Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Detecção de DDoS por aprendizado de máquina

Matheus Siade Ferreira

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Daniel Café

Brasília
2021

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Detecção de DDoS por aprendizado de máquina

Matheus Siade Ferreira

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Daniel Café (Orientador)
ENE/UnB

Prof. Dr. Alexandre Romariz
ENE/UnB

Prof. Dr. João José Costa Gondim
CiC/UnB

Prof. Dr. João José Costa Gondim
Coordenador do Curso de Engenharia da Computação

Brasília, 26 de maio de 2021

Dedicatória

Dedico esse trabalho à todos que caminharam comigo nesse jornada.

Agradecimentos

Seria impossível entregar esse trabalho sem agradecer às pessoas que participaram da história que me trouxe até aqui.

Aos meus pais, que sempre me apoiaram em minhas escolhas e fizeram o possível para que alcançasse meus objetivos, à minha madrinha por sempre se preocupar com meu bem estar e ao meu irmão por me aturar em todos os momentos.

Aos meus amigos, Clara, Eduardo e João, que desde o começo da faculdade estiveram comigo, nos melhores e piores momentos, e com quem sempre pude contar.

Aos meus professores, que me guiaram por esse percurso e me ensinaram não só conteúdos acadêmicos, mas também a buscar ir além do que achava ser capaz. Em especial ao professor, e orientador deste projeto, Daniel Café que por sua competência e experiência me conduziu da melhor forma possível, se mostrando sempre disponível a ajudar com o que fosse necessário.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Ataques DDoS são uma ameaça disruptiva ao funcionamento de diversos sistemas online em um mundo progressivamente mais dependente da internet em seu cotidiano. O aumento recente na intensidade e complexidade destes ataques torna o desenvolvimento de melhores práticas de prevenção e mitigação necessário para a manutenção da estabilidade do fornecimento de serviços online. Este trabalho tem como objetivo a análise de métodos de aprendizado de máquina na detecção de fluxos de rede anômalos, utilizando algoritmos como K-nn, SVM, florestas randômicas, árvores de decisão e redes neurais para a classificação de tráfego com base em dois conjuntos de dados distintos, o NSL_KDD e o CiCDDoS 2019. Para a criação dos modelos foram selecionados alguns atributos de cada dataset, com a performance sendo avaliada com base em subdivisões dos dados em conjuntos de treino e teste. Os resultados obtidos chegaram a uma acurácia de até 99.9% na separação dos dados anômalos contidos nos conjuntos, com algoritmos como Florestas Randômicas e Árvores de decisão alcançando os melhores valores. Para trabalhos futuros, um classificador com base em Machine Learning pode ser combinado a um analisador de tráfego em tempo real para a separação de fluxos anômalos.

Palavras-chave: DDoS, DoS, Machine Learning, Deep Learning, NSL_KDD, CiCDDoS2019

Abstract

DDoS attacks are a disruptive threat to the availability of online services in a world ever more dependent on the internet for performing daily tasks. The recent increase in complexity and frequency of these attacks points to the need of better and more reliable prevention and mitigation techniques as to maintain the stability of these services. This work aims to analyze the performance of machine learning techniques for anomaly detection in DDoS attacks by testing the classification performance of algorithms as K-nn, SVM, Random Forests, Decision Trees and Neural Networks in two separate datasets: NSL_KDD and CiCDDoS 2019. The process of training the ML models consisted of pre-processing, splitting the data in training and test sets, feature selection, training the models, and evaluating their performance as to the tests sets. Results of up to 99.9% accuracy were obtained in the detection of anomalous instances in the datasets in which Random Forests and Decision Trees were the highest rated classifiers. For future works, a machine learning based classifier could be implemented alongside a real-time traffic analyzer for DDoS detection.

Keywords: DDoS, DoS, Machine Learning, Deep Learning, NSL_KDD, CiCDDoS2019

Sumário

1 Introdução **1**

2 Revisão Teórica **4**

2.1 Ataques DDoS 4

2.1.1 Motivações 4

2.1.2 Estrutura 5

2.1.3 Classificação 6

2.1.4 Estratégias de Defesa 9

2.2 Aprendizado de máquina 11

2.2.1 Algoritmos 12

2.2.2 Conjuntos de treino, teste e validação 15

2.2.3 Análise de performance 16

2.3 Conjuntos de dados 17

2.3.1 NSLKDD 17

2.3.2 CiCDDoS2019 18

2.4 Ferramentas 19

2.4.1 TensorFlow 19

2.4.2 Weka 20

2.4.3 Scikit Learn [1] 20

3 Metodologia **21**

3.1 Processamento de dados 21

3.1.1 NSL_KDD 21

3.1.2 CiCDDoS 2019 22

3.2 Otimização de Parâmetros de Algoritmos 24

3.2.1 K-nn (iBK no Weka) 24

3.2.2 SVM (SMO no weka | svm pelo scikit-learn) 25

3.2.3 MLPs 27

3.2.4 C4.5 (J48 no weka) 30

3.2.5 Random Forests 31

4 Resultados **32**

4.1 NSL_KDD 32

4.1.1 Conjunto completo: 32

4.1.2 Subconjunto com atributos reduzidos 33

4.2 CiCDDoS2019 34

5 Conclusão **36**

Referências **37**

Anexo **40**

I Tabelas relacionadas ao NSL_KDD **41**

Lista de Figuras

2.1 Classificação de ataques de acordo com a largura de banda proposta por Gondin et al. 8
2.2 Ataques presentes no CiCDDoS2019. 19

3.1 Acurácia do algoritmo em relação ao número de vizinhos no NSL_KDD. . . 24
3.2 Acurácia do algoritmo em relação ao número de vizinhos no CICDDoS. . . . 25
3.3 Teste de escolha de hiperparâmetros para o NSL_KDD com 3 camadas densas e uma camada de "dropout". 28
3.4 Teste de escolha de hiperparâmetros para o NSL_KDD com 4 camadas densas e uma camada de "dropout". 28
3.5 Teste de uso de regularização de kernel no modelo. 29
3.6 Comparação de performance entre os otimizadores. 29
3.7 Topologia do modelo de rede neural para o NSL_KDD. 30
3.8 Topologia do modelo de rede neural para o CiCDDoS. 30

Lista de Tabelas

2.1 Matriz de confusão. 16
2.2 Proporção de classes dos dados presentes no NSL_KDD. 18

3.1 Features do NSL_KDD escolhidas na distribuição proposta 22
3.2 Distribuição dos dados no subconjunto inicial do CiCDDoS. 22
3.3 Composição do subconjunto do CiCDDoS em relação às classes de dados. . . 23
3.4 Features escolhidas do CiCDDoS 2019. 23
3.5 Tempo de execução do K-nn no NSL_KDD com diferentes algoritmos de busca de vizinhos. 25
3.6 Exemplos de varição de parâmetros na execução da SVM no NSL_KDD. . . 26
3.7 Exemplos de varição de parâmetros na execução da SVM no CiCDDoS . . . 27
3.8 Variação de acurácia de acordo com hiperparâmetros do J48 31

4.1 Resultados dos testes no NSL_KDD com todos os atributos. 32
4.2 Novo conjuto formado a partir do NSL_KDD. 33
4.3 Resultados dos testes no NSL_KDD com atributos reduzidos de acordo com o subconjunto proposto por DAS, Saikat et al [2]. 33
4.4 Resultados dos testes no NSL_KDD com divisão 70:30 para treino e teste . 34
4.5 Resultados do modelo criado a partir do CiCDDoS 34

I.1 Ataques DoS presentes no NSL_KDD. 41
I.2 Ataques probe presentes no NSL_KDD. 42
I.3 Ataques U2R presentes no NSL_KDD. 42
I.4 Ataques R2L presentes no NSL_KDD. 42
I.5 Features do NSL_KDD. 43

Capítulo 1

Introdução

É inegável o impacto que a pandemia do Covid-19 teve no uso de serviços online. A necessidade de mudança para um modelo work-from-home fez com que o uso de tecnologias de trabalho colaborativo, especialmente videoconferências, tivessem uma maior adoção. Ao mesmo tempo, o fechamento de estabelecimentos voltados para o entretenimento aumentou a busca por videogames e serviços de streaming de vídeo, contribuindo para o crescimento da dependência tecnológica. É nesse cenário, no qual o uso da internet se torna imprescindível ao cotidiano, que ataques que visam inutilizar serviços se tornam cada vez mais frequente.

Ataques de DoS (denial-of-service) são um tipo de ataque cibernético cujo principal objetivo é dificultar ou impossibilitar o acesso aos serviços atacados, sendo esses sites ou sistemas online. DDoS por sua vez, se refere a ataques de DoS de natureza distribuída, ou seja, que usam várias máquinas em um ataque coordenado. Em 2020 estes ataques cresceram tanto em intensidade como em complexidade, com empresas como NetStar registrando um aumento de 151% no número de ataques em comparação primeira metade do ano em relação ao período anterior.[3] Em comparação com o maior ataque já registrado anteriormente, 1.35Tbps no ataque ao Github em 2018, foram registrados ataques consideravelmente maiores, com Amazon e Google registrando ataques de 2.3Tbps e 2.5Tbps respectivamente. Na reportagem fornecida por A10Networks, as maiores causas identificadas para este aumento foi o fator de amplificação de ataques SSDP e a captura de dispositivos IoT para botnets. [4]

Ataques SSDP (Simple Service Discovery Protocol) exploram uma vulnerabilidade no protocolo de comunicação direta UPnP (Universal Plug and Play), usado para simplificar redes locais, de alguns roteadores para enviar uma quantidade de tráfego amplificada para dispositivos. O potencial de amplificação destes, somado ao número de dispositivos que fazem uso de UPnP, fez com que este tipo de ataque subisse para o topo da lista de frequência. Estes ataques conseguem usar o protocolo para amplificar requisições em até

30 vezes possibilitando ataques em grande escala.

Simultaneamente, a alta taxa de vulnerabilidades presentes em dispositivos IoT faz com que estes sejam alvos fácies para serem usados para DDoS. O bot Mirai responsável por alguns dos maiores ataques dos últimos anos se aproveitou da alta disponibilidade de dispositivos IoT vulneráveis para organizar ataques em massa, alcançando taxas que em 2016 já passavam de 600Gbps, chegando a infectar até 600,000 aparelhos. Em 2020 foi percebida uma queda no uso de Botnets Mirai em relação à ataques Mozi, mais comuns em territórios como a Índia.[4]

Além dos ataques supracitados existe uma enorme variedade de ataques DDoS, caracterizados geralmente em três tipos principais:[5]

- Ataques baseados em volume
- Ataques de protocolo
- Ataques da camada de aplicação

O primeiro tipo tem como objetivo saturar a largura de banda da vítima, sendo normalmente o mais comum dos 3, com ataques botnet e reflexivos fazendo parte deste grupo. O volume de informação gerado pelo ataque tem como objetivo impedir que requisições legitimas consigam chegar até o destino. Ataques de protocolo, por sua vez, tentam impedir o funcionamento através de requisições de protocolos de comunicação. Um exemplo são os ataques UDP flood, que envia requisições UDP para portas aleatórias de um servidor. Como no protocolo UDP estes pacotes devem ser respondidos, a vítima fica ocupada respondendo às requisições recebidas, atrasando seu funcionamento. O último tipo de ataque busca exaurir algum recurso da camada de aplicação, como CPU ou memória. Para isso, são feitas requisições à alguma aplicação de tal forma que o servidor ultrapasse suas capacidades.

A variabilidade de tipos de ataques, somada a imprevisibilidade de suas ocorrências, torna a detecção e a mitigação de ataques um desafio. São usados, na tentativa de lidar com estas ocorrências diversos métodos, que vão de mecanismos de prevenção à estratégias de mitigação.

Como ataques DDoS atuam diretamente nos recursos de suas vítimas, tornando a mitigação progressivamente mais difícil após o início do ataque, a prevenção acaba se tornando um mecanismo de defesa essencial no tratamento destes. Esta pode ser feita através da filtragem de pacotes de entrada/saída em uma rede segura usando listas de endereços IP suspeitos para validação, através do uso de métodos mais avançados para determinação de rotas suspeitas, ou até mesmo o histórico de tráfego normal para determinar tráfego anômalo. Métodos como o uso de honeypots, máquinas de baixa segurança usados como isca de ataques, e balanceamento de carga, distribuir a carga do ataque entre

diferentes máquinas de forma a evitar sobrecarga, também são frequentemente usados. No entanto, no caso de ocorrência de ataques a mitigação é essencial para a redução de danos à vítima.

