UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Detecção preditiva de anomalias em redes de computadores com utilização de aprendizagem de máquina.

Domingos Sávio Faria Paes

Itajubá, 22 de março de 2023

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Domingos Sávio Faria Paes

Detecção preditiva de anomalias em redes de computadores com utilização de aprendizagem de máquina.

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação.

Área de Concentração: Sistemas de Computação

Orientador: Prof. Dr. Bruno Guazzelli Batista
Coorientador: Prof. Dr. Carlos Henrique Valério de Moraes

22 de março de 2023
Itajubá

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Detecção preditiva de anomalias em redes de computadores com utilização de aprendizagem de máquina.

Domingos Sávio Faria Paes

Dissertação aprovada por banca examinadora em 27 de Fevereiro de 2023, conferindo ao autor o título de **Mestre em Ciência e Tecnologia da Computação.**

Banca Examinadora:
Prof. Dr. Rafael de Magalhães Dias Frinhani - UNIFEI
Prof. Dr. Lourenço Alves Pereira Junior - Instituto Tecnológico de Aeronáutica

Itajubá
2023

Domingos Sávio Faria Paes

Detecção preditiva de anomalias em redes de computadores com utilização de aprendizagem de máquina/ Domingos Sávio Faria Paes. – Itajubá, 22 de março de 2023-

100 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Bruno Guazzelli Batista

Dissertação (Mestrado)
Universidade Federal de Itajubá - UNIFEI
Programa de pós-graduação em CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO, 22 de março de 2023.

1. Redes de Computadores. 2. Aprendizado de Máquina. I. Bruno Guazzelli Batista. II. Carlos Henrique Valério de Moraes. III. Universidade Federal de Itajubá. IV. Mestre em Ciência e Tecnologia em Computação

CDU 07:181:009.3

Domingos Sávio Faria Paes

Detecção preditiva de anomalias em redes de computadores com utilização de aprendizagem de máquina

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação.

Trabalho aprovado. Itajubá, 27 de Fevereiro de 2023:

Prof. Dr. Bruno Guazzelli Batista
Orientador

Prof. Dr. Carlos Henrique Valério de Moraes
Coorientador

Prof. Dr. Rafael de Magalhães Dias Frinhani - UNIFEI

Prof. Dr. Lourenço Alves Pereira Junior - Instituto Tecnológico de Aeronáutica

Itajubá
22 de março de 2023

Agradecimentos

Agradeço a Deus e a minha família por todo o apoio, principalmente a minha esposa Mila, por todo o esforço e dedicação em cuidar de todos nós. Seu amor e companheirismo permitiu que eu pudesse me dedicar aos estudos, desde a graduação até a conclusão deste trabalho.

Agradeço também aos meus filhos Samuel, Mariana e Rafael por todo amor e carinho, principalmente pela paciência em não ter o papai em todas as brincadeiras. Eu os amo pra sempre!

A minha mãe Luzia, pela vida e por todos os ensinamentos que me fizeram a pessoa que sou hoje e a meu pai José*(in memorian)*, um entusiasta da tecnologia que sempre nos colocou em contato com o conhecimento, obrigado por me apresentar ao TK90X aos 9 anos de idade.

Não poderia deixar de agradecer também aos meus orientadores Bruno Guazzelli e Carlos Henrique Valério. Ao Bruno pela paciência e Carlos por toda a ajuda e disponibilidade, não poupando esforços e estando sempre disposto a colaborar, tenho certeza que estaremos juntos em outros trabalhos.

Aos colegas de trabalho, Adler, Anderson, Cláudio, Everaldo, Jader, João Martinho, Luiz Antônio e Pablo pelo companheirismo e apoio durante essa jornada, além da Diretoria de Tecnologia da Informação da Unifei pelas informações utilizadas nos experimentos.

Obrigado!

"Ensinar é um exercício de imortalidade. De alguma forma continuamos a viver naqueles cujos olhos aprenderam a ver o mundo pela magia da nossa palavra. O professor, assim, não morre jamais..."

(Rubem Alves)

Resumo

Com a dependência cada vez maior das tecnologias no dia a dia, fica evidente a preocupação em se manter as infraestruturas que dão suporte ao seu funcionamento, garantindo assim uma boa experiência ao usuário final. Assim, os ataques de negação de serviço, estão entre as principais causas de anomalias em redes de computadores, podendo causar uma degradação ou até mesmo a interrupção dos serviços. Nesse contexto, a aplicação de novas tecnologias, como a inteligência artificial ou o aprendizado de máquina, se tornam cada vez mais necessárias, para garantir mais agilidade na detecção dos problemas diminuindo seus impactos. Dessa forma, esse trabalho apresenta uma análise entre diferentes métodos de aprendizagem de máquina supervisionado classificador, aplicados aos dados coletados em equipamentos de rede, do tipo *switch*, de forma a detectar anomalias na infraestrutura de redes de uma instituição de ensino superior. Os métodos de aprendizado de máquina utilizados neste trabalho foram: *Decision Tree*, *Random Forest*, *Extra Tree*, *Gradient Boosting*, *Extreme Gradient Boosting* e *Histogram Gradient Boosting*. Os modelos gerados a partir desses métodos se mostraram promissores, sendo capaz de atingir resultados com 99.88% na métrica F1 Ponderada e 99,16% de Acurácia Balanceada. Outros pontos, como tempo de treinamento, tempo de predição e tamanho do arquivo de salvamento, também foram levados em consideração para a classificação do melhor método. Dada a importância das ferramentas de detecção de falhas, este trabalho contribui para a definição das melhores abordagens e assim permite que sejam elaboradas novas e mais eficientes ferramentas para esta finalidade.

Palavras-chaves: Aprendizado de Máquina, Detecção de anomalias, Redes de Computadores.

Abstract

With the increasing dependence on technologies on a daily basis, it is evident the concern to maintain the infrastructures that support its operation, thus guaranteeing a good experience for the end user. Thus, denial of service attacks are among the main causes of anomalies in computer networks, which can cause degradation or even interruption of services. In this context, the application of new technologies, such as artificial intelligence or machine learning, becomes increasingly necessary to ensure more agility in detecting problems, reducing their impacts. Thus, this work presents an analysis between different methods of classifier supervised machine learning, applied to data collected fromnetwork equipment, of the switch type, in order to detect anomalies in the network infrastructure of a higher education institution. The machine learning methods used in this work were: Decision Tree, Random Forest, Extra Tree, Gradient Boosting, Extreme Gradient Boosting and Histogram Gradient Boosting. The models generated from these methods showed promise, being able to achieve results with 99.88% in the Weighted F1 metric and 99.16% of Balanced Accuracy. Other points, such as training time, prediction time and save file size, were also taken into account for the classification of the best method. Given the importance of fault detection tools, this work contributes to the definition of the best approaches and thus allows the development of new and more efficient tools for this purpose.

Key-words: Anomaly detection, Computer Network, Machine Learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ACK	*Acknowledge*
ACL	*Access Control List*
ANN	*Artificial Neural Network*
BOOTP	*Bootstrap Protocol*
BPDU	*Bridge Protocol Data Units*
BPS	*Bits Per Second*
CAM	*Content Addressable Memory*
CBPDU	*Configuration Bridge Protocol Data Units*
CDP	*Cisco Discover Protocol*
CISA	*Cybersecurity and Infrastructure Security Agency*
CPU	*Central Processing Unit*
DDoS	*Distributed Denegation of Service*
DHCP	*Dynamic Host Configuration Protocol*
DMZ	*Demilitarized Zone*
DNS	*Domain Name System*
DoS	*Denegation of Service*
DRDoS	*Distributed Reflective Denegation of Service*
FIN	*Finalize*
Gbit/s	Gigabits por Segundo
GPU	*Graphics Processing Unit*
HTML	*Hyper Text Markup Language*
HTTP	*Hypertext Transfer Protocol*
ICMP	*Internet Control Message Protocol*
IDS	*Intrusion Detection System*
IOS	*Internetwork Operating System*
IoT	*Internet of Things*
IP	*Internet Protocol*
IPS	*Intrusion Prevention System*
KNN	*K-Nearest Neighbors*
LAN	*Local Area Network*
LLDP	*Link Layer Discover Protocol*
MAC	*Media Access Control*
MIB	*Management Information Base*
MTU	*Maximum Transmission Unit*
NGFW	*Next Generation Firewall*

NIDS	*Network Intrusion Detection System*
NTP	*Network Time Protocol*
OSI	*Open System Interconnection*
PCA	*Principal Component Analysis*
PDU	*Protocol Data Units*
PPS	*Packets Per Second*
QDA	*Quadratic Discriminant Analysis*
QoS	*Quality of Service*
RF	*Random Forest*
RFC	*Request for Comments*
RMON	*Remote Network Monitoring*
RST	*Reset*
RSTP	*Rapid Spanning Tree Protocol*
SCT	*Secure Core Technology*
SDN	*Software Defined Network*
SFP	*Small Form-factor Pluggable transceiver*
SNMP	*Simple Network Management Protocol*
SOHO	*Small Office Home Office*
STP	*Spanning Tree Protocol*
SYN	*Synchronize*
TCN	*Topology Change Notification*
TCP	*Transmission Control Protocol*
TFTP	*Trivial File Transfer Protocol*
TIC	Tecnologia da Informação e Comunicação
TLV	*Type-Length-Value*
UDP	*User Datagram Protocol*
VoIP	*Voice over Internet Protocol*
WAN	*Wide Area Network*

Sumário

1 Introdução

Em tempos atuais, a tecnologia ganha cada vez mais espaço e importância na vida da população mundial. Durante a pandemia de Covid-19 muitas empresas passaram a operar com a maior parte dos funcionários trabalhando de maneira remota. Essa situação se espalhou por muitos ramos de atividades, como na área médica, através do atendimento remoto ou tele-consultas, na área educacional desde a educação básica até o ensino superior, através do aumento substancial dos cursos a distância, incluindo os cursos presenciais. O governo também passou a oferecer mais serviços, antes exclusivamente presenciais, que agora podem ser feitos online.

No Brasil, um dos países mais afetados pela pandemia houve uma intensificação no uso das tecnologias e de acordo com a Agência Brasil[5], de 2019 para 2020, houve um salto de 61,8 milhões de domicílios com algum acesso à Internet. Assim, alguns dos desafios enfrentados para fazer frente a esse cenário, passaram pela expansão das infraestruturas com um aumento exponencial da capacidade dos servidores e das redes de comunicação.

Dessa forma, passaram a atender um número muito maior de usuários e também a alta demanda por performance das conexões de dados, o ponto de troca de tráfego de São Paulo (PTT-SP) chegou ao primeiro lugar no mundo em volume de tráfego, ultrapassando o PTT de Frankfurt.

De acordo com o site CISO Advisor[6], percebeu-se também o crescimento no número de cibercrimes, com o surgimento de vários grupos organizados realizando ataques, principalmente de *ransomware*, cobrando resgates para devolverem as informações que foram criptografadas, ou para não divulgarem os dados pessoais dos usuários.

Dados estes, que deveriam estar protegidos para atender a legislação vigente, como a LGPD[1] no Brasil ou ainda a GDPR[2] na Europa.

As fraudes também têm aumentado no setor financeiro e comercial, principalmente os bancos e comércio eletrônico têm sofrido bastante com o aumento das ocorrências de *Phishing*[9]. Segundo o CERT.br[10], no ano de 2020, 85,15% das fraudes envolvendo *Phishing* tiveram motivações financeiras. Porém, o governo também é impactado. A exemplo disso a Polícia Federal (PF) deflagrou em novembro de 2021 uma operação de combate a fraudes envolvendo o pagamento do auxílio emergencial, pago por um banco público, da ordem de dez milhões de reais[11].

Ainda nesse contexto, percebeu-se um aumento dos ataques de Negação de Serviço DoS (*Denegation of Service*). Segundo Moreno(2019)[12], esses ataques consistem na

[1] LGPD - Lei Geral de Proteção de Dados do Brasil. Lei nº13.709/2018[7]
[2] GDPR - *General Data Protection Regulation.* 2016/679 EEUU[8]

realização de um grande volume de acessos a um determinado servidor, serviço ou site, fazendo com que este receba uma quantidade de acessos muito maior que o normal ou maior que sua capacidade de responder. Para tal, os atacantes utilizam hosts infectados por vírus ou *exploits*[9], fazendo com que ele se torne um "zumbi". Uma vez contaminado, esse host passa a executar comandos determinados por um *master*, ou seja, o computador que irá orquestrar o ataque realizado pelos zumbis. Porém, não são só computadores que podem ser contaminados e se tornarem zumbis. Alguns ataques já foram feitos utilizando câmeras IP (*Internet Protocol*), telefones VoIP (*Voice over Internet Protocol*) e roteadores domésticos, que apresentavam alguma falha de segurança e estavam acessíveis na Internet. Essas redes de zumbis conhecidas como *Botnet's*[9], recebem comandos para dispararem contra um alvo causando uma sobrecarga nos serviços, fazendo com que os mesmos fiquem mais tempo fora do ar ou tenham sua performance prejudicada.

Alguns desses ataques são motivados por *ciberativismo*, que segundo Silveira(2010)[13], pode ser descrito como: "um conjunto de práticas em defesa de causas políticas, socioambientais, sociotecnológicas e culturais, realizadas nas redes cibernéticas, principalmente na Internet". A esse respeito, a BBC News Brasil[14] noticiou que o grupo de ciberativistas Anonymous realizou ataques a sites e sistemas do governo Russo em retaliação a guerra contra a Ucrânia, utilizando ataques de Negação de Serviço Distribuído DDoS (*Distributed Denegation of Service*), o que fez com que vários sites ficassem fora do ar por horas e até dias.

Dessa forma, todos os esforços feitos para detectar e mitigar os problemas que possam ocorrer em uma rede de dados são válidos, visto que a parada destes serviços de infra-estrutura cruciais para o funcionamento de diversas instituições nos mais diversos ramos de atuação, podem levar a prejuízos incalculáveis. No caso de instituições financeiras, de comércio eletrônico, de logística, entre outras, os prejuízos podem ser econômicos, uma vez que seus clientes podem deixar de utilizar seus serviços por conta da indisponibilidade. Já em um outro âmbito como o educacional, justiça, saúde e governos de maneira geral, os prejuízos são de cunho operacional fazendo com que os usuários fiquem sem acesso aos serviços públicos prestados por essas entidades.

Assim, segundo Kurose e Ross[3], o gerenciamento de falhas tem por objetivo detectar, reagir e registrar as falhas do sistema, que aponta também para uma possível confusão entre desempenho e falha. Pode-se dizer que o desempenho compreende uma condição de anomalia em que o sistema ainda está funcionando, mas fora das condições normais esperadas. Já com relação à falha, o dispositivo está fora de operação.

Então, pode-se dizer que uma parte das falhas na infraestrutura pode ser detectada, permitindo atuar de maneira a evitar interrupções, ou mesmo diagnosticando de maneira rápida a causa da interrupção para que o serviço seja reestabelecido com agilidade.

1.1 Motivação

Para o estudo proposto neste trabalho, a infraestrutura de rede de uma instituição de ensino de âmbito federal foi considerada. Ela oferece 34 cursos de graduação e 23 cursos de pós graduação, onde hoje são atendidos cerca de 7200 alunos entre o campus sede e avançado. Além disso, ela conta com 493 docentes e 468 funcionários entre servidores técnicos administrativos e funcionários públicos.

Assim, para que as atividades corriqueiras transcorram dentro da normalidade, tanto os docentes quanto os funcionários precisam ter acesso a alguns serviços básicos oferecidos pela infraestrutura de TIC (Tecnologia da Informação e Comunicação).

Hoje a instituição conta com cerca de 250 servidores, hospedados em seu data-center, além de equipamentos de conectividade à internet e telefonia. Estes serviços são essenciais para as tarefas do dia a dia, onde os usuários acessam os sistemas administrativos, e-mails, sites, redes sociais e plataformas de apoio ao ensino, que é a finalidade principal da instituição. Além das atividades administrativas, a instituição ainda mantém uma página web com funcionalidades voltadas aos alunos e a comunidade em geral que possa estar interessada nas informações veiculadas.

Assim, para que tudo transcorra da maneira correta a instituição deve manter, através de sua equipe de TIC, uma grande infraestrutura de redes de computadores composta por quase 300 *switchs* em operação, entre eles *switchs* de acesso, de distribuição e núcleo, além de roteadores, *firewalls* e de servidores DNS (*Domain Name System*), DHCP (*Dynamic Host Configuration Protocol*), NTP (*Network Time Protocol*). Todo esse complexo tecnológico ainda envolve atividades acessórias como o controle biométrico para acesso aos espaços físicos, como salas de aula e auditórios, câmeras IP para o video monitoramento do campus de maneira a apoiar a equipe de segurança, além de mais de 300 pontos de acesso sem fio à internet, que atende tanto alunos como professores do campus, bem como visitantes de outras instituições nacionais e internacionais, através da rede integrada Eduroam[3].

Em se tratando de uma estrutura desse porte, problemas são quase inevitáveis e podem ter as causas mais diferentes. Por diversos motivos não há como garantir que todos os equipamentos estejam seguros o tempo todo, pois por mais protegida que seja uma rede não há garantias de que alguém ou algum dispositivo da rede possa ter sido vítima de um ataque de Dia Zero[15] (do inglês *Zero Day*).

Inevitavelmente, nos casos em que há equipamentos contaminados sendo utilizados por uma *botnet* para realizar um ataque de negação de serviço, não só a vítima do ataque é prejudicada, mas a origem do ataque também sofre, como mostra a Figura 1.

[3] Eduroam - É uma rede de roaming internacional que provê acesso à pesquisadores, professores e alunos de qualquer instituição participante. <https://eduroam.org/what-is-eduroam>

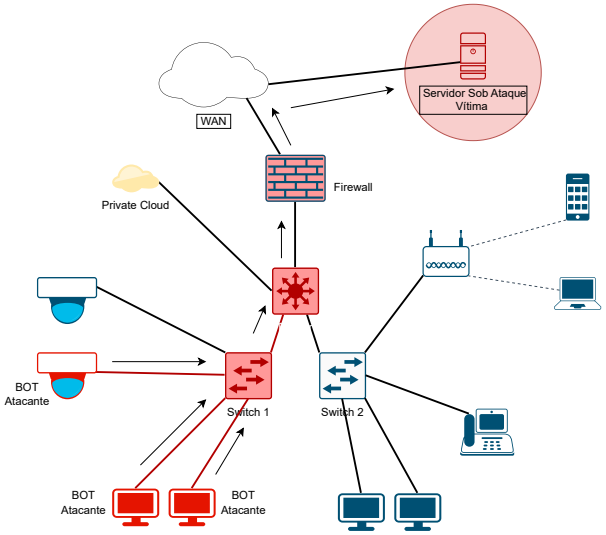


Figura 1 – *Infraestrutura com Falhas*

Observe que quando um ou mais equipamentos na rede estão sendo utilizados por uma *botnet* para atacar um alvo externo ou interno, ocorre a utilização de largura de banda de maneira espúria fazendo com que haja uma lentidão na rede interna em razão do aumento do fluxo, e isso faz com que todos os usuários dessa infraestrutura sejam prejudicados.

