e o processo começa novamente com o conjunto restante (KUHN; JOHNSON, 2013), o que caracteriza a natureza recursiva deste método.

3 TRABALHOS RELACIONADOS

Este capítulo se propõe a ilustrar como todos os temas relevantes para a realização das simulações computacionais são encontrados na literatura. Alguns casos de uso de ML na classificação de ataques DDoS são expostos. Em particular, também são selecionados trabalhos que usam técnicas de FS para o pré-processamento do conjunto de dados de entrada de seus modelos. O *dataset* CICDDoS2019 (SHARAFALDIN et al., 2019) é apresentado: seu método de geração, taxonomia e quais os ataques que são contemplados por ele. Por fim, alguns trabalhos que se utilizam deste conjunto de dados também são descritos.

3.1 *Machine Learning* para a Detecção de Ataques DDoS

Diversos trabalhos se valem do conceito de entropia de uma variável aleatória para detectar anomalias no comportamento de uma rede. A entropia, tal como introduzida por Shannon (SHANNON, 1948), mede a quantidade de informação contida em um conjunto de dados e está intimamente ligada com a Informação Mútua (IM), apresentada anteriormente. Informalmente, a entropia mede o grau de "surpresa" que é obtido com a realização de um evento descrito por uma variável aleatória. O trabalho de Bungart (BUNGART, 2018), por exemplo, contabiliza a entropia da distribuição dos intervalos de tempo entre requisições à um elemento de rede. Em um ataque DoS, há uma inundação de pacotes em direção ao alvo com intervalos de tempo regulares entre si. Desta maneira, a entropia medida, em situação de ataque, **diminui**. É possível determinar limiares de entropia a partir dos quais o tráfego é considerado um ataque. Outros trabalhos propõem o uso da entropia de outros parâmetros de rede (NOVAES et al., 2020), como IP de Origem, IP de Destino, Porta de Origem e Porta de Destino. É de grande interesse que esse cálculo seja feito *online* e atualizado continuamente (Li et al., 2020).

O artigo de Aamir e Zaidi (AAMIR; ZAIDI, 2019) lista uma série de *datasets* de ata-

ques DDoS populares na literatura. Em geral, as variáveis representadas nestes conjuntos podem ser chamadas de variáveis de fluxo (como contagens temporais. Ex.: Bytes/s) ou de pacotes (variáveis categóricas. Ex.: Porta TCP). Os autores, então, se utilizam do conjunto de dados de Alkasassbeh et al. (ALKASASSBEH et al., 2016) para o treinamento de modelos de ML. Utilizam a IM, Teste do Qui-Quadrado e Teste T de Student para selecionar as variáveis mais relevantes, assim como também o conhecimento do domínio de redes para descartar variáveis. Treinam diversos modelos para a classificação de ataques DDoS: *K-Nearest Neighbors* (KNN), *Naive Bayes* (NB), *Support Vector Machines* (SVM), *Random Forest* (RF) e Redes Neurais Artificiais. Usam um CV com 10 repetições. Em seu artigo, a combinação entre IM para o pré-processamento e KNN como algoritmo classificador traz os melhores resultadados.

Algoritmos baseados em árvores têm se mostrado eficientes como classificadores e regressores para dados tabulares (OMAR, 2018). No artigo de Dey, Rahman e Uddin (DEY; RAHMAN; UDDIN, 2018), que usa o *dataset* NSL-KDD, o melhor desempenho entre 7 tipos de classificadores de ataque é obtido com RF. Lopez, Mohan e Nair (LOPEZ; MOHAN; NAIR, 2019) usam o conjunto de dados CICIDS2017 para treinar classificadores multiclasse. Os autores se preocupam em fazer uma divisão de treino e teste **estratificada**, isto é: a distribuição de classes no *dataset* como um todo é a mesma distribuição de classes dos conjuntos de treino e teste. Treinam 6 tipos de classificadores e RF obtém o melhor desempenho neste cenário.

O algoritmo XGBoost (CHEN; GUESTRIN, 2016) tem chamado a atenção da indústria e academia, pelo seu bom desempenho (medido por métricas de classificação) e rápido tempo de treinamento, principalmente quando comparado com algoritmos mais complexos, como Redes Neurais. Seu uso é muito popular em plataformas de competição de ML, como o Kaggle (REINSTEIN, 2017). Além disso, é aplicado no problema de classificação de ataques DDoS com bons resultados (HUSAIN et al., 2019). O trabalho de Chen et al. (CHEN et al., 2018) se utiliza do algoritmo XGBoost, assim como SVM, RF e *Gradient-boosted Decision Trees* (GBDT), para desenvolver detectores de ataques com o *dataset* KDD 99. Ainda que XGBoost seja treinado em um tempo maior do que os demais, é, em contrapartida, superior em termos de desempenho de classificação. O tempo de treinamento pode ser abreviado com múltiplas *threads* no processador ou com o uso de GPUs.

O trabalho de Dhaliwal, Nahid e Abbas (DHALIWAL; NAHID; ABBAS, 2018) também se vale de um classificador XGBoost para a detecção de ataques DDoS. Para os autores, o algoritmo é adequado por vários motivos: é treinado rapidamente, pode ser paralelizado,

está disponível em várias linguagens de programação, o seu uso de *boosting* diminui a ocorrência de falsos positivos e a regularização na sua função custo penaliza o *overfitting* (sobreajuste). Utilizam o conjunto de dados NSL-KDD e convertem tipos de dados que requerem muita memória (como *strings*) em tipos mais econômicos em recursos (como números inteiros). No artigo, o desempenho do modelo, em termos de acurácia, é comparado com outros trabalhos na literatura. O modelo proposto é o que obtém o melhor resultado.

Como visto nos últimos exemplos, uma abordagem muito comum na literatura é a comparação das métricas de desempenho de uma série de classificadores de ataques independentes, estes sujeitos aos mesmos dados de entrada. Este formato também é encontrado em diversos outros artigos (KHAMAISEH et al., 2019) (TUAN et al., 2020b) (SILVA et al., 2021) (GOLCHIN et al., 2022). Uma outra alternativa é utilizar classificadores em conjunto em um *ensemble*. Desta maneira, a detecção da classe de tráfego de uma amostra é obtida, por exemplo, através de uma votação feita pelos classificadores individuais (DAS et al., 2019).

Os algoritmos de classificação também podem ser separados em paramétricos e não-paramétricos (KIOURKOULIS, 2020). Algoritmos paramétricos possuem um número fixo de atributos que devem ser definidos através de um processo de aprendizagem com os dados de treino. São computacionalmente eficientes, mas assumem que os dados de entrada possuem certas propriedades. Se estas propriedades não forem satisfeitas, o desempenho destes classificadores tende a ser insatisfatório. Dentre os algoritmos paramétricos estão SVM, NB, Regressão Logística e até mesmo Redes Neurais. Algoritmos não-paramétricos são mais flexíveis. Conforme o treinamento é executado, o número de parâmetros cresce conforme a necessidade. Exemplos de algoritmos não-paramétricos são Árvores de Decisão e RF.

A escolha do tipo de algoritmo de classificação também influencia no uso de seleção de variáveis. Em seu processo de treinamento, o algoritmo XGBoost já elenca quais *features* são mais importantes para a classificação que está sendo executada. O trabalho de Manju, Harish e Prajwal (MANJU; HARISH; PRAJWAL, 2019) se utiliza deste fato para realizar FS com XGBoost, além de também usá-lo como classificador de tráfego da Internet. Os autores reforçam que a classificação de tráfego é importante para o planejamento de alocação de recursos de rede e detecção de tráfego anômalo, como ataques DDoS. Fazem um CV de 10 repetições estratificado e medem o desempenho através da Precisão, *Recall* e *F1-score*. Treinam o modelo em dois conjuntos de dados: um balanceado e outro desbalanceado. O cenário balanceado e com menos variáveis possui o melhor desempenho.

Além disso, comparam as métricas de qualidade obtidas pelo XGBoost com uma Árvore de Decisão, RF e Adaboost, sendo que XGBoost se mostra superior aos demais.

O trabalho de Kwon et al. (KWON et al., 2019) emprega uma Rede Neural totalmente conectada para realizar a detecção de anomalias de redes de computadores, usando o *dataset* NSL-KDD. Como pré-processamento, codifica variáveis categóricas como numéricas e faz a normalização dos dados pelo *z-score*. Não faz FS. São raros os trabalhos na literatura que usam Redes Neurais como classificadores **e também** fazem seleção de variáveis. Isso é devido ao fato das camadas ocultas de uma Rede Neural executarem uma espécie de Extração de Variáveis (HAYKIN, 1999), em que o vetor multidimensional da entrada é projetado em um espaço menor intermediário. Esta representação mais compacta que é usada para problemas de classificação.

3.2 Seleção de Variáveis no contexto de Detecção de Ataques DDoS

Dentre as pesquisas que se propõe a usar ML para detectar ataques DDoS, algumas também usam certos tipos de FS como pré-processamento. No trabalho de Polat, Polat e Cetin (POLAT; POLAT; CETIN, 2020), são explorados, como métodos de FS, o uso de filtros, *wrappers* e LASSO (que é uma forma de executar FS embutida no treinamento de um modelo de ML) como etapa de pré-processamento para a classificação de ataques DDoS em *Software-Defined Networks* (SDN). Como classificadores os autores empregam SVM, KNN, NB e Redes Neurais Artificiais. Em todos os casos, ocorrem melhorias de acurácia quando os métodos de FS são aplicados.

Métodos com filtros envolvem atribuir um número para cada atributo e depois ordená-los, escolhendo-se os K atributos mais adequados. Nesta categoria enquadram-se o CCP, o Ganho de Informação e o Teste do Qui-Quadrado, por exemplo. Pattawaro e Polprasert (PATTAWARO; POLPRASERT, 2018) propõem uma técnica adequada a problemas de classificação, o Attribute Ratio (AR), que leva em conta apenas razões de contagens das observações pertencentes a cada classe do conjunto de dados, sendo assim um tipo de filtro.

Para os *wrappers*, escolhe-se uma métrica de qualidade (como a acurácia), treina-se um modelo de ML utilizando um subconjunto das variáveis e, conforme o resultado, novas variáveis são acrescidas ou subtraídas deste subconjunto, iterativamente. Um exemplo é o trabalho de Gupta (GUPTA, 2018) que aplica RFE para selecionar as variáveis relevantes

para a classificação de ataques DDoS. Contudo, *wrappers* são computacionalmente muito custosos, podendo tomar muito mais tempo para serem executados do que filtros ou técnicas de FS já embutidas no treinamento de classificadores. A escolha das melhores variáveis dentro do espaço de todas as possíveis combinações é considerada um problema NP-Hard (LINDQVIST; PRICE, 2018). Inclusive há pesquisas sobre maneiras de otimizar a execução dos *wrappers*. Por exemplo, uma forma é utilizar algoritmos genéticos (WANG; YAO; LIU, 2019), que incorporam a ideia de função objetivo (análoga ao conceito de seleção natural em biologia) para escolher a população de variáveis mais apta.

Classificadores baseados em árvore, como Árvores de Decisão, RF e XGBoost, já possuem embutidos no próprio treinamento métodos de quantificar a importância de cada atributo. Um exemplo desta característica está no trabalho de Devan e Khare (DEVAN; KHARE, 2020), em que é treinado um modelo XGBoost para elencar as variáveis mais importantes. As variáveis escolhidas servem de entrada para uma Rede Neural Profunda e outros algoritmos de classificação.

Um resumo dos principais trabalhos relacionados que usam FS pode ser encontrado na Tabela 1. Os *datasets* usados estão listados para referência. A Tabela 1 também mostra técnicas de processamento dos conjuntos de dados, como o tipo de amostragem. Os métodos usados para o treinamento do classificador de ataques DDoS também são expostos, como o algoritmo usado, o tipo de classificação que realiza e se CV foi usado ou não. Finalmente, métricas de qualidade também são listadas na Tabela 1. Além disso, os experimentos mostrados nesse trabalho são colocados no final da tabela para comparação com os demais. No geral, este trabalho procura aplicar uma ampla gama de técnicas de FS e diversas métricas de desempenho para quantificar o impacto dessas técnicas no problema de classificação de ataques DDoS. Um ponto a se notar é que este trabalho ataca o problema tanto do ponto de vista binário como multiclasse. As razões entre as métricas e os tempos de execução também são introduzidos, aqui chamadas de *Benefit-Cost Ratios* (BCR) e serão explicadas na seção 4.4.

Um ponto a se ressaltar é que, para classes diferentes de tráfego, conjuntos de variáveis **diferentes** podem ser melhores para identificá-las (SILVA et al., 2015). Trata-se, portanto, de uma decisão de projeto se optar em usar o mesmo conjunto de *features* para se detectar várias classes de ataque ou, então, introduzir mais complexidade no sistema com conjuntos especializados de variáveis (um conjunto distinto por ataque). Outro ponto a ser considerado é que *features* que trazem um bom desempenho de classificação para um modelo específico podem **não ter este bom desempenho** quando usadas de entrada para outro modelo (PILNENSKIY; SMETANNIKOV, 2020). O sistema, portanto, deve

Tabela 1: Resumo de trabalhos com FS

Trabalho	(POLAT; POLAT; CETIN, 2020)	(PATTAWARO; POLPRASERT, 2018)	(GUPTA, 2018)	(DEVAN; KHARE, 2020)	Este Trabalho
Dataset	Testbed dos Autores	NSL-KDD	KDD 99	NSL-KDD	CICDDoS2019
Feature Selection	Relief, Sequential Forward Selection (SFS), LASSO	Attribute Ratio	Ganho de Informação, Qui-Quadrado, RFE, Ensemble	Ganho do XGBoost	Baixa Variância, Alta Correlação, ANOVA, IM, ReliefF, Ganho do XGBoost, RFE, Ensemble
Cross Validation	10 repetições	Não	20 repetições	5, 10, 15, 20, 25 repetições	10 repetições
Amostragem	Não	Não	Sub-amostragem	Sub-amostragem e tamanhos variáveis de treino	Sub-amostragem e rebalanceamento de classes
Modelos	SVM, NB, KNN, Redes Neurais	XGBoost	NB, SVM, Árvore de Decisão, RF	LR, NB, SVM, Rede Neural Profunda	XGBoost
Tipo de Classificação	Multiclasse	Binária	Binária	Binária	Binária, Multiclasse, Multiclasse OvO, Multiclasse OvR
Métricas	Acurácia, Sensibilidade, Especificidade, Precisão, *F1-Score*	Acurácia, Precisão, *Recall*	Acurácia, Precisão, *Recall*	Acurácia, Precisão, *Recall*, *F1-Score*, ROC-AUC	Acurácia, Precisão, *Recall*, *F1-Score*, BCR

ser considerado de forma holística.

3.3 O Conjunto de Dados CICDDoS2019

Diversos *datasets* são usados ao longo dos anos como referência para a pesquisa em detecção de intrusões, com destaque para os mais populares: KDD 99 (GUPTA, 2018)(TUAN et al., 2020b)(MOHAMMADI et al., 2019) e NSL-KDD (HOSSEINI; AZIZI, 2019)(HOON et al., 2018)(DAS et al., 2019). Estes registros de ataques são utilizados em diversos estudos, muitos deles com o foco em detecção de comportamentos anômalos de rede utilizando ML. Contudo, tanto KDD 99, quanto NSL-KDD, estão defasados, uma vez que novos tipos de ataque, que não estão representados nestes *datasets*, foram e estão sendo introduzidos desde a publicação destes conjuntos de dados.

Portanto, faz-se necessário o uso de um *dataset* mais atualizado. O conjunto de dados Canadian Institute for Cybersecurity DDoS 2019 (CICDDoS2019), publicado por Sharafaldin et al. (SHARAFALDIN et al., 2019) é extraído de um *testbed* com equipamentos reais, como roteadores, *switches*, *firewalls* e servidores diversos. No cenário do trabalho

mencionado, há uma rede atacante totalmente apartada da rede vítima. Os autores geram fluxos de rede tanto de ataques como tráfego benigno (normal) de fundo. Os fluxos são definidos como uma janela de tempo em que o tráfego pertence à mesma n-upla **(IP de Origem, Porta de Origem, IP de Destino, Porta de Destino, Protocolo)**, sendo todas essas informações disponíveis nos cabeçalhos TCP/IP (BARZILAY et al., 2021). No total, 86 parâmetros são coletados para cada fluxo, entre variáveis identificadoras dos elementos presentes no *testbed* e estatísticas de rede. O tráfego dos ataques é significativamente mais volumoso que o normal neste conjunto de dados.

Sharafaldin et al. propõe uma taxonomia para os ataques executados na criação de seu conjunto de dados. Estes se dividem em dois grandes tipos: ataques de **Reflexão** e **Exploits** (MARVI; ARFEEN; UDDIN, 2021). Ataques de Reflexão constroem pacotes maliciosamente, de modo que o endereço IP de origem das requisições seja o endereço IP da máquina vítima. Exploits se aproveitam de alguma vulnerabilidade no dispositivo alvo para causar a exaustão de seus recursos computacionais. As amostras de ataques contemplados pelo CICDDoS2019 são rotuladas como sendo das seguintes classes:

- DNS (*Domain Name System*): servidores DNS são como dicionários, que fazem a tradução de *hostnames* para endereços IP. Ataques geralmente usam servidores DNS públicos como instrumento de amplificação de tráfego (YE; YE, 2013). É um ataque do tipo Reflexão.

- LDAP (*Lightweight Directory Access Protocol*): este é o protocolo usado por um sistema para acessar informações de usuário e senha armazenadas em servidores *Microsoft Active Directory* remotos (CHOI; KWAK, 2017). É um ataque do tipo Reflexão.

- MSSQL (*Microsoft Structured Query Language*): é um servidor de banco de dados. Atacantes se aproveitam de vulnerabilidades no protocolo *SQL Server Resolution Protocol* (SSRP) para se conectar no banco de dados. Dependo da quantidade de registros armazenados, o ataque pode ter um elevado fator de amplificação. É um ataque do tipo Reflexão.

- NetBIOS (*Network Basic Input/Output System*): é um protocolo que permite a comunicação entre dispositivos em uma rede local. O ataque consiste em adulterar o IP de origem para o da vítima e realizar um *broadcast* na rede local vulnerável. É um ataque do tipo Reflexão.

- NTP (*Network Time Protocol*): este é o protocolo usado para sincronizar os relógios

de elementos de uma rede. Ao responderem ao comando `monlst`, servidores NTP retornam informações das últimas 600 conexões que realizaram (ARUKONDA; SINHA, 2015). É um ataque do tipo Reflexão.

- SNMP (*Simple Network Management Protocol*): sistemas SNMP são usados para coletar informações de gerenciamento de equipamentos de rede e servidores. Atacantes podem usar o comando `GetBulkRequest` para enviarem uma quantidade de dados muito maior do que a requisição original para a vítima (SIEKLIK; MACFARLANE; BUCHANAN, 2016). É um ataque do tipo Reflexão.

- SSDP (*Simple Service Discovery Protocol*): é um protocolo usado para a descoberta de dispositivos *Universal Plug and Play* (UPnP). Caso estes dispositivos estejam expostos na Internet, um atacante pode modificar o IP de origem de uma requisição para o IP da vítima e usar o dispositivo como intermediário. É um ataque do tipo Reflexão.

- TFTP (*Trivial File Transfer Protocol*): servidores TFTP são usados para o *download* e *upload* de arquivos. Geralmente, se comunicam com outros elementos de rede através do protocolo UDP. Uma vez que o servidor pode armazenar diversos e volumosos arquivos, um ataque que os utilize de vetor pode implicar em um alto fator de amplificação (SIEKLIK; MACFARLANE; BUCHANAN, 2016). É um ataque do tipo Reflexão.