A mitigação de ataques DDoS consiste em 3 mecanismos principais: detecção, resposta e tolerância.[6] Perceber a ocorrência de um ataque é um processo relativamente fácil, uma vez que ataques tendem a rapidamente degradar o funcionamento de um sistema, com a maior dificuldade residindo na diferenciação do tráfego atacante do normal. Uma forma de detecção de fluxo maligno é a detecção baseada em assinaturas, a qual consiste no uso de assinaturas de ataques conhecidos para diferenciar o tráfego atacante. Outra forma é a detecção baseada em anomalias, que tem como maior vantagem a capacidade de detectar ataques com novas assinaturas. Tem como objetivo principal detectar padrões que fujam de um comportamento "normal"pré estabelecido, podendo ser realizado de diversas formas, as quais vão desde métodos estatísticos até o uso de aprendizado de máquina.Uma dificuldade comum a esses métodos é a definição de parâmetros de limite, uma vez que pode causar tanto uma baixa detecção de ataques quanto gerar falsos positivos, classificando fluxos normais como ataques. Um exemplo seria o uso de MULTOPS, uma técnica para detecção de ataques de volume com base na proporção de tráfego de entrada/saída, que em decorrência da desproporcionalidade gerada pelo aumento de fluxos de mensagens multimídia, acaba tendo um alto número de falsos positivos. Várias técnicas de aprendizado de máquina podem ser empregadas para a identificação de fluxos anômalos, variando tanto nos algoritmos utilizados quanto na forma de análise do fluxo de dados e tendo índices altos de acurácia, que vão de 85% a 99%. [7, 8]

Mecanismos de reposta, por outro lado, tendem a focar na redução do impacto do ataque utilizando técnicas como filtragem de pacotes ou limitação de banda. Por último, mecanismos de tolerância tentam, na eventual falha das medidas de prevenção e detecção, manter um nível base de funcionamento do serviço durante o ataque.

Este trabalho tem como objetivo a análise de performance de técnicas de aprendizado de máquina para a detecção de ataques DDoS, usando o treinamento de redes para a diferenciação de trafego normal de tráfego anômalo. Para isso serão feitos testes com diversos algoritmos em cima de dois conjuntos de dados com foco em ataques de denial of service, o NSL-KDD e o CiCDDoS2019.

Capítulo 2

Revisão Teórica

2.1 Ataques DDoS

De acordo com o apresentado na introdução, ataques DDoS são um tipo de ataque cibernético focado na inviabilização de uso de serviços online. Tem natureza distribuída, fazendo uso de diversas máquinas simultaneamente na tentativa de impossibilitar o funcionamento de suas vítimas. Os ataques tem enorme variabilidade tanto de motivação quanto de abordagem, tornando a prevenção e mitigação dos mesmos um desafio.

2.1.1 Motivações

As motivações para ataques DDoS normalmente se enquadram em 5 categorias principais:[9, 10]

- Ganho financeiro/econômico:

  Em sua maioria feitos por atacantes mais experientes tem como objetivo recompensas financeiras. São ataques mais técnicos e agressivos no geral, se destacando ataques a sites de comércio durante períodos de alta procura e ataques com objetivo de extorsão/chantagem em busca de pagamentos.

- Vingança:

  São ataques realizados por pessoas frustradas, que ao se ver como vítima uma injustiça vindo de uma corporação ou organização move um ataque contra esta. Estes ataques geralmente provem de pessoas com menor habilidade tecnológica.

- Crença ideológica:

  Também conhecido como hacktivismo, consiste de pessoas motivadas por suas crenças em atacar seus alvos. As vítimas normalmente são governos e os ataques consistem na derrubada de sites institucionais. Em 2020 o grupo anonymous retomou

suas atividades como forma de protesto em meio à turbulência política causada pelo homicídio de George Floyd.[11]

- Desafio intelectual

  Geralmente se trata de ataques realizados por pessoas mais jovens que encaram o sucesso de seus ataques como forma de validar suas habilidades como hackers. É facilitado pela disponibilidade ferramentas de uso simples e botnets para aluguel ao ponto que amadores possam causar ataques de sucesso.

- Guerra Virtual

  Usado por Estados com o objetivo de sabotar serviços estatais ou privados de outros países. É caracterizado por grupos com grande disponibilidade de recursos e alto conhecimento técnico. Podem causar grandes danos econômicos e paralisia de serviços essenciais dos países alvos.

2.1.2 Estrutura

Apesar da grande variabilidade de mecanismos e alvos de ataques DDoS é possível identificar uma estrutura básica na organização dos ataques. Estes são compostos por 4 entidades e ocorrem em 3 fases distintas.[6]

Os ataques podem ser estruturados em 4 elementos principais[12, 13]:

- Atacante
- Handlers
- Agentes
- Vítima

Neste modelo organizacional o atacante é a mente por trás do ataque, o organizador da estrutura. Os handlers são os controladores, coordenando os agentes e propagando os comandos do atacante para estes. Os agentes, também conhecidos como zumbis (Zombies no jargão em inglês) ou vítimas secundárias, são máquinas capturadas pelo atacante para a execução do ataque. Por último, a vítima é o alvo escolhido pelo atacante para receber o ataque. A propagação do ataque ocorre em 3 fases distintas, que correspondem a um processo de capturar, preparar e atacar.[6] Na primeira fase o atacante concentra seus recursos na captura de máquinas para seu exército de agentes, instalando malware nas vítimas e as usando para propagar código malicioso na captura de agentes. Quando o atacante consegue se munir de um número grande suficiente de agentes para seu ataque esse grupo passa a ser denominado uma Botnet. A segunda fase compete na propagação

de códigos de ataque para a botnet. Nessa fase são enviados aos agentes dados da vítima como endereço, além da hora de início do ataque e a duração do mesmo. Na última fase o ataque é proferido, com o atacante utilizando de diversos métodos para sua execução

2.1.3 Classificação

Na literatura há diversas formas de classificação de ataques DDoS, com critérios para a classificação variando de acordo com o autor. Atualmente empresas como Cloudfare[14] e A10Networks[5] dividem os ataques nas 3 categorias citadas anteriormente:

- Ataques Volumétricos

  Tem como objetivo congestionar a rede consumindo toda a largura de banda disponível entre o alvo e a internet ou até mesmo na rede interna do serviço [15]. São geralmente usados contra Provedores de Serviços ou clientes empresariais, podendo ser combinados com ataques da camada de aplicação na tentativa de aumentar o dano causado. Se enquadram nessa classificação ataques como UDP Flood, ICMP Floop e ataques de amplificação.

- Ataques de Protocolo

  Causam a exaustão de recursos da rede como firewalls e balanceadores de carga, usando vulnerabilidades das camadas de rede e aplicação para tal objetivo.[14, 16] São ataques de volume médio ou baixo, uma vez que é limitado pelo funcionamento dos protocolos abusados. Um dos principais ataques do tipo, DNS NXDOMAIN, envolve preencher o servidor DNS de uma rede com um grande volume de requisições com registros inválidos ou inexistentes, fazendo com que o server DNS se ocupe com requisições inválidas, deixando de servir as requisições legítimas. São outros ataques desta categoria: SYN Flood e SSL

- Ataques da camada de aplicação

  Tem como objetivo exaurir os recursos do alvo, atacando a aplicação em si através de vulnerabilidade específicas, impedindo o funcionamento da aplicação. Geralmente são usadas no ataque direto à serviços Web, usando chamadas HTTP mal intencionadas para sobrecarregar o servidor. O mecanismo do ataque pode incluir o envio requisições excessivas e ocupar a CPU de um servidor com o retornos de chamadas HTTP em HTTP flood ou o envio de requisições parciais na tentativa de manter estas em aberto o maior tempo possível e ocupar todas as conexões disponíveis do servidor em ataques SlowLoris. Dentre os diversos ataques presentes nessa categoria estão: HTTP Flood, Slowloris, Low and Slow e Slow Post

MAHJABIN, Tasnuva et al. [6], na tentativa de confeccionar uma nova classificação que contemplasse de forma simplificada as diversas classificações encontradas, sugeriu um modelo de classificação de ataques composto por 4 categorias principais definidas pelo impacto do ataque nos recursos da vítima. Esta é composta por:

- Ataques de exaustão de recursos

  Se enquadram aqui ataques que tem como objetivo exaurir diretamente os recursos da máquina como CPU, memória e sockets. Podem ser feitos tanto através de exploits das camadas de rede quanto pelo uso de pacotes malformados. Os ataques de protocolo dessa categoria usam de vulnerabilidades na implementação de protocolos de rede para elevar o uso de memória e cpu do sistema, incluindo ataques da camada de transporte como TCP Syn e ataques da camada de aplicação, como HTTP flood.

  TCP Syn se caracteriza por abusar do modelo de handshake em 3 etapas do protocolo TCP, enviando solicitações de conexões a um servidor e nunca terminando o processo de acknowledgement. Como o servidor mantém estas conexões em aberto à espera do último ACK até a ocorrência de um timeout, acaba lotando suas tabelas de conexões .

- Ataques de exaustão de banda

  São equivalentes aos ataques volumétricos da classificação anterior, enquadrando ataques que tem como objetivo a congestão da rede da vítima. O autor subdivide a categoria em duas subclasses de ataque: Ataques de protocolo, composta por ataques que usam protocolos da camada de transporte, como UDP e ICMP, e ataques de amplificação, os quais consistem em gerar uma grande reposta a partir de um pequeno número de requisições e redirecioná-la às vitimas.

  São ataques desta categoria: UDP Flood, ICMP Flood, DNS amplification attack, NTP amplification attack

- Ataques de infraestrutura

  Inclui ataques que tem como alvo a infraestrutura da rede de Internet em si. São ataques que almejam exaurir tanto a largura de banda quanto os recursos da vítima. Um exemplo são ataques diretos às zonas raiz de DNS, como o ataque de 2016 ao provedor de DNS Dyn que causou falhas de funcionamento generalizadas na América do Norte e em partes da Europa. Os atacantes normalmente fazem uso de técnicas de DNS Flood para a execução destes, usando grandes redes botnets para gerar requisições UDP ao servidor DNS até consumir seus recursos.

- Ataques Zero-day

São realizados no dia 0 utilizando de exploits e falhas desconhecidas. Enquadram ataques DDoS que usam vulnerabilidades e métodos desconhecidos até o momento do ataque. Podem se aproveitar de servidores e fontes que ainda não foram reconhecidas como suspeitas pela maioria dos provedores de serviços.

GONDIN, João et al. [17] por sua vez propõe uma classificação em duas categorias principais separadas pelo volume de dados utilizado pelo ataque, conforme pode ser visto na Figura 2.1.

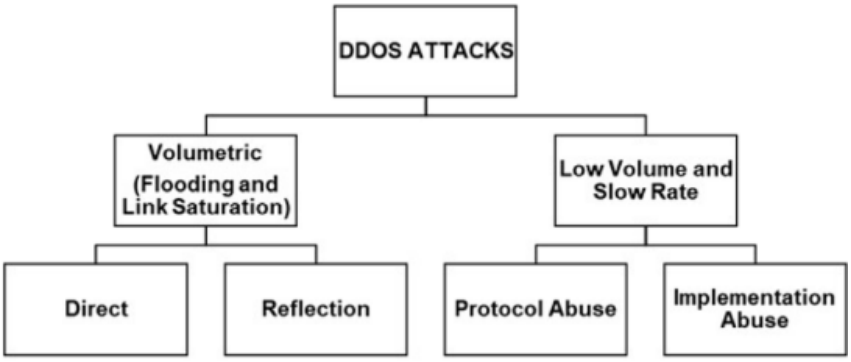

Figura 2.1: Classificação de ataques de acordo com a largura de banda proposta por Gondin et al. .