Já quando o ataque vem de fora da infraestrutura local, como da WAN (*Wide Area Network*) por exemplo, o *firewall* exerce um papel fundamental no sentido de realizar a filtragem do tráfego, o que quase sempre é suficiente para evitar a degradação do serviço por um ataque de pequeno porte. Porém, quando o ataque utiliza um fluxo de dados superior à largura de banda disponível, não há o que ser feito internamente, restando unicamente a possibilidade de contratar um serviço de provedor com capacidade de mitigar o ataque realizando uma filtragem, de modo que este não atinja a rede interna. Essa solução nem sempre é viável economicamente.

A principal motivação deste trabalho, advém da ocorrência recorrente na instituição de falhas ligadas exatamente à dispositivos que na maioria das vezes estavam infectados por vírus ou *exploits*, e que enviavam uma enorme quantidade de dados à rede. Como o *switch* de acesso, no qual o dispositivo está ligado, normalmente tem um *throughput* suficientemente alto e capaz de encaminhar os pacotes, não ocorre seu travamento e o tráfego é despejado no *switch* de distribuição através do *uplink*. Quando este tráfego chega ao *switch* de distribuição, se junta ao tráfego de outros *switchs* de acesso e este montante é suficientemente grande para causar uma falha ou degradação do serviço a ponto dos usuários perceberem que algo anormal está ocorrendo. Por vezes esse problema acaba atingindo um *switch* de núcleo, que está em um nível mais alto na hierarquia da infraestrutura de rede, algumas vezes concentrando os dados de mais de 100 *switchs* de acesso e milhares de dispositivos conectados, vide a Figura 2.

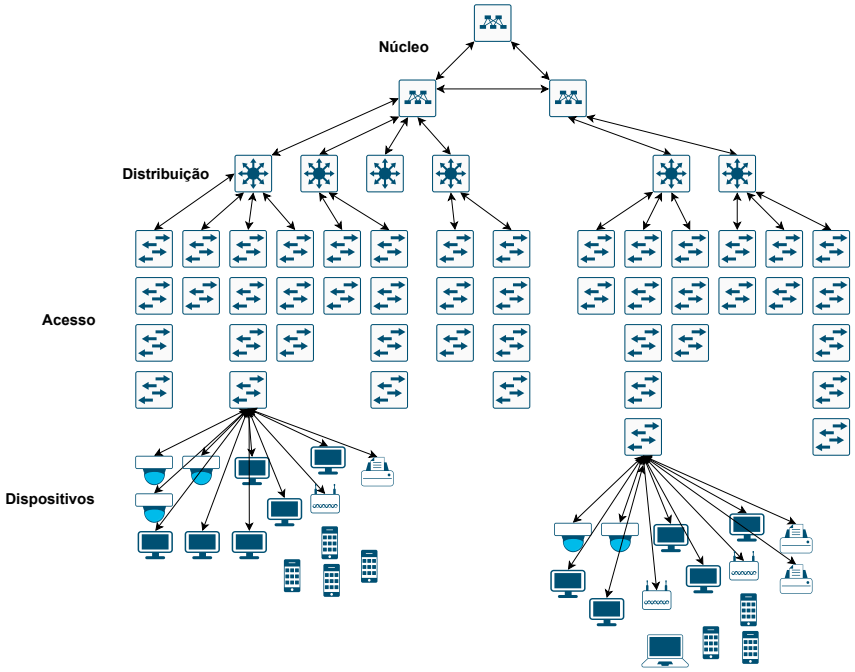


Figura 2 – *Infraestrutura por Camadas*

Levando em conta que cada *switch* de acesso tenha 24 portas, que é o usual na instituição, então um *switch* de núcleo está concentrando os pacotes de mais de 2400 dispositivos. Assim, a única forma de mitigar a falha, seria desligar as interfaces que atendem os *switchs* de distribuição, e ir religando uma a uma, até que se percebesse um aumento significativo do tráfego. Ali há um indicativo de qual pode ser o ramo da rede causador da falha. Quando esta causa é confirmada, a interface deste *switch* de distribuição é então ligada com as outras interfaces do *switch* de distribuição desligadas e o processo se repete até que se descubra qual é o *switch* de acesso onde está ligado o dispositivo com problemas. Descoberto o *switch* de acesso, então as portas são investigadas para que se chegue ao dispositivo ou *host* causador do problema.

Esse processo manual e penoso pode durar várias horas até que se descubra a causa raiz, levando a prejuízos operacionais enormes, e consumindo boa parte do tempo da equipe de campo.

1.1.1 Hipótese

Diante do exposto, a proposta deste trabalho, visa responder a seguinte questão: "Seria possível identificar/detectar anomalias nos dados advindos de equipamentos conectados aos *switchs*?"

Para responder a esta questão, este trabalho pretende validar que: "Ao aplicar uma variação do trabalho de Boyar, Özen e Metin[16], é possível identificar/detectar anomalias utilizando os dados das MIB's SNMP/RMON, obtidos dos *switchs* de acesso, submetendo-os a algoritmos de aprendizado de máquina supervisionado classificador".

1.2 Objetivos

O objetivo deste trabalho consiste em realizar um estudo a respeito da viabilidade da aplicação de técnicas de aprendizado de máquina aos dados, disponíveis nos *switchs* em uso na instituição, de forma a detectar os problemas advindos de ataques capazes de causar anomalias, levando a degradação ou interrupção das atividades que envolvem a infraestrutura de redes.

Dessa forma esse objetivo pode ser dividido em objetivos específicos como:

- A construção de modelos de aprendizado de máquina capazes de detectar as diferentes características das anomalias, com base nos dados disponíveis.
- Analisar de maneira qualitativa os diferentes métodos aplicados de forma a escolher o mais adequado para solução deste problema.

Entre as contribuições desta pesquisa estão: i) a aplicação de uma metodologia para detecção de falhas, e ou anomalias em *switchs* de acesso; ii) a análise comparativa dos diferentes métodos para a elaboração do modelo a ser utilizado; iii) a consolidação de um modelo capaz de detectar tais problemas de maneira eficiente, de forma que possa ser aplicado a instituição.

1.3 Organização do Documento

Nesse capítulo foram apresentadas as motivações que levaram ao desenvolvimento deste trabalho. Os capítulos seguintes estão organizados da seguinte forma:

- **Capitulo 2 - Referencial Teórico** Apresenta o embasamento teórico sobre o assunto, bem como as características intrínsecas à detecção de falhas em uma infraestrutura de rede local de grande porte.
- **Capitulo 3 - Aprendizado de Máquina** Apresenta um referencial teórico sobre as abordagens de aprendizado de máquina supervisionado classificador, além de uma revisão bibliográfica sobre os trabalhos mais relevantes quanto a detecção de falhas ou anomalias em redes de computadores.
- **Capitulo 4 - Metodologia** Apresenta as metodologias utilizadas para à aplicação das técnicas de aprendizado de máquina aos dados desta pesquisa, desde a coleta dos dados, a transformação, integração, análise e definição das bases de treinamento e testes. Apresenta ainda os métodos de aprendizado de máquina empregados e suas definições.

- **Capitulo 5 - Experimentos e Discussões** Apresenta a definição dos experimentos e os resultados obtidos quando da sua aplicação, bem como uma discussão a respeito.

- **Capitulo 6 - Conclusão** Apresenta uma conclusão a respeito do problema abordado, visando responder a questão principal deste trabalho explanada no Capítulo 1 e ainda propor outras ideias a respeito de trabalhos que podem ser realizados no futuro.

2 Referencial Teórico

Este capítulo abordará a importância da detecção de falhas em redes de computadores, além dos tipos mais comuns e alguns mecanismos importantes para essa finalidade. Em seguida serão apresentados algumas ferramentas relacionados ao tema.

2.1 Considerações Iniciais

Com o crescimento do número de usuários e das aplicações cada dia mais presentes no dia a dia das instituições, cresce também a importância da infraestrutura que suporta esse conjunto. Assim, as redes de computadores são uma peça fundamental para a viabilização do acesso a essas funcionalidades e facilidades oferecidas, aqui tratadas como serviços aos usuários.

Do ponto de vista de quem oferece os serviços, estão os servidores que proveem aplicações como páginas Web, servidores de arquivos, e-mails, entre outros. Por outro lado, esses servidores dependem de outros equipamentos para que se conectem aos usuários como roteadores e *switchs* e também dependem de outros servidores que proveem o acesso às redes como servidores DHCP (*Dynamic Host Configuration Protocol*), DNS (*Domain Name System*), NTP (*Network Time Protocol*), etc. Além dessa infraestrutura também é necessário prover segurança para que se garanta que não haverá invasões ou que as ocorrências sejam minimizadas e mitigadas o mais rápido possível, entrando em cena os *firewalls*, IDS (*Intrusion Detection System*), NIDS (*Network Intrusion Detection System*) e IPS (*Intrusion Prevention System*).

A medida que os serviços oferecidos aos usuários se tornam mais importantes e indispensáveis, também se tornam alvos de pessoas mal intencionadas e dispostas a realizar ataques contra essas instituições e empresas para que de alguma forma possam lucrar com isso.

No que tange as preocupações da instituição em questão, uma delas está relacionada às falhas causadas em sua infraestrutura de redes, principalmente *switchs* que proveem o acesso aos usuários internos. Usuários internos são todas as pessoas que acessam seus serviços utilizando a estrutura física de rede, seja por conexão cabeada ou por conexão sem fio.

Assim, grande parte dos problemas relacionados a degradação do desempenho em uma rede de computadores está ligada ao fluxo de pacotes. Um alto fluxo de pacotes levam os *switchs* a consumirem mais processamento de CPU (*Central Processing Unit*), além de consumir grande parte da largura de banda.

De acordo com o fabricante Versitron^{TM},[17], a largura de banda de um *switch* está relacionada às suas capacidades de chaveamento (do inglês *Switching capacity*) e taxa de encaminhamento (do inglês *Forwarding rate*) de pacotes.

Quando algum host ou vários deles são infectados por um vírus ou estão sendo utilizados como máquinas "zumbis" de uma *botnet*, por exemplo, ocorre uma mudança no padrão do fluxo de dados na porta onde o mesmo está ligado. Essa mudança no fluxo de dados é considerada anormal, uma vez que este fluxo não corresponde à utilização feita pelo próprio usuário.

Assim, as falhas podem ser caracterizadas pelo seu nível de criticidade, onde a perda de desempenho ou a degradação da rede têm um peso menor que a falha geral. Cabe ressaltar que não é objetivo deste trabalho analisar ou propor soluções para as falhas causadas por problemas externos como a falta de energia, rompimento de cabos ou danos físicos aos equipamentos.

Os tipos de falhas causadas por ataques *hacker* ou ações de vírus que visam a negação de serviço conhecidos como DoS (*Denegation of Service*) ou DDoS (*Distributed Denegation of Service*), talvez estejam entre as ocorrências, mais difíceis de serem combatidos.

De acordo com o CERT.br[18], os ataques de DoS ou DDoS têm sido utilizados como uma forma de dificultar o reestabelecimento dos serviços quando os mesmos são atacados mediante extorsão, sendo que alguns destes ataques não são executados necessariamente no servidor que provê o serviço e sim nos meios que fornecem a comunicação com a rede ou internet, como servidores DNS (*Domain Name System*), NTP (*Network Time Protocol*), além dos roteadores e *firewalls*.

As falhas tratadas neste trabalho estão relacionadas àquelas causadas por ataques de negação de serviço e suas variantes, principalmente os que fazem uso de vulnerabilidades intrínsecas ao funcionamento do protocolo DHCP pertencente a camada de aplicação, dos protocolos TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*) pertencentes a camada de transporte, do protocolo ICMP (*Internet Control Message Protocol*) pertencente a camada de rede e dos protocolos STP (*Spanning Tree Protocol*) e CDP (*Cisco Discover Protocol*) que pertencem a camada de enlace de dados.

2.2 Protocolos e suas Vulnerabilidades

Para entender melhor os ataques de negação de serviço é preciso entender o funcionamento dos protocolos utilizados para a realização dos mesmos. A seguir são detalhados os protocolos relacionados a este trabalho bem como suas características e importância.

2.2.1 TCP

O protocolo TCP é um protocolo orientado à conexão. Como explica Kurose and Ross[3], esse protocolo obriga que as partes envolvidas façam uma apresentação também conhecida como *3-Way Handshake*, e seu esquema pode ser visto na Figura 3.

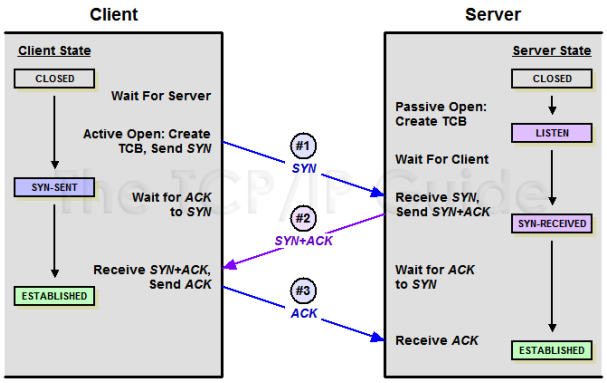


Figura 3 – *Three Way Handshake*
fonte: https://www.myfaqbase.com/
Acessado em 04/04/2022

Seu funcionamento têm início quando um *host* tenta estabelecer uma conexão com um servidor, assim, ele envia um pacote SYN (*Synchronize*) ao servidor que por sua vez responde com um pacote SYN-ACK ao remetente e este confirma com um pacote ACK (*Acknowledge*). Esse processo faz com que após essa troca de mensagens, os envolvidos estabeleçam algumas variáveis de controle utilizadas para que se garanta o desempenho e a entrega dos pacotes entre os hosts[3]. Conforme a RFC 793[1], o cabeçalho TCP possui os campos conforme ilustrado na Figura 4.

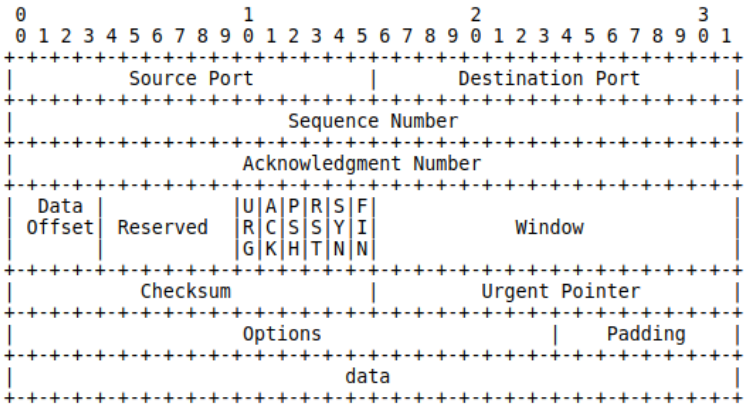


Figura 4 – *TCP Header*[1]

Com relação às explorações feitas por atacantes, as *flags* que compõe o cabeçalho têm uma grande importância quanto a forma como o alvo irá tratar o pacote. As *flags* SYN, ACK, RST e FYN estão entre as mais importantes e utilizadas.

De acordo com a empresa Cloudflare™[19] especialista em mitigação de ataques, a exploração da vulnerabilidade do protocolo TCP no ataque *TCP-SYN Flood*, ocorre quando o atacante envia uma grande quantidade de pacotes com a *flag* SYN, utilizando uma técnica de falsificação dos endereços IP (do inglês *spoofing*), dos remetentes. Dessa forma os destinatários dos pacotes de resposta SYN+ACK, enviados pelo alvo, nunca serão respondidos com o ACK final, que faria com que fosse estabelecida uma conexão legítima, forçando assim o alvo a aguardar por uma resposta que nunca chegará. Isso faz com que o servidor alvo mantenha pelo período de *timeout* a conexão no seu *buffer*, e caso esse *buffer* se esgote as conexões com os usuários legítimos são negadas, efetivando o sucesso do ataque.

2.2.2 UDP

O protocolo UDP, diferentemente do TCP, é um protocolo minimalista e também segundo Kurose and Ross[3], caracteriza-se por não ser confiável e não ser orientado à conexão. Não é confiável, no sentido de que não há garantias de entrega dos pacotes entre a origem e o destino. Assim, o protocolo UDP não faz o estabelecimento prévio de uma conexão, e os pacotes após receberem os campos inerentes ao protocolo, são simplesmente encaminhados à camada de rede para que possam ser enviados ao destino. Esses pacotes contam apenas com alguns mecanismos simples de verificação de erros, que permitem verificar se os pacotes recebidos pelo host de destino chegaram íntegros.

O protocolo UDP foi estabelecido pela RFC 768[20], que data de 1980, sendo bastante utilizado nos casos em que não há uma necessidade de garantias de entrega, como nos *streamings* de video ou telefonia VoIP por exemplo, sendo utilizado também para prover o transporte para os protocolos DNS, DHCP, SNMP, TFTP, além de outros, e possui um datagrama bastante simples conforme mostra a Figura 5.

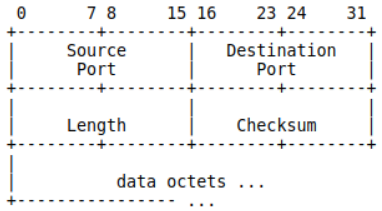


Figura 5 – *UDP Header*
fonte: IETF - RFC 768

Segundo o artigo *UDP-Based Amplification Attacks*[21], publicado no site da CISA (*Cybersecurity and Infrastructure Security Agency*)[1], o UDP não realiza nenhuma verificação prévia do campo de endereço IP do remetente, ficando a cargo da aplicação fazer isso, então é possível enviar uma solicitação a um serviço informando o IP da vítima como

[1] CISA - Agência do Governo dos EEUU que trata de Segurança Cibernética e Infraestrutura

destino da resposta. Como alguns serviços respondem muito mais dados do que o pacote original, este ataque acaba refletindo na vítima com uma carga maior, amplificando o ataque. Na Tabela 2[21], é possível verificar que alguns dos serviços mais comuns a utilizar UDP são capazes de refletir pacotes a uma taxa muito superior ao pacote inicial.