- SYN flood: é um ataque que se utiliza de uma vulnerabilidade do *Transmission Control Protocol* (TCP). O atacante envia uma mensagem SYN para abrir uma conexão com a vítima. Esta, por sua vez, responde com uma mensagem SYN-ACK e mantem uma conexão aberta. Contudo, o atacante nunca retorna uma mensagem ACK, mantendo a conexão aguardando e desperdiçando recursos da vítima (BOGDANOSKI; SUMINOSKI; RISTESKI, 2013). É um ataque do tipo Exploit.

- UDP flood: o atacante faz requisições a diversas portas UDP aleatórias contra o servidor alvo. Caso não possua nenhuma aplicação que escute nessa porta, o servidor manda mensagens de indisponibilidade, gastando seus recursos computacionais. É um ataque do tipo Exploit.

- UDP-Lag: é um ataque que utiliza vulnerabilidades do protocolo UDP para consumir a largura de banda de uma rede (SHARAFALDIN et al., 2019). É um ataque do tipo Exploit.

- WebDDoS: atacantes tem como alvo servidores HTTP expostos na Internet que, justamente pelo modo como operam, devem responder a requisições de clientes. Um alto volume de tráfego é disparado, de modo a consumir os recursos do servidor. É um ataque do tipo Exploit.

3.4 O Uso do Conjunto de Dados CICDDoS2019 na Literatura

Como parte do artigo que apresenta o conjunto de dados, Sharafaldin et al. propõem o uso de ML para realizar a detecção dos ataques, assim como elencar a importância de cada variável na tarefa de classificação destes. Usam RF para obter valores numéricos de importância das variáveis. Este resultado numérico pode ser usado como um método de FS, caso seja optado por não usar todas as variáveis no treinamento de um classificador. Os autores treinam, então, modelos com os algoritmos Iterative Dichotomiser 3 (ID3), RF, NB e Regressão Logística. Usam o *dataset* completo para realizar uma classificação multiclasse. O algoritmo que se sai melhor é o ID3, que é uma Árvore de Decisão. Este obtém os melhores valores das métricas de desempenho (Acurácia, Precisão e *Recall*) e dos tempos de ajuste entre os classificadores considerados.

Após sua publicação, o CICDDoS2019 já serviu de base para alguns trabalhos, como o de Hussain (HUSSAIN, 2020). Este autor usa os resultados de Sharafaldin et al. para selecionar apenas as 24 variáveis mais importantes (segundo o método de RF). Estas variáveis servem de entrada para diversos algoritmos que realizam classificação multiclasse. Para tratar o desbalanceamento entre as classes, Hussain se utiliza tanto de técnicas de subamostragem (para classes altamente representadas) quanto de sobreamostragem (para classes com menos registros). Cria 6 *datasets*, cada um com um número crescente de amostras totais. Para avaliar o treinamento, usa um CV de 5 repetições. Aqui, novamente, modelos baseados em Árvore de Decisão obtém melhores resultados que os demais.

Outro exemplo é o trabalho de Li (LI, 2020), que faz uso tanto de algoritmos de ML tradicionais como de *Deep Learning* (DP). Aqui, mais uma vez, é executado um rebalanceamento dos dados, de forma que não predominem amostras de ataque quando comparadas ao tráfego benigno. O balanceamento é feito por subamostragem. O autor também nota que existem algumas colunas com amostras vazias ou com valores infinitos, que devem ser tratadas antes do treinamento de um modelo de ML. Para a redução de dimensionalidade, são usadas tanto técnicas de seleção de variáveis (como remover variáveis com o módulo da Correlção de Pearson $|\rho| > 0,9$ entre si), como de extração de

variáveis (fazendo uso de *Autoencoders*). O autor treina algoritmos de Árvore de Decisão, RF, Regressão Logística e NB para executar uma classificação multiclasse. Treina também Redes Neurais profundas pra criar uma série de classificadores binários, um por cada tipo de ataque presente no conjunto de dados. Não há um tratamento dos dados que se saia melhor para todos os ataques, sendo que o autor propõe se façam experimentos para cada ataque individualmente.

O artigo de Marvi, Arfeen e Uddin (MARVI; ARFEEN; UDDIN, 2021) também faz uso do conjunto de dados CICDDoS2019. Em uma análise exploratória do *dataset*, os autores apontam que a maioria dos ataques é baseada no protocolo UDP, enquanto o tráfego normal é majoritariamente TCP. Como pré-processamento, os autores propõe diminuir a precisão dos dados numéricos, por exemplo, armazenando valores como pontos flutuantes de 32 bits ao invés de 64 bits. Desta maneira, frações maiores do conjunto de dados caberiam na memória de computadores com menos recursos. Chamam este processo de *Data Type Optimization* (DTO). Os autores fazem um *ensemble* de técnicas de FS. Combinam o resultado de um método de Filtro, ANOVA, e métodos *embedded*, RF e *Light Gradient Boosting Machine* (LGBM), para chegarem em um conjunto reduzido de variáveis. Treinam um classificador binário com o algoritmo LGBM usando como entrada amostras de apenas alguns ataques. Testam o classificador com amostras de todos os ataques. Mesmo os ataques que não participaram do conjunto de treino foram classificados como tráfego não-benigno, o que significa que um sistema com este modelo como base seria capaz de tomar alguma ação contra ataques aos quais nunca foi exposto.

4 PROPOSTA DE SIMULAÇÃO COMPUTACIONAL

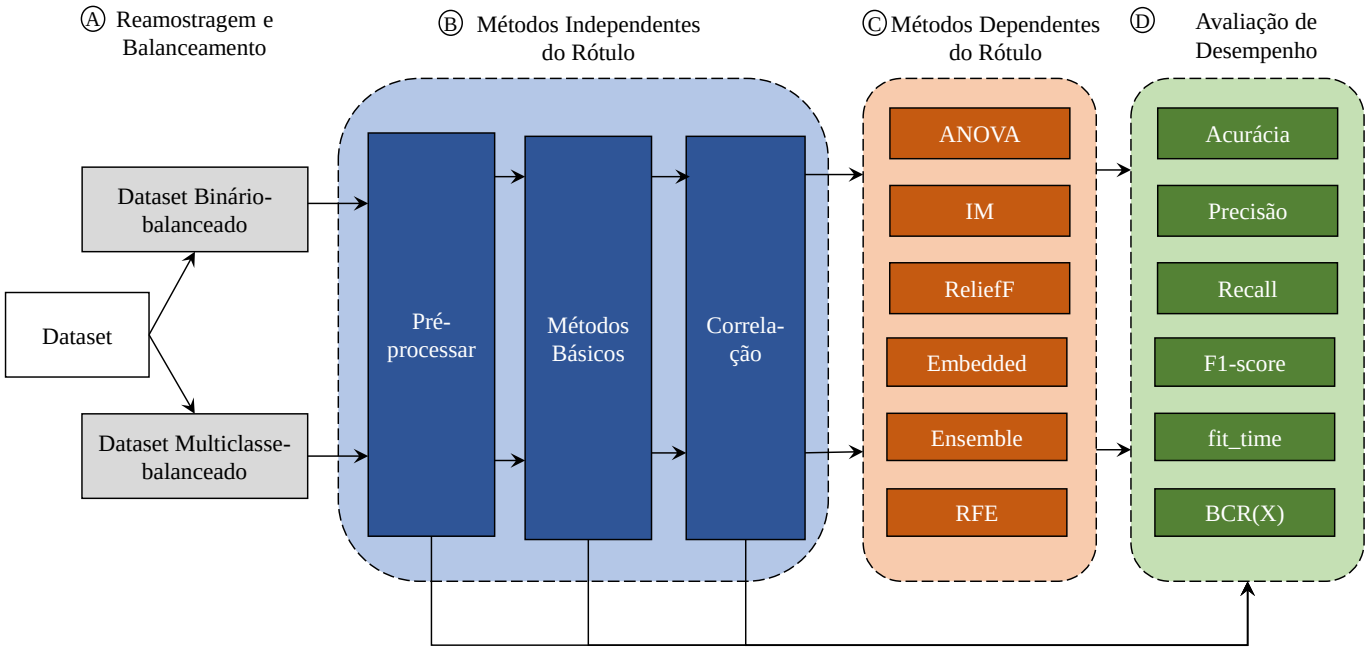
Para se mensurar a influência do uso de FS no problema de classificação de ataques DDoS, foi escolhido o conjunto de dados CICDDoS2019 (SHARAFALDIN et al., 2019). Este conjunto contempla amostras rotuladas de tráfego de 12 tipos de ataques DDoS (NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS, SYN, TFTP), além de tráfego benigno. O *dataset* é então processado, submetido a experimentos com diversos métodos de seleção de variáveis e serve de entrada para um classificador que utiliza o algoritmo XGBoost. Nenhum método de FS **novo** está sendo proposto nos experimentos que se seguem. Ao invés disso, técnicas de FS conhecidas na literatura são usadas antes do treinamento de classificadores de ataques DDoS, para que os desempenhos dos classificadores resultantes possam ser comparados numericamente.

Modelos baseados em árvores têm se mostrado consistentemente eficazes para problemas de classificação usando este conjunto de dados (HUSSAIN, 2020)(MARVI; ARFEEN; UDDIN, 2021). O classificador XGBoost é fixo no decorrer dos experimentos, uma vez que o objetivo do estudo é mensurar a influência das técnicas de FS, não do algoritmo de classificação escolhido.

As etapas das simulações computacionais realizadas são detalhadas no restante deste capítulo, bem como os principais conceitos necessários para avaliá-las. A Figura 10 ilustra o fluxo de dados geral dos experimentos deste trabalho que é dividido em quatro etapas: (A) Reamostragem e Balanceamento; (B) Métodos Independentes do Rótulo; (C) Métodos Dependentes do Rótulo; e (D) Avaliação de Desempenho.

É possível descrever a arquitetura da Figura 10 da seguinte maneira: como o *dataset* CICDDoS2019 é extremamente volumoso, é realizada uma amostragem dos dados, de forma que o conjunto resultante possa ser processado em um computador pessoal. Além disso, este conjunto de dados também é desbalanceado, o que pode interferir na interpretação das métricas de desempenho. A etapa (A) Reamostragem e Balanceamento, trata destes dois problemas. A seguir, métodos de FS independentes do rótulo são aplica-

Figura 10: Fluxo de dados proposto para os experimentos



dos de forma sequencial: o conjunto de saída de um método é o de entrada do próximo. Este processo é ilustrado pela etapa (B) Métodos Independentes do Rótulo. Para quantificar o desempenho destes métodos, classificadores treinados com os conjuntos de dados resultantes são avaliados segundo as métricas da etapa (D) Avaliação de Desempenho. O *dataset* resultante do último método de FS independente do rótulo é então submetido a diversas técnicas de FS que **dependem** do rótulo das amostras, de forma paralela. Estas técnicas são listadas na etapa (C) Métodos Dependentes do Rótulo e contam com uma característica que as distingue dos métodos independentes: escolhe-se de antemão quantas *features* serão selecionadas, antes do método ser executado. Os métodos da etapa (C) também são avaliados segundo as métricas da etapa (D), mas são executados diversas vezes, escolhendo-se um número progressivamente menor de variáveis em cada rodada. As próximas seções detalham a implementação da arquitetura proposta por este diagrama.

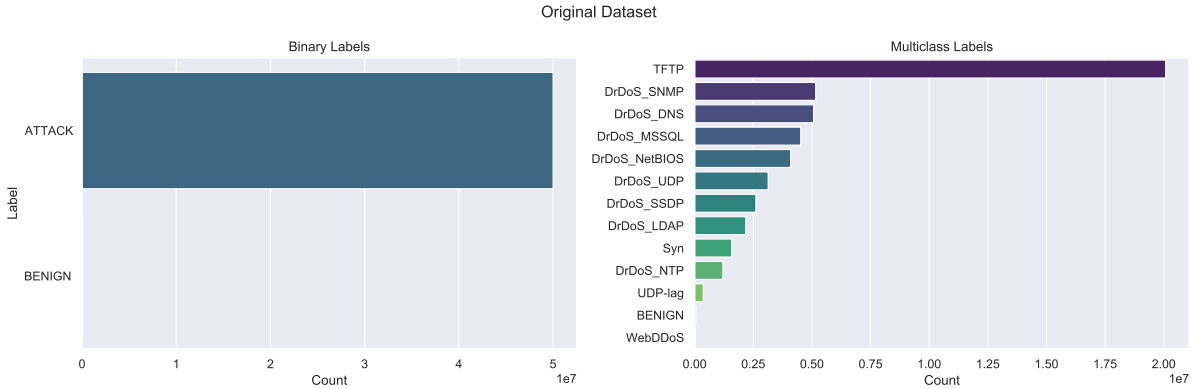
4.1 Reamostragem e Balanceamento

A Tabela 2 apresenta a distribuição das amostras entre as diferentes classes de tráfego para o *dataset* CICDDoS2019. Nela são expostas as representações binária e multiclasse deste conjunto. A Figura 11 apresenta a mesma informação, mas de forma gráfica.

Tabela 2: Distribuição de classes do *dataset* CICDDoS2019

Binário		Multiclasse	
Classe	Amostras	Classe	Amostras
ATAQUE	50006249	TFTP	20082580
		DrDoS_SNMP	5159870
		DrDoS_DNS	5071011
		DrDoS_MSSQL	4522492
		DrDoS_NetBIOS	4093279
		DrDoS_UDP	3134645
		DrDoS_SSDP	2610611
		DrDoS_LDAP	2179930
		Syn	1582289
		DrDoS_NTP	1202642
		UDP-lag	366461
		WebDDoS	439
BENIGNO	56863	BENIGNO	56863

Figura 11: Visualização da distribuição de classes do *dataset* CICDDoS2019



Intuitivamente, pode-se dizer que se trata de um conjunto de dados extremamente desbalanceado, com uma quantidade de ataques que supera em muito o tráfego benigno e, entre esses ataques, o de amplificação por TFTP é o que mais está representado. É possível formalizar e quantificar essa noção de desbalanceamento através do Índice de Pielou (PIELOU, 1966):

$$J = -\frac{1}{\ln S} \sum_{i=1}^{S} \frac{n_i}{N} \ln \frac{n_i}{N} = \frac{H}{\ln S} \tag{4.1}$$

onde há S classes distintas, cada classe C_i possui n_i elementos e o conjunto de dados completo possui N amostras. Na equação 4.1, H é a Entropia de Shannon (SHANNON, 1948). O Índice de Pielou normaliza a entropia pelo seu máximo (que é igual à $\ln S$), de forma que seus valores ficam confinados entre 0 e 1. Neste caso, 0 representa o maior desbalanceamento possível (todas as amostras são de uma única classe) e 1 o maior balanceamento possível (as amostras estão divididas igualmente entre as classes). Note que o Índice de Pielou é aplicável tanto a conjuntos com rótulos binários como multiclasse.

O conjunto de dados descrito na Tabela 2 e na Figura 11 é desbalanceado. Do ponto de vista binário, possui um Índice de Pielou de $J = 0,012748$. Do ponto de vista multiclasse, este mesmo conjunto possui $J = 0,764539$.

Um *dataset* desbalanceado tende a favorecer a(s) classe(s) majoritária(s) (LIU et al., 2018). Portanto, conforme ilustrado na etapa (A) da Figura 10, para os experimentos são gerados dois conjuntos de dados: um balanceado do ponto de vista binário (em que há uma mesma quantidade de ataques e tráfego benigno) e outro balanceado do ponto de vista multiclasse (com todas as classes com igual representação). Estas duas formas de se reamostrar o conjunto de dados fazem com que o novo Índice de Pielou, para ambos os casos, seja igual a 1. O procedimento de construção dos novos *datasets* é descrito em detalhes no Apêndice A. Por limitação de hardware, o processo de reamostragem se deu por uma sub-amostragem com reposição de cada uma das classes envolvidas, de forma a alocar os dados eficientemente na memória disponível. Um *dataset* maior ou menor não altera o Índice de Pielou, contanto que a proporção entre as classes seja a mesma, conforme demonstrado no Apêndice B. Logo, o conjunto de dados pode ser sub-amostrado sem perda de generalidade. Trabalhos como os de Hussain (HUSSAIN, 2020) e de Li (LI, 2020) também optam pela sub-amostragem. Por fim, a classe "WebDDoS" é descartada por contar com um número de amostras muito menor que os demais tipos de ataque.

Após o processo de reamostragem, os *datasets* binário e multiclasse servem de base

para o treinamento de classificadores binários e multiclasses, respectivamente.

4.2 Seleção de Variáveis Independente do Rótulo

A parte chamada de "Pré-processar" na etapa (B) da Figura 10 inclui uma série de operações. Foram descartadas todas as variáveis categóricas com base no conhecimento de domínio de redes e Segurança da Informação: atributos como ID do *Flow*, IP de Origem e IP de Destino são apenas identificadores do cenário de coleta dos dados, não possuindo valor preditivo. As variáveis Porta de Origem e Porta de Destino podem ser tratadas como numéricas, mas, como este trabalho foca em estatísticas de tráfego de rede, isto não será feito. Todas as variáveis restantes são numéricas, como contadores e razões desses contadores no tempo (como Bytes/s). As seguintes variáveis, mesmo representando contadores ou medidas de tamanho dos pacotes (ou seja, valores positivos), tem valores negativos espúrios no *dataset*: "Fwd Header Length", "Bwd Header Length", "Init_Win_bytes_forward", "Init_Win_bytes_backward", "min_seg_size_forward". Foi decidido ignorar o sinal negativo destas amostras (fazendo-as positivas). Uma descrição detalhada das variáveis do conjunto de dados completo pode ser encontrada em (LI, 2020). Valores infinitos foram substituídos pelo valor máximo da variável em questão. Os *datasets* depois destas operações servem de entrada para classificadores XGBoost. É usado um CV com 10 repetições, onde são coletadas as métricas ilustradas pela etapa (D) da Figura 10.

A segunda parte da etapa (B) realiza a seleção de variáveis através de duas operações: (1) remoção de colunas duplicadas entre si; e (2) remoção de colunas com baixa variância. Estes métodos são chamados de "Métodos Básicos" nas seções subsequentes. *Features* com baixa variância (e, consequentemente, também as constantes) não possuem poder discriminatório em uma Árvore de Decisão, e podem, então, ser descartadas (XIE et al., 2009). Note que a determinação do que é uma variância "baixa" é um hiperparâmetro do método de FS, que deve ser definido como um limiar no momento da execução. Um ponto de atenção é que, para verificar a influência das técnicas de seleção de variável nas métricas de classificação, é importante que estas técnicas sejam executadas **dentro do processo de CV** (sobre o conjunto separado como treino em cada rodada).

Com o conjunto de dados reduzido pelos "Métodos Básicos", é gerada uma Matriz de Correlação de Pearson entre todas as variáveis preditoras. A correlação entre as variáveis preditoras e os rótulos das amostras não é adequada, uma vez que as primeiras são numéricas e as últimas categóricas (KUHN; JOHNSON, 2013). A partir desta

operação, são formados grupos de variáveis que possuem correlação ρ maior (em módulo) que um limiar estabelecido no treinamento. Esse limiar, logo, é outro hiperparâmetro do experimento. Destes grupos, apenas uma variável é mantida e as demais são descartadas. O conjunto resultante também é usado para treinar um modelo XGBoost, dentro de um CV com 10 repetições. Este método é referido como "Correlação" nos resultados dos experimentos.