A divisão principal ocorre entre ataques lentos e de baixo volume (low and slow) e ataques volumétricos (Flooding and link saturation), podendo ser caracterizadas da seguinte forma:

- Ataques lentos e de baixo volume

  São enquadrados ataques que usem de tráfego legítimo com o objetivo de abusar de características de implementação de protocolos e sistemas com o intuito de exaurir recursos da vítima e consequentemente impedí-la de responder a requisições. Pode ser interpretada como a aglutinação dos Ataques de protocolo e Ataques da camada de aplicação da classificação sugerida pelo Cloudfare.

- Ataques Volumétricos

  Assim como nas classificações anteriores, engloba ataques que buscam exaurir a largura de banda da vítima, impedindo-a de responder a ataques. Aqui, o autor subdivide a categoria em duas subclasses: Ataques Diretos e Ataques Refletores. A primeira trata de ataques nos quais os nós afetados enviam pacotes à vitima diretamente, enquanto a segunda trata de ataques nós refletores são usados para esse fim. Ataques refletores se destacam por seu fator de amplificação, tendo como característica primordial a geração, pelos nós intermediários, de mais dados do que

foram enviados a eles. Nesses ataques é comum que o responsável envie aos refletores pacotes com o endereço da vítima como fonte, fazendo com que os pacotes que cheguem a ela não contenham o endereço original do atacante, dificultando o tratamento do ataque por meio de filtros tradicionais.

2.1.4 Estratégias de Defesa

As estratégias de defesa contra DDoS podem ser caracterizadas em duas áreas distintas: mecanismos de prevenção e mecanismos de mitigação. A primeira tem como foco tentar impedir que um ataque aconteça enquanto a segunda foca em lidar com o mesmo após seu início.

Prevenção

A partir do momento em que um ataque DDoS é iniciado com sucesso a vítima começa a perder recursos progressivamente, piorando a qualidade do serviço e até o impossibilitando. É por isso que no tratamento de ataques de ataques DDoS a prevenção é essencial para reduzir eventuais danos que poderiam ser causados pelo atacante, sendo composta por cinco categorias principais de mecanismos de ação:

- Filtragem

  Consiste no uso de técnicas de filtragem para impedir que pacotes maliciosos cheguem até a vítima. Estas técnicas tem como objetivo impedir uma vítima de sofrer um ataque assim como se tornar um atacante involuntário. Pode ser feito com base na filtragem de entrada/saída de pacotes, na qual regras são definidas para garantir que pacotes que cheguem não possuam endereços IP de origem falsos e que pacotes que saiam obedeçam a regras definidas para evitar que protocolos ou portas de destino sejam abusados ou sejam usados de forma desnecessária [18]. Outros métodos usados são: Hop-count filtering, Route-based packet filtering, path identifier.

- Overlay seguro

  Tem como objetivo permitir somente a comunicação entre um usuário confirmado e um alvo. O objetivo é a construção de uma rede overlay no topo de um provedor IP, tendo esta como porta de entrada para os pacotes que chegam à rede[6]. Os pacotes são validados nos pontos de entrada do overlay e uma vez dentro é guiado de forma segura até o destino.[19]

- Honeypots

  Honeypots são sistemas menos seguros que tentam enganar um atacante de que são o alvo do ataque, imitando o comportamento deste na tentativa de atrair o

atacante[6]. Além de ser uma forma de redirecionar o ataque também pode ser usado para entender as ferramentas, táticas e motivos do atacante, com as informações aprendidas sendo usadas no resto da rede como mecanismos de defesa[20].

- Balanceadores de carga

  Tem como objetivo balancear o tráfego de uma rede entre múltiplos servidores de forma a impedir que um deles fique sobrecarregado. No caso de ataques DDoS, consegue minimizar os danos ao fazer re-roteamento de tráfego para servidores não afetados[6, 21].

- Prevenção baseada em conhecimento [6]

  Medidas tomadas por usuários comuns que podem diminuir a ocorrência de ataques. Variam de manter dispositivos IoT atualizados a desabilitar serviços suspeitos em suas máquinas. Tem como objetivo reduzir o número de agentes disponíveis para um ataque.

Mitigação

Apesar do uso de métodos de prevenção, nem sempre é certo o sucesso dos mesmo, tornando a mitigação dos ataques imprescindível no controle de danos ocasionados. Um processo essencial à mitigação é a separação do fluxo de dados anômalos dos dados normais, com estes métodos se dividindo em duas categorias principais:

- Detecção baseada em assinaturas

  Usa um set de regras e assinaturas de padrão de ataques já identificadas guardadas em um banco de dados. Os padrões de tráfego são monitorados e comparados com assinaturas existentes para detecção de tráfego malicioso. Tem como maior desvantagem a inabilidade de detectar ataques que ainda não tenham suas assinaturas registradas, o que pode levar a um alto número de falsos negativos [7].

  Uma ferramenta open source normalmente usada baseada em assinaturas é SNORT, um sistema de prevenção de intrusão [6]. Pode ser usado para monitorar o tráfego de entrada e saída com o objetivo de encontrar os endereços IP usados no ataque e bloquear pacotes originados destes [7]. Combina detecção de assinaturas com regras de comportamento para detecção de ataques 0-day [22].

- Detecção baseada em anomalia

  Em contraposição à detecção baseada em assinaturas, se caracteriza pela definição de um perfil de tráfego normal em um período pré-determinado. Tem como objetivo principal detectar padrões que fujam do comportamento pré estabelecido.[7] A

definição dos parâmetros destes comportamentos é o principal desafio deste tipo de ferramenta, sendo crucial para o bom funcionamento. Pode ser feito usando métodos estatísticos, métodos de data mining, métodos de aprendizado de máquina e outros[6, 7].

Na detecção baseada em modelos estatísticos cria-se um modelo estatístico à partir de dados do fluxo normal de trafego que será usado na detecção de pacotes anômalos. Atribui-se um critério de score para cada pacote, tomando um valor limite para a decisão de classificação. A definição de limites ideais é uma das maiores dificuldades da implementação visto que pode causar falsos positivos/negativos. Como parte de observações relacionados ao trafego pode-se obter certo grau de precisão mesmo sem conhecimento prévio sobre o comportamento do alvo. Por isso, em certos casos, é dependente da validade dessas observações e hipóteses, o que pode levar a falhas na detecção.

No contexto de DDoS, Data Mining remete a extração de dados e relações de grandes conjuntos de dados, auxiliando na interpretação de grandes conjuntos de dados. Em conjunto com técnicas aprendizado de máquina/inteligência artificial possibilita alta acurácia de detecção de ataques.[23]

O uso de técnicas de aprendizado de máquina no campo de detecção de DDoS é frequente como técnica de deteção de fluxo anômalo. Estes métodos permitem que uma máquina se adapte de acordo com a construção de um modelo à partir de entradas de exemplo, de forma que se tornem capazes de prever a classe de novos dados à partir de inferências provenientes do modelo de treino. Em [8] o autor compara estudos cuja acurácia de detecção de tráfego anômalo varia entre 85% e 99% em diversos métodos como K-nn, SVM, árvores convolucionais, redes neurais e outros. Artigos como [24, 2] avaliam e propõe métodos para a detecção de fluxos anômalos utilizando como base de teste o dataset NSL_KDD provido pelo Canadian Institute of Cybersecurity, o qual é composto por diversos tipos de ataques de rede, incluindo DoS, e é usado para a avaliação destes modelos.

2.2 Aprendizado de máquina

Machine Learning é um subset de algoritmos de Inteligencia Artificial focados na construção de aplicações capazes de aprender a partir de dados e de aumentar sua acurácia ao longo do tempo, sem serem programados para tal [25]. Essencialmente, são algoritmos capazes de inferir padrões de comportamento e tomar decisões com base em grupos de dados sem contar com grande intervenção humana.

Métodos de aprendizado de máquina podem ser enquadrados em três categorias principais, dependendo de sua abordagem quanto ao nível de supervisão[26]. Estes são divididos em:

- Métodos Supervisionados

  São treinados em cima de conjuntos de dados rotulados (labeled data). Se configuram por uma relação de x -> y, na qual as entradas possuem uma saída esperada. O objetivo do algoritmo é inferir, a partir da variável de saída, as relações entre as diferentes entradas e saída, sendo capaz de inferir a saída de novos valores de entrada à partir deste processo. São normalmente usados para problemas de classificação e regressão. O primeiro trata da classificação de novos dados à partir da criação de um modelo, tendo como saída classes discretas de dados, enquanto o segundo problema trata de casos no qual a saída é um valor numérico real, estimando o valor de saída esperado para aquele caso [26].

- Métodos não supervisionados

  Trata de métodos treinados em conjuntos de dados não rotulado. O algoritmo deve entender os relacionamentos entre os dados e encontrar padrões entre estes. Também podem ser classificados em duas categorias: agrupamento e associação. [26] Agrupamento se refere a algoritmos que tentam separar um conjunto de dados em grupos distintos com base em características escondidas, possibilitando um maior entendimento do comportamento do conjunto através de interações antes não vistas entre os dados. Algoritmos deste grupo conseguem inferir grupos naturais presentes no conjunto e até a quantidade de grupos presentes [27]. Associação por sua vez trata de estabelecer regras entre os dados presentes nos conjuntos [27].

- Métodos Semi-supervisionados

  São métodos que existem entre abordagens não supervisionadas e supervisionadas. Nestes casos grande parte do conjunto de dados não é rotulado, com uma fração menor rotulada [26]. Métodos semi supervisionados podem ser usados como forma de diminuir o impacto de insuficiência de dados rotulados, usando um dataset rotulado menor como forma de guiar a classificação de um conjunto de dados não rotulado maior [25][27].

2.2.1 Algoritmos

Como o objetivo deste projeto é a classificação de fluxo de dados foram escolhidos algoritmos de classificação para teste, sendo estes:

- K-nn (iBK no Weka)

  É um algoritmo não paramétrico que classifica dados com base na sua proximidade e associação em relação a outros dados. Se baseia na ideia de que dados de mesma classes ficam próximos um dos outros. Atua calculando a distância entre pontos e determinando a classe de um dado através do voto de maioria de k vizinhos. A fórmula de distância mais usada normalmente é a de distância euclidiana, apesar que outras podem ser escolhidas.[28]

- Árvores de decisão:

  O aprendizado por árvores de decisão consiste em um modelo preditivo que usa nós de decisão com base em atributos para fazer a classificação/regressão de entidades. O processo de gerar a árvore de decisão consiste principalmente na decisão dos atributos e quais as condições de divisão.

  Para isso, em cada decisão de nó é escolhido o caminho de menor perda de acurácia. Para a tomada dessa decisão podem ser escolhidos diversos critérios como entropia, ganho informacional, índice de gini, variância, chi-quadrado e razão de ganho [29].

  O algoritmo C4.5(J48 no weka) utiliza uma modificação do ganho informacional que leva em conta o número de galhos resultantes antes da tomada de decisão.

  Um problema comum em algoritmos de árvore de decisão ocorre quando, na tentativa de reduzir o número de erros em relação ao conjunto de treino, o algoritmo continua a gerar hipóteses, reduzindo a acurácia em relação ao conjunto de teste. Este problema é conhecido como overfitting. Uma solução para este problema são algoritmos de poda.

  Algoritmos de poda dividem o processo de treino em duas etapas, construindo a árvore com base em uma parcela do conjunto de treino, e usando outra parcela para avaliar quais galhos cortar. O processo é feito "podando" galhos de forma que a acurácia geral não seja reduzida [29].

  Um dos maiores problemas com árvores de decisão é a instabilidade presente nessa classes, sendo que dependendo da forma com que são construídas, pequenas mudanças no conjunto de treino podem mudar completamente a forma com que são organizadas [30].

- Florestas Randômicas

  Outra forma de lidar com o problema de overfitting são os algoritmos de florestas randômicas. São algoritmos do tipo ensemble, utilizando de múltiplas árvores de decisão que atuam em conjunto na classificação/regressão dos dados.

Seu funcionamento é baseado em várias árvores de decisão não-correlacionadas atuando como um comitê, o que garante uma performance superior a de todos os participantes deste comitê [31]. O baixo nível de correlação entre as árvores é necessário para a garantia de bom funcionamento, uma vez que permite que árvores errem individualmente mas tenham um bom funcionamento em conjunto (desde que não apresentem os mesmos erros).