Tabela 2 – Fatores de Amplificação de ataques DDoS usando UDP

Protocolo	**Fator de Amplificação**
DNS	28 a 54
NTP	556.9
SNMPv2	6.3
CharGEN	358.8
mDNS	2 a 10
RIPv1	131.24
LDAP	46 a 55
TFTP	60
Memcached	10.000 a 51.000
WS-Discovery	10 a 500

Segundo o CERT.br [2] que é mantido pelo NIC.br [3], do Comitê Gestor da Internet no Brasil, até 07/2021 haviam 76.858 servidores SNMP, 73.758 servidores NTP e 46.520 servidores DNS com falhas que possibilitavam um ataque de DRDoS (*Distributed Reflective Denegation of Service*), conforme mostra a Figura 6.

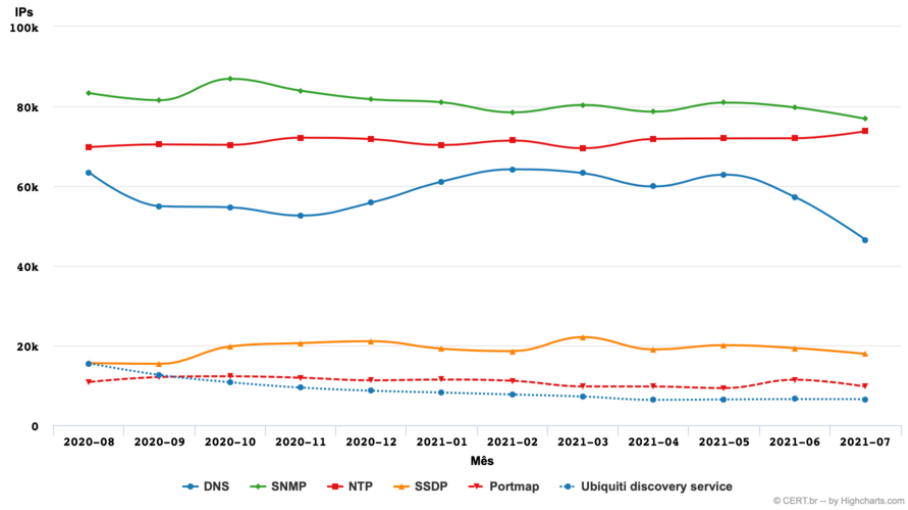


Figura 6 – *Serviços abertos permitindo amplificação*
fonte: CERT.br - https://www.cert.br/stats/amplificadores/
Acessado em 23/03/2022

Outro dado interessante, com relação aos ataques DoS, está na Figura 7, também veiculada pelo CERT.br, que mostra a quantidade de notificações sobre equipamentos participando de ataques deste tipo. Observa-se que houve um salto gigantesco a partir de 2014, mostrando que este tipo de ataque têm sido cada vez mais frequente e com potencial de causar muitos danos, principalmente econômicos.

[2] CERT.br - Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
[3] NIC.br - Núcleo de Informação e Coordenação do Ponto BR

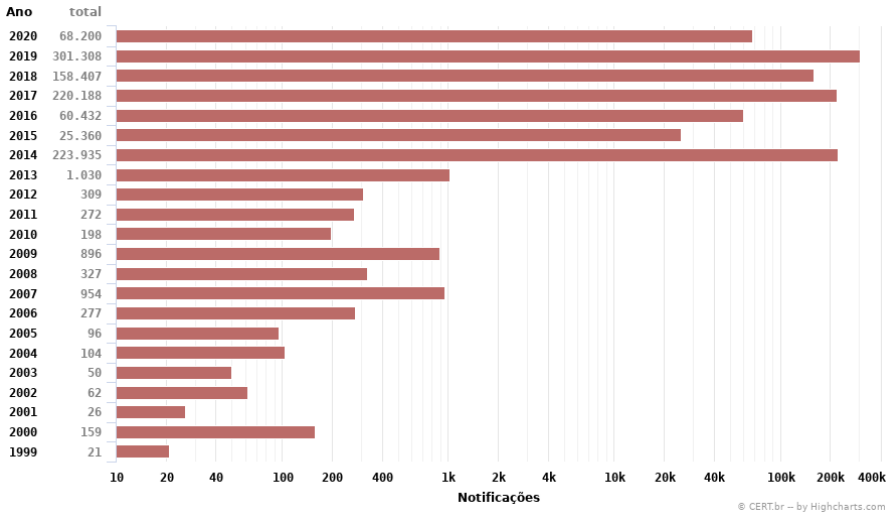


Figura 7 – *Notificações sobre Equipamentos Participando em Ataques DoS*
fonte: CERT.br - https://www.cert.br/stats/incidentes/2020-jan-dec/dos.html
Acessado em 23/03/2022

2.2.3 ICMP

Segundo White[22], o ICMP ou Protocolo de Mensagens de Controle da Internet (do inglês *Internet Control Message Protocol*) é um protocolo da camada de redes que complementa as operações envolvendo o protocolo IP. É utilizado por roteadores e *hosts* para enviar mensagens de alerta e controle quando alguma coisa acontece fora do esperado. Um exemplo é quando um pacote enviado a um servidor não consegue chegar ao destino, e pode ter ocorrido algumas situações envolvendo este pacote, como: o servidor não possuía nenhuma aplicação respondendo nessa porta, ou o servidor não estava disponível. Assim, é enviado ao *host* solicitante um pacote ICMP informando a ocorrência.

O ICMP está descrito na RFC 792[2] e possui um cabeçalho conforme a Figura 8, contendo os campos *Type* e *Code* que são utilizados para diferenciar as mensagens que se deseja enviar. O campo de dados possui um preâmbulo contendo o *Internet Header*, seguido de 64 bits de dados do datagrama que originou a mensagem.

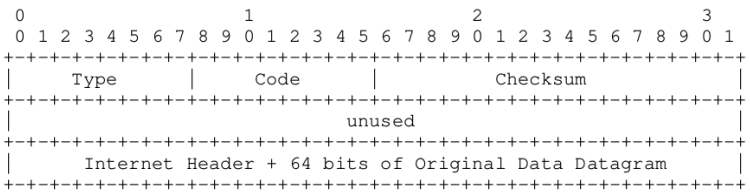


Figura 8 – *ICMP Header*[2]

Um exemplo de ataque que utiliza ICMP é o *Ping Flood* ou *ICMP Flooding*, que consiste no envio de uma grande quantidade de pacotes ECHO (*Type* 8 | *Code* 0), com endereços IP falsos, fazendo com que o alvo responda com pacotes ECHO REPLY (*Type* 0 | *Code* 0). De acordo com o CISA DDoS Quick Guide[23], um ataque deste tipo visa

consumir a largura de banda do alvo, bem como consumir recursos do *firewall* causando uma negação de serviço. Na Tabela 9, Kurose and Ross[3], apresenta um conjunto dos principais tipos e códigos das mensagens ICMP.

Tipo ICMP	Código	Descrição
0	0	resposta de eco (para *ping*)
3	0	rede de destino inalcançável
3	1	hospedeiro de destino inalcançável
3	2	protocolo de destino inalcançável
3	3	porta de destino inalcançável
3	6	rede de destino desconhecida
3	7	hospedeiro de destino desconhecido
4	0	repressão da origem (controle de congestionamento)
8	0	solicitação de eco
9	0	anúncio do roteador
10	0	descoberta do roteador
11	0	TTL expirado
12	0	cabeçalho IP inválido

Figura 9 – *Tipos e Códigos ICMP*[3]

2.2.4 DHCP

O protocolo DHCP (*Dynamic Host Configuration Protocol*) foi definido pela RFC 2131[24] em substituição ao antigo protocolo BOOTP (*Bootstrap Protocol*), RFC 951[25]. Conforme Kurose and Ross[3], o DHCP é um protocolo que têm a finalidade de prover endereços IP aos *hosts* em uma rede, de forma automática, fazendo uma analogia ao conecte e use (do inglês *Plug and Play*).

O uso do DHCP em uma rede, facilita muito a vida do administrador de redes, automatizando a distribuição dos endereços IP, além de outros parâmetros que podem ser enviados no campo *options* da mensagem. Conforme a RFC 2131[24], o protocolo possui oito diferentes mensagens DHCPDISCOVER, DHCPOFFER, DHCPREQUEST, DHCPACK, DHCPNAK, DHCPDECLINE, DHCPRELEASE e DHCPINFORM.

Quando um *host*, solicita um endereço IP para um servidor DHCP são utilizadas apenas as quatro primeiras mensagens, essenciais no processo de inicialização, descrito a seguir:

- O cliente envia uma mensagem DHCPDISCOVER ao endereço de *broadcast* da rede, 255.255.255.255 na porta 67/UDP;
- O servidor DHCP responde com uma mensagem DHCPOFFER com destino ao endereço de *broadcast* 255.255.255.255 na porta 68/UDP, informando o seu próprio endereço IP como origem, além de informar ao cliente o endereço IP que deverá ser atribuído a ele;

- O cliente por sua vez envia uma mensagem DHCPREQUEST ao endereço de *broadcast* da rede, 255.255.255.255 na porta 67/UDP, porém desta vez preenchendo o campo ID do servidor, para que outros possíveis servidores estejam cientes que o cliente escolheu responder ao DHCPOFFER de um servidor especifico;

- Por fim o servidor envia uma mensagem DHCPACK ao cliente via endereço de *broadcast* 255.255.255.255 na porta 68/UDP, confirmando que o cliente pode utilizar o endereço IP atribuído a ele. A partir desse processo de inicialização o cliente pode acessar a rede.

Do ponto de vista do administrador de rede, um servidor DHCP é muito útil e de grande valia em uma infraestrutura de rede, porém, algumas considerações devem ser feitas.

Quando mais de um servidor DHCP estiver respondendo requisições em um mesmo domínio de *broadcast*, o cliente decidirá a respeito de qual servidor ele vai aceitar o endereçamento.

Assim, de acordo com Wendell, O., and Sean, W. (2014)[26], se houver um servidor DHCP "Pirata", ele poderá atribuir endereços falsos aos *hosts*, fazendo com que os mesmos não consigam acessar a rede adequadamente ou ainda poderá aplicar configurações de maneira a se colocar no meio do tráfego entre o *host* e a rede, causando uma espécie de ataque *man-in-the-middle*. [4]

Outro problema, também pode ocorrer quando um atacante realizar uma grande quantidade de pedidos e, consequentemente, completa os processos de inicialização com informações falsas. Isso causa um esgotamento do servidor DHCP levando a um ataque conhecido como DHCP *Starvation*.

2.2.5 STP

O protocolo STP (*Spanning Tree Protocol*), de acordo com a IEEE 802.1D[27] de 1990, é um protocolo de camada 2 que permite que haja interligações entre *switchs* de forma redundante, de maneira que se evite *loops* de pacotes, típico de uma ligação em anel. O protocolo STP é um protocolo bastante antigo e vem sendo atualizado ao longo do tempo, o que gerou diversas revisões, uma delas e amplamente utilizada hoje em dia é o RSTP (*Rapid Spanning Tree Protocol*) ou IEEE 802.1w[28].

O funcionamento básico entre STP e RSTP é o mesmo, e consiste na construção de uma árvore do tipo grafo onde cada nó ou *switch* têm suas ligações mapeadas. Um *switch* na rede é eleito como ponte raiz (do inglês *root bridge*) baseado em sua prioridade

[4] Man-in-the-middle - Forma de ataque em que o atacante se coloca entre as partes que estão trocando dados de forma a interceptar ou alterar o conteúdo desses dados.

e endereço MAC. Em seguida cada *switch* interligado ao *root*, define qual será sua porta raiz (do inglês *root port*), analisando o menor custo até o *root* baseado na velocidade do *link*.

O protocolo STP é fundamental para o funcionamento da rede que utiliza *links* redundantes entre os *switchs* de modo a oferecer uma maior resiliência da infraestrutura em caso de falha em um de seus componentes.

Porém, se uma determinada porta do *switch* começar a receber mensagens BPDU (*Bridge Protocol Data Units*) do tipo CBPDU (*Configuration Bridge Protocol Data Units*) ou TCN (*Topology Change Notification*) advindos de um *host* malicioso, o *switch* pode ficar comprometido e ter seu funcionamento interrompido ou ter seu desempenho degradado em função do processamento massivo das mensagens falsas.

Aqui vale lembrar que, o protocolo STP é de camada 2 e para enviar mensagens STP, não é necessário nem mesmo um acesso à rede através da aquisição de um endereço IP. Bastaria apenas o acesso físico a uma tomada de rede que esteja conectada a uma porta do *switch*.

2.2.6 CDP/LLDP

O protocolo CDP (*Cisco Discover Protocol*), é um protocolo proprietário da empresa Cisco Systems™, ou seja, foi desenvolvido para uso nos equipamentos de rede fabricados por ela, não havendo a intenção de que fosse adotado por outros fabricantes, ou de tornar-se um padrão da indústria. Dessa forma, não há um RFC (*Request for Comments*) ou norma padrão para balizar sua implementação, porém o fabricante mantém documentos como o Catalyst 2960 - Software Configuration Guide[29], que detalha seu uso e suas possibilidades.

O CDP têm papel semelhante ao LLDP (*Link Layer Discover Protocol*), referenciado na norma IEEE 802.1AB[30], que conta com o esforço de vários fabricantes para que se torne um padrão da indústria e que seja amplamente utilizado. De acordo com a documentação técnica[31], da Cisco Systems™, o protocolo CDP foi criado em 1994 com a intenção de aprimorar o gerenciamento dos dispositivos conectados a rede, entre eles roteadores, *switchs*, telefones VoIP, câmeras IP, entre outros, e foi sendo desenvolvido até 2003. Atualmente está em uso a versão 2.

Estes protocolos atuam na camada de enlace (camada 2 do modelo OSI) e as mensagens são compostas por um preâmbulo obrigatório, diferente em cada um dos protocolos, seguidos de uma ou mais TLV (*Type-Length-Value*). Na Figura 10 observa-se o formato padrão do PDU (*Protocol Data Units*) referente ao protocolo CDP.

Conforme a Cisco Systems™cita no documento *"Cisco Discovery Protocol Configuration Guide, Cisco IOS Release 15M&T"*[32], o CDP é um protocolo vulnerável a

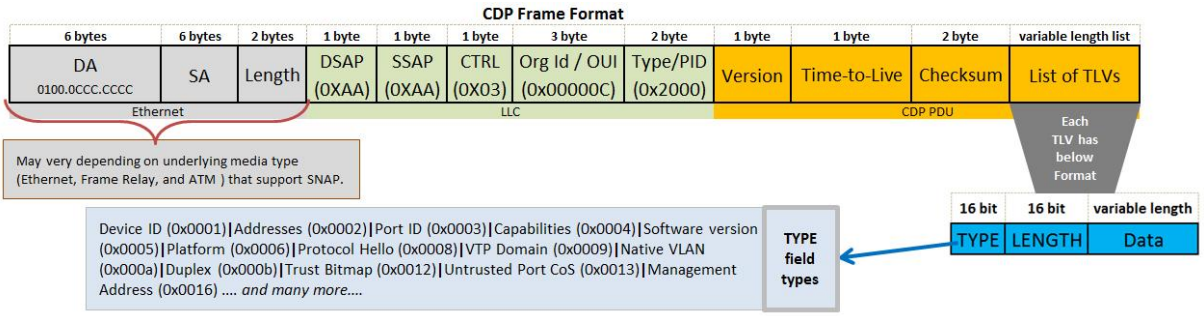


Figura 10 – *CDP PDU Format*[4]

ataques e oferece um caminho para amenizar os problemas. A sugestão passa pela criação de *TLV-Filters* e aplicação destes filtros às interfaces em que não se deseja propagar as informações CDP. Porém, essa opção oferecida não é desejável, uma vez que os dispositivos necessitam trocar essas informações para que seja facilitada a gestão dos equipamentos ligados aos *switchs* que é a finalidade da adoção desses protocolos.

Como exemplo, existem *switchs* que se utilizam das informações recebidas, via CDP ou LLDP, para aplicar configurações automáticas como *trunk* ou *voice vlan* às portas.

Existem diversos tipos de ataques de negação de serviço bem como inúmeras variações, além de outros tipos de ataques com outros propósitos. Especificamente para este trabalho, serão analisados os ataques que possuem alguma capacidade de causar impacto no consumo de banda de um *switch*, ou no uso de CPU, causando assim uma anomalia nos dados. A próxima seção trata sobre esses ataques.

2.3 Ataques

Segundo Santos, O., and Stuppi, J. (2015)[33], a preocupação com segurança quase sempre é focada na camada 3, do modelo OSI, ou superiores sendo que as camadas 1 e 2 acabam negligenciadas na maioria das vezes. Os ataques relacionados à infraestrutura de redes podem ser bastante danosos e de difícil diagnóstico, pois os *firewalls* não são capazes de identificar os dispositivos causadores dos ataques pelo fato de operar em camada 3.

Ainda que fosse possível, os ataques em camada 2 podem ter os endereços MAC (*Media Access Control*) dos dispositivos mascarados (*MAC Spoofing*) durante o ataque, dificultando ainda mais a detecção, ou ainda tratar-se de ataques que não se propagam a outros equipamentos, ficando restritos às interfaces de um *switch*, por exemplo.

Assim, entre os objetivos deste trabalho está a identificação de anomalias que sejam capazes de causar algum problema na infraestrutura de rede, seja degradando a qualidade do tráfego ou causando sua interrupção.

Então, com o intuito de elencar os ataques mais relevantes para este estudo, foram compiladas 7 assinaturas baseadas nas características comuns dos ataques referenciados em dois documentos. Um deles é o *Taxonomy of DDoS Attacks*[34], publicado pela empresa RioRey, onde são explicados 25 diferentes ataques, além de trazer uma tabela com suas principais características. O segundo documento é o *7 Popular Layer 2 Attacks*[35], que traz um conjunto de 7 diferentes ataques que afetam principalmente a camada 2 do modelo OSI.

Todos os ataques informados pelo documentos, tiveram suas características analisadas e foi observado que devido à anatomia de cada um, eles seriam iguais ou equivalentes quando observados diante das variáveis coletadas nos switchs.

Esse conjunto de variáveis composto pelas MIB's (*Management Information Base*) SNMP (*Simple Network Management Protocol*) e RMON (*Remote Network Monitoring*), estão disponíveis na maioria dos *switchs* disponíveis no mercado e particularmente presente em todos os *switchs* em uso na instituição e são listadas na Tabela 3.