4.3 Seleção de Variáveis Dependente do Rótulo

Os métodos da etapa (C) da Figura 10 levam em consideração relações entre os rótulos das amostras e as variáveis preditoras para selecionar os atributos que devem ser usados no modelo. De forma geral, todos eles são capazes de gerar *rankings* de importância das variáveis. Com os atributos ordenados, são escolhidos os K mais relevantes e os demais são removidos. Assim como nos métodos independentes do rótulo, aqui também é necessária a realização da seleção de variáveis **dentro do processo de CV**.

Pode-se gerar, então, curvas para cada métrica de qualidade de classificação desejada em função do valor de K (o número de variáveis selecionadas). Esta operação é repetida para cada método de FS dependente do rótulo considerado.

De acordo com a nomenclatura da seção 2.4, os métodos de FS escolhidos se dividem nas seguintes categorias: (1) filtros; (2) *wrappers*; (3) *embeddeds*; e (4) híbridos. Três filtros são considerados para os experimentos: (1) o teste F de ANOVA; (2) a Informação Mútua (IM); e (3) o método ReliefF. Como subproduto do treinamento do XGBoost, são criadas listas de importância de variáveis. Uma dessas listagens é baseada no Ganho do XGBoost (CHEN; GUESTRIN, 2016), que é então usado como método de FS *embedded*. O método RFE é usado como *wrapper* em torno também do XGBoost. Todas estas técnicas estão disponíveis na biblioteca Scikit-Learn (SCIKIT-LEARN, 2022), com exceção do ReliefF (OLSON, 2022), que possui biblioteca própria, mas também compatível com os Arrays do NumPy.

Por fim, é executado um *ensemble* de técnicas anteriores, assim como proposto por Gupta (GUPTA, 2018). Neste método, chamado de *Weighted Ranked Feature Selection* (WRFS), os *rankings* de cada variável para cada técnica de FS considerada são somados em um novo vetor e o valor desta soma é levado em conta no momento de ordenação das variáveis. Este método é detalhado no Algoritmo 1. Os experimentos tratam de um arranjo entre ANOVA, IM, ReliefF e o Ganho do XGBoost. O método RFE não foi

incluído no *ensemble* por conta do seu alto custo computacional.

Algoritmo 1: *Weighted Ranked Feature Selection* (WRFS)

Entrada: Vetores de valores atribuídos pelos métodos do conjunto FS = {ANOVA, IM, ReliefF, Ganho} para todas as variáveis do *dataset*
Saída: K variáveis selecionadas segundo WRFS
início
  para cada *método em FS* **faça**
    ordenar os valores do método do menor para o maior;
    atribuir *ranking* das variáveis segundo ordenação (menores primeiro);
    salvar resultado na lista método_rank
  fim
  para cada *variável* **faça**
    somar os *rankings* da variável particular em cada método_rank;
    salvar resultado no conjunto WRFS
  fim
  ordenar vetor WRFS (maiores primeiro);
  escolher as K primeiras variáveis
fim

4.4 Avaliação de Desempenho

Tanto os classificadores treinados na etapa (B) como na (C) são avaliados segundo as métricas de desempenho ilustradas na etapa (D) da Figura 10. As métricas diferem entre os casos binário e multiclasse.

Para um classificador binário, um dos rótulos é definido como positivo e o outro como negativo. Os possíveis resultados da classificação são organizados e recebem seus nomes a partir de uma **Matriz de Confusão** (JAMES et al., 2013) como a da Tabela 3.

Tabela 3: Elementos da Matriz de Confusão

		Rótulo Previsto	
		Positivo	Negativo
Rótulo Real	Positivo	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Negativo	Falso Positivo (FP)	Verdadeiro Negativo (VN)

A associação dos rótulos à uma classe é arbitrária, mas deve ser levada em conta na interpretação dos resultados. Escolhendo a classe "Benigno" como negativo e "Ataque" como positivo, temos a seguinte nomenclatura:

- VP: Ataque classificado como Ataque;
- FP: Benigno classificado como Ataque;

- FN: Ataque classificado como Benigno;
- VN: Benigno classificado como Benigno.

Para um classificador multiclasse, são consideradas N amostras no conjunto de teste e S classes C_i distintas. Esta situação é ilustrada na Tabela 4, que mostra uma **Matriz de Confusão Multiclasse**.

Tabela 4: Elementos da Matriz de Confusão Multiclasse

		Rótulo Previsto				
		C_1	$\cdots$	C_i	$\cdots$	C_S
Rótulo Real	C_1					
	$\vdots$					
	C_i			VP_i		
	$\vdots$					
	C_S					

Existem S^2 possíveis resultados de classificação, dos quais pode-se definir algumas métricas de interesse. VP_i é o número de elementos corretamente classificados com o rótulo de C_i, representado pela célula em verde da Matriz de Confusão. Os Falsos Positivos da classe i, chamados FP_i, são os elementos classificados como C_i, mas que pertencem a outra classe. Estes estão representados pelas células em amarelo da Matriz de Confusão e estão na mesma **coluna** que VP_i. Os Falsos Negativos da classe i, chamados FN_i, são os elementos que pertencem à C_i, mas erroneamente classificados em outra classe. Estes estão representados pelas células em vermelho da Matriz de Confusão e estão na mesma **linha** que VP_i.

A partir dessas considerações, podemos definir as seguintes métricas de desempenho:

- Acurácia (AC): é a razão entre todos os acertos do modelo e o total de previsões realizado. Para o caso binário:

$$AC = \frac{VP + VN}{VP + FP + VN + FN} \tag{4.2}$$

Para o caso multiclasse:

$$AC = \frac{1}{N} \sum_{i=1}^{S} VP_i \tag{4.3}$$

- Precisão (PR): é a razão das previsões corretas e o total de previsões para uma determinada classe. Um alto valor de PR está ligado a um baixo número de alarmes falsos. Para o caso binário:

$$PR = \frac{VP}{VP + FP} \tag{4.4}$$

Para o caso multiclasse, existem alguns métodos de agregação possíveis (SOKOLOVA; LAPALME, 2009). Para as simulações computacionais, é escolhida a Macro Média da Precisão (PR_M), que corresponde à média aritmética simples (sem ponderação pelo número de amostras) entre os valores de PR de cada classe individualmente. Desta forma, não se privilegia especificamente nenhuma classe. Esta métrica é dada por:

$$PR_M = \frac{1}{S}\sum_{i=1}^{S}\frac{VP_i}{VP_i + FP_i} = \frac{1}{S}\sum_{i=1}^{S} PR_i \tag{4.5}$$

- *Recall* (RC): é a razão das previsões corretas e o total de elementos que são originalmente de uma determinada classe. Um alto valor de RC implica que a maior parte das amostras de uma classe foi reconhecida. Para o caso binário:

$$RC = \frac{VP}{VP + FN} \tag{4.6}$$

Para o caso multiclasse, novamente com a ideia de Macro Média, desta vez do *Recall* (RC_M), tem-se:

$$RC_M = \frac{1}{S}\sum_{i=1}^{S}\frac{VP_i}{VP_i + FN_i} = \frac{1}{S}\sum_{i=1}^{S} RC_i \tag{4.7}$$

- *F1-score* ($F1$): as métricas PR e RC são conflitantes, uma vez que ao aumentar uma delas a outra é comprometida. O $F1$ é a média harmônica entre as últimas. Para o caso binário:

$$F1 = \frac{2 \cdot PR \cdot RC}{PR + RC} \tag{4.8}$$

Para o caso multiclasse, o cálculo mais recomendado (OPITZ; BURST, 2019) é dado por:

$$F1_M = \frac{1}{S}\sum_{i=1}^{S}\frac{2 \cdot PR_i \cdot RC_i}{PR_i + RC_i} = \frac{1}{S}\sum_{i=1}^{S} F1_i \tag{4.9}$$

As métricas de AC, PR, RC e $F1$ variam entre 0 (pior) e 1 (melhor), tanto para classificadores binários como multiclasse. Quando é usado o processo de CV, pode-se calcular a Média e o Desvio-Padrão de cada uma dessas métricas.

- Tempo de Ajuste (fit_time): é o tempo gasto para se obter um classificador treinado e com o conjunto de variáveis selecionado. É a soma do tempo usado para a execução da seleção de variáveis (fs_time) e o treinamento do classificador (train_time):

$$fit\_time = fs\_time + train\_time \tag{4.10}$$

- *Benefit-Cost Ratio* (BCR): é uma família de razões dada pela divisão de uma das métricas pelo fit_time necessário. Para uma métrica $M \in \{AC, PR, RC, F1\}$, a expressão é dada por:

$$BCR(M) = \frac{M}{fit\_time} \tag{4.11}$$

4.5 Demais considerações sobre o treinamento dos classificadores

Os experimentos realizados com CV o fazem de forma **estratificada** em cada repetição. Além disso, as amostras são embaralhadas (*shuffled*) antes da separação realizada pelo CV ocorrer. Como forma de colher resultados preliminares de forma mais ágil, os mesmos experimentos realizados com CV também são realizados apenas com *hold-out* em um momento anterior. Um resumo de todos os experimentos propostos pode ser encontrado no Apêndice C.

O presente trabalho tem por foco caracterizar numericamente a influência de técnicas de FS nas métricas de desempenho de um classificador de ataques DDoS. Para os resultados obtidos com CV de 10 repetições, existem, potencialmente, até 10 grupos de variáveis distintos para cada cenário de simulação. Esta situação é a mesma tanto para os métodos independentes do rótulo, quanto para os métodos dependentes dele. Está fora do escopo deste trabalho a listagem de quais variáveis em particular foram selecionadas em cada rodada dos experimentos.

4.6 Ambiente de Testes

Para todos os experimentos é usado um *notebook* com processador Intel Core i7 3630QM, 8GB de memória RAM DDR3, HD de 480GB SSD (*Solid State Drive*) e sistema operacional Windows 10. Através do gerenciador de pacotes Anaconda são instalados os seguintes programas (bem como suas dependências): Python (v3.8.5), NumPy (v1.19.5), Pandas (v1.2.0), Scikit-learn (v0.24.0), Seaborn (v0.11.1), ReliefF (v0.1.2) e XGBoost (v1.3.1). Trata-se, portanto, de um ambiente usual para pesquisa em ML baseado na linguagem Python e suas bibliotecas.

Os programas e gráficos criados para este projeto, bem como resultados adicionais mencionados e não ilustrados por questão de brevidade estão integralmente disponíveis em ⟨https://github.com/pedrohauy/ddos_feature_selection⟩.

5 RESULTADOS

Este capítulo ilustra os resultados dos experimentos obtidos com o *dataset* escolhido, o CICDDoS2019 (SHARAFALDIN et al., 2019). A construção dos conjuntos de dados utilizados para treinamento dos modelos de ML é ilustrada por tabelas e gráficos. Além disso, este capítulo é dedicado à análise das métricas de desempenho obtidas após a aplicação de técnicas de FS. São considerados tanto treinamentos por *hold-out*, como CV.

5.1 Reamostragem e Balanceamento

São gerados dois *datasets* **distintos**, sub-amostras do original, de tal forma que estes se tornem conjuntos de dados perfeitamente balanceados ($J = 1$). A amostragem é feita de maneira que os dois conjuntos de dados finais possuam a mesma quantidade de elementos, para que as métricas de tempo sejam comparadas de forma justa. Os critérios para a realização desta amostragem estão no Apêndice A.

Os *datasets* resultantes são mostrados na Tabela 5. Apesar de possuírem distribuições de classes distintas, ambos os conjuntos de dados gerados possuem 5280 elementos.

A Figura 12 mostra a quantidade de elementos em cada classe, resultando do processo de se balancear o *dataset* original para se obter uma igualdade entre amostras de ataque e tráfego benigno. Do ponto de vista binário, seu Índice de Pielou é $J = 1$. A título de completude temos que, do ponto de vista multiclasse, este *dataset* apresenta $J = 0,761435$. Contudo, este conjunto de dados será usado apenas para o treinamento de **classificadores binários**.

O segundo conjunto de dados gerado é mostrado na Figura 13. Todas as 12 classes de elementos (11 ataques e tráfego normal) estão igualmente representadas. Seu Índice de Pielou multiclasse é de $J = 1$, como foi projetado para ser. Do ponto de vista binário, este *dataset* possui $J = 0,413817$. Porém, analogamente ao conjunto de dados anterior, este será usado exclusivamente para o treinamento de **classificadores multiclasse**.

Tabela 5: *Datasets* sub-amostrados e balanceados

Novo Binário		Novo Multiclasse	
Classe	Amostras	Classe	Amostras
ATAQUE	2640	TFTP	440
		DrDoS_SNMP	440
		DrDoS_DNS	440
		DrDoS_MSSQL	440
		DrDoS_NetBIOS	440
		DrDoS_UDP	440
		DrDoS_SSDP	440
		DrDoS_LDAP	440
		Syn	440
		DrDoS_NTP	440
		UDP-lag	440
BENIGNO	2640	BENIGNO	440

Figura 12: Contagem de elementos do *dataset* Binário-balanceado

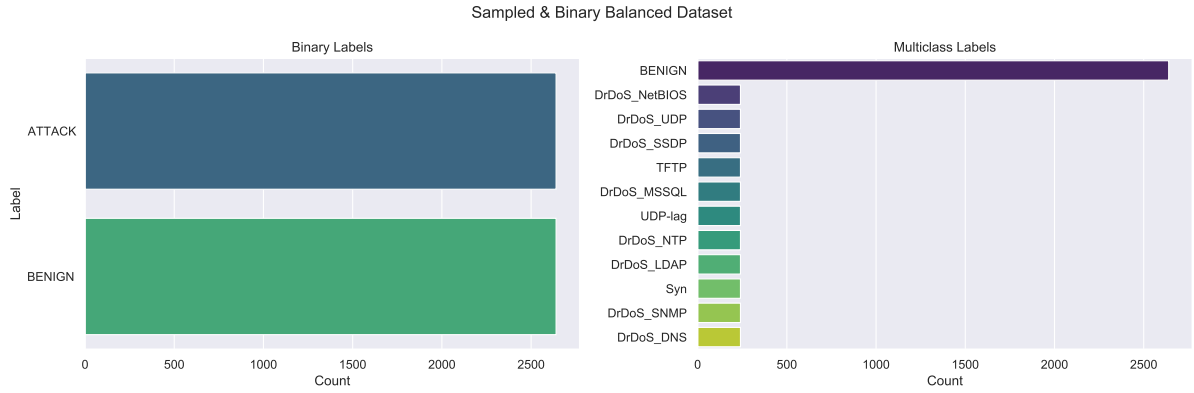
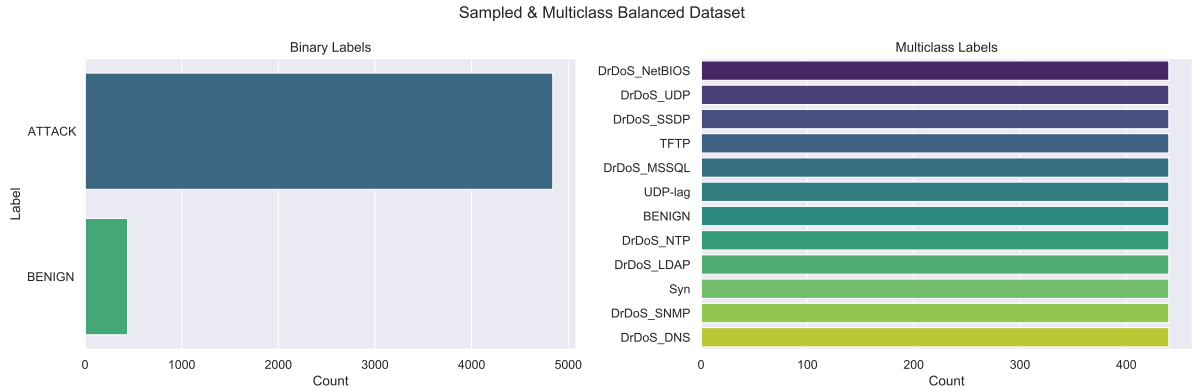


Figura 13: Contagem de elementos do *dataset* Multiclasse-balanceado



5.2 Seleção de Variáveis Independente do Rótulo

Após a construção dos dois *datasets* descritos na seção anterior, os dados seguem pelo processo mostrado na Figura 10 (que ilustra o fluxo de dados para os experimentos). Para a etapa de "Pré-processar", foram eliminadas 9 variáveis categóricas, a saber: "Flow ID", "Source IP", "Source Port", "Destination IP", "Destination Port", "Protocol", "Timestamp", "SimillarHTTP" e "Inbound".

Os endereços IP de origem (Source IP) e destino (Destination IP) servem apenas como identificadores dos dispositivos usados no *testbed* que deu origem ao conjunto de dados original. Os dados são coletados por Sharafaldin et al. (SHARAFALDIN et al., 2019) distribuindo ataques igualmente entre os servidores vítimas. Trabalhos como os de Parfenov et al. (PARFENOV et al., 2020) e Wei et al. (WEI et al., 2021) também optam por descartar endereços IP em suas análises preditivas, uma vez que é o comportamento estatístico dos parâmetros de rede que está sendo levado em consideração para a detecção de ataques.

Para os "Métodos Básicos" da Figura 10, são eliminados todos os atributos que tem variância $\sigma^2 < 0,01$. Para "Correlação", são agrupadas *features* com $|\rho| > 0,9$ entre si. Os valores de σ^2 e ρ são tomados tendo como base o método desenvolvido por Xie et al. (XIE et al., 2009) e se provam adequados para os conjuntos de dados usados nos experimentos.

Note que tanto os "Métodos Básicos", como a remoção de *features* altamente correlacionadas, são aplicados em dois cenários independentes, binário e multiclasse. Ainda que **não dependam dos rótulos**, estes processos resultam em conjuntos de variáveis diferentes para os dois cenários, pois são aplicados em *datasets* diferentes.

5.2.1 Resultados com *hold-out*

Como forma de ilustrar resultados intermediários no processamento dos dados, a Figura 14 mostra a Matriz de Correlação para o caso do *dataset* multiclasse. Uma matriz análoga é gerada para o caso binário e está disponível no repositório de códigos do projeto. Variáveis altamente correlacionadas em módulo aparecem na Figura 14 em tons de azul e vermelho mais intensos. Apenas um representante de cada grupo de *features* correlacionadas é o suficiente para ser usado nos modelos preditivos, sendo que o restante dos atributos pode ser descartado (já que estes contém praticamente a mesma informação).