Para garantir um baixo nível de correlação entre as árvores estes algoritmos se aproveitam da instabilidade de árvores de decisão e treinam cada um delas com amostras randômicas com substitutos.[31]

- SVMs

  Support Vector Machines, em português Maquinas de Vetores de Suporte, se refere a um modelo de aprendizado de máquina supervisionado que usa algoritmos de classificação de problemas de dois grupos. [32]

  De forma simplificada SVMs tentam seccionar um dataset de outro trançando limites entre as classes nos planos. Esses limites podem ser tanto uma linha quanto um hiperplano, dependendo da complexidade do problema. O algoritmo deve encontrar então o separador que garante a maior distância simultânea entre as duas classes, definindo as regiões do espaço de cada lado do como pertencente a cada uma destas. Como a dimensão de um dataset é equivalente ao número de features(n) do mesmo, a dimensão de um hiperplano acaba sendo n - 1, tornando a identificação de um hiperplano um problema não linear. Como forma de auxiliar a definição de um hiperplano são usados vetores de suporte, pontos de dados próximos ao hiperplano e que influenciam sua posição e orientação.[33]

  Para a definição de um hiperplano em conjuntos não lineares, em que a separação das classes não pode ser feita por uma linha, são usados kernels para a separação. Kernels são transformações do espaço de análise de um estado inicial (conhecido como espaço de entrada) para um estado de dimensão maior (conhecido como feature space). Os métodos buscam então por funções lineares neste espaço que separem as classes, que vão se tornar funções não lineares no plano original [34]. Alguns kernels normalmente usados são linear, polinomial, sigmoid e a função de base radial (RBF).

- Redes Neurais

  Redes Neurais são uma subclasse de algoritmos de aprendizado de máquina composta por algoritmos que tem a estrutura e funcionalidade inspiradas pelo cérebro humano, podendo ser utilizada de forma supervisionada ou não supervisionada [35].

Algoritmos baseados em redes neurais tentam simular o funcionamento de neurônios biológicos. São redes de nós interconectados, com estes sendo chamados neurônios. As conexões entre dois neurônios, análogas aos axônios de uma rede neural biológica, são responsáveis por atribuir os pesos entre esses[36]. Sempre que um valor chega a um neurônio, ele é multiplicado pelo peso da conexão com o neurônio conectado a ele. Dessa forma, o valor propagado por um neurônio é resultante da soma do produto de cada valor recebido em relação ao peso da conexão de que ele origina. O processo de aprendizado de uma rede consiste na atribuição de pesos entre neurônio de forma que a rede seja capaz de realizar o processo de classificação. O conjunto de pesos de uma rede é único para cada tarefa e cada conjunto de dados [36].

Este processo de aprendizado ocorre em dois mecanismos principais: propagação direta e propagação reversa. Na propagação direta a rede recebe os dados de entrada e propaga estes até a saída. As saídas resultantes são então avaliadas e comparadas ao resultado esperado através de funções de perda [37]. Uma função de perda é uma medida numérica de comparação entre o resultado esperado e o resultado obtido, sendo exemplos comuns desse tipo de função a de média quadrática de erro e a de entropia cruzada [36]. Uma vez calculados os erros estes são propagados para trás como forma de reduzir o erro da rede, com este processo sendo chamado de propagação reversa. Para isso, podemos usar um processo de gradiente descendente, o qual altera os pesos de forma progressiva, de acordo com uma taxa de aprendizado que define a velocidade com a qual é modificado. Com um gradiente descendente calculado à partir da função de perda, seu resultado é usado para ajustar o valor dos pesos entre neurônios, aumentando assim a acurácia da rede. Este processo é repetido sequencialmente até obter-se uma boa acurácia [37].

Alguns algoritmos que permitem a implementação de redes neurais são os Perceptrons de Multiplas Camadas. Estes geram uma ou mais camadas de neurônios, consistindo de uma camada de entrada, camadas escondidas e uma camada de saída. Aparte da camada de entrada, as camadas possuem ativação por funções não lineares. Usa retro propagação para treino, no qual os nós de decisão são reajustados. É recomendado para problemas de classificação onde os dados são rotulados. Apesar disso, é bastante flexível e pode ser usado em variados tipos de aplicação.

2.2.2 Conjuntos de treino, teste e validação

Um outro fator determinante no desempenho do modelo é a divisão dos dados utilizados entre conjuntos de treino e teste. O conjunto de treino é aquele que será utilizado para treinar o modelo e consequentemente serão os dados utilizados para classificação/regressão.

O conjunto de validação, será usado para a avaliação do modelo de forma que seja possível fazer ajuste dos parâmetros do modelo ou simplesmente seja testada sua acurácia. Por fim o conjunto de teste é usado somente após o modelo ter sido finalizado e tem como objetivo testar a performance deste.

Esta divisão é normalmente feita de forma aleatória, com o tamanho do split sendo decidido com base na quantidade de dados disponíveis e na quantidade de dados necessária para um bom treino do algoritmo.

2.2.3 Análise de performance

Matriz de confusão

Uma matriz de confusão tem como objetivo comparar os dados esperados de um teste de um modelo com os dados previstos. Para isso são calculados os dados em que ambos concordam e em que discordam, formando uma matriz quadrada que contém em sua diagonal principal os dados "verdadeiros" e no restante dos campos os dados "falsos" Em um caso de classificação binária na diagonal principal ficam os verdadeiros positivos e negativos, enquanto nos outros campos ficam os falsos positivos e falsos negativos, como consta na tabela 2.1

	Predição A	Predição B
Esperado A	Verdadeiros Positivos	Falso Negativo
Esperado B	Falsos Positivos	Verdadeiros Negativos

Tabela 2.1: Matriz de confusão.

A partir destes dados serão calculados diversas métricas de validação do modelo, relacionando sua capacidade preditiva.

Métricas de validação

- Acurácia[38]:

  É a razão entre o número total de predições corretas e o número total de predições, sendo descrito por:

  $$\frac{VP + VN}{VP + VN + FP + FN}$$

  É um bom medidor de performance em casos em que o dataset é balanceado, mas falha em demonstrar de forma precisa a performance do mesmo quando as classes são discrepantes em números de instâncias.

- Precisão, Recall e F1-Score [39]:

  É uma medida de relevância do resultado, sendo representada pela razão entre os verdadeiros positivos de uma classe e a soma dos verdadeiros positivos com os falsos positivos enquanto "recall" é uma medida de quantos dos dados resultantes são relevantes e é calculado pela razão entre verdadeiros positivos e a soma de verdadeiros positivos com falsos negativos. São usados para avaliação de modelos com classes desbalanceadas, representando melhor sua performance. Estes podem ser assimilados em conjunto através do F1-Score, que é a média harmônica entre os dois indicadores.

$$Precisão : \frac{VP}{VP + FP}$$

$$Recall : \frac{VP}{VP + FN}$$

  Por último, o F1-Score é média ponderada entre a precisão e o "recall", tendo seu maior valor como 1 e o menor como 0. Pode ser calculado por:

$$F1 - Score : 2 * \frac{precisão * recall}{precisão + recall}$$

2.3 Conjuntos de dados

2.3.1 NSLKDD

O conjunto de dados escolhido para os testes iniciais do desenvolvimento deste trabalho foi o NSL-KDD, provido pela UNB (University of New Brunswick). Foi desenvolvido para teste de sistemas de detecção de intrusão baseados em anomalias. Este é estruturado em múltiplos conjuntos, incluindo conjuntos de treino, teste e subsets dos mesmos.

O dataset é uma versão aprimorada do KDD99, contando com remoção de duplicatas e em torno menor redundância. Este, em si, é baseado no DARPA, sendo composto por features extraídas dos pacotes TCP/IP do conjunto. É composto por aproximadamente 126 mil dados que compõem o conjunto de treino e outros 22,5 mil que compõem o conjunto de teste, os quais são fornecidos em arquivos separados. Além destes conjuntos há ainda uma versão reduzida do conjunto de treino composto por 20% dos dados, e um subset do conjunto de teste que exclui os testes de dificuldade 21.

O KDDNSL contém 41 features. Este dataset é formado a partir de dados provenientes de pacotes normais, DoS, Probe, U2R e R2L. Informações mais detalhadas podem ser encontradas no anexo I, nas tabelas I.1, I.2, I.3 e I.4. Nas tabelas citadas é importante

Classe	Treino	Teste
Normal	67343 (54.25%)	9711 (43.08%)
DoS	45035 (36.28%)	7448 (33.04%)
Probe	11656 (9.39%)	2421 (10.74%)
U2R	52 (0.04%)	67 (0.30%)
R2L	995 (0.80%)	2885 (12.80%)
Total	125081	22532

Tabela 2.2: Proporção de classes dos dados presentes no NSL_KDD.

notar que existe uma grande disparidade entre as classes de dados presentes nos conjuntos de treino e teste.

Dos 40 tipos de ataque representados no dataset apenas 23 estão presentes no conjunto de treino contrastando com 38 tipos no conjunto de teste. Também é importante ressaltar que a proporção de representação de cada uma dessas classes é desproporcional, com classes como normal e neptune compondo 87% do conjunto de treino e 63% do conjunto de teste fornecido.

Um outro fato importante a ser levado em conta é que 16% dos dados do conjunto de teste não tem representação alguma no conjunto de treino, enquanto outros 9.7% estão presentes na casa de dezenas de registros, somando à disparidade. Em uma observação geral, 26% do conjunto de teste não possui representação efetiva no conjunto de treino, o que pode ser observado nas tabelas de tipos de ataques, sendo especialmente notável em classes como mscan, saint, warezmaster e guess_passwd que equivalem sozinhas a 15.48% do conjunto. As features que compõem o dataset estão presentes no anexo, na tabela: I.5

2.3.2 CiCDDoS2019

O segundo dataset escolhido para análise foi o CiCDDoS2019, também provido pelo instituto de cybersegurança da University of New Brunswick. Consiste em dados de tráfego que tentam simular o uso normal da rede em conjunto com ataques intercalados. Os pacotes foram gerados em dois dias diferentes e contém dados referentes a ataques volumétricos com uso de mecanismos de reflexão e ataques de protocolo/aplicação que os autores caracterizam como ataques exploração (exploitation), sendo organizados conforme a figura 2.2.

O dataset foi organizado pelos dias de ataque, sendo disponibilizado tanto na forma de dados não tratados como em um modelo de 80 características extraídas a partir do uso do CICFlowMeterV3, sendo esta a versão usada nesse projeto.

O primeiro dia de análise contém 8.15GB de dados referentes ao tráfego simulado, sendo divididos em arquivos de acordo com o ataque realizado no intervalo de tempo da análise. Neste dia, os testes feitos foram: LDAP, MSSQL, NetBIOS, Portmap, Syn, UDP

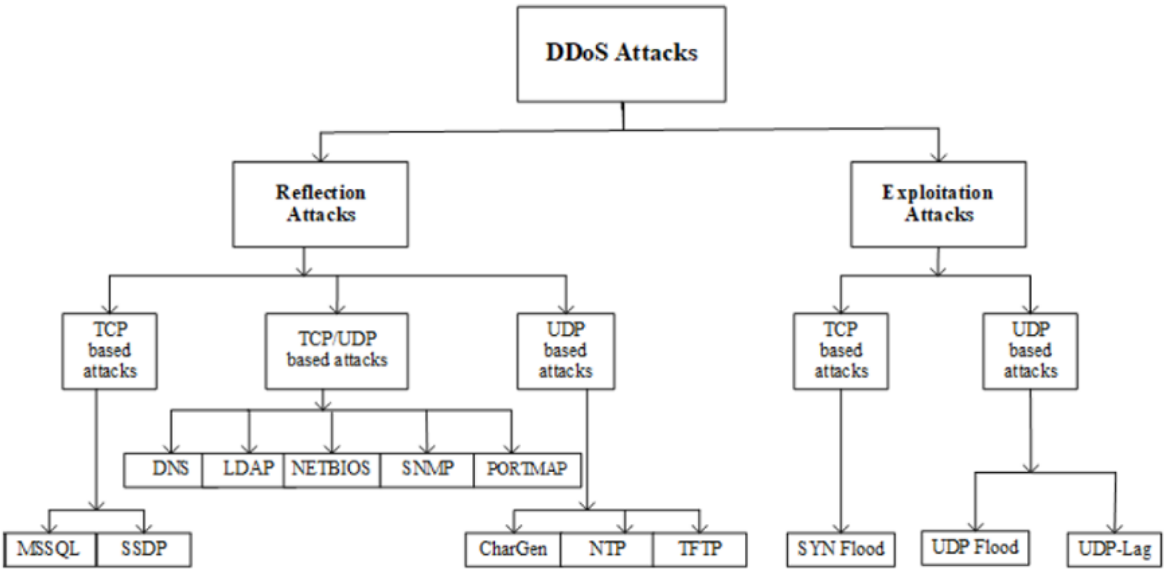

Figura 2.2: Ataques presentes no CiCDDoS2019.

e UDPLag, somando 8.15GB de dados de tráfego No segundo dia foram feitos todos os ataques identificados na figura 2.2 com exceção do ataque Portmap, totalizando 20GB de dados de tráfego.