Tabela 3 – Objetos MIB SNMP/RMON

MIB Object	**Descrição**
Grupo A	
ifInOctets	Total de octetos transmitidos pela interface
ifOutOctets	Total de octetos recebidos pela interface
Grupo B	
ifInUcastPkts	Total de pacotes Unicast transmitidos pela interface
ifOutUcastPkts	Total de pacotes Unicast recebidos pela interface
ifInBroadcastPkts	Total de pacotes Broadcast transmitidos pela interface
ifOutBroadcastPkts	Total de pacotes Broadcast recebidos pela interface
ifInMulticastPkts	Total de pacotes Multicast transmitidos pela interface
ifOutMulticastPkts	Total de pacotes Multicast recebidos pela interface
ifInDiscards	Total de pacotes descartados transmitidos pela interface
ifOutDiscards	Total de pacotes descartados recebidos pela interface
ifInErrors	Total de pacotes com erro transmitidos pela interface
ifOutErrors	Total de pacotes com erro recebidos pela interface
Grupo C	
etherStatsUndersizePkts	Total de pacotes com tamanho menor que 64 octetos
etherStatsPkts64Octets	Total de pacotes com tamanho até 64 octetos
etherStatsPkts65to127Octets	Total de pacotes com tamanho entre 65 e 127 octetos
etherStatsPkts128to255Octets	Total de pacotes com tamanho entre 128 e 255 octetos
etherStatsPkts256to511Octets	Total de pacotes com tamanho entre 256 e 511 octetos
etherStatsPkts512to1023Octets	Total de pacotes com tamanho entre 512 e 1023 octetos
etherStatsPkts1024to1518Octets	Total de pacotes com tamanho entre 1024 e 1518 octetos
etherStatsOversizePkts	Total de pacotes com tamanho maior que 1518 octetos

As variáveis podem ser separadas em três grupos básicos de acordo com as informações que elas trazem:

- **A - Volume de Octetos** - representa a quantidade de octetos que são recebidos ou transmitidos pela interface;
- **B - Tipo de Pacote** - representa a quantidade de pacotes que são recebidos ou transmitidos pela interface, de acordo com cinco tipos, que são: *Unicast, Broadcast, Multicast, Discards e Errors*;

- **C - Tamanho dos Pacotes** - os pacotes são classificados de acordo com o seu tamanho em octetos.

Após a análise, foram descartados 11 ataques, entre os listados pela RioRey[34], classificados como baixo ou muito baixo *Packet Rate*, pois quando executados em um ambiente de laboratório, seu impacto nas variáveis é imperceptível, não causando impacto na largura de banda nem no uso de CPU dos *switchs*.

Foram descartados também os ataques do grupo *TCP HTTP BASED*, pois além de sua maioria se enquadrar na condição anterior, são ataques de sessão HTTP (*Hypertext Transfer Protocol*) e causam problemas especificamente nos servidores atacados, não causando impacto nos quesitos de largura de banda e CPU dos *switchs*.

No grupo *TCP BASED e ICMP BASED* foram descartados os ataques *Misused Application* e *ICMP Flood*, pois possuíam uma característica em que o *Packet Rate* ou o *Packet Size* deveriam ser variáveis e não foi possível gerar tal característica com as ferramentas utilizadas. Os ataques descartados foram: *Fake Session*, *Session Attack*, *Misused Application*, *HTTP Fragmentation*, *Excessive VERB*, *Excessive VERB Session*, *Multiple VERB Single Request*, *Recursive GET*, *Random Recursive GET*, *Faulty Application* e *ICMP Flood*.

Com relação aos ataques de camada 2, listados em *7 Popular Layer 2 Attacks*[35], em alguns casos, não haviam parâmetros bem definidos para se gerar um ataque, portanto foi realizado um ataque baseado no protocolo de forma a causar o efeito sugerido e alguns ataques foram descartados por não causar impacto nos quesitos analisados. Os ataques descartados foram: *Address Resolution Protocol (ARP) Attacks*, *Media Access Control (MAC) Spoofing* e *Virtual LAN (VLAN) Hopping*.

Dessa forma, a Tabela 4, traz a classificação dos ataques, bem como o protocolo que cada um utiliza e a camada OSI em que operam. As características, em comum, que foram as responsáveis pelo agrupamento dos vários ataques em assinaturas estão descritas na sequência.

2.3.1 Assinatura A

A assinatura A, é baseada no protocolo DHCP, que pode ser feita de diversas maneiras, quando a questão é um ataque *hacker*. O ataque sugerido no documento *7 Popular Layer 2 Attacks*[35], *DHCP Spoofing*, não é capaz de causar uma negação de serviço, portanto foi feita uma adaptação para se gerar um ataque conhecido como *DHCP Starvation/Flooding*. Inclusive a mitigação deste ataque é tema central do trabalho de Mukhtar et al.[36].

Este ataque realiza uma inundação (*Flooding*) de pedidos de endereço IP ao ser-

Tabela 4 – Ataques por Assinatura

Camada OSI	Assinatura	Protocolo	Ataque
7	**A**	**DHCP**	DHCP Flooding
3	**B**	**TCP**	SYN Flood
			SYN-ACK Flood
			ACK & PUSH ACK Flood
			RST or FIN Flood
			Synonymous IP
		UDP	UDP Flood
			DNS Flood
			VoIP Flood
			Non-Spoofed UDP Flood
		ICMP	Ping Flood
	C	**TCP**	Fragmented ACK
		UDP	UDP Fragmentation
		ICMP	ICMP Fragmentation
	D	**UDP**	Media Data Flood
	E	**TCP**	CAM Overflow/MAC Flooding
2	**F**	**STP**	STP Flooding
	G	**CDP**	CDP Flooding

vidor, falsificando (*Spoofing*) o endereço MAC do solicitante de maneira aleatória. Dessa forma o servidor realizará a entrega de todo o seu *pool* de endereços e passará a negar todos os pedidos legítimos, causando a falha.

O ataque com esta assinatura têm como característica gerar um tráfego em torno de 320 Mbit/s, com pacotes do tipo Broadcast de tamanho entre 256 e 511 octetos. No uso de CPU percebeu-se alterações significativas, sendo registrados picos de até 68%.

2.3.2 Assinatura B

A assinatura B é a que engloba a maior parte dos ataques, dez ao todo, e caracteriza-se por utilizar três protocolos distintos (TCP, UDP e ICMP), todos operando em camada 3 do modelo OSI.

São ataques que realizam uma inundação (do inglês *Flooding*) de pacotes e mesmo quando utiliza TCP, não realiza o processo de *Handshake*. Possui uma taxa de pacotes (*Packet Rate*) alta ou muito alta de tamanho (*Packet Size*) pequeno, normalmente 64Kb.

Nesta assinatura, utilizando o protocolo TCP, pode-se exemplificar detalhando o ataque TCP-SYN, uma vez que os outros ataques que utilizam TCP apenas fazem uso de *flags* diferentes no cabeçalho do protocolo, o que não causa impacto nas variáveis analisadas.

Assim, o ataque TCP-SYN[37], consiste em explorar uma característica do proto-

colo TCP. Conforme mencionado, na seção 2.2.1, este ataque explora uma característica do *3-Way Handshake*.

Quando um atacante realiza um ataque baseado nessa fragilidade do protocolo, ele envia uma grande quantidade de pacotes SYN ao servidor. O servidor, então envia um pacote SYN-ACK para cada pacote SYN recebido, com destino ao endereço IP informado como remetente, e fica aguardando uma resposta ACK de volta ao servidor para fechar a apresentação do *3-Way Handshake*. Porém, esta resposta ACK nunca é enviada, fazendo com que estas conexões sejam mantidas abertas na pilha de memória do servidor, que acaba esgotada devido ao excesso de pedidos, passando assim a negar os pedidos legítimos por falta de recursos.

Este ataque, pode ser dividido em três formas: um Ataque Direto, um Ataque com IP Falso, ou ainda um Ataque Distribuído. O ataque com IP falso é o mais usual, onde o atacante envia pacotes com destino a um alvo, utilizando endereços IP falsificados (do inglês *Spoofed IP*) como origem. Assim, o servidor responde os pacotes SYN+ACK a remetentes desconhecidos, nesse caso, o atacante não recebe os pacotes de resposta do servidor.

Então, quando um *host* recebe um pacote SYN+ACK sem ter enviado previamente um SYN, ele responderá com um pacote RST (*Reset*), o que causaria o fechamento do processo de apresentação liberando o servidor para uma nova solicitação, diminuindo o poder do ataque. Assim, segundo a RFC 4987[37], um ataque eficiente utilizando IP falso, não utiliza endereços falsos escolhidos aleatoriamente. São utilizados endereços previamente testados, de maneira que os destinatários da resposta SYN+ACK não respondam com um pacote RST.

Os ataques distribuídos geralmente fazem uso de uma *botnet*, onde o atacante assume o controle ou envia comandos a um grupo de computadores "zumbis" que geram o ataque DDoS. Nesses casos, detectar a origem do ataque é muito difícil, tornando-se quase impossível quando os computadores que originam o ataque fazem uso de endereços IP falsos.

De acordo com o site SegInfo[38], a *botnet* Mirai foi responsável pelo maior ataque de negação de serviço registrado em 2016 e utilizou mais de 145 mil pequenos dispositivos domésticos como roteadores, câmeras IP e dispositivos do tipo Internet das Coisas ou IoT (*Internet of Things*), que devido à falhas de segurança, foram invadidos e passaram a enviar ataques coordenados.

Já o ataque UDP-Flood[39], funciona causando uma inundação de pacotes UDP em várias portas conhecidas por oferecer serviços como DHCP, NTP, DNS, entre outros.

Nesse caso, o atacante envia os pacotes UDP a um alvo e por consequência, o servidor faz o processamento destes pacotes realizando dois passos. Primeiro, ele verifica

em suas portas se há algum processo respondendo pacotes UDP, não havendo, é enviado um pacote ICMP ao remetente informando que o servidor não está disponível nesta porta ou destino inalcançável (do inglês *Destination Unreachable*). Então, caso haja um processo que responda nessa porta, o servidor devolverá a resposta ao endereço IP do remetente.

Para este tipo de ataque, o atacante informa um endereço IP de origem diferente do seu, pois, como ocorre no TCP, receberia os pacotes de resposta. Nos ataques utilizando os protocolos citados na Tabela 2, o endereço do remetente é propositalmente o da vítima, para que as respostas amplificadas causem a inundação do alvo.

Por fim, o ataque *Ping Flood* ou *ICMP Flood*[40], utiliza-se do protocolo ICMP, enviando pacotes ECHO (*Type* 8 | *Code* 0) com endereços falsos.

Os ataques incluídos nesta assinatura, têm como característica gerar um tráfego em torno de 104 Mbit/s, com pacotes do tipo Unicast de 64 octetos. O uso de CPU permaneceu, sem alterações significativas, em torno de 5%.

2.3.3 Assinatura C

A assinatura C, agrupa três ataques e também utiliza os protocolos TCP, UDP e ICMP, todos operando em camada 3 do modelo OSI.

Estes ataques, são caracterizados pela fragmentação (do inglês *Fragmentation*) de pacotes e consiste em enviar uma alta taxa de pacotes (*Packet Rate*), de tamanho (*Packet Size*) grande, normalmente acima do maior MTU possível, neste caso foi utilizado 65.495Kb. Dessa forma, para que os pacotes consigam atravessar os dispositivos de rede e atingir o alvo, ele é dividido em pacotes menores, para serem comportados no MTU definido pelo *switch*. Esses pacotes, divididos em partes menores, são enviados ao alvo do ataque, de forma que o mesmo deve armazená-los em um buffer até que o último pacote chegue para então recompor o pacote original.

Os equipamentos de rede e segurança como *switchs*, roteadores, *firewalls* além dos IDS e IPS não barram estes pacotes pois se assemelham a um tráfego legítimo. Este ataque, além de consumir uma grande largura de banda ainda causa sobrecarga nos servidores fazendo com que os mesmos utilizem processamento remontando pacotes com carga de dados inúteis.

Os ataques incluídos nesta assinatura têm como característica gerar um tráfego em torno de 740 Mbit/s, com pacotes do tipo Unicast com 97,98% de tamanho entre 1024 e 1518 octetos e 2,12% entre 256 e 511 octetos. O uso de CPU permaneceu sem alterações significativas, em torno de 5%.

2.3.4 Assinatura D

A assinatura D, composta apenas pelo ataque *Media Data Flood*, basicamente é um ataque *UDP Flood*, como na assinatura B, porém neste caso o tamanho do pacote (*Packet Size*) é moderado. Para este ataque, foi estabelecido um tamanho de pacote de 750 Bytes pois corresponde a metade do tamanho do MTU, que é de 1500 Bytes.

Este ataque, tenta imitar um tráfego de mídia do tipo áudio ou vídeo, porém também se parece com um tráfego de voz (*VoIP*), exceto pelo tamanho dos pacotes.

O ataque desta assinatura, têm como característica gerar um tráfego em torno de 513 Mbit/s, com pacotes do tipo Unicast de tamanho entre 512 e 1023 octetos. O uso de CPU permaneceu sem alterações significativas, em torno de 5%.

2.3.5 Assinatura E

A assinatura E, é composta apenas pelo ataque *CAM Overflow* ou *MAC Flooding*[41], mas para entender melhor esse ataque é preciso entender o funcionamento desse mecanismo presente nos *switchs*.

Os *switchs* possuem uma característica particular que permite a realização deste ataque, e baseia-se no funcionamento da tabela CAM (*Content Addressable Memory*) ou tabela MAC (*Media Access Control*). Essa tabela é responsável por registrar os endereços MAC dos equipamentos ligados às suas portas, e ele faz isso inspecionando o endereço MAC contido nos pacotes que trafegam por ele.

Assim, quando o *switch* é ligado, sua tabela MAC está vazia e para que ele garanta que o primeiro pacote chegue ao seu destino, ele faz uma difusão deste pacote para todas as suas portas, exceto a que originou o pacote, realizando uma inundação (*flooding*). O *switch* faz isso uma única vez e registra na tabela MAC a qual porta determinado endereço está ligado. Na próxima vez que um pacote com destino a este endereço passar pelo *switch* ele encaminha diretamente à porta correta.

Essa característica, faz com que a largura de banda do *switch* seja otimizada. Porém a tabela MAC têm uma quantidade finita de endereços que podem ser aprendidos, a depender do modelo do *switch*, neste ponto é que ocorre o ataque. Dessa forma, o atacante realiza o envio de uma grande quantidade de pacotes, falsificando o endereço MAC de origem e destino. Então o *switch* tenta aprender todos os endereços que chegam, e sua tabela MAC se esgota. Todas as vezes, que a tabela está esgotada o *switch* é obrigado a realizar uma inundação (*flooding*) para entregar o pacote de um MAC desconhecido. Assim a performance do *switch* cai e sua largura de banda é afetada, causando lentidão, o que pode ocasionar o reinício do *switch*.

O ataque desta assinatura têm como característica gerar um tráfego em torno de

11 Mbit/s, com pacotes do tipo Unicast e Multicast, com tamanho de 64 octetos. No uso de CPU percebeu-se alterações significativas, sendo registrados picos de 61%.

2.3.6 Assinatura F

Assim como na Assinatura E, para entender o ataque *STP Flooding*[42], único ataque que compõe a assinatura F, também é preciso entender o funcionamento desse mecanismo presente nos *switchs*.

Os *switchs* fazem uso deste protocolo para evitar *loops* em redes de camada 2, que possuem *links* redundantes, estabelecendo, por exemplo, uma maneira de ligação de *switchs* em topologia de anel sem que ocorra uma inundação de pacotes *broadcast*.

Esse efeito indesejado, ocorre quando um *switch* X encaminha um pacote ao *switch* Y, através do link A, e o mesmo devolve o pacote ao *switch* X através do link redundante B. Quando isso acontece, esse ciclo não é quebrado, e os pacotes nesta situação ficam sendo trocados entre os switchs indefinidamente, causando uma degradação da largura de banda e levando ao colapso do tráfego.

Quando o STP (*Spanning Tree Protocol*) está presente e habilitado no *switch*, ele estabelece quem será o *root bridge* ou o *switch* principal da estrutura. A partir das portas deste switch, são enviados CBPDU de maneira a informar os outros switchs quem é o root e qual o melhor caminho para atingi-lo, levando em consideração a métrica estabelecida pelo STP.

Dessa maneira, um ataque do tipo *STP Flooding*, consiste em enviar uma quantidade muito grande de pacotes, do tipo TCN ou CBPDU, a um *switch*, de forma que o mesmo processe esses pacotes de maneira a modificar sua configuração ou topologia conforme a sinalização dos pacotes. Esse tipo de ataque pode causar uma inconsistência na hierarquia das redes de camada 2, levando ao colapso ou a configurações errôneas que prejudicam o tráfego dos usuários, causando assim uma negação de serviço.

O ataque desta assinatura têm como característica gerar um tráfego em torno de 111 Mbit/s, com pacotes do tipo Unicast e Multicast com tamanho de 64 octetos. No uso de CPU percebeu-se alterações significativas, sendo registrados picos de 63%.

2.3.7 Assinatura G

O CDP (*Cisco Discover Protocol*), como dito anteriormente, é um protocolo vulnerável a ataques. Ainda assim é muito utilizado, principalmente em redes composta por equipamentos Cisco^{TM}em sua maioria.

O ataque que representa a assinatura G é o *CDP Flooding*[42], que têm uma estrutura bem parecida com o *STP Flooding* com relação a seus efeitos. Os switchs podem

modificar dinamicamente suas configurações, baseados nas informações que recebem do protocolo CDP.

Quando um atacante envia uma grande quantidade de mensagens CDP, através de uma das portas do *switch*, o mesmo registra uma entrada, em sua tabela de entradas CDP, para cada mensagem recebida. Essa tabela têm uma quantidade finita de entradas e realizar uma inundação (*flooding*), neste caso, pode fazer com que o *switch* deixe de receber entradas legítimas de equipamentos que estejam necessitando informar o *switch* para que a porta receba uma configuração adequada.

O ataque desta assinatura, têm como característica gerar um tráfego em torno de 150 Mbit/s, com pacotes do tipo Multicast com tamanho de 65 a 127 octetos. No uso de CPU percebeu-se alterações significativas, sendo registrados picos de 77%.

As falhas e ataques podem ocorrer de diversas maneiras, sejam total ou parcialmente, a depender da maneira como as organizações estão estruturadas. Na infraestrutura analisada neste trabalho, as falhas ocorridas em *switchs* de acesso podem ser responsáveis por se propagar a *switchs* de camada superior afetando assim outros ramos da rede e podendo levar a paralisação total ou parcial da estrutura. Assim, a próxima seção trata sobre alguns mecanismos de predição de falhas.