A Matriz de Confusão da Figura 15 é resultado do treinamento de um classificador

Figura 14: Matriz de Correlação para *dataset* multiclasse

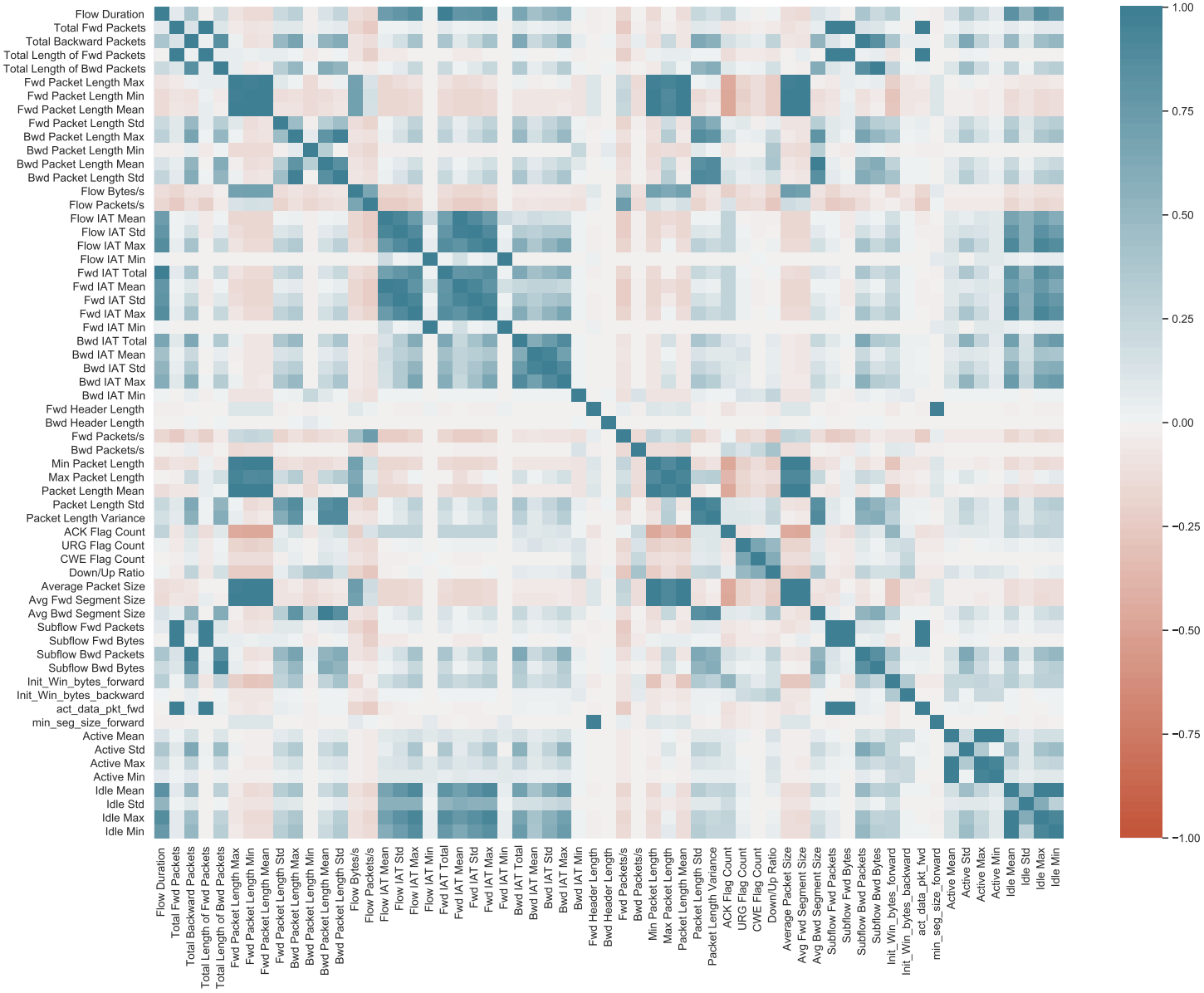
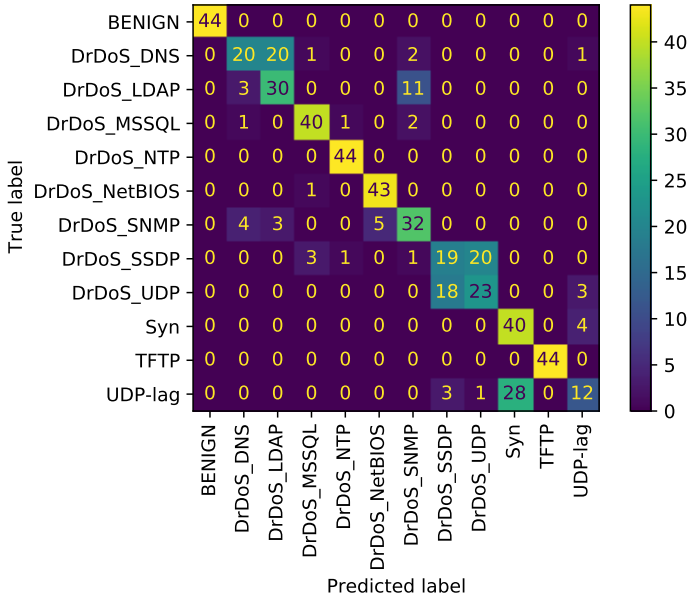


Figura 15: Matriz de Confusão para modelo treinado após remoção de variáveis altamente correlacionadas



XGBoost para o conjunto de dados multiclasse. É utilizado um procedimento de *hold-out*, com 90% dos dados no conjunto de treino e 10% no conjunto de teste. A divisão é feita de forma estratificada (isto é, mantendo as proporções entre as classes). Novamente, temos um resultado análogo para o caso binário, disponível no repositório do projeto. Além disso, também são treinados modelos intermediários para as etapas de "Pré-processar" e de "Métodos Básicos", também omitidos aqui por simplicidade (a relação de todos os experimentos está listada no Apêndice C).

A partir da Matriz de Confusão, podem ser calculadas diversas métricas de desempenho, como descritas na seção 4.4: $AC = 0,7405$, $PR_M = 0,7395$, $RC_M = 0,7405$, $F1_M = 0,7288$. Estas métricas são levantadas para todos os classificadores treinados nas simulações computacionais.

A Tabela 6 a seguir ilustra os resultados obtidos treinando-se modelos e testando-os através de métodos de *hold-out*. Para cada conjunto de dados (binário e multiclasse), são utilizadas 90% das amostras para treino e 10% para teste. Estas proporções são escolhidas para coincidir com as proporções de CV de 10 repetições. O *hold-out* possui limitações (JAMES et al., 2013): pode-se, por exemplo, escolher um conjunto de testes aleatório especialmente favorável ao treino realizado. Desta forma as métricas de desempenho podem indicar um bom classificador, mas que falha ao tentar generalizar o resultado para casos muito diferentes do conjunto de treino.

A primeira parte da Tabela 6 contém os resultados para os classificadores binários,

Tabela 6: Desempenho em *hold-out* dos métodos de FS independentes do rótulo

Classificadores Binários			
Métrica	**Pré-processar**	**Métodos Básicos**	**Correlação**
Acurácia	0,998106	0,998106	0,998106
Precisão	0,996226	0,996226	0,996226
Recall	1,000000	1,000000	1,000000
F1-score	0,998110	0,998110	0,998110
fit_time	0,402000	0,361994	0,344000
fs_time	0	0,019000	0,076010
train_time	0,402000	0,342995	0,267990
Features na Entrada	86	77	63
Features na Saída	77	63	42
Classificadores Multiclasse			
Métrica	**Pré-processar**	**Métodos Básicos**	**Correlação**
Acurácia	0,731061	0,731061	0,740530
Precisão	0,729548	0,729548	0,739516
Recall	0,731061	0,731061	0,740530
F1-score	0,720807	0,720807	0,728800
fit_time	5,317004	4,798003	3,136010
fs_time	0	0,017003	0,073002
train_time	5,317004	4,781000	3,063008
Features na Entrada	86	77	61
Features na Saída	77	61	35
Classificadores Multiclasse - One vs Rest			
Métrica	**Pré-processar**	**Métodos Básicos**	**Correlação**
Acurácia	0,740530	0,740530	0,736742
Precisão	0,740498	0,740498	0,738319
Recall	0,740530	0,740530	0,736742
F1-score	0,728200	0,728200	0,724470
fit_time	7,610002	7,429997	4,396002
fs_time	0	0,020002	0,075000
train_time	7,610002	7,409995	4,321002
Features na Entrada	86	77	61
Features na Saída	77	61	35
Classificadores Multiclasse - One vs One			
Métrica	**Pré-processar**	**Métodos Básicos**	**Correlação**
Acurácia	0,740530	0,740530	0,744318
Precisão	0,741742	0,741742	0,747879
Recall	0,740530	0,740530	0,744318
F1-score	0,730063	0,730063	0,734853
fit_time	7,544997	7,287995	6,588996
fs_time	0	0,018000	0,080002
train_time	7,544997	7,269995	6,508994
Features na Entrada	86	77	61
Features na Saída	77	61	35

sujeitos aos métodos de FS independentes do rótulo das amostras. De forma geral, as métricas de AC, PR, RC e F1 Score se mostraram altamente satisfatórias (próximas de 1) para todos os métodos utilizados. As diferenças mais significativas estão nos tempos gastos no processo.

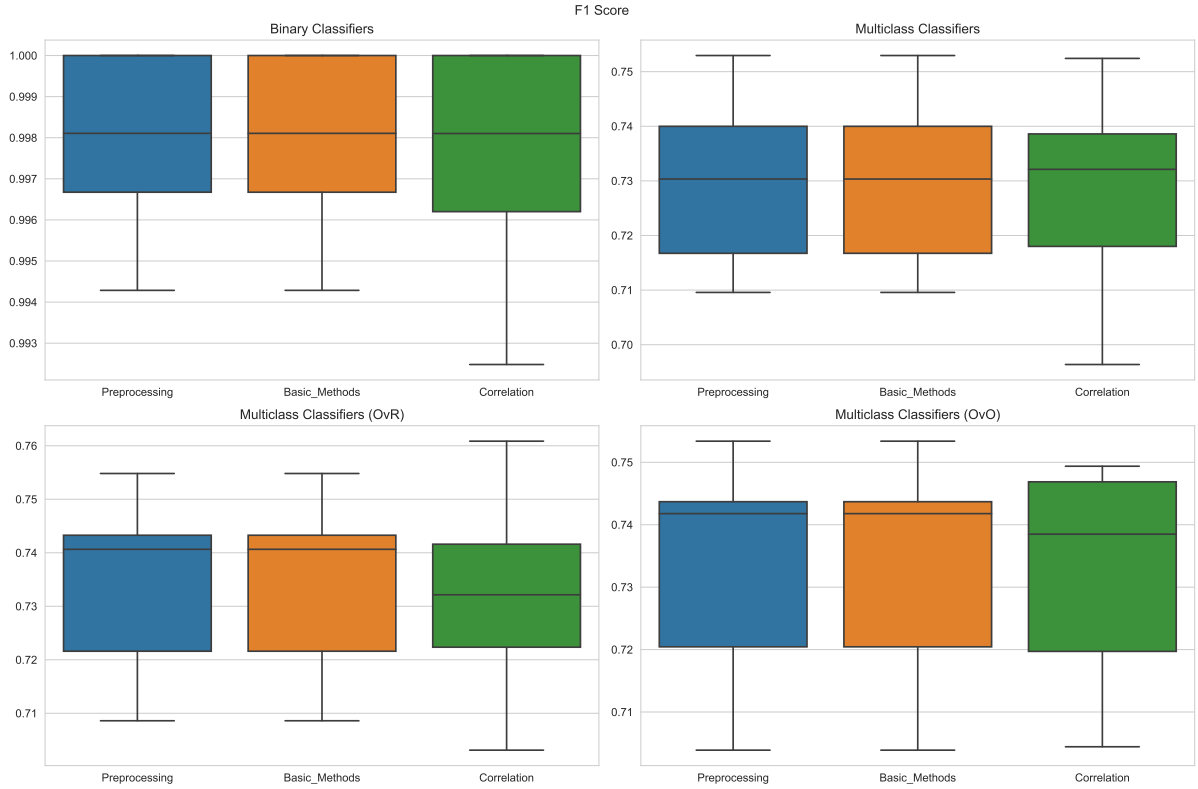
Neste ponto, é preciso diferenciar as técnicas de FS independentes do rótulo das demais. Pela arquitetura proposta na Figura 10, as *features* de saída de um método são as de entrada do seguinte. Só se sabe a quantidade de variáveis eliminadas quando o método termina de ser executado. Por exemplo, ainda que os "Métodos Básicos" sejam executados em $0,019000s$ e a remoção de *features* correlacionadas gaste $0,076010s$, os primeiros eliminam 14 variáveis, enquanto os últimos eliminam 21. Por se dar de forma sequencial na arquitetura proposta, contando progressivamente com menos variáveis de saída, os métodos independentes do rótulo também treinam o classificador propriamente dito (train_time) cada vez mais rápido.

A segunda parte da Tabela 6 apresenta os resultados dos classificadores multiclasse. Os algoritmos treinados apresentam métricas de desempenho inferiores às dos classificadores binários, ficando em torno de $0,7$ para todos os indicadores. O tempo de treinamento do modelo para o caso multiclasse também aumentou consideravelmente (em torno de 10 vezes).

O desempenho dos classificadores OvR está exposto na terceira parte da Tabela 6. As mesmas observações sobre o treinamento do modelo e dos métodos de FS dos classificadores multiclasse tradicionais também se aplica aqui. Existe um pequeno aumento médio nas métricas de qualidade, mas às custas de um aumento médio de 46% no tempo de ajuste. Neste caso, o aumento se deve a parcela do treinamento do modelo (train_time).

Por fim, o desempenho dos classificadores multiclasse OvO está detalhado na quarta parte da Tabela 6. Aqui valem as observações feitas acerca dos classifcadores multiclasse OvR, com a diferença deste tipo de treinamento de modelo ser o mais computacionalmente custoso de todos os considerados até o momento. Os tempos de treinamento são os mais extensos dentre as técnicas utilizadas. As métricas de desempenho ficam, na média, melhores que as do multiclasse tradicional e também melhores do que as do multiclasse OvR.

Figura 16: *F1-Score* para métodos de seleção de variáveis independentes do rótulo



5.2.2 Resultados com CV

Os resultados com CV se diferem dos com *hold-out* na medida em que são conjuntos de pontos experimentais e, portanto, podem ser descritos por médias e desvios-padrões. Também podem ser representados graficamente por *boxplots*, como o da Figura 16, que mostra o *F1-Score* para os experimentos com FS independente do rótulo das amostras. Analogamente, também foram mensuradas a Acurácia, a Precisão e o *Recall*. O comportamento de todas estas métricas foi muito semelhante em relação ao uso de seleção de variáveis.

Visualmente, os resultados de "Pré-processar" e da aplicação dos "Métodos Básicos" foram os mesmos, indicando que a remoção de *features* com baixa variância não modifica a performance de um classificador (tanto binário, como multiclasse). O desempenho ao se retirar as variáveis altamente correlacionadas também não se alterou muito, com um leve aumento no intervalo interquertil das métricas ao se utilizar a CV para os classificadores binário e multiclasse OvO.

Os resultados dos classificadores submetidos a métodos de FS independentes do rótulo das amostras estão resumidos na Tabela 7 a seguir.

Tabela 7: Desempenho em CV dos métodos de FS independentes do rótulo

Classificadores Binários			
Métrica	**Pré-processar**	**Métodos Básicos**	**Correlação**
Acurácia	99,79 ± 0,23	99,79 ± 0,23	99,75 ± 0,27
Precisão	99,85 ± 0,36	99,85 ± 0,36	99,81 ± 0,47
Recall	99,73 ± 0,40	99,73 ± 0,40	99,7 ± 0,39
F1-score	99,79 ± 0,23	99,79 ± 0,23	99,75 ± 0,27
fit_time	0,4994 ± 0,0353	0,4759 ± 0,01569	0,472 ± 0,02
fs_time	0,0 ± 0,0	0,0417 ± 0,00488	0,144 ± 0,012
train_time	0,4994 ± 0,0353	0,4342 ± 0,01117	0,328 ± 0,01
Features na Entrada	86	77	63
Features na Saída (média)	77	63	41
Classificadores Multiclasse			
Métrica	**Pré-processar**	**Métodos Básicos**	**Correlação**
Acurácia	73,94 ± 1,64	73,94 ± 1,64	73,81 ± 1,85
Precisão	74,3 ± 2,04	74,3 ± 2,04	74,17 ± 2,21
Recall	73,94 ± 1,64	73,94 ± 1,64	73,81 ± 1,85
F1-score	72,97 ± 1,47	72,97 ± 1,47	72,8 ± 1,70
fit_time	5,5701 ± 0,1481	5,2737 ± 0,215	3,8897 ± 0,0719
fs_time	0,0 ± 0,0	0,0429 ± 0,0061	0,1314 ± 0,0071
train_time	5,5701 ± 0,1481	5,2308 ± 0,2118	3,7583 ± 0,0687
Features na Entrada	86	77	61
Features na Saída (média)	77	61	36
Classificadores Multiclasse - One vs Rest			
Métrica	**Pré-processar**	**Métodos Básicos**	**Correlação**
Acurácia	74,38 ± 1,67	74,38 ± 1,67	74,05 ± 1,8
Precisão	74,81 ± 2,16	74,81 ± 2,16	74,44 ± 2,25
Recall	74,38 ± 1,67	74,38 ± 1,67	74,05 ± 1,8
F1-score	73,43 ± 1,55	73,43 ± 1,55	73,1 ± 1,67
fit_time	6,1049 ± 0,6005	6,158 ± 0,1912	4,4419 ± 0,0633
fs_time	0,0 ± 0,0	0,0411 ± 0,0068	0,1279 ± 0,0177
train_time	6,1049 ± 0,6005	6,1169 ± 0,1853	4,314 ± 0,0579
Features na Entrada	86	77	61
Features na Saída (média)	77	61	36
Classificadores Multiclasse - One vs One			
Métrica	**Pré-processar**	**Métodos Básicos**	**Correlação**
Acurácia	74,3 ± 1,84	74,3 ± 1,84	74,28 ± 1,7
Precisão	74,53 ± 2,21	74,53 ± 2,21	74,57 ± 2,11
Recall	74,3 ± 1,84	74,3 ± 1,84	74,28 ± 1,7
F1-score	73,33 ± 1,67	73,33 ± 1,67	73,28 ± 1,61
fit_time	8,3781 ± 0,3469	7,9289 ± 0,1978	7,0986 ± 0,2957
fs_time	0,0 ± 0,0	0,0415 ± 0,0053	0,128 ± 0,0051
train_time	8,3781 ± 0,3469	7,8874 ± 0,1973	6,9706 ± 0,2964
Features na Entrada	86	77	61
Features na Saída (média)	77	61	36

No geral, os classificadores binários obtiveram desempenhos superiores aos multiclasse. Entre os classificadores multiclasse, os que se utilizaram das estratégias de meta-aprendizado (OvO e OvR) obtiveram desempenhos levemente superiores. Pode-se notar, na Tabela 7 que, mesmo havendo estabilidade nas métricas de classificação, a quantidade de variáveis pôde ser reduzida consideravelmente. Há uma redução **média** de 46% (de 77 para 41) no número de variáveis para os classificadores binários, enquanto os classificadores multiclasse tem suas variáveis reduzidas em 53% (de 77 para 36).

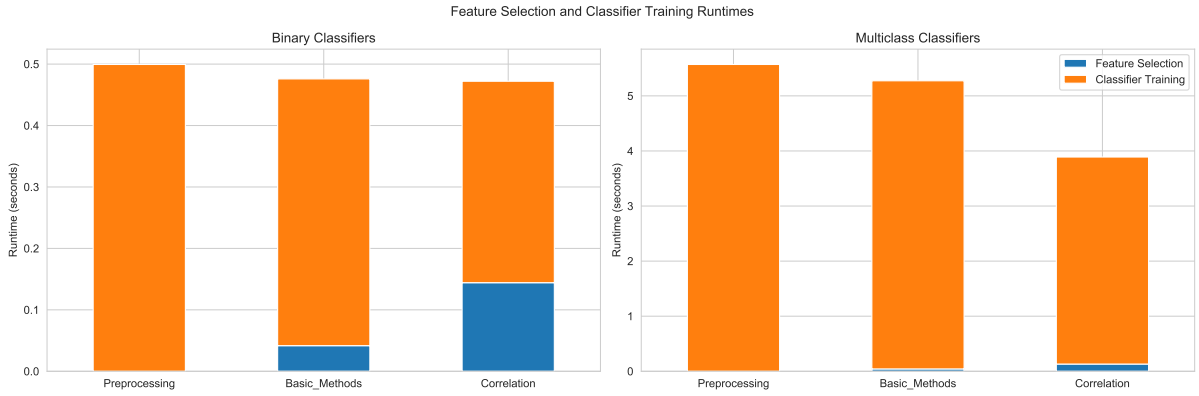
Ainda que o número total de amostras seja o mesmo, os *datasets* binário e multiclasse foram construídos de maneiras diferentes (como já ilustrado na Tabela 5). O conjunto de dados binário possui mais elementos de tráfego benigno. Por ter mais amostras deste tipo de tráfego, que é mais próximo do comportamento normal de usuários da Internet (que, por sua vez, é imprevisível), o *dataset* binário tem menos colunas com baixa variância, sendo que estas colunas seriam removidas pelos "Métodos Básicos" descritos anteriormente. O tráfego benigno também não possui os padrões repetitivos dos ataques, que seriam agrupados e removidos em uma análise de correlação.