2.4 Ferramentas

2.4.1 TensorFlow

Criado pelo Google para uso interno, acabou sendo distribuído em uma licença Apache em 2015, o Tensorflow é um FrameWork de Machine Learning, com o maior foco sendo algoritmos de Deep Learning. Utiliza Python como front-end através da API Keras, apesar de ser implementado em C++.

Possui suporte nativo a aceleração com uso de GPUs, o que no caso de modelos maiores se mostra imprescindível. O nome da plataforma faz alusão aos Tensores, unidade básica de seu funcionamento, e que atuam como arrays multidimensionais imutáveis com tipagem uniforme.

Foi utilizado também o plugin "tensorboard" para a escolha de hiperparâmetros. Este gera gráficos comparativos de performance dos parâmetros escolhidos, com as cores dos traços sendo indicativas da performance de cada combinação de parâmetros.

2.4.2 Weka

Weka (Waikato Environment for Knowledge Analysis) é uma ferramenta de código aberto para data-mining e aprendizado de máquina. É uma coletânea de ferramentas de visualização e algoritmos de análise de dados e modelagem preditiva [40].

Contém múltiplos algoritmos de aprendizado de máquina como K-nn, SVM, Random Forests, Decision Trees (incluindo C4.5 como J48) e métodos de Deep Learning como MultiLayer Perceptron. Possibilita também fácil visualização de características de conjuntos de dados, possuindo diversos métodos de processamento de dados como filtragem por atributo e análise de ganho informacional por classe.

Utiliza um formato de arquivo próprio para conjuntos de dados, chamado arff (Atribute-Relation File Format), que inclui no cabeçalho do arquivo informações de tipagem dos atributos, sendo que em atributos nominais são designados os valores aceitos em cada. O cabeçalho é composto pelo nome da relação, e uma lista dos atributos com seus tipos. ,

2.4.3 Scikit Learn [1]

É um framework gratuito para python e inclui diversas ferramentas de aprendizado de máquina, sendo desenhado para operar em conjunto com bibliotecas como Numpy. Possui acesso à métodos como SVM, K-nn, florestas randômicas e diversos outros. Foi usado no projeto pela implementação nativa de um "Baggin Classifier" capaz de usar subamostragem para gerar n classificadores que atuam em conjunto para a paralelização da implementação de uma SVM.

A complexidade de uma SVM com kernel não linear, que varia de $O(n^2)$ a $O(n^3)$ dependendo da implementação torna o tempo de execução extremamente longo em conjuntos de dados maiores. Subdivindo o problema e usando múltiplos núcleos foi possível diminuir bastante o tempo de execução.

Capítulo 3

Metodologia

O processo de execução dos experimentos passou por dois mecanismos principais: processamento de dados e tunagem de algoritmos, com a tomada de decisões de processamento de dados também sendo consequência dos resultados obtidos na execução dos algoritmos. Os testes foram executados em um ambiente de execução Windows, com uso de 3 plataformas principais: Weka, SciKit-learn e Tensorflow, com o primeiro sendo usado também para a análise e visualização dos dados.

3.1 Processamento de dados

3.1.1 NSL_KDD

O conjunto de dados é fornecido em sua totalidade em duas formas principais: em arquivos .txt com dados tabelados em formato csv, que incluem a dificuldade e tipos de ataque, apesar de não conter rótulos para os atributos, e em arquivos .arff com atributos rolutados e classificados entre pacotes normais e pacotes anômalos, ambos com arquivos pré definidos de treino e teste. A divisão entre treino e teste proposta pelos autores do algoritmo tem como objetivo padronizar a análise de resultados entre diferentes estudos. Os testes foram feitos tanto com a divisão proposta pelos autores como em uma nova divisão realizada à partir da randomização do conjunto.

Os dados fornecidos no conjunto não necessitam de grandes medidas de pré tratamento uma vez que estão devidamente padronizados, não possuem valores vazios nem destoantes em relação ao tipo do atributo e estão divididos entre 34 atributos numéricos e 8 nominais (com um destes sendo a classe), com 4 dos valores nominais no conjunto atuando como flags binárias.

As análises feitas usando o conjunto de dados na totalidade foram realizadas através da junção dos dois conjuntos iniciais de treino e teste.

A seleção dos atributos foi feita em relação à análise de DAS, Saikat et al [2], na qual os autores identificam atributos relacionados à cada tipo de ataque e os selecionam para seu modelo. Os atributos escolhidos estão na tabela 3.1.

Ataque	Features relevantes
Land	7
Smurf	2, 3, 5, 23, 24, 27, 28, 36, 40, 41
Neptune	4, 25, 26, 29, 30, 33, 34, 35, 38, 39
Terdrop	8
Black	10, 13
Novo conjunto	2, 3, 4, 5, 7, 8, 10, 13, 23, 24, 25, 26, 27, 28, 29, 30, 33, 34, 35, 36, 38, 39, 40, 41

Tabela 3.1: Features do NSL_KDD escolhidas na distribuição proposta

O novo conjunto passa a ser composto por 24 atributos, sendo 4 categóricos, 19 numéricos e a classe dos dados, tendo um foco maior nos dados mais relevantes aos ataques DoS presentes, excluindo atributos voltados a tentativas de acesso, verificações de login, verificações de permissão, entre outros que não são relevantes no contexto deste trabalho. No geral foram mantidos os atributos relacionados ao comportamento do pacote na rede.

3.1.2 CiCDDoS 2019

Devidos às enormes quantidades de dados fornecidas pelo conjunto, as quais totalizam quase 30GB de dados, foi tomada a decisão de utilizar um subconjunto para a realização de testes sobre o conjunto de dados. Como o tamanho dos arquivos tornava a análise dos mesmos difícil, uma vez que alguns arquivos passavam facilmente da casa de GB em tamanho, estes foram subdivididos, randomizados e a partir disso selecionados conjuntos menores de dados.

DNS	38977	UDP	44987
LDAP	69949	Syn	43967
MSSQL	47977	TFTP	47992
NetBIOS	41977	UDPLag	47035
NTP	31984	Portmap	23961
SNMP	34356	Benign	76249
SSDP	53980		

Tabela 3.2: Distribuição dos dados no subconjunto inicial do CiCDDoS.

No fim foram selecionados em torno de 40 mil instancias de cada classe atacante e em torno de 75 mil instâncias de tráfego normal. É importante ressaltar a baixa disponibilidade de dados de tráfego normal no conjunto, os quais são apresentadas em quantidades

muito inferiores às classes atacantes. Enquanto os dados adquiridos com o subamostragem do conjunto de dados completo continham mais de 500 mil instâncias de dados de ataques, apenas 3000 instâncias de dados benignos foram capturados, fazendo com que fosse necessária a seleção destes de forma separada.

Para o uso destes em conjunto com algoritmos de machine learning os valores precisaram ser tratados, substituindo valores que constavam como infinitos por -1 e tratando valores indefinidos e vazios como 0. Atributos textuais, como Similar HTTP, ou que contenham endereços IP, como FlowID, Source IP, Source Port, e relacionados foram removidos para garantir que o modelo fosse generalizado. Foram retiradas também atributos cujo valor era uniforme em todos os dados separados, com alguns dos bit de Flag, que eram sempre 0 em todo o conjunto. Como a quantidade de dados se mostrou muito grande para o treino eficiente em alguns algoritmos e a proporção de dados de ataque para dados benignos estava em disparidade, foi tomada a decisão de reduzir os dados ataque presentes no conjunto.

	Ataque	Benigno	Proporção
Inicial	527142	76249	14.46%
Reduzido	263571	76249	28.9%

Tabela 3.3: Composição do subconjunto do CiCDDoS em relação às classes de dados.

Depois, em sequência, foi calculado o ganho informacional de cada atributo, os quais foram utilizados para determinar um dataset composto por um subconjunto reduzidos dos atributos, chegando a uma versão composta por 23 atributos numéricos e a classe dos dados, os quais foram separados entre “benigno” e “ataque”. As features escolhidas estão contidas na tabela 3.4

1	Flow duration	13	Bwd header lenght
2	Total Backward packets	14	Fwd packets/s
3	Total Length of fwd packets	15	Bwd packets/s
4	Fwd packet length max	16	Min packet length
5	Fwd packet length min	17	Max packet length
6	Fwd packet length mean	18	Packet lenght std
7	Flow Bytes/s	19	Packet lenght variance
8	Flow Packets/s	20	Average fwd segment size
9	Flow IAT std	21	Average packet size
10	Flow IAT max	22	Subflow fwd bytes
11	Bwd IAT max	23	Init Win Bytes Fwd
12	Bwd IAT std	24	Class

Tabela 3.4: Features escolhidas do CiCDDoS 2019.

3.2 Otimização de Parâmetros de Algoritmos

O processo de escolha de hiperpâmetros foi diferente para cada algoritmo executado, com base nos parâmetros disponíveis para escolha em cada um destes. O processo é dependente também do framework utilizado.

3.2.1 K-nn (iBK no Weka)

O processo de otimização do K-nn envolveu a escolha de dois parâmetros principais:

- Número de vizinhos (K):

  O número de vizinhos a ser considerados na classificação de um item

- Algoritmo de busca:

  O algoritmo utilizado para a busca do vizinho mais próximo.

O número de vizinhos foi variado entre os números impares contidos entre 2 e 40, enquanto os algoritmos de busca testados foram o de busca linear, a implementação padrão do algoritmo, e o KDTree, que implementa uma árvore de busca para encontrar um vizinho mais próximo. Como exemplos do processo iterativo de busca, nas figuras 3.3 e 3.4 estão a relação entre a acurácia e o número de vizinhos no CiCDDoS e no NSL_KDD (distribuição sugerida):

Figura 3.1: Acurácia do algoritmo em relação ao número de vizinhos no NSL_KDD.

O algoritmo de busca escolhido foi o KDTree pela maior eficiência no tempo de execução se comparado a uma abordagem linear sem haver perda de acurácia. O exemplo na tabela 3.5 é representativo do tempo de execução do K-nn no NSL_KDD(na distribuição sugerida) com 3 vizinhos:

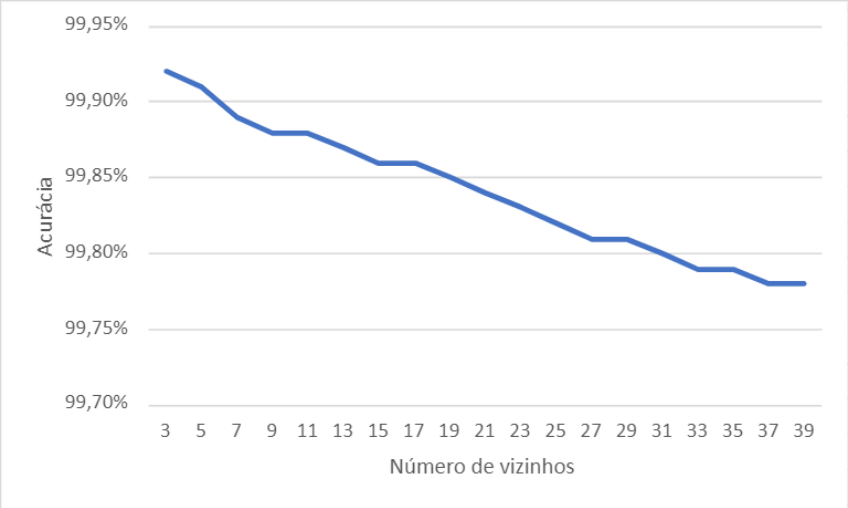

Figura 3.2: Acurácia do algoritmo em relação ao número de vizinhos no CICDDoS.