2.4 Mecanismos de predição de falhas

Existem na literatura e no mercado alguns recursos que se dispõem a amenizar ou solucionar os problemas com ataques DoS/DDoS. Nesta seção serão abordados alguns mecanismos disponíveis na instituição utilizada como caso de uso neste projeto, bem como a sua aplicabilidade no contexto do dia a dia.

2.4.1 Cisco Storm Control

A empresa Cisco Systems™, norte americana, fabricante de uma ampla gama de equipamentos de conectividade e segurança, têm em seu portfólio vários modelos de *switchs*, que compõe tanto a classe empresarial *Catalyst* quanto a classe SOHO (*Small Office Home Office*). Os switchs de classe *Catalyst* são conhecidos pela sua robustez e alta capacidade, além de utilizarem o sistema operacional IOS (*Internetwork Operating System*), um sistema operacional proprietário e destinado a uso embarcado que conta com recursos avançados de gerência e segurança.

Dentre os recursos disponíveis, há um que merece maior atenção com relação a este trabalho, o *Storm Control*[43]. Este recurso está disponível nos switchs da classe *Catalyst* e também em alguns *switchs* mais avançados da classe SOHO, como o modelo SG500, bastante utilizado pela instituição. Ainda assim cabe ressaltar que existem muitos

switchs não compatíveis com o *Storm Control* atualmente em operação no parque de equipamentos da instituição. Com relação aos equipamentos mais avançados da linha *Catalyst*, como a série *Catalyst 4500E* que são equipamentos de agregação, esse recurso é implementado em *hardware* para que se obtenha uma performance adequada e consiga verificar com muita rapidez o estado de cada interface configurada no tempo necessário para responder aos limites definidos na configuração. Nos equipamentos mais simples esse recurso é implementado em software.

O manual técnico dos modelos *Catalyst 4500 Series* se refere ao *Storm Control* como um controle de uma tempestade de *broadcast* que pode inundar a rede criando um tráfego excessivo, assim, degradando ou impedindo sua utilização.

Por padrão, este recurso é definido como desabilitado e para que possa ser utilizado deve ter seu funcionamento configurado. O mecanismo de funcionamento como ilustrado na Figura 11, se da através da análise da quantidade de pacotes ou bytes trafegados na interface entre dois instantes. Por exemplo, se a quantidade de pacotes, ou bytes, ultrapassa o valor definido no limite (do inglês *threshold*) entre os instantes T1 e T2, então uma ação é disparada. As ações a serem tomadas em caso de violação da regra, são:

- ***Shutdown***: nesse caso, a interface é desabilitada até que o valor volte a ficar abaixo do limite estabelecido após um determinado tempo pré-estabelecido, ou no caso desse tempo não ser definido, a interface permanecerá desativada até que um operador ou algum sistema à habilite novamente.
- ***Trap***: quando a ação escolhida é um *trap*, a interface não sofre qualquer alteração e uma mensagem de alerta é enviada a algum sistema de gerenciamento capaz de tratar essa informação.

Os limites da regra podem ser definidos em BPS (*Bits Per Second*), PPS (*Packets Per Second*) ou em porcentagem sobre a largura de banda da interface.

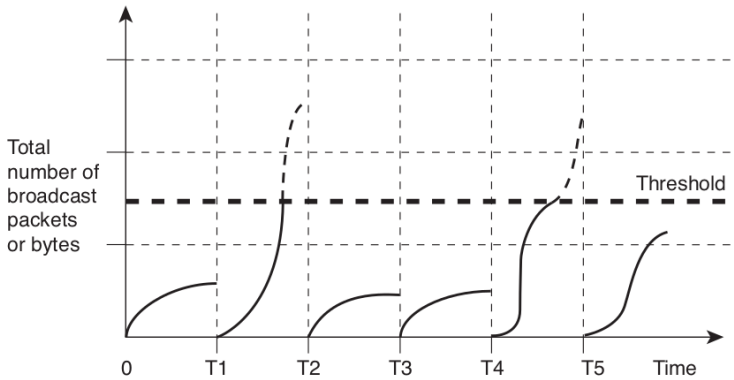


Figura 11 – *Storm Control Example—Hardware-based Implementation*
fonte: Configuring Storm Control[43]

Esse recurso, bastante interessante, é amplamente difundido pelo fabricante, que inclusive incorpora *hardware* específico para tal, porém cabe aqui ressaltar que este recurso não esta disponível em todos os equipamentos gerenciáveis do fabricante, pois trata-se de um recurso avançado, conforme mencionado anteriormente.

É necessário entender que a configuração deste recurso ocorre por interface, uma a uma ou em grupos, porém de acordo com o tipo de equipamento ligado à interface podemos ter comportamentos bastante diferentes e um limite definido pra uma interface pode não ser adequado a outra, causando uma grande dificuldade de se realizar a configuração.

Há ainda uma situação, comum em infraestruturas de rede, em que os usuários ou equipamentos podem mudar de localização passando a se conectar em uma interface diferente do *switch*, fazendo com que os limites configurados deixem de fazer sentido, uma vez que não são dinâmicos e não acompanham as mudanças sem a intervenção técnica adequada.

Cabe ainda dizer, que com a análise pura e simples da quantidade de dados trafegados, seja *broadcast*, *multicast* ou *unicast*, há o risco de bloquear um tráfego legítimo ou o que seria pior, permitir um tráfego anômalo sem que a regra seja violada. Em última análise, trata-se de um recurso caro e não muito vantajoso do ponto de vista econômico e operacional, dada sua baixa eficiência.

2.4.2 Cisco Security Suite

Os *switchs* mais robustos da linha SOHO, como o SG500 ou versões mais recentes, dispõe de um recurso bastante interessante chamado *Security Suite*[44]. No tocante a este trabalho, ele chama a atenção particularmente por prover a detecção e a filtragem do tráfego anômalo, causado por ataques do tipo DoS ou DDoS.

Vale ressaltar, que mesmo configurados para atuarem em camada 2, esses *switchs* possuem capacidade para atuar em camada 3, permitindo assim a análise de datagramas dos protocolos TCP e UDP. Segundo o manual de administração do *switch*[44], as regras de ACL (*Access Control List*) e as políticas de QoS (*Quality of Service*) devem estar desabilitadas na interface em que se deseja utilizar o recurso de detecção de ataques.

Esses ataques estão entre os principais causadores de falhas que são objetivo deste trabalho e entre as possibilidades de detecção do *Security Suite* estão:

- ***Stacheldraht Distribution*** O ataque *stacheldraht*, "Arame Farpado" do alemão, é um ataque do tipo DDoS e faz uso de um cliente para se conectar a um ou mais manipuladores (do inglês *handlers*) de maneira a coordenar um ataque distribuído em direção a um alvo. Quando habilitado, são descartados os pacotes TCP com porta de origem igual a 16660.

- ***Invasor Trojan*** Um ataque deste tipo permite que um *host* se torne um "zumbi" fazendo com que sejam descartados os pacotes do tipo TCP com destino a porta 2140 cuja origem seja a porta 1024.

- ***Back Orifice Trojan*** Este ataque é uma variação do *trojan* que permite que seja explorado um *back orifice* ou "porta dos fundos", de maneira a infectar o *host*. Neste caso são descartados os pacotes UDP cujo destino seja a porta 31337 e a origem seja a porta 1024.

- ***Martian Address*** Os "Endereços Marcianos" são endereços IP definidos em uma lista como endereços individuais ou por faixas inseridas, manualmente, e que deverão ter os pacotes descartados todas as vezes que o endereço de origem coincidir com esta lista. Alguns exemplos podem ser os endereços IP que não pertençam a rede interna.

- ***SYN Filtering*** A filtragem de pacotes SYN é feita de acordo com a configuração na interface específica, levando em conta o endereço IP especificado, que pode ser único, uma faixa ou qualquer endereço e ainda a porta TCP que se deseja realizar o filtro, podendo ser entre uma porta, todas as portas ou portas mais conhecidas.

- ***SYN Rate Protection*** A proteção contra ataques que utilizam pacotes SYN ainda pode ser feita configurando a taxa de pacotes entrantes na interface. Isso permite que se estabeleça o fluxo máximo de pacotes de maneira a evitar uma inundação em um servidor, ligado a essa porta. Porém, isso apenas evitaria a sobrecarga no servidor, uma vez que o *switch* passaria a descartar os pacotes excedentes causando uma negação de serviço de qualquer maneira.

- ***ICMP Filtering*** A filtragem de pacotes ICMP é feita nos mesmos moldes da filtragem dos pacotes SYN, porém neste caso o protocolo passa a ser ICMP, que é um protocolo de controle importante e muitas vezes essencial em algumas operações de redes.

- ***IP Fragmented*** Os pacotes fragmentados podem ser bloqueados com essa opção, porém aqui cabe uma ressalva, algumas vezes os pacotes cujo tamanho definido pelo MTU (*Maximum Transmission Unit*) do pacote, pode ultrapassar os convencionais 1500 bytes, devendo ser dividido em pacotes menores para que possam atravessar a rede. Assim pacotes fragmentados podem não representar uma ameaça de fato e deve-se ter muita cautela ao utilizar esse filtro sob o risco de bloquear pacotes realmente uteis.

Pode-se habilitar o recurso em dois níveis, *System Level Prevention* ou *System Level and Interface Level Prevention*. No nível *System Level Prevention* a proteção se dá

a nível de sistema. Estes *switchs* utilizam um servidor de páginas HTML, interno, para a sua interface de configuração e a mesma está sujeita a ataques como qualquer servidor web. A CPU do *switch* está protegida pelo SCT (*Secure Core Technology*) que é habilitado por padrão e não pode ser desabilitado.

Já quando habilitado o nível mais alto, *System Level and Interface Level Prevention*, também é elevada a proteção a nível de interface, de acordo com as proteções descritas acima. Cabe ressaltar que o *switch* têm seu nível de utilização de CPU aumentado à medida que o nível de proteção é aumentado, pois para que os datagramas sejam analisados, é necessário processamento e isso pode levar a uma sobrecarga de uso da CPU. Esse fato pode levar a superaquecimento, maior consumo de energia e possível lentidão nos processos internos de encaminhamento de pacotes.

Em relação a utilidade do *Security Suite*, pode-se destacar que as configurações se dão a nível de *switch*, que assim como no *Storm Control*, devem ser realizadas de acordo com o uso de cada *switch*, deparando-se com as mesmas especificidades de ambiente. Outro ponto relevante é que o *Security Suite* traz proteções contra ataques específicos, como *Stacheldraht Distribution*, *Invasor Trojan* e *Back Orifice Trojan*, que são problemas conhecidos e possivelmente serão deixados de lado por atacantes, pois a cada dia descobre-se novas falhas de segurança e novos métodos de realizar os ataques de maneira menos custosa e mais eficiente.

Assim, os switchs estariam protegidos de ataques obsoletos e nos mesmos moldes para os quais foram idealizados, por exemplo: o ataque *Stacheldraht Distribution* faz o bloqueio dos pacotes TCP advindos da porta 16660, caso a porta seja modificada, esta proteção se torna inútil.

2.4.3 FortiOS DoS/DDoS Protection

Os *firewalls* são equipamentos fundamentais para a proteção de qualquer rede privada ou DMZ (*Demilitarized Zone*), para proteção interna ou externa, quando estas são conectadas à internet. Assim será analizada a solução presente na instituição, com relação a sua capacidade de mitigar ataques DoS/DDoS presente no NGFW (*Next Generation Firewall*) FortiGate 1500D[45].

Este modelo é fabricado pela Fortinet™e possui capacidade de processamento de até 80 Gbit/s, quando utilizado no modo *Firewall*, 13 Gbit/s para IDS, 7 Gbit/s quando no modo NGFW e 5 Gbit/s para proteção de ameaças. Ele possui ainda múltiplas interfaces de 1 a 10 Gigabits por segundo com conexões em cobre RJ45 ou fibra óptica utilizando SFP/SFP+.

O equipamento está conectado à rede conforme a Figura 12, e atua como NGFW, analisando e filtrando todo o tráfego que entra e sai da rede privada, LAN e da rede DMZ.

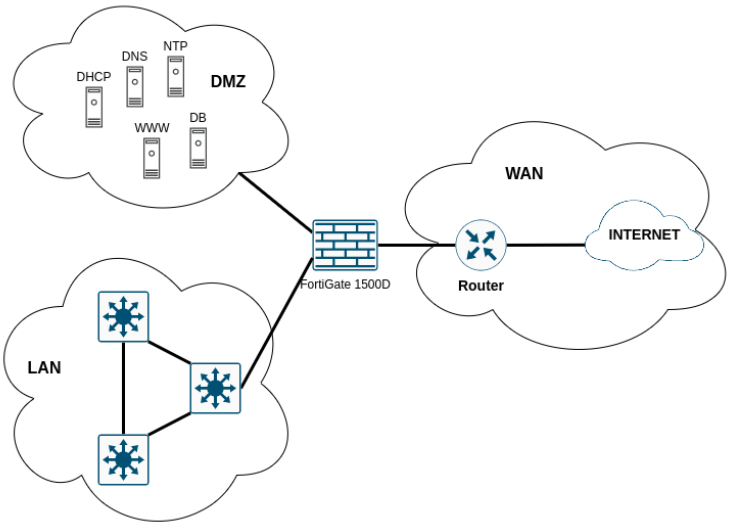


Figura 12 – *Infraestrutura de Conexão à Internet*

Como se trata de um equipamento com muitos recursos, o foco deste trabalho é apenas os recursos contra DoS. Assim como informado na documentação oficial[46], este equipamento é capaz de realizar a análise de pacotes a nível de *hardware*, possuindo 2 processadores dedicados denominados **NP6** (*Network Processor*), o que torna seu desempenho bastante rápido e trás robustez ao equipamento. Uma vez habilitada a proteção contra DoS, este tráfego passa pelo NP6, que examina os pacotes suspeitos antes mesmo de entregá-los a CPU, então, para os casos de detecção positiva para ataque o pacote é descartado logo na entrada.

Os tipos de ataques detectados pelo *firewall* e a forma como ele trabalha para mitigar os problemas são detalhados a seguir:

- ***SYN-Flood*** Para minimizar os problemas causados por uma inundação de pacotes SYN, o *firewall* utiliza algumas técnicas baseadas nas opções de configuração. Ao se definir um limite de conexões abertas, o *firewall* passa a monitorar as conexões *tcp_syn_flood* e pode tomar duas ações quando este limite é atingido, aprovar ou bloquear.

  Ao aprovar, os pacotes são liberados para atingir o IP de destino e o evento é registrado em log, mas ao bloquear, esses pacotes são descartados antes de serem encaminhados ao alvo. Para o caso de bloqueio, ocorre um fato interessante que a regra aproveita muito bem: quando um cliente legítimo tenta fazer uma conexão e não consegue, quase sempre tenta novamente, então o *firewall* classifica essa tentativa como legítima e passa a permitir a conexão. Isso se dá pelas características das ferramentas de ataque, que ao enviarem pacotes com endereços IP aleatórios, quase nunca repetem a tentativa de conexão com o mesmo endereço. Porém, vale salientar que os limites ou *thresholds* são definidos por quem configura o sistema.

Assim, valores muito altos podem deixar os alvos mais suscetíveis aos ataques e valores muito baixos, apesar de deixarem o sistema mais seguro, podem gerar mais falsos positivos.

Um outro recurso consiste em fazer com que o *firewall* trabalhe como um *Proxy SYN*. Nessa condição, ao atingir o limite o *firewall* passa a interceptar os pacotes SYN, respondendo o pacote SYN+ACK ao cliente, e caso receba o pacote ACK, como esperado em conexões legítimas, o firewall completa a conexão para o IP de destino, caso contrário descarta os pacotes.

- ***Blocking External Probes*** Uma abordagem utilizada pelo *firewall* vem de encontro com a prevenção de um ataque e parte de um princípio simples. Antes de iniciar um ataque, o *hacker* pesquisa sobre seu alvo em potencial, assim uma varredura de endereços e portas quase sempre é o passo inicial. Sabendo disso, o *firewall* dispõe de regras que ajudam a ocultar informações preciosas sobre sua infraestrutura, como: *AddressMask, Traceroute, ICMP.Invalid.Packet.Size e ICMP.Oversized.Packet*. Essas regras podem ser configuradas para permitir somente o que é necessário.

  Outra regra que também pode ser usada atende a uma situação em que o atacante tenta realizar conexões em diversas portas para determinar quais delas respondem a um pedido. Essas conexões geralmente partem de um endereço IP único, que é onde o atacante espera receber a resposta ao pacote que inicia a conexão. Assim, as regras *tcp_port_scan e udp_port_scan* podem bloquear endereços que tentam realizar muitas conexões em portas diferentes, o que caracteriza um escaneamento malicioso.

- ***Bad Flags*** Também podem ser verificados se os pacotes TCP não possuem erros nos bits de sinalização, como por exemplo um pacote não pode conter o sinalizador SYN, que marca o início de uma sessão, juntamente com um sinalizador FIN usado para encerrar uma sessão. Esses artifícios podem ser explorados para testar o comportamento dos dispositivos diante de situações inesperadas.

- ***UDP Flood e ICMP Flood*** Há também parâmetros que limitam os pacotes UDP e ICMP com a finalidade de bloquear uma inundação de pacotes destes protocolos. A documentação do equipamento informa que esses ataques são menos prevalentes do que os *TCP-Flood*, e que os possíveis alvos podem resolver isso de maneira simples, apenas descartando os pacotes uma vez que pode-se bloquear as respostas aos pacotes ICMP e descartar os pacotes UDP que não sejam destinados a nenhuma porta que ofereça um serviço válido.

  Em parte isso pode ser considerado verdadeiro, uma vez que os pacotes para conexões TCP podem fazer com que os alvos sejam afetados com menos esforço, porém a prevenção aos ataques utilizando UDP ou ICMP, é muito utilizada quando se

trata de uma ataque volumétrico, ou seja, com a intenção de se esgotar a largura de banda do alvo.

Além destes pontos abordados, cabe ressaltar que o *firewall*, não é capaz de bloquear o fluxo de pacotes na rede interna, entre *switchs*, causado por um equipamento originando um DoS/DDoS, o que acaba afetando o ramo da rede ao qual este dispositivo está ligado. Ainda assim, as vantagens de poder contar com um equipamento deste nível é indiscutível.