É possível notar que os três classificadores multiclasse possuem o mesmo número de *features* na saída (36 em média). Os experimentos foram programados usando a mesma semente aleatória para dividir o conjunto de dados em 10 dobras, de forma a tornar os experimentos reprodutíveis. Desta maneira, como o *dataset* de entrada é o mesmo para os três classificadores multiclasse, é dividido da mesma forma no processo de CV e passa pela FS usando os mesmos hiperparâmetros, os classificadores multiclasse acabam sendo treinados com o mesmo número de variáveis.

A diferença no número de variáveis se reflete no Tempo de Ajuste (fit_time), conforme a Figura 17 a seguir. O Tempo de Ajuste foi definido como o tempo para se executar tanto a seleção de variáveis, como o treinamento do classificador. A Figura 17 mostra duas situações distintas: para os classificadores binários, o Tempo de Ajuste permanece aproximadamente no mesmo patamar com o uso dos métodos de seleção de variáveis, ainda que a quantidade de *features* tenha diminuído. Já para os classificadores multiclasse, podemos observar uma acentuada diminuição do Tempo de Ajuste. Para os classificadores multiclasse OvO e multiclasse OvR, os resultados são semelhantes aos dos classificadores multiclasse tradicionais (e, portanto, foram omitidos na Figura 17).

Ao separarmos o tempo para a realização de FS do tempo de treinamento do modelo de ML, podemos verificar a influência de cada etapa do processo. A Tabela 7 também nos mostra que o tempo de treinamento dos classificadores binários já é baixo nos modelos

Figura 17: Tempos de Ajuste: seleção de variáveis e treinamento do classificador



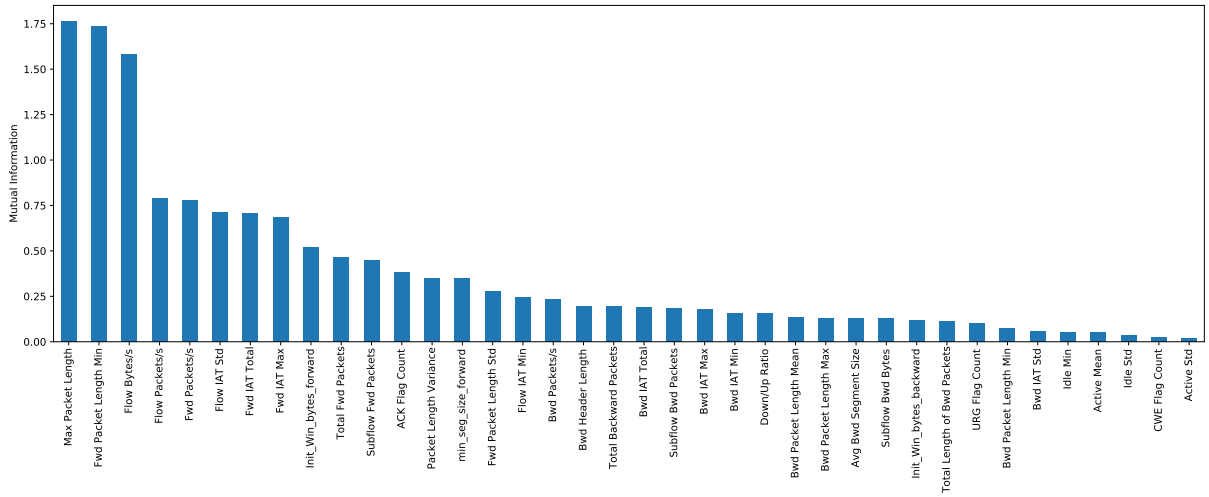
iniciais. Ao retiramos variáveis através dos demais métodos, o Tempo de Ajuste cai apenas $5,5\%$ (de $0,4994s$ para $0,472s$) para os classificadores binários. Para os classificadores multiclasse, essa queda é de $30,1\%$ (de $5,5701s$ para $3,8897s$).

Contudo, as etapas de seleção variáveis são executadas em tempos com ordens de grandeza relativamente próximos aos tempos de treinamento dos classificadores binários, mas significativamente menores que os dos classificadores multiclasse. Isso faz com que o Tempo de Ajuste dos classificadores binários não se altere tanto entre os "Métodos Básicos" (que retiram *features* de baixa variância) e "Correlação": o aumento de complexidade para a remoção de variáveis altamente correlacionadas, que necessita de mais operações aritméticas para ser realizada, é contrabalanceado pelo treinamento de um classificador com menos variáveis. Para o caso multiclasse, o tempo de treinamento do classificador (train_time) é a parcela dominante no Tempo de Ajuste. Note que, por serem métodos de seleção de variáveis independentes dos rótulos, sua execução depende somente da quantidade de variáveis preditoras.

5.3 Seleção de Variáveis Dependente do Rótulo

Os métodos de seleção de variáveis dependentes do rótulo das amostras produzem *rankings* dos atributos conforme uma estatística bem estabelecida. A partir disso, é possível escolher as K melhores variáveis segundo este critério.

Figura 18: Informação Mútua entre as variáveis e o rótulo para *dataset* Multiclasse



5.3.1 Resultados com *hold-out*

A Figura 18 mostra os valores de Informação Mútua entre todas as variáveis do conjunto de dados multiclasse e o rótulo das amostras. Conforme a arquitetura da Figura 10, este processo se dá após a aplicação dos métodos independentes do rótulo. Os valores atribuidos a cada variável estão ordenados do maior para o menor. Arbitrariamente, podem ser escolhidas as K *features* mais relevantes (no caso da Informação Mútua, as com maior valor) e descartar as demais. Os demais métodos de FS dependentes do rótulo (com exceção do RFE) produzem gráficos de importância de variáveis semelhantes.

Escolhendo-se as $K = 20$ variáveis mais relevantes do *ranking* anterior e treinando um modelo XGBoost chega-se à Matriz de Confusão da Figura 19, que por sua vez, gera as seguintes métricas de qualidade: $AC = 0,7311$, $PR_M = 0,7309$, $RC_M = 0,7311$, $F1_M = 0,7198$.

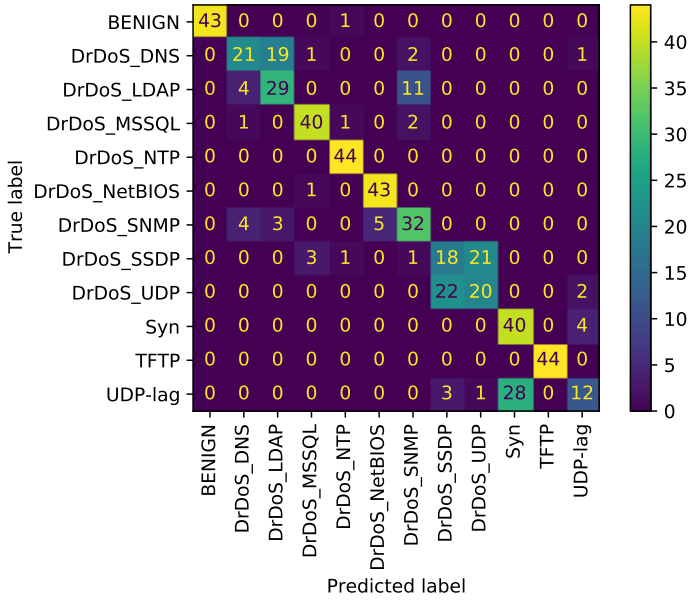
Os resultados da Tabela 8 são análogos aos da Tabela 6, uma vez que também foram obtidos por *hold-out*, mas se utilizando de métodos dependentes do rótulo das amostras. Para todos estes experimentos, os classificadores foram treinados após serem escolhidas as $K = 20$ variáveis mais importantes segundo cada método.

A primeira parte da Tabela 8 mostra as métricas de desempenho dos classificadores binários. Assim como no caso dos métodos de FS independentes do rótulo das amostras, os métodos dependentes do rótulo obtém resultados todos próximos de 1. Os métodos de FS dependentes do rótulo gastam um tempo comparável entre si para treinar o classificador, uma vez que removem a mesma quantidade de variáveis no experimento (20 variáveis).

Tabela 8: Desempenho em *hold-out* dos métodos de FS dependentes do rótulo

Classificadores Binários						
Métrica	**ANOVA**	**IM**	**ReliefF**	**Ganho XGB**	**Ensemble**	**RFE**
Acurácia	1,000000	0,99811	0,99621	0,998106	1,000000	1,000000
Precisão	1,000000	0,99623	0,99248	0,996226	1,000000	1,000000
Recall	1,000000	1,000000	1,000000	1,000000	1,000000	1,000000
F1-score	1,000000	0,998110	0,99623	0,998110	1,000000	1,000000
fit_time	0,204001	1,661996	1,415000	0,544994	3,394994	8,038990
fs_time	0,011000	1,390004	1,149001	0,354991	3,197000	7,775993
train_time	0,193	0,27199	0,266	0,190003	0,197994	0,262996
Features na Entrada	40	40	40	40	40	40
Features na Saída	20	20	20	20	20	20
Classificadores Multiclasse						
Métrica	**ANOVA**	**IM**	**ReliefF**	**Ganho XGB**	**Ensemble**	**RFE**
Acurácia	0,721591	0,73106	0,71212	0,715909	0,719697	0,715909
Precisão	0,725369	0,730900	0,7234	0,720729	0,724781	0,720036
Recall	0,721591	0,731061	0,712121	0,715909	0,719697	0,715909
F1-score	0,710557	0,719758	0,69482	0,703391	0,707310	0,702948
fit_time	2,422000	4,422004	3,544999	7,542996	9,735003	74,783010
fs_time	0,015000	1,905996	1,420998	5,090000	7,200004	72,217008
train_time	2,407	2,51601	2,124	2,452996	2,534998	2,566002
Features na Entrada	38	38	38	38	38	38
Features na Saída	20	20	20	20	20	20
Classificadores Multiclasse - One vs Rest						
Métrica	**ANOVA**	**IM**	**ReliefF**	**Ganho XGB**	**Ensemble**	**RFE**
Acurácia	0,719697	0,73864	0,71212	0,723485	0,723485	0,723485
Precisão	0,720980	0,737646	0,71843	0,729764	0,727361	0,731717
Recall	0,719697	0,738636	0,712121	0,723485	0,723485	0,723485
F1-score	0,708374	0,725696	0,69481	0,711870	0,711679	0,711275
fit_time	3,336993	5,056992	4,003999	8,015990	10,513992	75,669993
fs_time	0,019000	1,759995	1,362004	4,885996	7,063996	72,420000
train_time	3,31799	3,297	2,642	3,129994	3,449996	3,249993
Features na Entrada	38	38	38	38	38	38
Features na Saída	20	20	20	20	20	20
Classificadores Multiclasse - One vs One						
Métrica	**ANOVA**	**IM**	**ReliefF**	**Ganho XGB**	**Ensemble**	**RFE**
Acurácia	0,715909	0,73864	0,70644	0,721591	0,725379	0,719697
Precisão	0,715601	0,740444	0,71172	0,725854	0,729644	0,723383
Recall	0,715909	0,738636	0,706439	0,721591	0,725379	0,719697
F1-score	0,704428	0,729004	0,6884	0,710447	0,714235	0,707585
fit_time	5,380995	7,829000	6,923990	11,074990	12,986992	78,085993
fs_time	0,013999	1,855009	1,283996	5,214999	7,077000	72,380001
train_time	5,367	5,97399	5,63999	5,85999	5,909992	5,705992
Features na Entrada	38	38	38	38	38	38
Features na Saída	20	20	20	20	20	20

Figura 19: Matriz de Confusão para modelo treinado com as $K = 20$ *features* com maior Informação Mútua com o rótulo

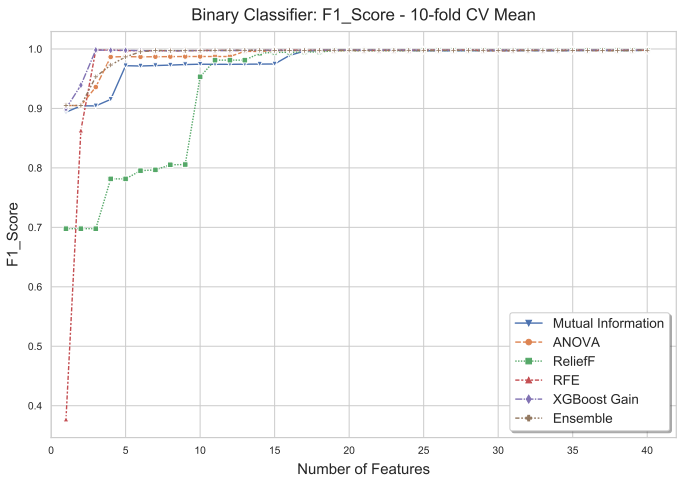


Dentre estes, pode-se observar que ANOVA e o Ganho do XGBoost apresentam melhor desempenho de tempo (fs_time). RFE, que é mais custoso computacionalmente, leva um tempo significativamente maior para ser executado.

Pode-se observar na segunda parte da Tabela 8 que o tempo de **treinamento** aumenta significativamente para o classificador multiclasse (em relação ao binário). Por causa disso, os métodos de FS que incorporam a realização de um treinamento de classificação para serem executados (Ganho do XGBoost, *Ensemble* e RFE) tiveram seus tempos extendidos. Em particular, RFE, que executa uma série de treinamentos de modelo, recursivamente, demorou 10 vezes mais (em termos de fs_time) para ser executado que no caso binário. O tempo de FS de ANOVA, Informação Mútua e ReliefF também crescem com o número de rótulos distintos, mas não na mesma proporção. Já os métodos independentes do rótulo (analisados anteriormente), por definição, não tem seu tempo alterado com a mudança no número de classes no variável rótulo.

Para as duas últimas partes da Tabela 8, que tratam dos classificadores multiclasse OvR e OvO, nota-se uma estabilidade nas métricas de desempenho em relação aos classificadores multiclasse tradicionais. Este resultados podem sugerir que o uso destas estratégias não vale a pena e acabam introduzindo complexidade (e consequentemente, tempo de execução) no sistema. Contudo, uma análise mais profunda se faz necessária, para que este mesmo comportamento seja verificado com diferentes valores de K variáveis escolhidas.

Figura 20: Média do *F1-Score* em função do número de variáveis (Binário)



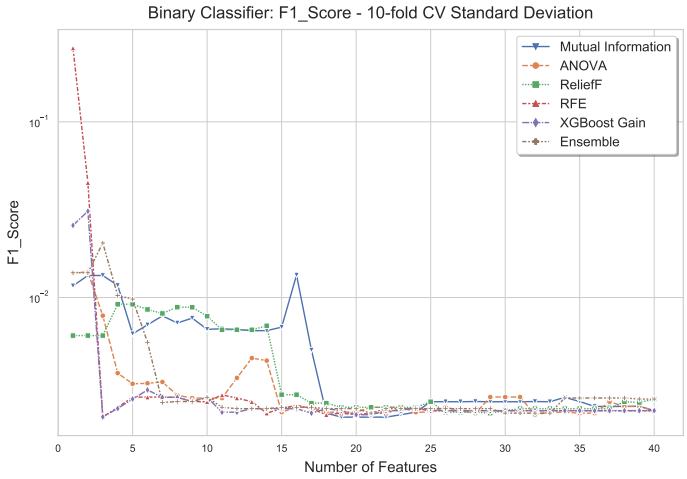
5.3.2 Resultados com CV

Os métodos de seleção de variáveis dependentes do rótulo das amostras produzem *rankings* dos atributos conforme uma estatística bem estabelecida. A partir disso, é possível escolher as K melhores variáveis segundo este critério. As Figuras 20 a 27 ilustram a variação de métricas de qualidade em função do valor de K.

Para classificadores binários, foram levantadas curvas de AC, PR, RC e F1 para os métodos de seleção de variáveis considerados. A Figura 20 ilustra as curvas da Média do *F1-Score* (dentro de um processo de CV). As demais métricas possuem curvas com comportamento semelhante e, portanto, foram omitidos. Todos os gráficos estão disponíveis no repositório do projeto.

As métricas de qualidade se mantêm estáveis até aproximadamente 17 variáveis restantes para os classificadores binários, em todos os métodos de FS considerados. Isso representa uma redução de 78% no número de *features*, comparado aos primeiros treinamentos, em que apenas "Pré-processar" havia sido realizado e 77 haviam sido usadas. Ao escolhermos menos de 17 variáveis, notamos uma degradação de performance em quase todos os métodos, em particular para ReliefF. Conforme diminuímos o espaço multidimensional removendo variáveis, os pontos no subespaço resultante se tornam mais próximos. Ou seja, os valores obtidos com medidas de distância diminuem. Desta maneira, a escolha dos pontos em particular que ReliefF faz é mais sensível à flutuações nos dados de entrada, o que causam erros que são refletidos no *F1-Score*. Para menos de 5 variáveis, RFE sofre uma abrupta piora nas métricas, que se mantém altas ao utilizarmos mais atributos. A execução de RFE é feita considerando as variáveis individualmente. É possível que

Figura 21: Desvio-Padrão do *F1-Score* em função do número de variáveis (Binário)

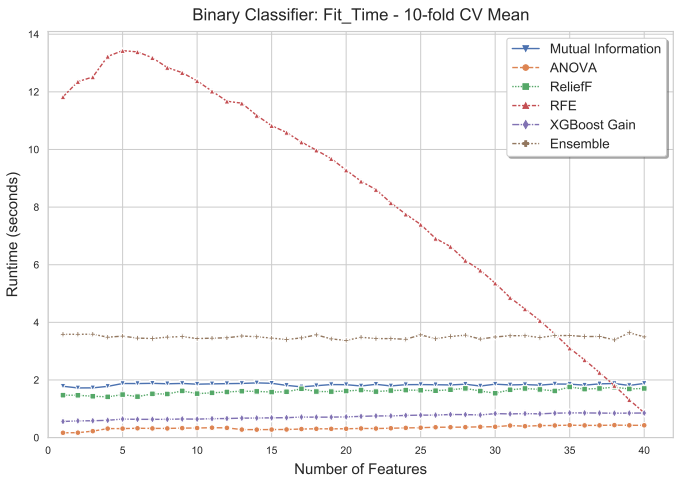


uma variável tenha um poder preditivo maior quando usada em conjunto com outra no treinamento de um modelo de ML. Contudo, no RFE, se esta outra *feature* é descartada prematuramente, os resultados da classificação serão piores quando um pequeno número de variáveis for utilizado.

A Figura 21 mostra o Desvio-Padrão do *F1-Score* para todas as técnicas de seleção de variáveis. O gráfico foi plotado em escala logarítmica no eixo vertical. Valores mais baixos para o Desvio-Padrão indicam menores flutuações para a métrica em questão. Novamente, ao escolhermos mais de 17 *features*, todas as técnicas produzem resultados com baixas variações. Abaixo desse valor, a IM e ReliefF possuem variações maiores. No caso da IM e sua adaptação para valores numéricos (isto é, não categóricos), seu cálculo é influenciado pelo processo de compartimentalização (*binning*) dos dados da coluna em questão. Esse processo, por sua vez, é influenciado pelas etapas de CV. Como não é possível garantir a homogeneidade nos conjuntos gerados por CV, as variáveis com mais informação também irão diferir, o que acaba aumentando a variância das métricas de qualidade. Para ReliefF, deve ser levado em consideração que a escolha de pontos que terão sua vizinhança checada é aleatória, o que contribui para um aumento na variância.