	Tempo(s)	Acurácia
KDTree	9.38s	78.695%
Busca Linear	173.8s	78.695%

Tabela 3.5: Tempo de execução do K-nn no NSL_KDD com diferentes algoritmos de busca de vizinhos.

3.2.2 SVM (SMO no weka | svm pelo scikit-learn)

Para uma execução otimizada do algoritmo foi necessária a binarização das classes nominais, transformando-as em múltiplas classes binárias equivalentes a cada um dos seus possíveis valores. Esse fator teve maior impacto no NSL_KDD, no qual a classe service possui 71 valores possíveis, a classes flag possui 11 e a classe protocol possui 3. O subset utilizado do CiCDDOS2019 é composto basicamente por atributos numéricos, então não teve de ser processado da mesma forma.

O algoritmo possui 3 hiperparâmetros principais:

- Kernel:

  O kernel a ser usado para a separação das classes.

- C (custo):

  É a penalidade induzida por um erro de classificação. Um valor menor reduz a penalidade de erros de classificação e leva a uma maior área de decisão, enquanto um valor maior tende à reduzí-la

- Gamma

Controla a distância de influência de um único ponto de decisão. Um gamma maior tende a reduzir a área de decisão de uma classe, enquanto um gamma menor tende a aumentá-la.

A escolha dos parâmetros foi feita com base em um algoritmo de GridSearch, escolhendo valores discretos para o teste de cada hiperparâmetro e percorrendo as possíveis permutações entre eles.

Como a performance do algoritmo cresce bastante de acordo com o número de elementos, especialmente dependendo do kernel e implementação do algoritmo, foi tomada a decisão de combinar o SVM com o BaggingClassifier presente no SciKit Learn. A função gera um sistema de voto por ensemble baseado em uma abordagem de Bagging, similar a presente em florestas randômicas. Com isso foi possível reduzir o número de dados em uma única SVM, gerando múltiplas SVM menores e ainda paralelizar o processo, sendo possível controlar tanto o número de SVM a serem geradas quanto o números de jobs de processamento. Essa abordagem permitiu acelerar a execução no NSL_KDD, que dependendo dos parâmetros passava de 1 hora de execução, para alguns minutos, além de permitir que testes no CiCDDoS fossem feitos de forma eficiente.

Foram testados para ambos os conjuntos de dados diferentes valores de C e Gamma, variando de 0.1 à 100, com os resultados provenientes de testes com valores mais altos se sobressaindo. Os kernels testados foram “RBF” e “polinomial”. Na tabela 3.6 segue uma comparação da performance dos kernels com alguns hiperparâmetros no NSL_KDD nos testes feitos com uma divisão 70:30 para treino e teste: No caso do CiCDDoS o tempo de

Kernel	C	Gamma	Acurácia
RBF	100	1	97,81%
RBF	100	10	97,03%
Poly	100	1	98,20%
Poly	100	10	98,27%

Tabela 3.6: Exemplos de varição de parâmetros na execução da SVM no NSL_KDD.

duração do treino da SVM com kernel polinomial dificultou a execução de testes, com o tempo de execução para alguns parâmetros ultrapassando a casa de 5 horas de duração. Por esse motivo foi escolhido o kernel “RBF” por oferecer, para esse conjunto de dados, uma performance de detecção similar enquanto seu tempo de treino ficava na casa de alguns minutos. Abaixo segue uma tabela comparando a acurácia da SVM no CiCDDoS para alguns parâmetros:

Kernel	C	Gamma	Acurácia
RBF	1	0,1	85,89%
RBF	10	1	92,73%
Poly	100	1	98,84%
RBF	100	10	99,45%

Tabela 3.7: Exemplos de varição de parâmetros na execução da SVM no CiCDDoS

3.2.3 MLPs

O framework escolhido para a implementação de uma rede neural profunda foi o Tensorflow uma vez que os testes com MLP no weka foram lentos e a implementação no tensorflow, com aceleração por GPU, foi múltiplas vezes mais rápida. Os gráficos gerados pelo tensorboard apresentam diferentes combinações de hiperparâmetros, com as cores das linhas sendo resultantes de gradientes de azul para vermelho, com valores de acurácia mais próximos de azul indicando valores mais baixos enquanto linhas vermelhas se aproximam das acuáracias mais altas.

Para a implementação do modelo foi feito o tratamento dos dados, gerando à partir dos arquivos .csv estruturas do tipo Dataset do framework. A partir destas foi feito o uso de camadas de normalização para dados númericos e o uso de camadas de condificação one-hot para dados categóricos. Codificação one-hot é uma técnica que gera um dicionário binário para um atributo categórico, fazendo com que um atributo de 70 categorias se torne 70 atributos binários, possibilitando uma melhor interpretação por uma rede neural. Também foi feito o uso de "dropout", técnica de regularização que tenta evitar "overfitting" pelo abandono aleatório de neurônios com base em uma taxa de abandono. A técnica permite que o modelo seja treinado em diferentes subconjuntos de suas features, inferindo novas relações que aumentam a generalização da rede.

O modelo em si foi testado com 3, 4 e 5 camadas densas somadas a uma camada de "dropout" antes da camada densa de saída. Cada camada densa tem como parâmetro o número de unidades que a compõe, enquanto a camada de "dropout" tem como parâmetro a probabilidade de um neurônio ser deixado de lado no processo de treino em um subconjunto.

No total foram testadas diversas topologias para a rede neural, com diversos números de camadas e diversos números de unidades. Para cada uma destas foi variado também a taxa de "dropout". Os resultados dos testes referentes ao NSL_KDD com 3 e 4 camadas densas podem ser vistos nas figuras 3.3 e 3.4.

Como a topologia com 4 camadas densas e uma camada de "dropout" teve uma performance levemente superior, esta foi escolhida para os testes a serem feitos.

Devido à diferença na distribuição dos dados entre treino e teste, foi avaliado também

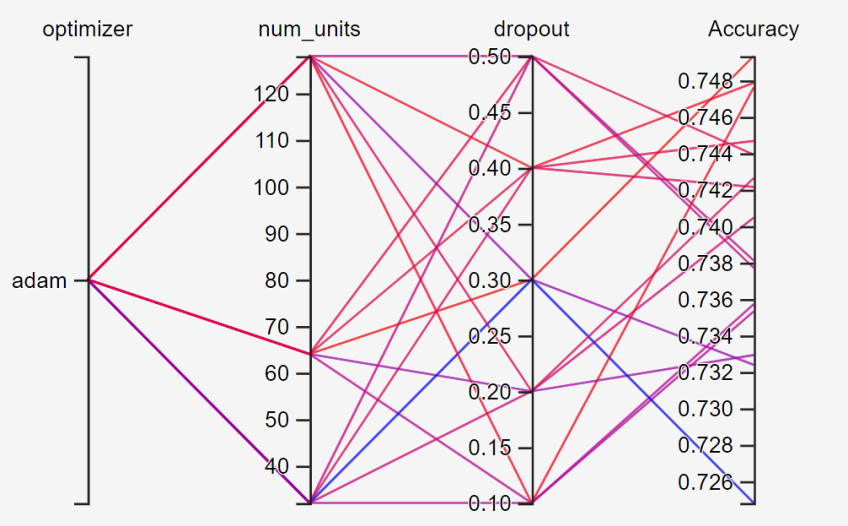

Figura 3.3: Teste de escolha de hiperparâmetros para o NSL_KDD com 3 camadas densas e uma camada de "dropout".

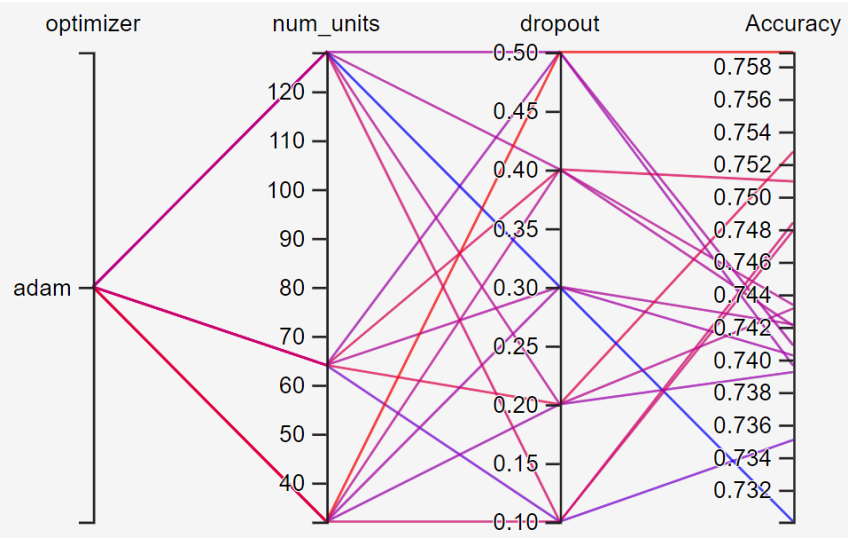

Figura 3.4: Teste de escolha de hiperparâmetros para o NSL_KDD com 4 camadas densas e uma camada de "dropout".

a possibilidade de uso regularização de kernel no modelo, com esta sendo inserida na camada densa que antecede a camada de "dropout". O uso desta técnica, assim como dropout, tem como objetivo evitar overfitting, evitando que o modelo se torne específico demais à distribuição dos dados do conjunto de treino e perca performance em relação a um contexto generalizado. A abordagem consiste em modificar a função de perda do modelo de forma a evitar que os pesos da rede fiquem muito grandes, atribuindo uma penalidade ao modelo no caso de crescimento destes.

Os resultados obtidos com o uso de regularização podem ser vistos na figura 3.5.

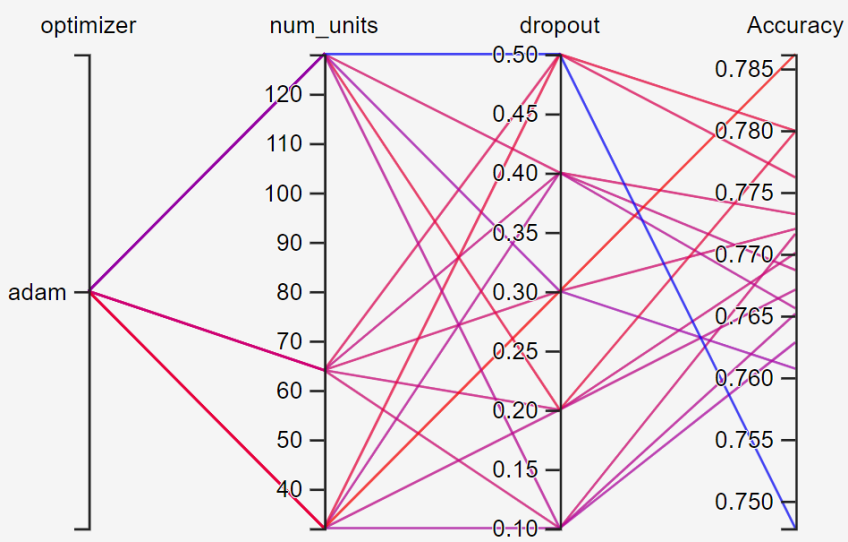

Figura 3.5: Teste de uso de regularização de kernel no modelo.

Também foi feita a comparação entre o uso de dois otimizadores para o modelo, “adam” e “sgd’, com os resultados presentes na imagem 3.6.