Este equipamento foi colocado em operação na instituição em 2019, a um custo inicial aproximado de R$ 480.000,00, valor este que incluía uma licença de 3 anos, atualizações e garantia, devendo ter essa licença renovada em 2022 a um custo estimado de R$ 380.000,00 por mais 3 anos. Valor proibitivo para muitas instituições e/ou empresas menores.

2.5 Considerações finais

Este capítulo abordou os principais protocolos de rede utilizados para realização dos ataques mais comuns, bem como algumas de suas vulnerabilidades que são exploradas por atacantes. Foram abordados também os ataques que compõe as assinaturas das anomalias que servirão como insumos a serem utilizados nos algoritmos de aprendizado de máquina. Por fim, foram analisadas algumas ferramentas já disponíveis na instituição utilizada como caso de uso neste trabalho, que poderiam ser utilizadas para mitigar os problemas.

O próximo capítulo abordará as técnicas de aprendizado de máquina que podem ser aplicadas para detectar as anomalias que levam às falhas.

3 *Aprendizado de Máquina*

3.1 Considerações Iniciais

Segundo o dicionário Michaelis[47], Inteligência é definida como "a faculdade de entender, pensar, raciocinar e interpretar", porém outras definições podem ser aplicadas. O dicionario Dicio Online[48] define inteligência como a "habilidade para entender e solucionar adversidades ou problemas, adaptando-se a circunstâncias novas". No entanto, outra definição de inteligência, do mesmo dicionário, diz que a inteligência provém de intelecto e é a característica que distingue o homem do animal. Por outro lado isso não é uma justa verdade, visto que a literatura traz muitos algoritmos de inteligência artificial que imitam justamente o comportamento dos animais, como colônia de formigas, enxame de abelhas entre outros.

Assim, a inteligência artificial surgiu da abordagem computacional para resolução de problemas, utilizando imitações da inteligência humana ou outras formas de inteligência natural, sendo considerada um ramo da ciência da computação, tornando-se tema de estudos. Alguns atribuem seu surgimento como tema de pesquisa ao matemático britânico Alan Turing, a partir de seu artigo *"Computing Machinery and Intelligence"*[49] de 1950.

Desde então, outros pesquisadores trouxeram à luz várias definições, ou interpretações, para inteligência artificial assim como algumas discussões filosóficas a respeito do que de fato pode ser chamado de inteligente ou não. A definição de inteligência artificial, ou simplesmente IA, veio mudando e se alterando com o passar do tempo, e filosoficamente poderia ser discutida em um livro dedicado ao assunto.

3.2 Inteligência Artificial

Até a década de 80, os sistemas de IA não podiam fazer muito mais do que realizar o processamento de milhares de cálculos por segundo, calcular as milhares de possibilidades para as jogadas de xadrez, como foi feito pelo IBM Deep Blue^{TM}ao jogar contra o enxadrista Garry Kasparov, ou algo do tipo. A capacidade de processamento, principalmente para cálculos matemáticos, é inegavelmente superior ao de um ser humano, todavia, um computador nesta época necessitava de um software específico para jogar xadrez.

No tocante a este trabalho, a IA é compreendida como um sistema especialista, uma vez que não se pretende criar um sistema de propósito geral (GPS - *General Purpose System*) e sim um sistema capaz de prever e detectar falhas em infraestruturas de rede de computadores. Porém, não é possível utilizar um sistema programado especificamente

para tal, como em um jogo de cartas, onde todas as possíveis jogadas são conhecidas e não se alteram com o tempo, por exemplo, do contrário este sistema teria que ser modificado cada vez que fosse percebida alguma falha desconhecida ou situação nova.

Pode-se dizer, que a inteligência artificial evoluiu à medida que os computadores foram aumentando a capacidade de processamento, e após os anos 80, como mostra a Figura 13, dando origem ao aprendizado de máquina (do inglês *machine learning*).

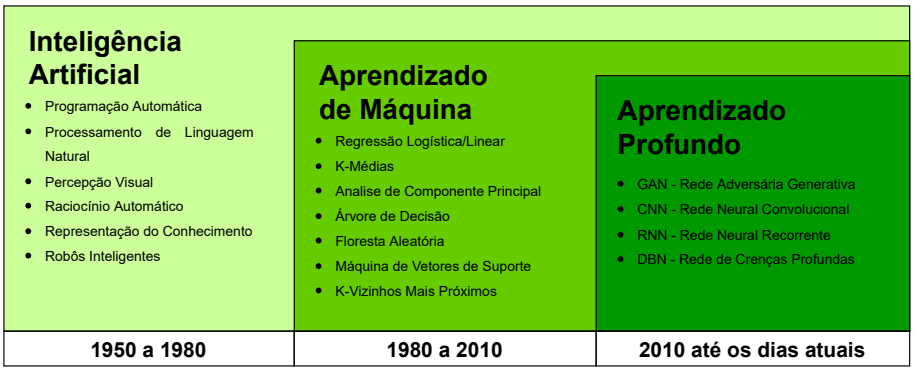


Figura 13 – Linha temporal da Inteligência Artificial
fonte: Adaptado de Copeland, M.(2016)[50]

3.3 Machine Learning

O aprendizado de máquina surgiu da necessidade de se resolver problemas sem que houvesse a necessidade de uma programação especifica para o problema abordado, Arthur Samuel(1959) disse que: "O aprendizado de máquina é o campo de estudo que dá aos computadores a habilidade de apreender sem ser explicitamente programado".

E ainda de acordo com Monard, M. C., & Baranauskas, J. A. (2003)[51], "Aprendizado de Máquina é uma área de IA cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado bem como a construção de sistemas capazes de adquirir conhecimento de forma automática".

Como visto na Figura 13, várias abordagens são apresentadas para resolver algumas dessas questões e podem ser utilizadas de maneira simples ou combinadas.

No caso do jogo de xadrez, por exemplo, que tem suas regras bem definidas, podemos pedir ao computador que calcule todas as possíveis combinações entre as jogadas, levando em conta a forma como as peças podem se movimentar no tabuleiro e isso de fato não é uma tarefa fácil para uma pessoa, mas um computador poderia fazê-lo em um tempo razoável.

No entanto, se for utilizada uma técnica de aprendizado de máquina para analisar todas as jogadas executadas por mestres enxadristas em uma grande quantidade de par-

tidas, seria possível gerar um modelo capaz de realizar jogadas mais rapidamente e mais assertivamente.

Já quando se trata de falhas em uma infraestrutura de rede de computadores, há um sistema dinâmico que pode ter suas características modificadas ou ainda deve se levar em conta que uma falha ainda não descoberta venha a ser explorada por um atacante e comece a causar problemas. Como não se pode prever todas as possibilidades, então é necessário um sistema que seja capaz de "aprender" com os dados históricos e de alguma maneira se adaptar a novas situações, fazendo com que não seja necessário modificar o sistema ou que essas modificações sejam minímas.

Os métodos de aprendizado de máquina, segundo Monard, M. C., & Baranauskas, J. A. (2003)[51], podem ser divididos em: supervisionados ou não supervisionados, sendo subdivididos ainda em classificadores ou regressores. No caso deste estudo foram utilizados métodos supervisionados classificadores.

3.3.1 Classificação

Quando se utiliza aprendizado de máquina para procedimentos de classificação de dados, é necessário que sejam seguidos alguns procedimentos de maneira a garantir que os algoritmos escolhidos funcionem da melhor maneira possível para os dados que estão em uso. Os procedimentos para utilização de aprendizado de máquina podem variar de acordo com os dados e o problema a ser abordado. Schröer et al. [52] trás uma revisão sistemática do modelo CRISP-DM que está bastante consolidado na prática e no meio acadêmico.

De maneira geral, é possível guiar este processo adaptando o que já está preconizado pela literatura. Kumar et al. [53], trás um procedimento bastante similar ao modelo CRISP-DM e de maneira prática, é possível aplicá-lo, conforme ilustrado na Figura 14.

- **Problema:** Nesta etapa é necessário ter em mente de forma bastante clara qual é o problema a ser resolvido. É nesta fase que se define qual o tipo de abordagem será aplicada;
- **Identificação dos Dados:** Uma análise deve ser feita com os dados disponíveis de forma a identificar se os mesmos podem trazer respostas que ajudem a responder a questão principal do problema. Assim, estabelece-se as correlações entre os atributos e é possível conhecer melhor os dados que são necessários e os que podem ser dispensados, tornando o conjunto de dados mais enxuto e otimizado possível;
- **Limpeza e Rotulação:** Nesta etapa é realizada uma limpeza de maneira a eliminar os dados repetidos erroneamente, os atributos nulos ou com valores zerados por

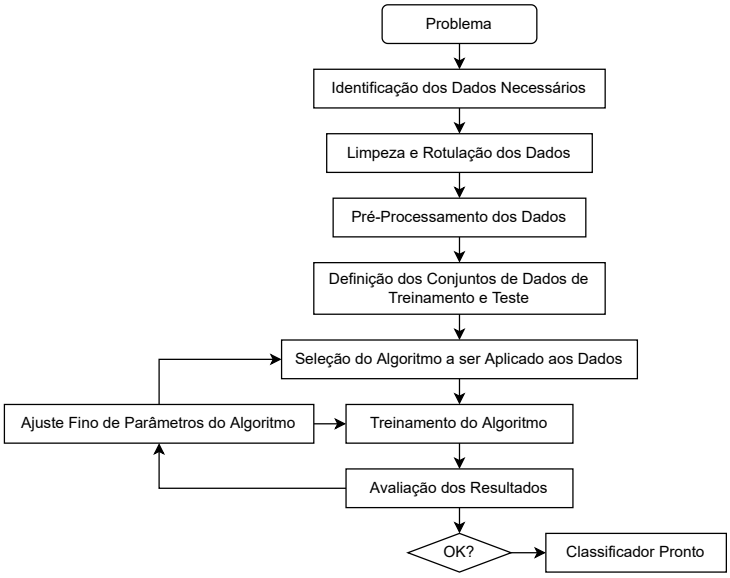


Figura 14 – *Procedimento de Classificação*
fonte: Adaptado de Kumar et al.(2017)[53]

exemplo. É nesta etapa que também realiza-se a rotulação dos dados diferenciando-os entre os dados normais e as anomalias;

- **Pré-Processamento:** O pré-processamento pode ser usado para ajustar os dados de maneira que o algoritmo consiga utilizá-los, assim pode ser necessário substituir campos de texto por valores de referência ou ainda normalizar valores muito grandes ou pequenos demais;

- **Definição dos Conjuntos de Treinamento e Teste:** A etapa de definição dos conjuntos utilizados para treinamento e teste talvez seja uma das mais importantes, pois, nesta etapa os dados devem ser analisados com relação a sua proporcionalidade entre dados normais ou anomalias, dessa forma é possível eleger a melhor maneira de se submeter os dados ao algoritmo.

  Uma das maneiras de separar os dados de treino e teste é realizar uma validação cruzada (do inglês *Cross-Validation*), isso permite que se ajuste os tamanhos dos conjuntos e que se faça um embaralhamento dos dados de forma a evitar que o conjunto de treino fique muito desproporcional ao de teste, ocasionando um *Overfitting* e fazendo com que o classificador se torne tendencioso;

- **Seleção do algoritmo:** Os algoritmos devem ser selecionados de acordo com o objetivo que se pretende alcançar e os tipos de dados utilizados. Na seção 3.4 é feita uma revisão da literatura mostrando os principais algoritmos utilizados para dados muito semelhantes aos tratados aqui, além de outros algoritmos semelhantes disponíveis no Scikit-Learn[54];

- **Treinamento:** O treinamento é a execução da função *Fit* do algoritmo utilizando o conjunto de dados destinados ao treinamento e posteriormente validar os resultados tentando classificar os dados do conjunto de teste;

- **Avaliação dos Resultados:** Após o treinamento, obtém-se um modelo treinado e os resultados são avaliados baseado nas métricas definidas na seção 3.3.3. Caso seja aceitável, o classificador está pronto para ser utilizado. Caso contrário, o modelo pode passar por ajustes finos de modo a ter sua avaliação melhorada ou ser desconsiderado como possibilidade de ser usado;

- **Ajuste Fino dos Parâmetros:** Os algoritmos de classificação possuem diversos parâmetros a depender de cada um que podem ser modificados, melhorando assim sua performance. Esta é uma fase que pode ser considerada empírica, e deve-se experimentar diversos parâmetros diferentes para que se possa extrair o melhor de cada algoritmo.

3.3.2 Balanceamento dos Dados

Muitas vezes, os conjuntos de dados ou *datasets*, utilizados para treinamento dos modelos de classificação, podem conter uma quantidade de amostras desbalanceadas em relação aos dados rotulados como normais e as diversas classes de anomalias. Segundo Sun, Yanmin et al.(2009)[55], este desbalanceamento ocorre por diversas vezes em conjuntos onde as amostras são desproporcionais entre dados normais e anômalos e inclusive entre anomalias.

No caso do *dataset* KDD99[56], utilizado amplamente para treinamento de detecções de anomalias em redes de computadores e sistemas de detecção de intrusão, existem 4 grupos de anomalias, ou ataques, em que R2L (*remote-to-local*) e U2R (*user-to-root*) são proporcionalmente menores em relação aos outros, como mostra a Figura 15.

Classe	Registros	Proporção %
Normal	97.277	16.69
DoS	391.458	79.24
Probe	4.107	0.83
U2R	52	0.01
R2L	1.126	0.23
Total	**494.020**	**100**

Figura 15 – *Balanceamento dos Dados do Dataset KDD99*
fonte: Adaptado de Nguyen, Huy Anh and Choi, Deokjai.(2008)[57]

Ainda de acordo com Sun, Yanmin et al.(2009)[55], uma quantidade de dados desbalanceadas entre as diversas classes pode levar o algoritmo de classificação a se tornar

tendencioso para a classe mais abundante, chamada de classe majoritária. Existem algumas alternativas para evitar esse problema, onde o conjunto de dados pode ser redefinido de maneira a equilibrar a proporção entre as classes.

Assim, duas abordagens podem ser adotadas para ajuste do conjunto de dados. A primeira é a redução da classe majoritária de maneira a equacionar a proporção dos dados, técnica conhecida como *Random Undersampling*, abordada por Smith et al.(2012)[58], que consiste no descarte aleatório de dados da classe majoritária. Um ponto desfavorável dessa abordagem é quando a classe majoritária representa uma grande parte do conjunto, então ocorreria uma redução drástica do conjunto de dados que pode levar a uma situação indesejável chamada *Underfitting*, em que o modelo se torna incapaz de reproduzir os dados do conjunto.

A segunda abordagem é a utilização de uma técnica de *Oversampling* chamada SMOTE (do inglês *Synthetic Minority Over-sampling Technique*). Conforme Chawla, Nitesh V., et al.(2002)[59], essa técnica consiste em aumentar artificialmente a quantidade de dados de classes minoritárias. Porém, ao adotar essa técnica pode ocorrer outra situação, contrária a primeira técnica, que é a ocorrência do *Overfitting*, em que o modelo fica super ajustado.

3.3.3 Métricas para Classificadores

As métricas são medidas importantes para que se possa avaliar quantitativamente os resultados. Para calcular essas equações são necessários alguns valores básicos que servem de insumo para equações importantes. Esses são os dados que compõem a matriz de confusão, conforme ilustrado na Figura 16. Os dados da matriz de confusão são descritos na sequência:

Matriz de Confusão		PREDITO	
		SIM	NÃO
REAL	SIM	Verdadeiro Positivo	Falso Negativo
	NÃO	Falso Positivo	Verdadeiro Negativo

Figura 16 – *Matriz de Confusão*
fonte: Adaptado de Scikit-Learn(2011)[54]

- **VP - Verdadeiro Positivo:** Representa a quantidade de acertos do modelo ou seja, quantas vezes o modelo foi capaz de classificar um dado como anomalia quando de fato se tratava de uma anomalia.
- **FP - Falso Positivo:** É a quantidade de vezes em que um dado foi classificado como anomalia quando de fato era normal.

- **VN - Verdadeiro Negativo:** É a quantidade de vezes que o modelo classificou uma dado como normal e de fato se tratava de um dado normal.
- **FN - Falso Negativo:** É a quantidade de vezes em que o modelo classificou um dado como normal quando na verdade se tratava de uma anomalia.

Com os valores obtidos pela matriz de confusão, é possível calcular algumas métricas que serão úteis em uma avaliação mais abrangente, que são:

- **Pr - Precisão**: A medida da precisão é o quociente da divisão de VP pela soma de VP com FP e é dada pela equação 3.1. Para o cenário de detecção de anomalias, essa medida é importante para se conhecer o percentual de acertos do modelo ao predizer corretamente a classificação de uma anomalia.

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{3.1}$$

- **Rc - *Recall*:** A sensibilidade também chamada de *Recall*, é uma taxa de verdadeiros positivos ou seja, o quociente entre VP pela soma dos valores de VP e FN, dada pela equação 3.2. Essa taxa mostra qual o percentual de acertos das predições corretas(VP) em relação à todas as corretas, representadas pelos acertos e pelos erros.

$$\text{Sensibilidade} = \frac{VP}{VP + FN} \tag{3.2}$$

- **F1 - *F1 Score*:** A métrica F1-Score é dada pela média harmônica entre Pr e Rc, tendo seu melhor valor em 1 e o pior em 0. Essa métrica tem sido amplamente utilizada para medir de maneira mais abrangente a capacidade de um modelo de acertar suas predições, uma vez que leva em consideração a média entre outras métricas, fazendo com que F1-Score seja usado como referência para classificação dos diversos algoritmos quando comparados entre si.

$$\text{F1} = 2 \times \frac{\text{Precisão} \times Sensibilidade}{\text{Precisão} + Sensibilidade} \tag{3.3}$$

  Na análise multi-classe, como a tratada nesta pesquisa, é necessária a utilização de médias entre as classes classificadas para o valor final das métricas. As médias possibilitam analisar problemas de desbalanceamento entre classes e suas influências nos resultados. As médias que podem ser utilizadas nas métricas classificadoras são:

  - Média Micro: calcula a métrica considerando o total de VP, FN e FP independentemente da previsão para cada rótulo no conjunto de dados;

  - Média Macro: calcula a métrica para cada rótulo e retorna a média sem considerar a proporção de cada rótulo no conjunto de dados;
  - Média Ponderada: calcula a métrica para cada rótulo e retorna a média considerando a proporção de cada rótulo no conjunto de dados.

- ***Acurácia*:** A acurácia é uma métrica de desempenho muito comum em vários trabalhos, porém ela avalia os métodos de maneira generalizada, que por vezes não ajuda a medir de maneira eficiente o desempenho do mesmo. Essa métrica é definida pela razão entre a quantidade de acertos do método e a quantidade de dados avaliados. Ela é dada pela equação 3.4.