O Tempo de Ajuste de cada um dos métodos de FS é mostrado na Figura 22. Existem poucos cruzamentos neste gráfico, uma vez que a maioria das técnicas de seleção de variáveis dependente do rótulo leva um tempo fixo para escolher K atributos (qualquer que seja o valor de K). A exceção é o método RFE, que, dada a sua natureza recursiva, leva mais tempo quando precisa eliminar mais variáveis. Nos piores casos, chega a demorar 3 vezes mais que os demais métodos. Podemos destacar também o método *En-*

Figura 22: Tempo de Ajuste em função do número de variáveis (Binário)



semble (WRFS), que é executado aproximadamente na soma dos 4 tempos dos métodos que o compõe. O método ANOVA possui a execução mais rápida de todos os métodos considerados. Isto ocorre pois ANOVA executa menos operações aritméticas, além de não realizar nenhum processo iterativo ou de compartimentalização, como os outros métodos de FS dependentes do rótulo considerados.

O gráfico da BCR para a média do *F1-Score* é plotado em escala logarítmica na Figura 23. Uma vez que o *F1-Score* (e, analogamente, as demais métricas), se mantém aproximadamente constante ao retirarmos atributos, é vantajoso escolhermos os métodos de seleção de variáveis que sejam executados mais rapidamente para chegar a esse resultado. Para os classificadores binários, ANOVA é o que melhor atende este critério, independentemente do valor de K. O Ganho do XGBoost como seletor de variáveis aparece como segunda opção, uma vez que a classificação binária possui um baixo tempo de treinamento. Conforme retiramos variáveis, o RFE vai progressivamente degradando sua BCR pois, ainda que não sofra uma queda tão acentuada em AC, PR, RC e F1, seu custo em tempo de execução cresce rapidamente.

No caso dos classificadores multiclasse, também foram geradas curvas de AC, PR, RC e F1 para os métodos de seleção de variáveis dependentes do rótulo utilizados. A Figura 24 traz a Média do *F1-Score*, em funcão do número de variáveis escolhidas. As demais métricas possuem gráficos análogos com resultados parecidos. Um ponto a se notar é que, em um *dataset* perfeitamente balanceado, o valor de AC é sempre igual ao de RC, independente do número de classes. Esse resultado é demonstrado no Apêndice D.

Há mais variedade de resultados multiclasse do que no caso binário. Para classificado-

Figura 23: BCR(F1) em função do número de variáveis (Binário)

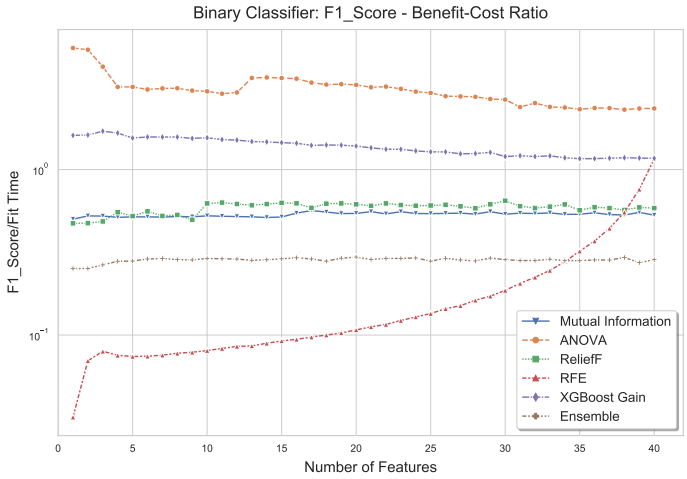
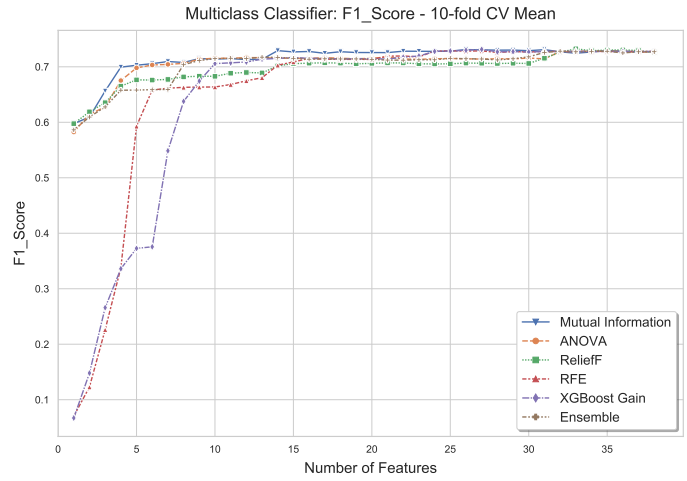


Figura 24: Média do *F1-Score* em função do número de variáveis (Multiclasse)

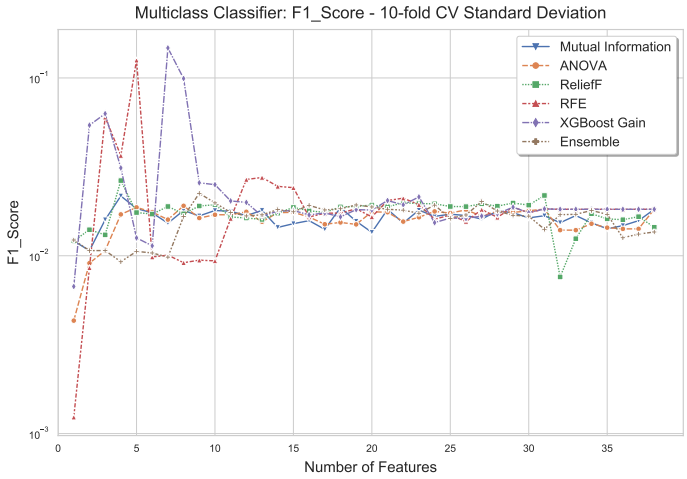


res multiclasse com menos de 31 *features*, o método de seleção de variáveis escolhido passa a ter mais influência. Note que isso representa uma diminuição de 60% na quantidade de *features* usada no modelo original (77 variáveis).

IM é o método que obtém os melhores resultados para a média do *F1-Score*. O uso de IM é particularmente benéfico para a classificação multiclasse, uma vez que o maior número de classes diminui a probabilidade individual de ocorrência de cada rótulo. O logarítmo que existe no cálculo da IM evidencia estas pequenas diferenças, fazendo com que os valores obtidos por cada classe se distanciem e estejam mais adequados ao processo de *ranking* das variáveis.

Também observa-se que o Ganho do XGBoost tem seu desempenho degradado rapi-

Figura 25: Desvio-Padrão do *F1-Score* em função do número de variáveis (Multiclasse)



damente ao escolhermos menos de 10 *features*. Os Ganhos são calculados com o *dataset* mais amplo (resultado da filtragem feita pelos métodos independentes do rótulo das amostras). A contribuição individual de cada variável pode não ser tão determinante para a realização da classificação de uma nova amostra quanto a contribuição conjunta de mais de uma variável, resultando em um desempenho pior com menos *features*.

Finalmente, RFE novamente sofre uma acentuada queda de desempenho com menos de 5 variáveis. Isso ocorre por conta da eliminação prematura de variáveis (assim como no caso binário) e também porque RFE incorpora o Ganho do XGBoost no cálculo da importância das *features*, sendo que este último tem seu desempenho prejudicado com menos variáveis.

A Figura 25 apresenta o Desvio-Padrão do *F1-Score* para todas as técnicas de seleção de variáveis, em escala logarítmica. Como há mais classes do que no caso binário, os cortes aleatórios de amostras e colunas realizados pelo XGBoost (CHEN; GUESTRIN, 2016) influenciam mais os métodos de seleção de variáveis que se baseiam nele. O RFE e o Ganho do XGBoost, que possuem esta característica, acabam apresentando um Desvio-Padrão mais elevado (de todas as métricas) ao selecionarem menos de 15 variáveis.

O tempo de treinamento para um classificador multiclasse também influência diretamente o aumento do Tempo de Ajuste do RFE, como pode ser observado na Figura 26. Ao ser utilizado para escolher menos de 5 variáveis, esse método chega a demorar 12 vezes mais tempo para ser executado do que os demais. A complexidade computacional do classificador multiclasse também faz com que o Ganho do XGBoost tenha um tempo de execução mais alto do que todos os métodos de Filtros (ANOVA, IM e ReliefF).

Figura 26: Tempo de Ajuste em função do número de variáveis (Multiclasse)

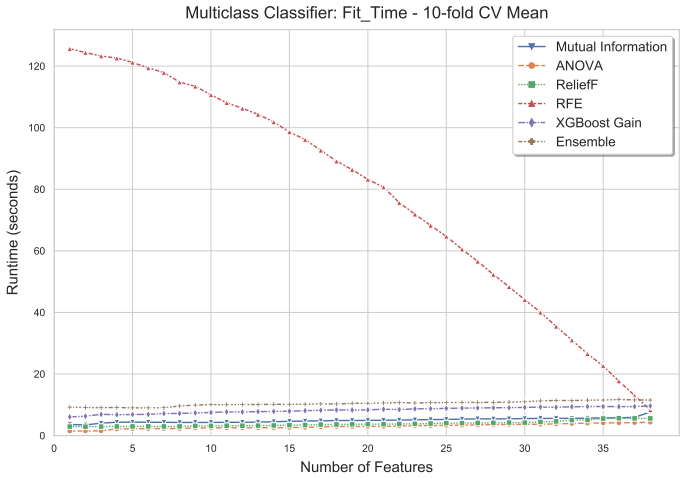
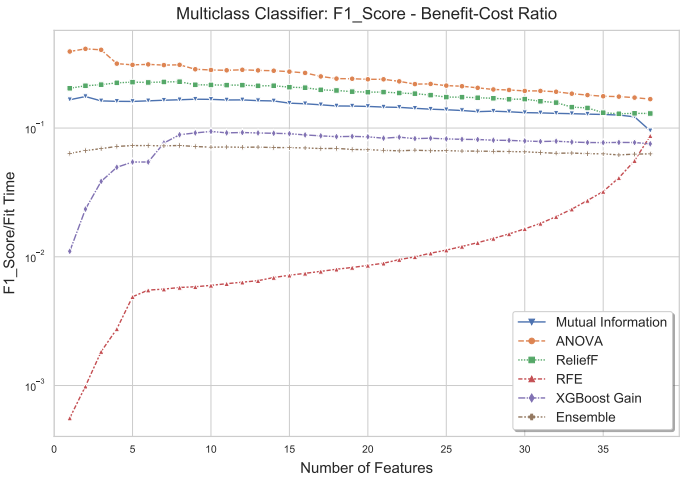


Figura 27: BCR(F1) em função do número de variáveis (Multiclasse)



A diferença nos Tempos de Ajuste entre os classificadores binário e multiclasse influenciam diretamente a curva da BCR. Na Figura 27, vemos esta razão para o *F1-Score* dos classificadores multiclasse, em escala logarítmica. RFE exige um tempo de execução muito maior do que os demais, ficando abaixo de todos os outros métodos (em termos da BCR(F1)) para quase todos os valores de K. O Ganho do XGBoost e, consequentemente, o *Ensemble* que o contém, acabam ficando piores que todos os Filtros segundo esta medida. Os melhores desempenhos são, nessa ordem, de ANOVA, ReliefF e IM, uma vez que **não se utilizam do treinamento de um classificador** para realizar a seleção de variáveis. A ordem relativa entre os três Filtros é a mesma da classificação binária.

Assim como no caso de classificadores multiclasse tradicionais, para classificadores multiclasse OvR e multiclasse OvO também foram construídas curvas de AC, PR, RC e

F1 para os métodos de seleção de variáveis dependentes do rótulo. As métricas, no geral, se aproximaram daquelas obtidas com classificadores multiclasse tradicionais (que tem como função custo a *Categorical Cross-Entropy*). Estas curvas, portanto, serão omitidas, mas disponibilizadas no repositório do projeto.

Para ilustrar o desempenho de classificadores multiclasse OvR e multiclasse OvO, a Tabela 9 apresenta suas métricas e Tempos de Ajuste para todos os métodos de FS dependentes do rótulo, focando a análise para $K = 15$ variáveis selecionadas (quantidade arbitrariamente escolhida). Também foram incluídos, para este recorte, o desempenho de classificadores multiclasse tradicionais, bem como de classificadores binários.

De forma geral, cada método de FS provê resultados de classificação próximos para os três tipos de classificadores multiclasse. Contudo, o Tempo de Ajuste pode diferir significativamente entre eles. Dado o grande número de classes ($S = 12$), as estratégias OvO e OvR se tornam computacionalmente mais custosas que o treinamento multiclasse tradicional, ainda que executem somente treinamentos binários do XGBoost. Em diversos métodos de seleção de variáveis (como ANOVA e IM), foi gasto um Tempo de Ajuste 2 vezes maior para se treinar um classificador multiclasse OvO em relação a um multiclasse tradicional. Resultados análogos foram obtidos para os demais valores de K.

Tabela 9: Desempenho em CV dos métodos de FS dependentes do rótulo

Classificadores Binários						
Métrica	**ANOVA**	**IM**	**ReliefF**	**Ganho XGB**	**Ensemble**	**RFE**
Acurácia	99,73 ± 0,22	97,44 ± 0,69	99,43 ± 0,28	99,77 ± 0,23	99,75 ± 0,24	99,77 ± 0,23
Precisão	99,77 ± 0,36	96,28 ± 0,96	99,03 ± 0,56	99,85 ± 0,36	99,81 ± 0,36	99,85 ± 0,36
Recall	99,7 ± 0,39	98,71 ± 0,6	99,85 ± 0,26	99,7 ± 0,39	99,7 ± 0,39	99,7 ± 0,39
F1-score	99,73 ± 0,22	97,48 ± 0,68	99,43 ± 0,28	99,77 ± 0,23	99,75 ± 0,24	99,77 ± 0,23
fit_time	27,9 ± 0,76	188,44 ± 1,1	158,02 ± 14,82	68,58 ± 1,8	345,27 ± 22,2	1082,57 ± 31,35
Features na Entrada	40	40	40	40	40	40
Features na Saída	15	15	15	15	15	15
Classificadores Multiclasse						
Métrica	**ANOVA**	**IM**	**ReliefF**	**Ganho XGB**	**Ensemble**	**RFE**
Acurácia	73,0 ± 1,7	73,7 ± 1,8	72,3 ± 1,8	73,1 ± 1,7	73,0 ± 1,7	72,5 ± 2,0
Precisão	73,7 ± 2,3	73,9 ± 2,0	73,5 ± 2,4	73,8 ± 2,4	73,7 ± 2,3	73,8 ± 2,2
Recall	73,0 ± 1,7	73,7 ± 1,8	72,3 ± 1,8	73,1 ± 1,7	73,0 ± 1,7	72,5 ± 2,0
F1-score	71,6 ± 1,8	72,7 ± 1,5	70,6 ± 1,9	71,7 ± 1,8	71,5 ± 1,8	70,9 ± 2,4
fit_time	260,9 ± 2,7	465,6 ± 2,9	339,6 ± 2,9	792,9 ± 13,4	1015,4 ± 6,8	9861,3 ± 108,3
Features na Entrada	38	38	38	38	38	38
Features na Saída	15	15	15	15	15	15
Classificadores Multiclasse - One vs Rest						
Métrica	**ANOVA**	**IM**	**ReliefF**	**Ganho XGB**	**Ensemble**	**RFE**
Acurácia	73,3 ± 1,7	73,9 ± 1,8	72,3 ± 1,6	73,2 ± 1,5	73,3 ± 2,0	73,0 ± 1,8
Precisão	74,2 ± 2,2	74,2 ± 2,2	73,3 ± 2,2	74,1 ± 2,1	74,2 ± 2,8	74,4 ± 2,0
Recall	73,3 ± 1,7	73,9 ± 1,8	72,3 ± 1,6	73,2 ± 1,5	73,3 ± 2,0	73,0 ± 1,8
F1-score	71,9 ± 1,8	72,9 ± 1,6	70,5 ± 1,6	71,7 ± 1,6	71,8 ± 2,1	71,3 ± 2,2
fit_time	311,0 ± 3,1	527,4 ± 7,2	416,4 ± 12,0	870,0 ± 12,8	1118,3 ± 8,3	9874,4 ± 106,8
Features na Entrada	38	38	38	38	38	38
Features na Saída	15	15	15	15	15	15
Classificadores Multiclasse - One vs One						
Métrica	**ANOVA**	**IM**	**ReliefF**	**Ganho XGB**	**Ensemble**	**RFE**
Acurácia	72,8 ± 1,8	73,9 ± 1,9	72,1 ± 1,5	73,0 ± 1,7	72,8 ± 1,9	72,6 ± 1,9
Precisão	73,3 ± 2,5	74,1 ± 2,4	73,1 ± 2,1	73,7 ± 2,3	73,3 ± 2,5	73,9 ± 2,4
Recall	72,8 ± 1,8	73,9 ± 1,9	72,1 ± 1,5	73,0 ± 1,7	72,8 ± 1,9	72,6 ± 1,9
F1-score	71,3 ± 1,9	72,9 ± 1,8	70,3 ± 1,6	71,5 ± 1,8	71,3 ± 2,0	70,9 ± 2,3
fit_time	655,0 ± 10,5	829,0 ± 11,2	732,2 ± 4,4	1202,7 ± 21,2	1389,6 ± 11,2	10233,4 ± 185,3
Features na Entrada	38	38	38	38	38	38
Features na Saída	15	15	15	15	15	15

6 CONCLUSÃO

Neste trabalho, o conjunto de dados CICDDoS2019 (SHARAFALDIN et al., 2019) foi usado como base para se verificar o impacto do uso de FS nas métricas de qualidade de classificação de ataques DDoS e o classificador de referência adotado foi o XGBoost. Foram consideradas técnicas de FS que são independentes do rótulo das amostras, como o uso de conhecimento de domínio, remoção de atributos com baixa variância e alta correlação. Além disso, também foram utilizados métodos que faziam o uso da informação contida no rótulo das amostras, como ANOVA, Informação Mútua, ReliefF, Ganho do XGBoost e RFE, além de uma técnica de *Ensemble* envolvendo os *rankings* de variáveis gerados pelos outros métodos.

A retirada de atributos com baixa variância não alterou as métricas de classificação. Pôde-se verificar a robustez do XGBoost, que manteve estáveis seus resultados de desempenho mesmo sem 78% das variáveis originais, para o classificador binário, e 60%, no caso multiclasse. O treinamento de classificadores multiclasse levou consideravelmente mais tempo que o treinamento de classificadores binários (para a mesma quantidade de amostras e atributos). Métodos de seleção de variáveis que incorporaram um classificador em sua execução, como *wrappers* (RFE) e *embeddeds* (Ganho do XGBoost), precisaram de mais tempo para serem executados no caso multiclasse. Devido ao rápido tempo de execução, o ANOVA se mostrou a técnica de FS dependente de rótulo mais vantajosa, tanto para classificadores binários, como para multiclasse.