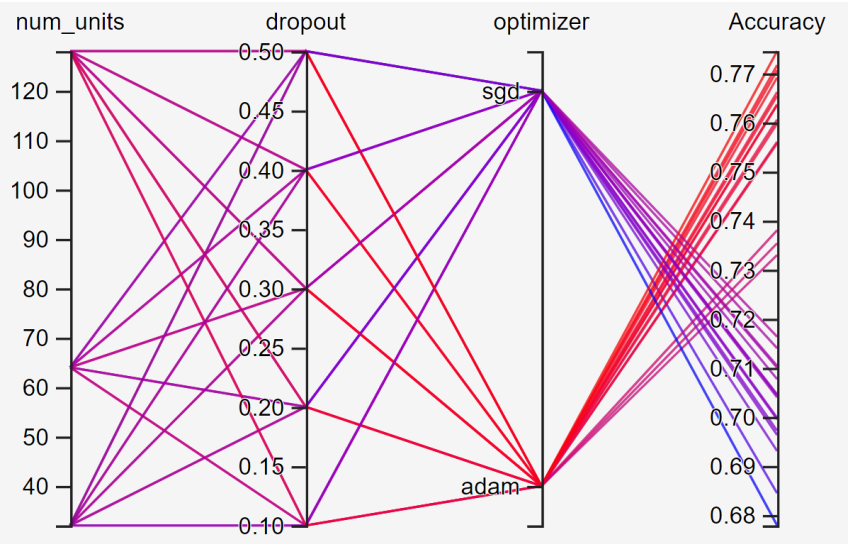

Figura 3.6: Comparação de performance entre os otimizadores.

Foi avaliado também o uso de early stopping, técnica que utiliza as métricas internas do modelo em relação ao conjunto de validação para evitar overfitting parando o treino quando atinge um checkpoint. Esta técnica se mostrou ineficaz no caso do NSL_KDD, uma vez que o conjunto de validação, por surgir de uma divisão do conjunto de treino, não oferecia um bom parâmetro de comparação em relação à performance do modelo.

A partir da definição desses fatores foi definida a topologia da rede, que pode ser conferida na figura 3.7.

Como o CiCDDoS possui grandes quantidades de dados relativos ao tráfego e testes iniciais de classificação mostraram bons resultados, com estes na casa de 99% de acurácia quando avaliado em relação ao conjunto de teste, em especial sem apresentar diferenças entre as métricas internas de acurácia em relação ao conjunto de validação, não foi necessário o uso de regularização ou early stopping. Fora essa questão, a tomada de decisões sobre o modelo seguiu o mesmo processo utilizado para o outro conjunto de dados, culminando no modelo presente na figura 3.8.

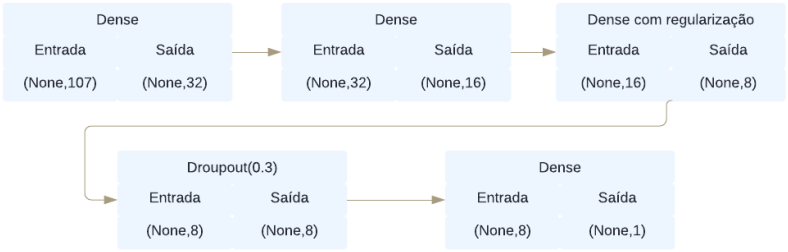

Figura 3.7: Topologia do modelo de rede neural para o NSL_KDD.

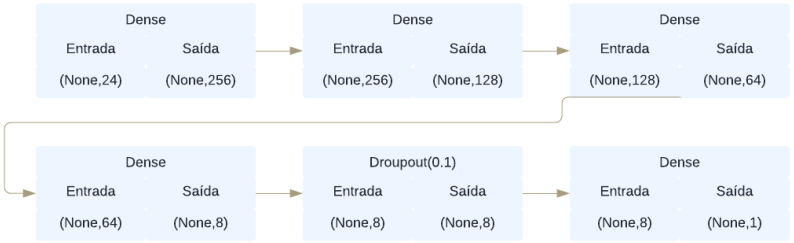

Figura 3.8: Topologia do modelo de rede neural para o CiCDDoS.

3.2.4 C4.5 (J48 no weka)

Os fatores de maior impacto sobre o funcionamento do algoritmo são as variáveis confidence factor e minNumObj, que configuram respectivamente o fator de confiança para a poda de uma árvore e o menor número de instâncias em uma folha.

O primeiro coeficiente foi testado em valores de 0.1 a 1 e o segundo de 2 a 10. Os testes realizados mostraram baixa variação nos valores de acurácia em relação aos parâmetros, com a diferença do melhor para o pior caso ficando na casa de 0.07%. Como consta na figura 3.8

Dataset	Acurácia do melhor Caso	Acurácia do pior caso
CiCDDoS	99.94%	99.91%
NSL_KDD(divisão 70:30)	99.45%	99.37%

Tabela 3.8: Variação de acurácia de acordo com hiperparâmetros do J48

3.2.5 Random Forests

Assim como no caso do algoritmo J48, apresentou pouca variação de performance em relação à mudanças nos hiperparâmetros. O hiperparâmetro alterado foi o número de árvores à serem geradas e sua variação variou pouco a acurácia final do algoritmo.

Capítulo 4

Resultados

4.1 NSL_KDD

4.1.1 Conjunto completo:

Os resultados obtidos com os testes feitos no conjunto de dados com todas as suas features e de acordo com a divisão entre KDDTrain+ e KDDTest+, proposta pela formatação inicial do conjunto, se encontram na tabela 4.1.

Algoritmo	J48	RF	K-nn	SVM	MLP (weka)
Acurácia	81.53%	80.45%	79.80%	77.56%	80.05%
Precisão (normal)	0.708	0.695	0.691	0.674	0.692
Recall (normal)	0.973	0.973	0.961	0.923	0.968
F1-Score	0.820	0.811	0.804	0.779	0.807
Precisão (anomalia)	0.971	0.971	0.958	0.916	0.965
Recall (anomalia)	0.696	0.677	0.674	0.661	0.674
F1-Score	0.811	0.798	0.791	0.768	0.794

Tabela 4.1: Resultados dos testes no NSL_KDD com todos os atributos.

O índice de acurácia dos algoritmos ficou entre 77.5% e 81.5% e uma tendência grande a falsos positivos no fluxo de tráfego normal. Essa tendência pode ser explicada pela distribuição disforme das classes entre os conjuntos de treino e teste, uma vez que essencialmente o modelo não teria sido treinado para lidar com os tipos de dados que estão sendo testados.

Dito isso, e levando em conta que até 26% dos dados de ataque não tem representação efetiva no conjunto de treino, é considerável afirmar que os algoritmos conseguiram uma taxa razoável de acerto, em especial na identificação de dados de tráfego benignos, já que tiveram um alto nível de Recall para esta classe. A identificação de pacotes anômalos

teve uma taxa de sucesso consideravelmente inferior, se situando na casa de 0,67 a 0,695, indicando uma dificuldade do modelo em detectar estes pacotes corretamente.

4.1.2 Subconjunto com atributos reduzidos

Em seguida foram feitos testes com a redução no número de atributos proposta por DAS, Saikat et al. [2], reduzindo o conjunto a 24 atributos, seguindo a tabela 3.1, que em sua totalidade pode ser conferido na tabela 4.2.

Ataque	Features relevantes
Novo conjunto	2, 3, 4, 5, 7, 8, 10, 13, 23, 24, 25, 26, 27, 28, 29, 30, 33, 34, 35, 36, 38, 39, 40, 41

Tabela 4.2: Novo conjuto formado a partir do NSL_KDD.

Distriubição Proposta

Algoritmo	J48	RF	K-nn	SVM	MLP(Tensorflow)
Acurácia	82,87%	80,18%	80,72%	80,70%	77,13%
Precisão (normal)	0.725	0.693	0.700	0.697	0.670
Recall (normal)	0.971	0.971	0.965	0.975	0.926
F1-Score	0.830	0.809	0.811	0.813	0.777
Precisão (anomalia)	0.970	0.968	0.963	0.973	0.921
Recall (anomalia)	0.721	0.674	0.687	0.680	0.579
F1-Score	0.827	0.795	0.802	0.800	0.711

Tabela 4.3: Resultados dos testes no NSL_KDD com atributos reduzidos de acordo com o subconjunto proposto por DAS, Saikat et al [2].

A performance, que pode ser observada na tabela 4.3, foi semelhante para a maioria dos algoritmos, com um leve aumento de precisão no Knn e no J48 e uma pequena perda em Random Forests. O MLP implementado no tensorflow teve uma queda de performance com a retirada dos dados, implicando na redução de precisão na identificação de dados não referentes à ataques DoS. A performance dos outros algoritmos foi similar, com a mesma tendência de falsos positivos em relação à classe “normal” e um bom recall para a mesma. A exceção à regra foi o SVM, que pela simplificação do modelo teve uma performance 3% superior.

Distribuição 70:30

Por último foi feita uma análise da performance sobre um novo conjunto formado pelos dados combinados de treino e teste apresentados anteriormente, o qual foi randomizado e

dividido em novos conjuntos de treino e teste com 70% e 30% dos dados respectivamente. Os resultados podem ser observados na tabela 4.4.

Algoritmo	J48	RF	K-nn	SVM	MLP(Tensorflow)
Acurácia	99,46%	99,54%	98,62%	97,81%	97,40%
Precisão (normal)	0.994	0.994	0.986	0.953	0.962
Recall (normal)	0.995	0.996	0.965	0.941	0.986
F1-Score	0.994	0.995	0.975	0.947	0.974
Precisão (anomalia)	0.995	0.995	0.963	0.937	0.986
Recall (anomalia)	0.993	0.993	0.987	0.951	0.958
F1-Score	0.994	0.994	0.975	0.944	0.972

Tabela 4.4: Resultados dos testes no NSL_KDD com divisão 70:30 para treino e teste

Os resultados obtidos mostram um grande aumento de performance em relação à divisão anterior, ficando acima de 97% de acurácia em todos os casos. O aumento de performance para a nova distribuição aleatorizada dos dados corrobora para a hipótese de que a baixa performance obtida anteriormente foi relacionada à distribuição escolhida dos dados. Dividir os dados de forma aleatória permitiu que as classes anteriormente não representadas no conjunto de treino pudessem participar do processo de treino, aumentando a acurácia verificada. Os métodos J48 e RF novamente acalçaram os maiores índices de performance, se sobressaindo em relação aos demais e mantendo uma performance superior à 99%.

4.2 CiCDDoS2019

Os resultados obtidos com a criação de modelos a partir do subconjunto selecionado do CiCDDoS, segue na tabela 4.5

Algoritmo	J48	RF	K-nn	SVM	MLP(Tensorflow)
Acurácia	99,95%	99,90%	99,90%	99,58%	99,70%
Precisão (normal)	0.999	0.998	0.998	0.980	0.991
Recall (normal)	0.999	0.998	0.998	0.980	0.997
F1-Score	0.999	0.998	0.998	0.980	0.994
Precisão (anomalia)	0.998	0.999	0.999	0.998	0.999
Recall (anomalia)	0.999	0.999	0.999	0.999	0.997
F1-Score	0.998	0.999	0.999	0.998	0.998

Tabela 4.5: Resultados do modelo criado a partir do CiCDDoS

Os algoritmos tiveram uma excelente performance, com a acurácia de todos acima de 99.5%. Assim como no caso do NSL_KDD, os algoritmos baseados em árvores de decisão, o J48 e RF, tiveram os melhores resultados, assim como o K-nn. A performance

de perceptron de múltiplas camadas e a máquina de vetor de suporte ficaram um pouco abaixo em desempenho mas ainda tiveram bons resultados.

Mesmo utilizando apenas 23 dos 87 atributos presentes no conjunto os algoritmos ainda mantiveram a eficácia na detecção dos fluxos anômalos, se mostrando capazes de separar os ataques do fluxos benignos.

Capítulo 5

Conclusão

Ataques DDoS são uma ameaça constante e crescente à estabilidade e ao fornecimento de serviços online. A capacidade disruptiva destes, junto à frequência com que ocorrem e os esforços necessários para sua mitigação reforçam à necessidade de constante melhora nos métodos disponíveis tanto para a prevenção quanto para a mitigação destes.

Dentre os problemas enfrentados pela mitigação a detecção de fluxo atacante se mostra um dos maiores desafios, com ataques zero-day e variações de ataques conhecidos podendo passar despercebidos por sistemas de detecção baseados em assinaturas enquanto sistemas de detecção baseados em anomalias necessitando de análises do tráfego normal para conseguir diferenciar padrões anômalos no seu comportamento.