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN} \tag{3.4}$$

- ***Acurácia Balanceada:*** Quando existe um desbalanceamento dos dados, como é o caso dessa pesquisa, a métrica da acurácia pode não ser útil ou até mesmo fornecer uma medida equivocada, uma vez que os valores classificados como verdadeiros negativos podem mascarar as classificações baixas de verdadeiros positivos, fornecendo assim uma classificação não confiável.

  Dessa forma, é utilizada a acurácia balanceada, que consiste em calcular a taxa de verdadeiros positivos e verdadeiros negativos, como demonstrado na equação 3.5.

$$\text{Acurácia Balanceada} = \frac{1}{2}\left(\frac{VP}{VP + FN} + \frac{VN}{VN + FP}\right) \tag{3.5}$$

  Ao utilizar-se a acurácia balanceada, no qual é levada em conta os acertos de cada classe de forma igualitária, é mostrado um valor mais próximo do quanto o modelo consegue acertar de cada classe, penalizando assim a métrica quando alguma classe tem uma alta taxa de erros.

- ***Tempo de Execução*:** O tempo de execução, apesar de não estar ligado a nenhuma métrica envolvendo a quantidade de acertos ou erros de um modelo, é importante que seja levada em consideração, uma vez que em caso de utilização prática do modelo, o tempo passa a ter um impacto significativo.

3.3.4 Técnicas de Classificação

Inicialmente foram consideradas 15 técnicas de aprendizado de máquina supervisionado classificador, disponíveis no rol de métodos do *Scikit-Learn*[54], algumas também presentes na revisão da literatura, para a análise da base de dados que será apresentada no Capítulo 4.

De forma a reduzir o campo de investigação foram realizados experimentos preliminares e as técnicas foram submetidas ao processo de treinamento, aplicados aos dados

do local A, utilizando parâmetros padrão para todos os métodos. Assim, os valores obtidos são resultado das médias obtidas no processo de validação cruzada, explicado em 4.4.

As técnicas avaliadas inicialmente são mostradas na Tabela 5, em ordem decrescente pela métrica **f1_weighted** ou F1 ponderada.

Tabela 5 – Técnicas de Classificação Avaliadas

Técnica	Tempo	f1_weighted	balanced_accuracy	precision_weighted	recall_weighted
RandomForestClassifier	2.208354	0.999488	0.994792	0.999506	0.999503
ExtraTreesClassifier	0.536193	0.999488	0.994792	0.999506	0.999503
HistGradientBoostingClassifier	2.659715	0.999291	0.993736	0.999322	0.999304
XGBClassifier	6.208331	0.999095	0.991653	0.999152	0.999105
GradientBoostingClassifier	29.942042	0.999086	0.991653	0.999116	0.999105
DecisionTreeClassifier	0.146615	0.998899	0.991626	0.998954	0.998906
RidgeClassifier	0.022884	0.981876	0.870833	0.976093	0.987766
SVC	0.345599	0.981280	0.865611	0.975533	0.987170
KNeighborsClassifier	0.003447	0.978883	0.866976	0.975808	0.982100
MLPClassifier	5.280509	0.955956	0.662147	0.950801	0.965195
AdaBoostClassifier	1.271703	0.955474	0.655100	0.948211	0.966279
LinearSVC	7.284898	0.918053	0.325000	0.906213	0.935547
PassiveAggressiveClassifier	0.097306	0.911670	0.248918	0.902032	0.927989
DummyClassifier	0.002410	0.876498	0.125000	0.839883	0.916451
SGDClassifier	0.581844	0.870325	0.293261	0.907283	0.868784

Como critério, foram escolhidas as técnicas com **f1_weighted** igual ou maior que 0.99 e **balanced_accuracy** maior ou igual a 0.98, chegando assim a 6 técnicas que foram as mais bem avaliadas como mostra a Tabela 6.

Tabela 6 – Técnicas de Classificação Escolhidas

Técnica	Tempo	f1_weighted	balanced_accuracy	precision_weighted	recall_weighted
RandomForestClassifier	2.208354	0.999488	0.994792	0.999506	0.999503
ExtraTreesClassifier	0.536193	0.999488	0.994792	0.999506	0.999503
HistGradientBoostingClassifier	2.659715	0.999291	0.993736	0.999322	0.999304
XGBClassifier	6.208331	0.999095	0.991653	0.999152	0.999105
GradientBoostingClassifier	29.942042	0.999086	0.991653	0.999116	0.999105
DecisionTreeClassifier	0.146615	0.998899	0.991626	0.998954	0.998906

Um fato interessante é que as técnicas escolhidas possuem uma característica em comum, todas são baseadas em árvores de decisão, o que nos mostra uma afinidade natural desse tipo de técnica com os dados do problema abordado.

Dentre as técnicas escolhidas, observa-se que elas podem ser divididas em dois grupos, o primeiro é baseado em árvore de decisão pura e duas variações que utilizam a técnica de *bagging* aplicadas a aleatorização, e o segundo grupo que utiliza técnicas de *boosting* para minimizar a função de perda.

Segundo Géron, Aurélien(2019)[60], a técnica de *Ensemble Learning* consiste em utilizar vários métodos diferentes em um conjunto, arranjados como *bagging* ou *boosting*, e podem ser utilizadas com diferentes métodos, não só com árvores de decisão.

A técnica de *bagging* realiza o treinamento de vários previsores, e em seguida realiza uma votação entre os resultados obtidos, de forma a verificar qual valor obteve o maior número de votos, decidindo assim pelo voto da maioria. Como pode ser visto na Figura 17, a execução dos previsores pode ser paralelizada.

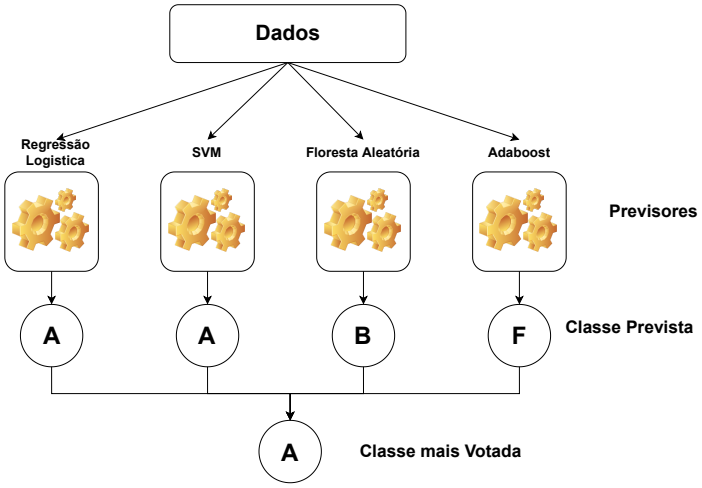


Figura 17 – *Estrutura de um Ensemble do tipo Bagging*
fonte: Adaptado de Géron, Aurélien(2019)[60]

A técnica de *boosting*, consiste na execução de uma sequência de previsores em que os dados submetidos ao próximo previsor da sequência, é o erro residual do previsor anterior. Os passos dessa técnica podem ser vistos na Figura 18.

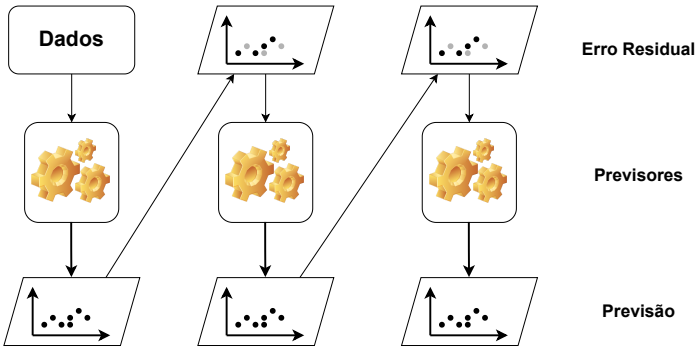


Figura 18 – *Estrutura de um Ensemble do tipo Boosting*
fonte: Adaptado de Géron, Aurélien(2019)[60]

A seguir são detalhadas as técnicas escolhidas.

3.3.4.1 *Decision Tree Classifier*

O algoritmo de árvore de decisão (do inglês *Decision Tree*), consiste em uma abordagem bastante simples e podemos fazer uma analogia às estruturas de dados como árvores B, B+, Rubro Negra, entre outras. Trata-se de uma estrutura recursiva, que segundo Géron, Aurélien(2019)[60], pode ser definida da seguinte maneira: Um nó folha representa uma classe no processo de classificação e um nó de decisão contém um teste condicional

sobre o atributo analisado, onde cada aresta pode levar a um nó folha ou a outro nó de decisão, conforme ilustrado na Figura 19.

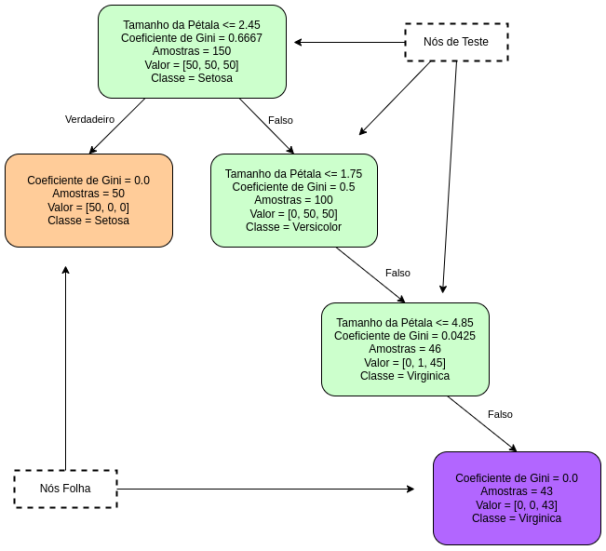


Figura 19 – *Estrutura de Árvore de Decisão*
fonte: Adaptado de Géron, Aurélien(2019)[60]

Esse algoritmo, utiliza uma medida de impureza para os nós, essa medida é o coeficiente de Gini, dado pela equação 3.6, onde $P_{i,k}$ é a média das instâncias da classe k entre as instancias do nó i.

$$G_i = 1 - \sum_{k=1}^{n} P_{i,k^2} \tag{3.6}$$

Dessa forma, a Figura 19, nos mostra que o nó direito de profundidade 2 tem o coeficiente de Gini igual a $1 - (0/46)^2 - (1/46)^2 - (45/46)^2 \approx 0.0425$. Já quando o coeficiente de Gini é igual a 0, no caso das folhas, podemos dizer que ele é puro e isso ocorre quando a classificação está definida.

Como critério de divisão dos nós, também pode ser utilizada a entropia de Shannon, que equivale a minimizar a perda de log entre os rótulos verdadeiros e as previsões probabilísticas do modelo.

O algoritmo de árvore de decisão é um dos mais simples, tornando-se popular a partir do trabalho de Breiman, L., et al.(1984)[61], é também a base para a construção do algoritmo de floresta aleatória (do inglês *Random Forest*) tratado mais adiante.

A construção da árvore, inicia pelo particionamento do conjunto de dados de treinamento, em dois subconjuntos separados por um limite ou *threshold*, esse limite será a utilizado para o teste condicional dó primeiro nó, minimizando a entropia dos subconjuntos. Esse processo, se repetirá para cada subconjunto gerado e um novo atributo será escolhido a cada vez que o processo se repetir.

Todas as vezes que um novo atributo for escolhido para particionar um conjunto, será criado um nó de teste para a condição deste atributo. Esse processo se repetirá até que restem somente nós folha, representando a classe a qual o subconjunto pertence, porém, como alerta Oshiro (2013)[62], caso a árvore cresça demais, se tornará muito específica para o conjunto de dados utilizado para o treinamento, causando assim, um problema de sobre-ajuste ou do inglês *overfitting*. A definição de um número mínimo de amostras necessárias em um nó folha bem como definir a profundidade máxima da árvore ajudam a evitar esse problema.

Outra desvantagem de se utilizar árvores de decisão apontada em Scikit-Learn (2011)[54], é a criação de uma árvore utilizando dados que contenham classes dominantes, sob o risco de se obter um modelo tendencioso. Portanto recomenda-se o balanceamento dos dados antes de realizar os ajustes, conforme a subseção 3.3.2.

3.3.4.2 *Random Forest Classifier*

As árvores de decisão, são modelos de alta variância, ou seja, uma pequena mudança nos dados de treinamento podem gerar modelos totalmente diferentes. Isso se deve a uma tendência natural de super ajustagem ou *overfitting*, assim Breiman, Leo.(1996)[63], propõe uma técnica de *bagging* como forma de reduzir o viés e otimizar a variância do modelo.

O modelo de floresta aleatória, proposto em Breiman, Leo.(2001)[64], é uma variação da árvore de decisão que faz uso da técnica de *bagging* aliada a aleatorização (do inglês *Randomization*). A floresta aleatória é composta por um número de k árvores aleatórias, identicamente distribuídas e independentes onde cada árvore apresenta um resultado para a entrada desejada.

Cada árvore aleatória é gerada a partir de um conjunto de árvores possíveis, onde os atributos a são aleatoriamente distribuídos entre m possíveis sendo $a \leq m$. Para realizar a classificação, cada árvore elege uma classe para uma determinada entrada, assim a classificação ocorre utilizando a classe mais votada entre as árvores que compõe a floresta, como ilustrado na Figura 20.

Pode-se perceber, que os registros de entrada é o mesmo para diversas árvores, mesmo que o aspecto final das árvores sejam ligeiramente diferentes, algumas classes podem ser idênticas. A partir das classes de cada árvore é escolhida a mais votada. Segundo Breiman (2001) e Shiyang, Xuan et al.(2018), a eficiência deste modelo depende de dois fatores: a força individual e a correlação de duas árvores quaisquer.

- **Força individual**: A força de cada árvore depende de sua chance de acerto, dessa forma uma árvore com uma taxa de acerto alta, possui uma força maior e quanto maior a força individual das árvores, maior a eficiência do classificador;

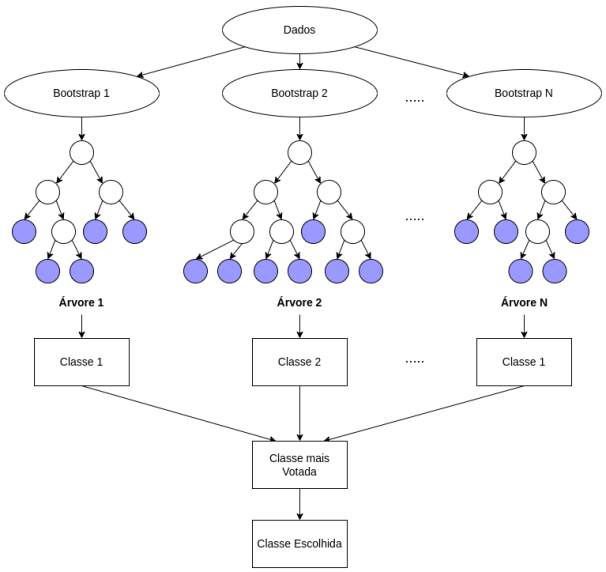


Figura 20 – *Floresta Aleatória*
fonte: Adaptado de Shiyang, Xuan et al.(2018)[65]

- **Correlação entre as árvores**: Dada a forma aleatória como as árvores são geradas, existe uma tendência de que a correlação entre elas seja baixa e a eficiência do classificador tende a ser maior quando a correlação entre as árvores é menor.

3.3.4.3 *Extra Trees Classifier*

O surgimento do modelo *Extra Trees* (*Extremely randomized trees*) ou florestas extremamente aleatórias proposto por Geurts, P. et al.(2006)[66], trás um nível a mais de aleatoriedade ao "randomizar" fortemente tanto a escolha de atributo quanto o ponto de corte enquanto divide um nó da árvore.

Assim, como nas florestas aleatórias, um subconjunto aleatório de características candidatas é usado, porém em vez de usar os limites mais discriminativos, os limites são definidos de maneira aleatória para cada característica, sendo escolhido o melhor para a regra de divisão, reduzindo um pouco mais a variância porém com um aumento no viés.

Ao diminuir a variância em detrimento do viés, as árvores extremamente aleatórias se tornam mais rápidas para treinar do que os métodos anteriores, isso se da ao fato de que encontrar os limites de um recurso para cada nó consome um recurso computacional considerável, que nesse caso é minimizado. Na Figura 21 vemos a estrutura de uma floresta extremamente aleatória.

3.3.4.4 *Gradient Boosted Decision Trees*

O *Gradient Boosting*, é uma técnica de aprendizado de máquina supervisionado que pode ser utilizado em problemas de classificação, como neste caso.

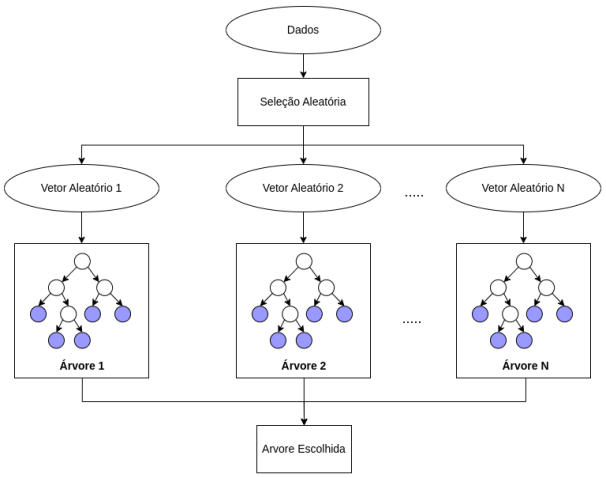


Figura 21 – *Floresta Extremamente Aleatória*
fonte: Adaptado de Bhati, Bhoopesh Singh, e C. S. Rai.(2020)[67]

Segundo Friedman(2001)[68], esta técnica produz um modelo em forma de conjunto ou *ensemble* de árvores de decisão, construindo o modelo de forma consecutiva, permitindo a otimização de uma função de perda.

De forma prática, podemos dizer que o algoritmo GBDT (do inglês *Gradient Boosted Decision Trees*) é composto por uma sequência de treinamentos e predições, que são os iteradores. O algoritmo utiliza um gradiente descendente e a cada iteração é calculado o erro residual, que é utilizado para treinar e calcular o erro da próxima iteração, como ilustra a Figura 22.