Entre as principais contribuições realizadas neste trabalho, pode-se destacar:

O uso do Índice de Pielou para formalizar a noção de desbalanceamento de um *dataset*. A forma usual, na indústria e na academia, para se quantificar o grau de desbalanceamento de um conjunto de dados é descrever a frequência de elementos da classe majoritária. Por exemplo, mencionar que X% das amostras pertencem à uma determinada classe. Esta noção funciona bem para *datasets* com rótulos binários, mas é falha em descrever rótulos multiclasse com diferentes distribuições. O Índice de Pielou é aplicável tanto no caso binário, como no multiclasse, sendo uma métrica adequada para as duas

situações.

A condução do problema de classificação de ataques DDoS **tanto do ponto de vista binário como multiclasse** pode ser vista como mais uma contribuição deste trabalho. Como verificado experimentalmente, através das métricas de desempenho, a classificação multiclasse é um problema significativamente mais difícil que a binária. Os trabalhos que serviram de referência para esta dissertação (POLAT; POLAT; CETIN, 2020)(PATTAWARO; POLPRASERT, 2018)(GUPTA, 2018)(DEVAN; KHARE, 2020)(HUSSAIN, 2020)(MARVI; ARFEEN; UDDIN, 2021) atacaram, de uma forma geral, o problema do ponto de vista binário ou multiclasse, mas não ambos. Para as vítimas de ataques DDoS, a informação binária (se há ataque ou não) já é valiosa para mobilizar as equipes de segurança. Ainda assim, a detecção de qual ataque específico está sendo executado, poderia direcionar os esforços de mitigação para uma estratégia em particular, tornando a classificação multiclasse ainda mais importante para a proteção do sistema alvo.

Como ilustrado no Capítulo 2, Referencial Teórico, existem diversas maneiras de se implementar um classificador multiclasse. O uso de estratégias de **meta-aprendizado como One-Vs-One (OvO) e One-Vs-Rest (OvR)** raramente é abordado na literatura de detecção de ataques DDoS. De forma à serem comparados com métodos de treinamento tradicionais, também foram realizados experimentos com as estratégias OvO e OvR. Apesar dos ganhos em termos de métricas de desempenho, o treinamento se mostrou computacionalmente mais custoso. Trata-se, portanto, de um *trade-off* que deve ser levado em conta no momento de se implementar um classificador de ataques DDoS, principalmente se a intenção é realizar retreinamentos periódicos do modelo.

Uma ampla gama de técnicas de seleção de variáveis foi usada neste trabalho, introduzidas a partir de diferentes áreas do conhecimento, como Estatística e Teoria da Informação. Foi criado um ***benchmark* de métricas de classificação** (AC, PR, RC, F1) de ataques DDoS para o algoritmo XGBoost e o *dataset* CICDDoS2019 sujeitos aos métodos de FS listados anteriormente. Este *benchmark*, bem como os programas desenvolvidos para gerá-los e os gráficos que o ilustram estão integralmente disponíveis no repositório do projeto:

⟨https://github.com/pedrohauy/ddos_feature_selection⟩

Este estudo não esgotou o tema de FS na classificação de ataques DDoS, mas trouxe uma perspectiva que pode servir de base para trabalhos futuros. Uma limitação do trabalho foi a utilização de apenas um *dataset* (CICDDoS2019), gerado artificialmente em

um *testbed* por Sharafaldin et al. (SHARAFALDIN et al., 2019). Um conjunto de dados extraído de um ambiente de produção ou que fosse mais balanceado poderia ter gerado resultados diferentes. Todos os experimentos foram executados via CPU, mas a paralelização de diversas operações em GPU poderia trazer diversos ganhos em tempo de execução (DEVELOPERS, 2022). Um ponto interessante seria elencar quais variáveis foram escolhidas em cada rodada de CV, listar quais foram as mais frequentes e verificar se esta escolha dependeu ou não do método de FS usado. Desta forma, poderiam ser filtradas variáveis intrinsecamente importantes para o problema em questão.

A maior parte dos algoritmos de seleção de variáveis considerados estava disponível no Scikit-Learn, mas existem outras técnicas interessantes descritas na literatura como mRMR (PENG; LONG; DING, 2005), BORUTA (KURSA; RUDNICKI, 2010) ou mesmo o uso de valores SHAP (LUNDBERG; LEE, 2017) para ranquear os atributos. Para verificar a capacidade de generalização dos resultados aqui descritos, propõe-se usar a mesma metodologia com outros classificadores de base, como KNN e SVM, ou modelos não-lineares, como Redes Neurais.

Por fim, vale destacar que o desenvolvimento desta dissertação e das simulações computacionais que a acompanham ampliou as discussões realizadas pelo autor e demais membros do grupo de pesquisa na seguinte publicação:

- **Hauy Netto de Araujo, P. H.**, Silva, A., Ferraz Junior, N., Cabrini, F., Santiago, A., Guelfi, A., Kofuji, S. (2021). Impact of Feature Selection Methods on the Classification of DDoS Attacks using XGBoost. *Journal of Communication and Information Systems*, 36(1), 200-214. ⟨https://doi.org/10.14209/jcis.2021.22⟩

REFERÊNCIAS

AAMIR, M.; ZAIDI, S. M. A. Ddos attack detection with feature engineering and machine learning: the framework and performance evaluation. *International Journal of Information Security*, Springer, v. 18, n. 6, p. 761–785, 2019.

ALJAWARNEH, S.; ALDWAIRI, M.; YASSEIN, M. B. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*, v. 25, p. 152–160, 2018. ISSN 1877-7503.

ALKASASSBEH, M. et al. Detecting distributed denial of service attacks using data mining techniques. *International Journal of Advanced Computer Science and Applications*, Citeseer, v. 7, n. 1, 2016.

ARUKONDA, S.; SINHA, S. The innocent perpetrators: reflectors and reflection attacks. *Advanced Computer Science*, v. 4, p. 94–98, 2015.

AWADA, W. et al. A review of the stability of feature selection techniques for bioinformatics data. In: *INTERNATIONAL CONFERENCE ON INFORMATION REUSE & INTEGRATION (IRI), 13. Proceedings...* [S.l.: s.n.], 2012. p. 356–363.

BARZILAY, A. et al. Anubisflow: A feature extractor for distributed denial of service attack classification. In: IEEE. *INTERNATIONAL CONFERENCE ON NETWORK OF THE FUTURE (NoF), 12. Proceeedings...* [S.l.], 2021. p. 1–8.

BELLOVIN, S. M. A look back at security problems in the tcp/ip protocol suite. In: IEEE. *ANNUAL COMPUTER SECURITY APPLICATIONS CONFERENCE, 20. Proceedings...* [S.l.], 2004. p. 229–249.

BINDRA, N.; SOOD, M. Detecting ddos attacks using machine learning techniques and contemporary intrusion detection dataset. *Automatic Control and Computer Sciences*, v. 53, n. 5, p. 419–428, Set 2019. ISSN 1558-108X. doi: ⟨https://doi.org/10.3103/S0146411619050043⟩.

BISHOP, C. M.; NASRABADI, N. M. *Pattern recognition and machine learning*. [S.l.]: Springer, 2006. v. 4.

BOGDANOSKI, M.; SUMINOSKI, T.; RISTESKI, A. Analysis of the syn flood dos attack. *International Journal of Computer Network and Information Security (IJCNIS)*, MECS Publisher, v. 5, n. 8, p. 1–11, 2013.

BUNGART, J. W. *Detecção e bloqueio de ataques DoS e Low-rate DoS em redes de sensores sem fio utilizando redes definidas por software*. Dissertação (Mestrado) — Instituto de Pesquisas Tecnológicas do Estado de São Paulo, São Paulo, SP, Brasil, 2018.

CAI, J. et al. Feature selection in machine learning: A new perspective. *Neurocomputing*, Elsevier, v. 300, p. 70–79, 2018.

CHANDRASHEKAR, G.; SAHIN, F. A survey on feature selection methods. *Computers & Electrical Engineering*, v. 40, n. 1, p. 16–28, 2014. ISSN 0045-7906.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: *SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 22. Proceedings...* New York, NY, USA: Association for Computing Machinery, 2016. p. 785–794. ISBN 9781450342322. Disponível em: ⟨https://doi.org/10.1145/2939672.2939785⟩.

CHEN, Z. et al. Xgboost classifier for ddos attack detection and analysis in sdn-based cloud. In: IEEE. *INTERNATIONAL CONFERENCE ON BIG DATA AND SMART COMPUTING (BIGCOMP). Proceedings...* [S.l.], 2018. p. 251–256.

CHERIF, I. L.; KORTEBI, A. On using extreme gradient boosting (xgboost) machine learning algorithm for home network traffic classification. In: IEEE. *2019 WIRELESS DAYS (WD)*. [S.l.], 2019. p. 1–6.

CHOI, S.-J.; KWAK, J. A study on reduction of ddos amplification attacks in the udp-based cldap protocol. In: IEEE. *INTERNATIONAL CONFERENCE ON COMPUTER APPLICATIONS AND INFORMATION PROCESSING TECHNOLOGY (CAIPT), 4. Proceedings...* [S.l.], 2017. p. 1–4.

CISCO. *Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper - Cisco*. 2018. Disponível em: ⟨https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html⟩. Acesso em: 14.06.2022.

DAS, S. et al. Ddos intrusion detection through machine learning ensemble. In: IEEE. *INTERNATIONAL CONFERENCE ON SOFTWARE QUALITY, RELIABILITY AND SECURITY COMPANION (QRS-C), 19. Proceedings...* [S.l.], 2019. p. 471–477.

DEVAN, P.; KHARE, N. An efficient xgboost–dnn-based classification model for network intrusion detection system. *Neural Computing and Applications*, Springer, v. 32, n. 16, p. 12499–12514, 2020.

DEVELOPERS xgboost. *XGBoost GPU Support*. 2022. Disponível em: ⟨https://xgboost.readthedocs.io/en/latest/gpu/index.html⟩. Acesso em: 30.01.2023.

DEY, S. K.; RAHMAN, M. M.; UDDIN, M. R. Detection of flow based anomaly in openflow controller: Machine learning approach in software defined networking. In: IEEE. *INTERNATIONAL CONFERENCE ON ELECTRICAL ENGINEERING AND INFORMATION & COMMUNICATION TECHNOLOGY (iCEEiCT), 4. Proceedings...* [S.l.], 2018. p. 416–421.

DHALIWAL, S. S.; NAHID, A.-A.; ABBAS, R. Effective intrusion detection system using xgboost. *Information*, Multidisciplinary Digital Publishing Institute, v. 9, n. 7, p. 149, 2018.

DHINGRA, A.; SACHDEVA, M. Ddos detection and discrimination from flash events: a compendious review. In: IEEE. *INTERNATIONAL CONFERENCE ON SECURE CYBER COMPUTING AND COMMUNICATION (ICSCCC). Proceedings...* [S.l.], 2018. p. 518–524.

DUNHAM, K.; MELNICK, J. *Malicious Bots: An Inside Look into the Cyber-Criminal Underground of the Internet*. 1. ed. USA: Auerbach Publications, 2008. ISBN 1420069039.

ELSAYED, M. S. et al. Ddosnet: A deep-learning model for detecting network attacks. In: IEEE. *INTERNATIONAL SYMPOSIUM ON "A WORLD OF WIRELESS, MOBILE AND MULTIMEDIA NETWORKS" (WoWMoM), 21. Proceedings...* [S.l.], 2020. p. 391–396.

GOLCHIN, P. et al. Improving ddos attack detection leveraging a multi-aspect ensemble feature selection. In: IEEE. *NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM (NOMS). Proceedings...* [S.l.], 2022. p. 1–5.

GUPTA, A. *Distributed Denial of Service Attack Detection Using a Machine Learning Approach*. Dissertation (Masters) — University of Calgary, Calgary, AB, Canada, 2018.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. *Journal of machine learning research*, v. 3, n. Mar, p. 1157–1182, 2003.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The elements of statistical learning: data mining, inference and prediction*. 2. ed. Springer, 2009. Disponível em: ⟨http://www-stat.stanford.edu/~tibs/ElemStatLearn/⟩.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. [S.l.]: Prentice Hall, 1999.

HOON, K. S. et al. Critical review of machine learning approaches to apply big data analytics in ddos forensics. In: IEEE. *INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATION AND INFORMATICS (ICCCI). Proceedings...* [S.l.], 2018. p. 1–5.

HOSSEINI, S.; AZIZI, M. The hybrid technique for ddos detection with supervised learning algorithms. *Computer Networks*, Elsevier, v. 158, p. 35–45, 2019.

HUSAIN, A. et al. Development of an efficient network intrusion detection model using extreme gradient boosting (xgboost) on the unsw-nb15 dataset. In: IEEE. *INTERNATIONAL SYMPOSIUM ON SIGNAL PROCESSING AND INFORMATION TECHNOLOGY (ISSPIT). Proceedings...* [S.l.], 2019. p. 1–7.

HUSSAIN, Y. S. *Network Intrusion Detection for Distributed Denial-of-Service (DDoS) Attacks using Machine Learning Classification Techniques*. Dissertation (Masters) — University of Victoria, Victoria, BC, Canada, 2020.

IMTHIYAS, M. et al. Ddos mitigation: A review of content delivery network and its ddos defence techniques. *International Journal on Perceptive and Cognitive Computing*, v. 6, n. 2, p. 67–76, 2020.

JAMES, G. et al. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. Disponível em: ⟨https://faculty.marshall.usc.edu/gareth-james/ISL/⟩.

JUNIOR, N. F. et al. Iot6sec: reliability model for internet of things security focused on anomalous measurements identification with energy analysis. *Wireless Networks*, v. 25, n. 4, p. 1533–1556, Mai 2019. ISSN 1572-8196. doi: ⟨https://doi.org/10.1007/s11276-017-1610-2⟩.

KHAMAISEH, S. et al. Detecting saturation attacks in sdn via machine learning. In: IEEE. *INTERNATIONAL CONFERENCE ON COMPUTING, COMMUNICATIONS AND SECURITY (ICCCS), 4. Proceedings...* [S.l.], 2019. p. 1–8.

KIOURKOULIS, S. *DDoS datasets: Use of machine learning to analyse intrusion detection performance*. Dissertation (Masters) — Luleå University of Technology, Luleå, Suécia, 2020.

KUHN, M.; JOHNSON, K. *Applied predictive modeling*. New York, NY: Springer, 2013. ISBN 9781461468493.

KUHN, M.; JOHNSON, K. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. [S.l.]: CRC Press, 2019. (Chapman & Hall/CRC Data Science Series). ISBN 9781351609463.

KURSA, M. B.; RUDNICKI, W. R. Feature selection with the boruta package. *Journal of statistical software*, v. 36, p. 1–13, 2010.

KWON, D. et al. A survey of deep learning-based network anomaly detection. *Cluster Computing*, Springer, v. 22, n. 1, p. 949–961, 2019.

LI, J. *Detection of DDoS attacks based on dense neural networks, autoencoders and Pearson Correlation Coefficient*. Dissertation (Masters) — Dalhousie University, Halifax, NS, Canada, 2020.

Li, J. et al. Rtvd: A real-time volumetric detection scheme for ddos in the internet of things. *IEEE Access*, v. 8, p. 36191–36201, 2020. ISSN 2169-3536.

LIANG, X.; ZNATI, T. An empirical study of intelligent approaches to ddos detection in large scale networks. In: IEEE. *INTERNATIONAL CONFERENCE ON COMPUTING, NETWORKING AND COMMUNICATIONS (ICNC). Proceedings...* [S.l.], 2019. p. 821–827.

LINDQVIST, N.; PRICE, T. *Evaluation of Feature Selection Methods for Machine Learning Classification of Breast Cancer*. 2018.

LIU, H. et al. Weighted gini index feature selection method for imbalanced data. In: *INTERNATIONAL CONFERENCE ON NETWORKING, SENSING AND CONTROL (ICNSC), 15. Proceedings...* [S.l.: s.n.], 2018. p. 1–6.

LOPEZ, A. D.; MOHAN, A. P.; NAIR, S. Network traffic behavioral analytics for detection of ddos attacks. *SMU data science review*, v. 2, n. 1, p. 14, 2019.

LOWE, C. *Cybersecurity and Reputation — Stronger International Inc. |Cyber Security Training |IT Training*. 2019. Disponível em: ⟨https://stronger.tech/cybersecurity-and-reputation/⟩. Acesso em: 16.04.2021.

LUNDBERG, S. M.; LEE, S.-I. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, v. 30, 2017.

MANJU, N.; HARISH, B.; PRAJWAL, V. Ensemble feature selection and classification of internet traffic using xgboost classifier. *International Journal of Computer Network and Information Security*, Modern Education and Computer Science Press, v. 10, n. 7, p. 37, 2019.

MARVI, M.; ARFEEN, A.; UDDIN, R. A generalized machine learning-based model for the detection of ddos attacks. *International Journal of Network Management*, v. 31, n. 6, p. e2152, 2021. E2152 nem.2152. Disponível em: ⟨https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.2152⟩.

MOHAMMADI, S. et al. Cyber intrusion detection by combined feature selection algorithm. *Journal of information security and applications*, Elsevier, v. 44, p. 80–88, 2019.

MONTGOMERY, D. C.; RUNGER, G. C. *Applied Statistics and Probability for Engineers*. [S.l.]: John Wiley and Sons, 2003.

MURPHY, K. P. *Machine learning: a probabilistic perspective*. [S.l.]: MIT press, 2012.

NETSCOUT. *Global DDoS Summary, May 2022 |NETSCOUT Cyber Threat Horizon*. 2022. Disponível em: ⟨https://horizon.netscout.com/?atlas=summary⟩. Acesso em: 10.06.2022.

NETSCOUT. *Real-Time DDoS Attack Map | NETSCOUT Omnis Threat Horizon*. 2022. Disponível em: ⟨https://horizon.netscout.com/⟩. Acesso em: 10.06.2022.

NEWMAN, L. H. *´DDoS-For-Hire´ Is Fueling a New Wave of Attacks |WIRED*. 2020. Disponível em: ⟨https://www.wired.com/story/ddos-for-hire-fueling-new-wave-attacks⟩. Acesso em: 16.04.2021.

NICHOLSON, P. *Five Most Famous DDoS Attacks and Then Some — A10 Networks*. 2022. Disponível em: ⟨https://www.a10networks.com/blog/5-most-famous-ddos-attacks/⟩. Acesso em: 14.06.2022.

NOORIBAKHSH, M.; MOLLAMOTALEBI, M. A review on statistical approaches for anomaly detection in ddos attacks. *Information Security Journal: A Global Perspective*, Taylor & Francis, v. 29, n. 3, p. 118–133, 2020.

NORVIG, P.; RUSSELL, S. *Inteligência artificial: Tradução da 3a Edição*. [S.l.]: Elsevier Brasil, 2014. ISBN 9788535251418.

NOVAES, M. P. et al. Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. *IEEE Access*, v. 8, p. 83765–83781, 2020.

OLSON, R. S. *ReliefF · PyPI*. 2022. Disponível em: ⟨https://pypi.org/project/ReliefF/⟩. Acesso em: 30.05.2022.

OMAR, K. B. A. Xgboost and lgbm for porto seguro's kaggle challenge: A comparison. *Preprint Semester Project*, 2018.

OPITZ, J.; BURST, S. *Macro F1 and Macro F1*. arXiv, 2019. Disponível em: ⟨https://arxiv.org/abs/1911.03347⟩.

PARFENOV, D. et al. Research application of ensemble machine learning methods to the problem of multiclass classification of ddos attacks identification. In: IEEE. *INTERNATIONAL CONFERENCE ENGINEERING AND TELECOMMUNICATION (EN&T). Proceedings...* [S.l.], 2020. p. 1–7.

PATTAWARO, A.; POLPRASERT, C. Anomaly-based network intrusion detection system through feature selection and hybrid machine learning technique. In: IEEE. *INTERNATIONAL CONFERENCE ON ICT AND KNOWLEDGE ENGINEERING (ICT&KE), 16. Proceedings...* [S.l.], 2018. p. 1–6.