Este trabalho teve como objetivo a análise de métodos de aprendizado de máquina na detecção de fluxos atacantes, avaliando dois conjuntos de dados distintos e algoritmos como K-nn, árvores de decisão e redes neurais e suas atuações como sistemas de detecção. Os resultados obtidos foram positivos, mostrando uma eficácia de até 99% na identificação de fluxos anômalos, confirmando a eficácia dos algoritmos testados e indicando técnicas de ML como eficazes na detecção de ataques DDoS.

A performance dos algoritmos baseados em árvores de decisão, tanto J48 quanto Florestas Randômicas, se destacou tanto pela baixa necessidade de escolha de hiperparâmetros, que tiveram baixa influência no resultado, quanto pela acurácia resultante de sua execução. São algoritmos que mostraram ser simples de implementar e, no caso de dados com presença de atributos categóricos necessitar de menor esforço com pré-processamento se comparado à redes neurais e máquinas de suporte.

Para trabalhos futuros pode ser avaliado a implementação de um sistema de análise de tráfego em tempo real para separar fluxos anômalos de benignos. Para a implementação seria necessário o uso de uma analisador de tráfego de tempo real que produza os dados necessários para a caracterização dos fluxos em conjunto a um classificador baseado em aprendizado de máquina para a separação destes.

Referências

[1] *Scikit-learn*. https://sklearn.org/. vii, 20

[2] Das, Saikat, Ahmed M. Mahfouz, Deepak Venugopal e Sajjan Shiva: *Ddos intrusion detection through machine learning ensemble*. Em *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, páginas 471–477, 2019. x, 11, 22, 33

[3] *Ddos attacks increase by 151% in first half of 2020*. "https://www.businesswire.com/news/home/20200916005046/en/DDoS-Attacks-Increase-by-151-in-First-Half-Of-2020". 1

[4] a10Networks: *The state of ddos weapons*. 2020. https://www.a10networks.com/wp-content/uploads/A10-EB-The-State-of-DDoS-Weapons-Report.pdf. 1, 2

[5] *Understanding ddos attacks*. "https://www.a10networks.com/blog/understanding-ddos-attacks/". 2, 6

[6] Mahjabin, Tasnuva, Yang Xiao, Guang Sun e Wangdong Jiang: *A survey of distributed denial-of-service attack, prevention, and mitigation techniques*. International Journal of Distributed Sensor Networks, 13:155014771774146, dezembro 2017. 3, 5, 7, 9, 10, 11

[7] Osanaiye, Opeyemi, Kim Kwang Raymond Choo e Mqhele Dlodlo: *Distributed denial of service (ddos) resilience in cloud: Review and conceptual cloud ddos mitigation framework*. Journal of Network and Computer Applications, 67:147–165, 2016, ISSN 1084-8045. https://www.sciencedirect.com/science/article/pii/S1084804516000023. 3, 10, 11

[8] Tayyab, Mohammad, Bahari Belaton e Mohammed Anbar: *Icmpv6-based dos and ddos attacks detection using machine learning techniques, open challenges, and blockchain applicability: A review*. IEEE Access, 8:1–19, setembro 2020. 3, 11

[9] Zargar, Saman Taghavi, James Joshi e David Tipper: *A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks*. IEEE Communications Surveys Tutorials, 15(4):2046–2069, 2013. 4

[10] *The psychology behind ddos: Motivations and methods*. "https://www.perimeter81.com/blog/network/the-psychology-behind-ddos-attacks". 4

[11] *Anonymous volta à ativa contra bolsonaro e trump; conheça o grupo hacker.* "https://www.uol.com.br/tilt/noticias/redacao/2020/06/02/anonymous-volta-a-ativa-contra-bolsonaro-e-trump-conheca-o-grupo-hacker.htm". 5

[12] Jun, Jae Hyun, Hyunju Oh e Sung Ho Kim: *Ddos flooding attack detection through a step-by-step investigation.* Em *2011 IEEE 2nd International Conference on Networked Embedded Systems for Enterprise Applications*, páginas 1–5, 2011. 5

[13] Chhabra, Meghna, B B Gupta e Dr.Ammar Almomani: *A novel solution to handle ddos attack in manet.* Journal of Information Security, 04:165–179, janeiro 2013. 5

[14] *What is a ddos attack?* "https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/". 6

[15] *What is a volumetric ddos attack?* "https://www.netscout.com/what-is-ddos/volumetric-attacks". 6

[16] *https://www.netscout.com/what-is-ddos/state-exhaustion-attacks.* "https://www.netscout.com/what-is-ddos/state-exhaustion-attacks". 6

[17] Gondim, João J.C., Robson de Oliveira Albuquerque e Ana Lucila Sandoval Orozco: *Mirror saturation in amplified reflection distributed denial of service: A case of study using snmp, ssdp, ntp and dns protocols.* Future Generation Computer Systems, 108:68–81, 2020, ISSN 0167-739X. https://www.sciencedirect.com/science/article/pii/S0167739X19322745. 8

[18] *Ncsc: Ingress and egress filtering.* "https://www.ncsc.gov.ie/emailsfrom/DDoS/Ingress-Egress/index.html". 9

[19] Chowriwar, Shalaka, Madhulika Mool, Prajyoti P.Sabale, Sneha S.Parpelli e Nilesh Sambhe: *Mitigating denial-of-service attacks using secure service overlay model.* International Journal of Engineering Trends and Technology, 8:479–483, fevereiro 2014. 9

[20] Weiler, N.: *Honeypots for distributed denial-of-service attacks.* Em *Proceedings. Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, páginas 109–114, 2002. 10

[21] *Dns, load balancing and ddos attacks.* "https://kemptechnologies.com/blog/load-balancing-and-ddos-attacks/". 10

[22] *Snort frequently asked questions.* "https://www.snort.org/faq". 10

[23] Kaur, Parneet, Manish Kumar e Abhinav Bhandari: *A review of detection approaches for distributed denial of service attacks.* Systems Science & Control Engineering, 5(1):301–320, 2017. "https://doi.org/10.1080/21642583.2017.1331768". 11

[24] "Osanaiye, O., Cai H. Choo KK.R. et al.": *Ensemble-based multi-filter feature selection method for ddos detection in cloud computing. j wireless com network.* EURASIP Journal on Wireless Communications and Networking, 2016, 2016. 11

[25] *What is machine learning?* "https://www.ibm.com/cloud/learn/machine-learning". 11, 12

[26] *Supervised and unsupervised machine learning algorithms.* "https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/". 12

[27] *Unsupervised machine learning: What is, algorithms, example.* "https://www.guru99.com/unsupervised-machine-learning.html#7". 12

[28] *What is supervised learning.* "https://www.ibm.com/cloud/learn/supervised-learning". 13

[29] *Decision tree algorithm, explained.* "https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html#:~:text=Decision%20trees%20use%20multiple%20algorithms,homogeneity%20of%20resultant%20sub%2Dnodes.&text=The%20decision%20tree%20splits%20the,in%20most%20homogeneous%20sub%2Dnodes.". 13

[30] *The indecisive decision tree — story of an emotional algorithm (1/2) | by jay trivedi | towards data science.* "https://towardsdatascience.com/the-indecisive-decision-tree-story-of-an-emotional-algorithm-1-2-8611eea7e397#:~:text=Decision%20tree%20is%20unstable%20because,importance%20order%20to%20change%20significantly.". 13

[31] *Understanding random forest. how the algorithm works and why it is... | by tony yiu | towards data science.* "https://towardsdatascience.com/understanding-random-forest-58381e0602d2". 14

[32] *An introduction to support vector machines (svm) (monkeylearn.com).* "https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/#:~:text=A%20support%20vector%20machine%20(SVM,able%20to%20categorize%20new%20text.". 14

[33] *Support vector machines(svm) — an overview | by rushikesh pupale | towards data science.* "https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989". 14

[34] Apsemidis, Anastasios, Stelios Psarakis e Javier M. Moguerza: *A review of machine learning kernel methods in statistical process monitoring.* Computers Industrial Engineering, 142:106376, 2020, ISSN 0360-8352. https://www.sciencedirect.com/science/article/pii/S0360835220301108. 14

[35] *What is deep learning? (machinelearningmastery.com).* "https://machinelearningmastery.com/what-is-deep-learning/". 14

[36] *What is deep learning and how does it work? | towards data science.* "https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac". 15

[37] *Learning process of a deep neural network | by jordi torres.ai | towards data science.* "https://towardsdatascience.com/learning-process-of-a-deep-neural-network-5a9768d7a651". 15

[38] *Classification: Accuracy.* https://developers.google.com/machine-learning/crash-course/classification/accuracy. 16

[39] *Classification: Precision and recall.* https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall. 17

[40] *Weka (machine learning).* https://en.wikipedia.org/wiki/Weka_(machine_learning). 20

Anexo I

Tabelas relacionadas ao NSL_KDD

Ataque	Treino		Teste	
apache2		0.00%	737	3.27%
back	956	0.77%	359	1.59%
land	18	0.01%	7	0.03%
mailbomb		0.00%	293	1.30%
neptune	41214	33.20%	4657	20.66%
pod	201	0.16%	41	0.18%
processtable		0.00%	685	3.04%
smurf	2646	2.13%	665	2.95%
udpstorm		0.00%	2	0.01%
worm		0.00%	2	0.01%
Total	45035	36.28%	7448	33.04%

Tabela I.1: Ataques DoS presentes no NSL_KDD.

Ataque	Treino		Teste	
ipsweep	3599	2.90%	141	0.63%
mscan		0.00%	996	4.42%
nmap	1493	1.20%	73	0.32%
portsweep	2931	2.36%	157	0.70%
saint		0.00%	319	1.42%
satan	3633	2.93%	735	3.26%
Total	11656	9.39%	2421	10.74%

Tabela I.2: Ataques probe presentes no NSL_KDD.

Ataque	Treino		Teste	
buffer_overflow	30	0.02%	20	0.09%
loadmodule	9	0.01%	2	0.01%
perl	3	0.00%	2	0.01%
ps		0.00%	15	0.07%
rootkit	10	0.01%	13	0.06%
sqlattack		0.00%	2	0.01%
xterm		0.00%	13	0.06%
Total	52	0.04%	67	0.30%

Tabela I.3: Ataques U2R presentes no NSL_KDD.

Ataque	Treino		Teste	
ftp_write	8	0.01%	3	0.01%
guess_passwd	53	0.04%	1231	5.46%
httptunnel		0.00%	133	0.59%
imap	11	0.01%	1	0.00%
multihop	7	0.01%	18	0.08%
named		0.00%	17	0.08%
phf	4	0.00%	2	0.01%
sendmail		0.00%	14	0.06%
snmpgetattack		0.00%	178	0.79%
snmpguess		0.00%	331	1.47%
spy	2	0.00%	0	0.00%
warezclient	890	0.72%	0	0.00%
warezmaster	20	0.02%	944	4.19%
xlock		0.00%	9	0.04%
xsnoop		0.00%	4	0.02%
Total	995	0.80%	2885	12.80%

Tabela I.4: Ataques R2L presentes no NSL_KDD.

No.	Feature Name		
1	Duration	22	is_guest_login
2	protocol type	23	count
3	service	24	srv_count
4	flag	25	serror_rate
5	src_bytes	26	srv_serror_rate
6	dst_bytes	27	rerror_rate
7	land	28	srv_rerror_rate
8	wrong_fragment	29	same_srv_rate
9	urgent_	30	diff srv_rate
10	hot	31	srv diff host_rate
11	num_failed logins	32	dst_host_count
12	logged_in	33	dst_host_srv_count
13	num_compromised	34	dst_host_same_srv rate
14	root_shell	35	dst_host_diff_srv_rate
15	su_attempted	36	dst_host_same_src_port_rate
16	num_root	37	dst host_srv_diff_host_rate
17	num_file_creations	38	dst_host_serror_rate
18	num_shells	39	dst_host_srv_serror_rate
19	num_access_files	40	dst_host_rerror_rate
20	num_outbound_cmds	41	dst_host_srv_rerror_rate
21	is_host_login		

Tabela I.5: Features do NSL_KDD.