PENG, H.; LONG, F.; DING, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, IEEE, v. 27, n. 8, p. 1226–1238, 2005.

PIELOU, E. The measurement of diversity in different types of biological collections. *Journal of Theoretical Biology*, v. 13, p. 131–144, 1966. ISSN 0022-5193.

PILNENSKIY, N.; SMETANNIKOV, I. Feature selection algorithms as one of the python data analytical tools. *Future Internet*, MDPI, v. 12, n. 3, p. 54, 2020.

POLAT, H.; POLAT, O.; CETIN, A. Detecting ddos attacks in software-defined networks through feature selection methods and machine learning models. *Sustainability*, Multidisciplinary Digital Publishing Institute, v. 12, n. 3, p. 1035, 2020. ISSN 2071-1050.

POST, M. J.; PUTTEN, P. van der; RIJN, J. N. van. Does feature selection improve classification? a large scale experiment in openml. In: BOSTRÖM, H. et al. (Ed.). *Advances in Intelligent Data Analysis XV*. Cham: Springer International Publishing, 2016. p. 158–170. ISBN 978-3-319-46349-0.

REINSTEIN, I. *XGBoost, a Top Machine Learning Method on Kaggle, Explained*. 2017. Disponível em: ⟨https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html⟩. Acesso em: 01.06.2022.

ROSS, B. C. Mutual information between discrete and continuous data sets. *PloS one*, Public Library of Science San Francisco, USA, v. 9, n. 2, p. e87357, 2014.

SCIKIT-LEARN. *scikit-learn: machine learning in Python — scikit-learn 1.1.1 documentation*. 2022. Disponível em: ⟨https://scikit-learn.org/stable/index.html⟩. Acesso em: 30.05.2022.

SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, v. 27, n. 3, p. 379–423, 1948.

SHARAFALDIN, I. et al. Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In: *INTERNATIONAL CARNAHAN CONFERENCE ON SECURITY TECHNOLOGY (ICCST). Proceedings...* [S.l.: s.n.], 2019. p. 1–8.

SIEKLIK, B.; MACFARLANE, R.; BUCHANAN, W. J. Evaluation of tftp ddos amplification attack. *computers & security*, Elsevier, v. 57, p. 67–92, 2016.

SILVA, A. et al. Grouping detection and forecasting security controls using unrestricted cooperative bargains. *Computer Communications*, v. 146, p. 155–173, 2019. ISSN 0140-3664. doi: ⟨https://doi.org/10.1016/j.comcom.2019.07.022⟩.

SILVA, A. et al. Energy-efficient node position identification through payoff matrix and variability analysis. *Telecommunication Systems*, v. 65, n. 3, p. 459–477, Jul 2017. ISSN 1572-9451. doi: ⟨https://doi.org/10.1007/s11235-016-0245-4⟩.

SILVA, A. S. D. et al. Identification and selection of flow features for accurate traffic classification in sdn. In: IEEE. *INTERNATIONAL SYMPOSIUM ON NETWORK COMPUTING AND APPLICATIONS, 14. Proceedings...* [S.l.], 2015. p. 134–141.

SILVA, B. R. et al. A comparative analysis of undersampling techniques for network intrusion detection systems design. *Journal of Communication and Information Systems*, v. 36, n. 1, p. 31–43, 2021.

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.

TANENBAUM, A. S.; WETHERALL, D. *Computer networks, 5th Edition*. [S.l.]: Pearson, 2011. ISBN 0132553171.

TUAN, N. N. et al. A ddos attack mitigation scheme in isp networks using machine learning based on sdn. *Electronics*, MDPI, v. 9, n. 3, p. 413, 2020.

TUAN, T. A. et al. Performance evaluation of botnet ddos attack detection using machine learning. *Evolutionary Intelligence*, Springer, v. 13, n. 2, p. 283–294, 2020.

TUNG, L. *Microsoft: Here's how we stopped the biggest ever DDoS attack — ZDNet*. 2022. Disponível em: ⟨https://www.zdnet.com/article/microsoft-heres-how-we-stopped-the-biggest-ever-ddos-attack/⟩. Acesso em: 16.04.2021.

URBANOWICZ, R. J. et al. Relief-based feature selection: Introduction and review. *Journal of biomedical informatics*, Elsevier, v. 85, p. 189–203, 2018.

WANG, C.; YAO, H.; LIU, Z. An efficient ddos detection based on su-genetic feature selection. *Cluster Computing*, Springer, v. 22, n. 1, p. 2505–2515, 2019.

WANG, J. et al. An ensemble feature selection method for high-dimensional data based on sort aggregation. *Systems Science & Control Engineering*, Taylor & Francis, v. 7, n. 2, p. 32–39, 2019.

WEI, Y. et al. Ae-mlp: A hybrid deep learning approach for ddos detection and classification. *IEEE Access*, IEEE, v. 9, p. 146810–146821, 2021.

XIE, J. et al. A combination of boosting and bagging for kdd cup 2009 - fast scoring on a large database. In: DROR, G. et al. (Ed.). *Proceedings of KDD-Cup 2009 Competition*. New York, New York, USA: PMLR, 2009. v. 7, p. 35–43.

YE, X.; YE, Y. A practical mechanism to counteract dns amplification ddos attacks. *Journal of Computational Information Systems*, v. 9, n. 1, p. 265–272, 2013.

YUSOF, A. R.; UDZIR, N. I.; SELAMAT, A. Systematic literature review and taxonomy for ddos attack detection and prediction. *International Journal of Digital Enterprise Technology*, Inderscience Publishers (IEL), v. 1, n. 3, p. 292–315, 2019.

ZHANG, Z. et al. Empowering one-vs-one decomposition with ensemble learning for multi-class imbalanced data. *Knowledge-Based Systems*, Elsevier, v. 106, p. 251–263, 2016.

APÊNDICE A – AMOSTRAGEM DO DATASET CICDDOS2019

O *dataset* CICDDoS2019 (SHARAFALDIN et al., 2019) possui amostras de 12 tipos de ataque DDoS, além de tráfego benigno, como mostra a Tabela 2. O conteúdo da Tabela 2 é adaptado na Tabela 10 a seguir, mas com valores percentuais em relação ao total de elementos:

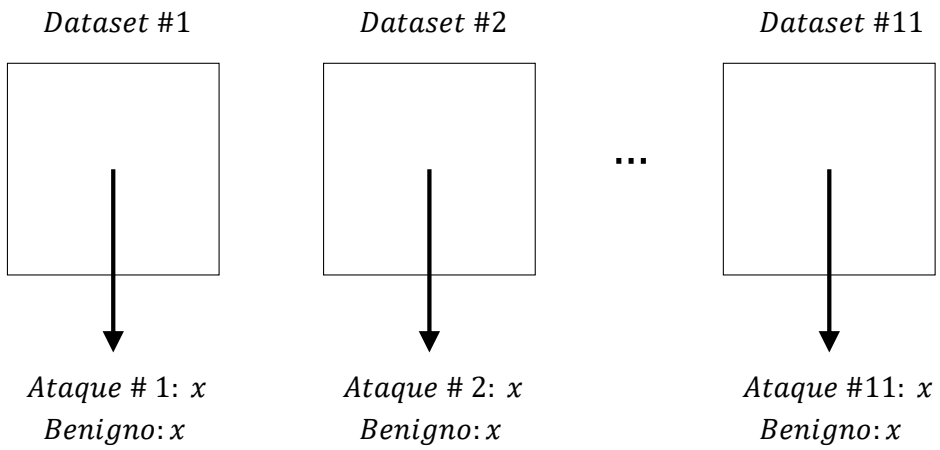
Tabela 10: Distribuição de classes do *dataset* CICDDoS2019 (Percentual)

Binário		Multiclasse	
Classe	Percentual	Classe	Percentual
ATAQUE	99,886%	TFTP	40,115%
		DrDoS_SNMP	10,307%
		DrDoS_DNS	10,129%
		DrDoS_MSSQL	9,034%
		DrDoS_NetBIOS	8,176%
		DrDoS_UDP	6,261%
		DrDoS_SSDP	5,215%
		DrDoS_LDAP	4,354%
		Syn	3,161%
		DrDoS_NTP	2,402%
		UDP-lag	0,732%
		WebDDoS	0,001%
BENIGNO	0,114%	BENIGNO	0,114%

O ataque “WebDDoS” representa apenas 0,001% do total de elementos do conjunto de dados. Como este valor é muito menor do que o percentual da próxima menor classe de ataques (UDP-lag), a classe “WebDDoS” foi **descartada** do conjunto de dados. Os tratamentos posteriores levam em conta, então, 11 tipos de ataques e o tráfego benigno.

As amostras do CICDDoS2019 estão divididas em 11 *datasets* menores (em arquivos CSV). Cada um desses arquivos possui entradas tanto de tráfego malicioso, como tráfego normal. Para evitar viéses na coleta do tráfego benigno, foram retiradas amostras desta classe de cada um dos arquivos do conjunto completo. A seguir, procurou-se criar dois conjuntos de dados balanceados e distintos.

Figura 28: Amostragem para criação de conjunto balanceado do ponto de vista binário



A.1 Balanceamento Binário

O primeiro *dataset* é balanceado do ponto de vista binário, contando com um mesmo número de amostras de ataques (de todos os tipos) e tráfego benigno. Para a criação deste conjunto de dados, foi retirado um mesmo número x de amostras de ataque e tráfego normal de cada *dataset*, como ilustra a Figura 28.

O número total de amostras é dado por:

$$\underbrace{(x + x + \cdots + x)}_{\text{11 ataques}} + \underbrace{(x + x + \cdots + x)}_{\text{11 partes de tráfego benigno}} = TOTAL \tag{A.1}$$

Que, agrupando os termos, pode ser escrito como:

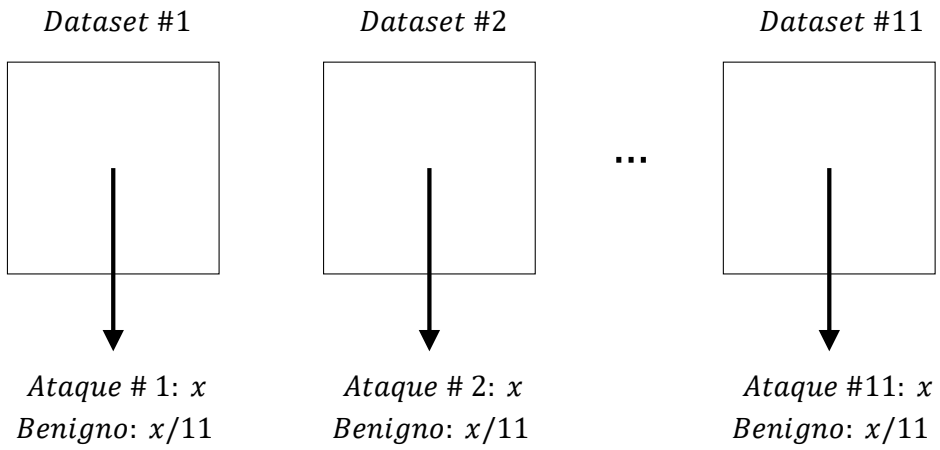
$$11x + 11x = TOTAL \Rightarrow 22x = TOTAL \Rightarrow x = \frac{TOTAL}{22} \tag{A.2}$$

A.2 Balanceamento Multiclasse

O segundo *dataset* é balanceado pela perspectiva multiclasse, onde todas as classes (os 11 ataques e o tráfego normal) tem igual representação. Sendo x o número de amostras de cada ataque, foram coletados $x/11$ exemplares de tráfego benigno de cada arquivo para chegarmos na proporção desejada. A Figura 29 ilustra o processo.

A quantidade total de amostras é obtida por:

Figura 29: Amostragem para criação de conjunto balanceado do ponto de vista multiclasse



$$\underbrace{(x + x + \cdots + x)}_{\text{11 ataques}} + \underbrace{\left(\frac{x}{11} + \frac{x}{11} + \cdots + \frac{x}{11}\right)}_{\text{11 partes de tráfego benigno}} = TOTAL \tag{A.3}$$

Colecionando os termos, vem:

$$11x + 11\left(\frac{x}{11}\right) = TOTAL \Rightarrow 11x + x = 12x = TOTAL \Rightarrow x = \frac{TOTAL}{12} \tag{A.4}$$

A.3 Cardinalidade dos Datasets

Para compararmos a performance das técnicas de seleção de variáveis nos *datasets* binário e multiclasse, é interessante que eles tenham a mesma quantidade de elementos. Pelas equações A.2 e A.4, vemos que o número $TOTAL$ de amostras deve ser múltiplo de 22 e 12 ao mesmo tempo (para que x seja um número inteiro). Logo:

$$TOTAL = 22 \cdot 12 \cdot k \Rightarrow TOTAL = 264 \cdot k, k \in \mathbb{Z} \tag{A.5}$$

APÊNDICE B – INVARIÂNCIA DO ÍNDICE DE PIELOU EM RELAÇÃO À AMOSTRAGEM

O Índice de Pielou é uma forma de mensurar o desbalanceamento de um conjunto de dados (PIELOU, 1966). É definido pela seguinte expressão:

$$J = -\frac{1}{\ln S} \sum_{i=1}^{S} \frac{n_i}{N} \ln \frac{n_i}{N} = \frac{H}{\ln S} \tag{B.1}$$

onde S é o número de classes do conjunto, cada classe C_i possui n_i elementos, o conjunto de dados completo possui N elementos e H é a Entropia de Shannon (SHANNON, 1948).

Note que N pode ser escrito como a soma da quantidade de elementos em cada uma das S classes, ou seja:

$$N = \sum_{i=1}^{S} n_i \tag{B.2}$$

Suponha que seja realizada uma amostragem do conjunto de dados, de modo que cada classe tenha seu número de elementos reduzida por um fator a, sendo este fator igual para todos os rótulos do conjunto. Desta maneira, cada classe C_i teria agora $n'_i = n_i/a$ elementos.

Seja N' o novo total de elementos obtido após este processo. Levando em conta que o número S de classes não foi alterado, temos que o novo total também sofre uma redução por um fator de a:

$$N' = \sum_{i=1}^{S} n'_i = \sum_{i=1}^{S} \frac{n_i}{a} = \frac{1}{a} \sum_{i=1}^{S} n_i = \frac{N}{a} \tag{B.3}$$

Como resultado deste procedimento, o Índice de Pielou J' do novo conjunto amostrado é dado por:

$$\begin{aligned} J' &= -\frac{1}{\ln S} \sum_{i=1}^{S} \frac{n'_i}{N'} \ln \frac{n'_i}{N'} \\ &= -\frac{1}{\ln S} \sum_{i=1}^{S} \frac{n_i/a}{N/a} \ln \frac{n_i/a}{N/a} \\ &= -\frac{1}{\ln S} \sum_{i=1}^{S} \frac{n_i}{N} \ln \frac{n_i}{N} \\ &= J \end{aligned} \tag{B.4}$$

E, portanto, o Índice de Pielou não se altera com uma amostragem pelo mesmo fator de todas as classes simultaneamente.

APÊNDICE C – EXPERIMENTOS COM O DATASET CICDDOS2019

As Tabelas 11 e 12 a seguir listam todos os experimentos realizados para se avaliar o desempenho dos métodos de FS no problema de classificação de ataques DDoS. Para todos os classificadores treinados, o *dataset* de referência foi o CICDDoS2019 (SHARAFALDIN et al., 2019).

Tabela 11: Resumo dos experimentos realizados com *hold-out*

Divisão do Dataset	**Tipo de Classificador**	**Tipo de FS**	**Método de FS**
Hold-Out	Binário	Independente do Rótulo	Pré-processar
			Básicos
			Correlação
		Dependente do Rótulo	ANOVA
			Informação Mútua
			ReliefF
			Ganho do XGBoost
			Ensemble
			RFE
	Multiclasse	Independente do Rótulo	Pré-processar
			Básicos
			Correlação
		Dependente do Rótulo	ANOVA
			Informação Mútua
			ReliefF
			Ganho do XGBoost
			Ensemble
			RFE
	Multiclasse OvR	Independente do Rótulo	Pré-processar
			Básicos
			Correlação
		Dependente do Rótulo	ANOVA
			Informação Mútua
			ReliefF
			Ganho do XGBoost
			Ensemble
			RFE
	Multiclasse OvO	Independente do Rótulo	Pré-processar
			Básicos
			Correlação
		Dependente do Rótulo	ANOVA
			Informação Mútua
			ReliefF
			Ganho do XGBoost
			Ensemble
			RFE

Tabela 12: Resumo dos experimentos realizados com CV

Divisão do Dataset	**Tipo de Classificador**	**Tipo de FS**	**Método de FS**
Cross-Validation	Binário	Independente do Rótulo	Pré-processar
			Básicos
			Correlação
		Dependente do Rótulo	ANOVA
			Informação Mútua
			ReliefF
			Ganho do XGBoost
			Ensemble
			RFE
	Multiclasse	Independente do Rótulo	Pré-processar
			Básicos
			Correlação
		Dependente do Rótulo	ANOVA
			Informação Mútua
			ReliefF
			Ganho do XGBoost
			Ensemble
			RFE
	Multiclasse OvR	Independente do Rótulo	Pré-processar
			Básicos
			Correlação
		Dependente do Rótulo	ANOVA
			Informação Mútua
			ReliefF
			Ganho do XGBoost
			Ensemble
			RFE
	Multiclasse OvO	Independente do Rótulo	Pré-processar
			Básicos
			Correlação
		Dependente do Rótulo	ANOVA
			Informação Mútua
			ReliefF
			Ganho do XGBoost
			Ensemble
			RFE

APÊNDICE D – QUANDO O *RECALL* É IGUAL À ACURÁCIA?

É dito que um *dataset* com N amostras e S classes é balanceado quando o número de elementos de cada uma das classes (n_i) é igual número de elementos dos demais. É, portanto, o resultado da divisão do número de amostras total pelo número de classes do conjunto de dados. Ou seja:

$$n_i = \frac{N}{S}, \forall i \tag{D.1}$$

Para este caso particular, pode-se considerar a Matriz de Confusão Multiclasse da Tabela 13:

Tabela 13: Proporção de amostras por classe

		Rótulo Previsto					
		C_1	$\cdots$	C_i	$\cdots$	C_S	
Rótulo Real	C_1	VP_1					$\rightarrow$ N/S
	$\vdots$						
	C_i			VP_i			$\rightarrow$ N/S
	$\vdots$						
	C_S					VP_S	$\rightarrow$ N/S

Na tabela anterior, o número de elementos de cada linha é **constante** e igual a N/S. Do ponto de vista da classificação multiclasse, este número é igual a soma dos Verdadeiros Positivos da classe considerada (célula verde de cada linha) com os Falsos Negativos desta classe (demais células vermelhas de cada linha). Esta propriedade de um *dataset* balanceado pode ser expressa por:

$$VP_i + FN_i = n_i = N/S, \forall i \tag{D.2}$$

Desta forma, ao se calcular a Macro Média do *Recall*, temos:

$$
\begin{aligned}
RC_M &= \frac{1}{S}\sum_{i=1}^{S}\frac{VP_i}{VP_i + FN_i} \\
&= \frac{1}{S}\sum_{i=1}^{S}\frac{VP_i}{N/S} \\
&= \frac{1}{S}\frac{1}{(N/S)}\sum_{i=1}^{S} VP_i \\
&= \frac{1}{N}\sum_{i=1}^{S} VP_i \\
&= AC
\end{aligned}
\tag{D.3}
$$

E, portanto, toda vez que é calculada a Macro Média do *Recall* para um conjunto de dados **balanceado**, esta é sempre igual à Acurácia.