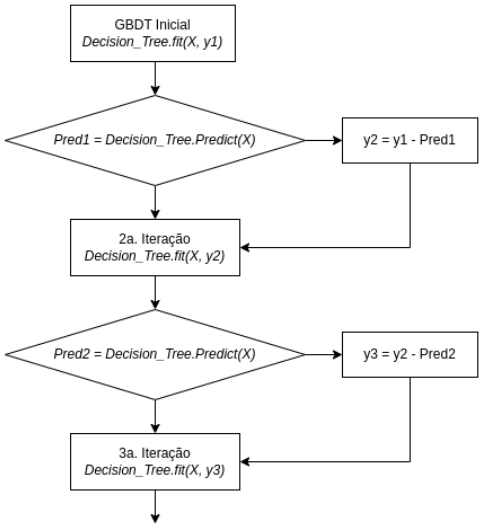


Figura 22 – *Gradient Boosting Decision Tree*

3.3.4.5 *Histogram-Based Gradient Boosting*

O método *Histogram-Based Gradient Boosting*, implementado no conjunto de métodos do *Scikit-Learn*, é inspirado no *LightGBM* proposto por Ke, Guolin, et al.(2017)[69],

porém não traz uma implementação completa do *LightGBM*. Este método aprimora o *Gradient Boosting* com relação ao tempo de aprendizagem, uma vez que os dados de entrada são agrupados em *bins* ou unidades do histograma, dessa forma reduz drasticamente os pontos de divisão do algoritmo.

Segundo Ke, Guolin, et al.(2017)[69] o método baseado em histogramas foi capaz de realizar o treinamento até 20 vezes mais rápido que a implementação tradicional do *Gradient Boosting*, com praticamente a mesma precisão. Este método ainda traz a vantagem de fazer uso do processamento paralelo, tanto em CPU quanto em GPU.

O aumento do desempenho neste caso é devido a forma como as árvores são construídas. Ao utilizar essa abordagem em algumas partes como a construção dos histogramas, a definição do melhor ponto de divisão ou o mapeamento das amostras entre folhas esquerda e direita, é possível paralelizar a execução do método ganhando performance.

O *Histogram-Based Gradient Boosting* é especialmente eficaz com grandes quantidades de dados, acima de 10.000, e quando utilizado em pequenas amostras tem o desempenho semelhante ao *Gradient Boosting*, porém apresenta um uso de memória e tamanho do arquivo de salvamento consideravelmente menor.

3.3.4.6 *Extreme Gradient Boosting*

O método *Extreme Gradient Boosting* ou *XGBoost* foi proposto por Chen, T., & Guestrin, C.(2016)[70] e trouxe uma capacidade de lidar com uma quantidade massiva de dados e foi projetado originalmente para ser escalável.

O *Extreme Gradient Boosting* assim como o *Gradient Boosting* é um *ensemble* de árvores de decisão, porém tem a capacidade de paralelizar algumas partes do algoritmo, como ocorre no *Histogram-Based Gradient Boosting*.

De acordo com a documentação oficial do *XGBoost*[71], este método implementa uma função de otimização de perda que quando integrada ao critério de divisão, é utilizada como estrategia de pré-poda na construção da árvore, controlando assim a complexidade. O método *Extreme Gradient Boosting* ainda permite que seja utilizada uma função de perda definida pelo usuário. Além de possuir a capacidade de lidar com dados esparsos e implementar alguns hiperparâmetros de controle de crescimento das árvores.

Um deles é o controle de encolhimento, que regula o passo da função de otimização da perda. Um passo mais lento, como 0.1, pode fazer com que o modelo se ajuste demais aos dados, causando um *overfitting*, já um passo mais rápido pode levar o modelo a generalizar mal.

Outros parâmetros podem ser ajustados de forma a controlar a profundidade máxima da árvore, fazendo com que o modelo treine mais rapidamente e utilize menos espaço no arquivo de salvamento, e ainda é possível definir um critério de parada antecipada.

Técnicas de randomização também são utilizadas para reduzir o *overfitting* e aumentar a velocidade.

É possível ainda definir o tipo de "*booster*" que será empregado pelo método: os tipos baseados em árvore *gbtree* ou *dart*, e *gblinear* baseado em uma função linear.

3.4 Trabalhos Relacionados

Muitos algoritmos de aprendizado de máquina, já foram utilizados para o propósito de detecção de anomalias em redes de dados, porém não foram encontrados artigos relacionados especificamente a detecção de anomalias que afetam os *switchs* a nível de camada 2, de toda forma o conhecimento adquirido nesse campo de investigação mostrou-se bastante relevante, pois as características dos *datasets* utilizados é semelhante. Dessa forma, abordaremos os principais trabalhos expondo os pontos fortes e fracos dos algoritmos utilizados, bem como as métricas obtidas ao tentar detectar as anomalias.

No trabalho de Yu, Jaehak, et al.(2008)[72], foi utilizada uma abordagem em duas etapas em que a primeira consistia na aplicação de um classificador SVM de uma classe, responsável por separar os dados normais dos dados de ataques. Na segunda etapa foi utilizado um SVM multiclasse com o intuito de identificar o ataque quanto a sua assinatura. Os resultados obtidos, para a classificação utilizando SVM de uma classe, foram 97.07% de acertos, os falsos positivos foram de 2.9% e os falsos negativos de 2.93% com o uso de 13 atributos, já para a segunda etapa que faz a classificação dos ataques obteve-se 99.4% de acertos, os falsos positivos foram de 1.8% e os falsos negativos de 0,6% com o uso de 5 atributos.

Feng, Wenying et al.(2014)[73] aborda as vantagens de se combinar um algoritmo SVM puro com o algoritmo CSOACN (do inglês *Clustering based on Self-Organized Ant Colony Network*), propondo o algoritmo CSVAC (do inglês *Combining Support Vectors with Ant Colony*) que foi aplicado ao *dataset* KDD99Cup[56], amplamente conhecido e utilizado para treinamento de algoritmos de detecção de anomalias. Esse algoritmo visa colher o melhor dos dois anteriores, onde SVM e CSOACN são duas fases interativas que são realizadas várias vezes. O SVM é usado para encontrar os vetores de suporte, e para gerar o hiperplano que separa dados normais e anormais, enquanto um CSOACN é usado para encontrar dados adicionados ao conjunto de treinamento SVM ativo, e finalmente gerar modelos para dados normais, bem como para cada classe de dados anormais. Os resultados obtidos com essa abordagem foram: taxa de acertos de 78.18% com 2.776% de falsos positivos e 0.3% de falsos negativos em um tempo de treinamento de 3.388 segundos.

Tao, Peiying et al.(2018)[74], trás em seu trabalho uma proposta de melhoria no uso de SVM para detecção de anomalias, que consiste na adoção de um algoritmo Genético para otimizar o desempenho quando se utiliza uma grande base de informações.

Assim, foi proposta uma otimização da probabilidade de cruzamento e a probabilidade de mutação do algoritmo genético, gerando a população utilizada para acelerar a busca na evolução inicial da população e acelerou-se a convergência do algoritmo na evolução posterior da população. Na etapa de seleção do conjunto ótimo de recursos, uma nova função de aptidão é proposta para diminuir a taxa de erro enquanto aumenta a taxa de verdadeiros positivos. Finalmente, os pesos dos recursos e os parâmetros do SVM são otimizados simultaneamente e a robustez do SVM é aprimorada.

Neste caso, a utilização de SVM associado ao algoritmo genético, para selecionar o subconjunto de recursos ideal mostrou uma acurácia de 99.75% e um tempo de classificação de 5.0781 segundos usando apenas 19 parâmetros, contra 99.56% de acurácia e 5.1875 segundos quando utiliza 41 parâmetros. Ainda foi apresentada uma taxa de detecção de 100% com 0.7% de falsos negativos e 0% de falsos positivos, para o melhor caso.

Devi, BS Kiruthika, et al.(2018)[75], trás um comparativo entre vários algoritmos e ao final foram elencados os algoritmos que mostraram os melhores resultados para 4 categorias. A primeira, fez uma comparação entre diferentes tipos de kernel para o algoritmo SVC, na segunda, foram testados diferentes algoritmos baseados em árvores de decisão, na terceira, foram analisadas diversas abordagens para o classificador MLP (do inglês *Multi-Layer Perceptron*) com relação a quantidade de camadas ocultas (do inglês *Hidden Layers*) e finalmente para representar a quarta categoria, apenas o algoritmo de Regressão Logística (do inglês *Logistic Regression*). Os resultados, levaram em conta apenas a acurácia dos algoritmos elencados como melhores em suas categorias, são eles: ExtraTreesClassifier, Logistic Regression, MLP com 50 camadas ocultas e Linear SVC, obtendo respectivamente, uma acurácia de 96.7%, 82%, 77,2% e 71%.

Sharafaldin, Iman, Arash Habibi Lashkari, e Ali A. Ghorbani.(2018)[76], ao proporem um novo conjunto de dados chamado CICIDS2017[77], fazem uso de alguns algoritmos de classificação para à avaliação e validação deste conjunto de dados. Os algoritmos utilizados são: KNN (*K-Nearest Neighbors*), RF (*Random Forest*), ID3, Adaboost, MLP, Naive-Bayes e QDA (*Quadratic Discriminant Analysis*) e foram avaliadas as métricas Precision(Pr), Recall(Rc) e F1-Measure(F1), o tempo de execução e teste também foi levado em consideração.

Foi observado, que com base na execução o KNN levou 1908,23 segundos e é o mais lento, mas ao contrário RF, é o mais rápido com tempo de execução de 74,39 segundos. Além disso, segundo a média ponderada das três métricas de avaliação (Pr, Rc, F1), a maior precisão pertence a KNN, RF e ID3. Considerando o tempo de execução e as métricas de avaliação RF é o melhor algoritmo com o menor tempo de execução e maior precisão, obtendo 98% na métrica Precision, 97% em Recall e F1-Measure, além de um tempo de 74.39 segundos.

Al-Naymat, Ghazi, Mouhammd Al-Kasassbeh, and Eshraq Al-Harwari.(2018)[78],

fazem uma análise do uso de três algoritmos em um conjunto de dados proposto pelos mesmos autores em 2016[79]. Este conjunto de dados, foi dividido em 5 grupos de acordo com a afinidade dos dados coletados e os ataques a serem identificados, assim foi proposto o uso de MLP, Random Forest e AdaboostM1. As métricas utilizadas foram Precision(Pr), Recall(Rc), F1-Measure(F1) e Acurácia, sendo que a avaliação dos autores foi dada principalmente sobre os valores de F1-Measure, pois representa uma média ponderada de Precision e Recall, ficando assim implícitas nos resultados.

O classificador Random Forest, obteve o maior índice de acerto com o grupo IP (100%) e com o grupo Interface (99,93%). A partir dos resultados, também constatou-se que entre os cinco grupos MIB, os grupos Interface e IP foram os grupos mais afetados por todos os tipos de ataque, enquanto os grupos ICMP, TCP e UDP foram os menos afetados. A conclusão geral, é que o uso dos algoritmos AdaboostM1 e Random Forest é uma abordagem muito eficaz para detecção de anomalias de rede com uma pequena vantagem para Random Forest no contexto global.

Boyar, O., M. E. Özen, and B. Metin.(2018)[16], apresenta em seu artigo uma abordagem de detecção de anomalias utilizando dados mais semelhantes aos utilizados neste trabalho. Fazendo uso de dados de camada 2 presentes nos *switchs* e coletados através do protocolo SNMP, coleta-se informações sobre estatísticas de pacotes trafegados por faixa de tamanho fornecidas pelo grupo RMON1. Para realizar a análise, foram utilizados três algoritmos: ANN (*Artificial Neural Network*), KNN e C5.0 em um conjunto de dados simulados, alternando entre baixa e alta carga de dados, simulando o funcionamento de uma rede. Os resultados deste experimento foram divididos em duas partes, uma para baixa outra para alta carga de dados. Para baixa carga, os resultados foram de 100% para todos os algoritmos, já para o cenário com alta carga de dados, que trás uma semelhança melhor com a realidade, nenhum algoritmo foi capaz de atingir 100%, sendo o melhor resultado obtido com ANN, que apresentou 99.45593% de acurácia, 100% de sensibilidade e 97.47475% de especificidade.

3.5 Considerações Finais

Conforme apresentado neste capítulo, os métodos de aprendizado de máquina podem ser classificadores ou regressores e existem vários métodos de aprendizado de máquina que podem ser utilizados para a classificação das anomalias em redes de computadores ou sistemas de detecção de ameaças.

Foram apresentados também formas de superar os problemas com relação aos conjuntos de dados desbalanceados, bem como um processo para tratamento dos dados a serem utilizados. Apresentou-se também métricas diferentes, para avaliação das técnicas elegendo as métricas mais adequadas aos dados e ao problema abordado, afim de ser

realizada uma comparação adequada à diferentes estudos. Ao final, são apresentados os trabalhos relacionados mais relevantes.

Todo o desenvolvimento deste estudo será detalhado no Capítulo 4.

4 Desenvolvimento

4.1 Considerações Iniciais

A questão principal deste trabalho, abordada no Capítulo 1, é a aplicação de técnicas de inteligência artificial, mais especificamente aprendizado de máquina, para realizar a detecção de anomalias em redes de computadores.

A abordagem utilizada faz uso de dados coletados em equipamentos em uso na instituição. Esses dados são compostos por dados normais e também por dados de ataques, que podem provocar as anomalias no ambiente de rede. Os dados foram cedidos pela Diretoria de Tecnologia da Informação, através de sua equipe de infraestrutura de telecomunicações.

Neste estudo de caso foi adotada uma abordagem de aprendizado de máquina com uma técnica de treinamento supervisionado classificador, ou seja, os dados utilizados para treinar os algoritmos escolhidos estavam etiquetados, ou classificados, como normais ou com uma das assinaturas de ataques citados na Seção 2.3. Dada a natureza da aplicação, optou-se por técnicas de classificação, onde o resultado dos algoritmos é sempre uma das classes à qual pertence determinado dado ao ser submetido à predição.

Devido à natureza das árvores de decisão e suas variações, existe uma afinidade intrínseca entre estes algoritmos e o problema abordado. Assim os algoritmos *Decision Tree*, *Random Forest*, *Extra Tree*, *Gradient Boosting*, *Extreme Gradient Boosting* e *Histogram Gradient Boosting* foram escolhidos para esta pesquisa.

Neste capítulo são descritos o problema abordado, a metodologia e a abordagem proposta utilizando os algoritmos de aprendizado de máquina.

4.2 Problema

O problema abordado neste trabalho consiste em diagnosticar de maneira preditiva as anomalias que podem ocorrer em *switchs* de uma infraestrutura de redes. Essas anomalias podem levar ao desencadeamento de uma série de problemas que escalam os níveis da hierarquia da infraestrutura de redes, levando a um deterioramento parcial ou completo desta infraestrutura.

A instituição, atualmente, não conta com nenhum sistema de alertas a respeito dessas falhas, sendo detectadas apenas quando os usuários manifestam alguma ocorrência ou insatisfação à equipe técnica. Já a detecção do ponto de falha, é ainda mais precário,

fazendo com que a equipe de campo faça o desligamento físico das conexões, afim de detectar qual ramo da rede está causando o problema ao ser religado.

Em face a essa situação, foi implementado um sistema de monitoramento dos ativos de rede, de forma a coletar informações dos *switchs*, porém este sistema não é capaz de perceber um problema antes que o mesmo comprometa o funcionamento da rede, ou que possa identificar qual o tipo de problema está causando a falha. Não há também um histórico a respeito dos problemas já ocorridos e nem relatórios a respeito do diagnóstico e das medidas de mitigação que foram adotadas em cada ocorrência.

Dessa forma, este estudo sugere que os dados coletados pelo sistema de monitoramento sejam submetidos a avaliação de um algoritmo de aprendizado de máquina, treinado de maneira a detectar alguns problemas já conhecidos, melhorando assim a detecção dessas falhas antes que comprometam a infraestrutura.

Os dados disponíveis nos *switchs* estão listados na Tabela 3, e são referentes a cada interface do *switch*. Dessa forma, de posse desses dados pode-se verificar a ocorrência de um problema de maneira mais granular. As interfaces de um *switch* podem ser do tipo entroncamento (do inglês *Trunk*), ou de acesso (do inglês *Access*). As interfaces de acesso são mais comuns nos *switchs* localizados mais próximos aos usuários, e podem ter no máximo dois dispositivos ativos conectados. As interfaces de entroncamento, são utilizadas para interligação de *switchs* a outros *switchs*, e normalmente são as conexões de *Up-Link* de um equipamento à um nível superior da hierarquia de infraestrutura de rede. No caso dos *switchs* de acesso, os *Up-Links* interliga-os a um *switch* de distribuição ou concentrador, onde geralmente só há conexões entre *switchs*.

Os esforços deste estudo estão concentrados em detectar as anomalias advindas das interfaces de acesso, uma vez que partem delas os dados dos dispositivos finais, que podem estar sendo utilizados para gerar um ataque, ou que possam estar comprometidos com algum tipo de código malicioso.

Este estudo ainda levou em consideração que tecnologias mais modernas como SDN (*Software Defined Network*), são inviáveis neste momento, devido às características dos equipamentos do parque atualmente em uso na instituição, visto que menos de 10% dos *switchs* possuem capacidade de utilizar SDN.

Os dados coletados via SNMP são processados e armazenados em um banco de dados pelo software de gerenciamento Zabbix[80], que atualmente está em uso na instituição. Para treinamento dos algoritmos, foram disponibilizados os dados coletados entre os dias 07/11/2022 e 14/11/2022. Foi feita a coleta de uma amostra por minuto, esses dados foram analisados e classificados como normais por um especialista.

Já os dados referentes aos ataques foram gerados em um ambiente controlado de laboratório em uma quantidade de 120 amostras por ataque, ou o equivalente a duas

horas, também a uma taxa de uma amostra por minuto.

A coleta, o pré-processamento, a transformação e análise dos dados coletados serão melhor explicados na Seção 4.3.

4.3 Metodologia

De acordo com o procedimento de classificação descrito na subseção 3.3.1, alguns passos devem ser seguidos para que os dados estejam adequados antes de serem utilizados. Para esta pesquisa, foi proposto um procedimento mais objetivo e alinhado com os recursos disponíveis.

Assim, como mostra a Figura 23, após o problema ser modelado é preciso entender se os dados possuem a relevância necessária para fornecer respostas à esse problema. Sabendo que o trabalho de Boyar, O., M. E. Özen, and B. Metin.(2018)[16] e também a ferramenta Cisco Storm Control[43] fazem uso das informações coletadas nas tabelas MIB dos *switchs* listadas na Tabela 3, então, pode-se inferir que esta relevância está confirmada.

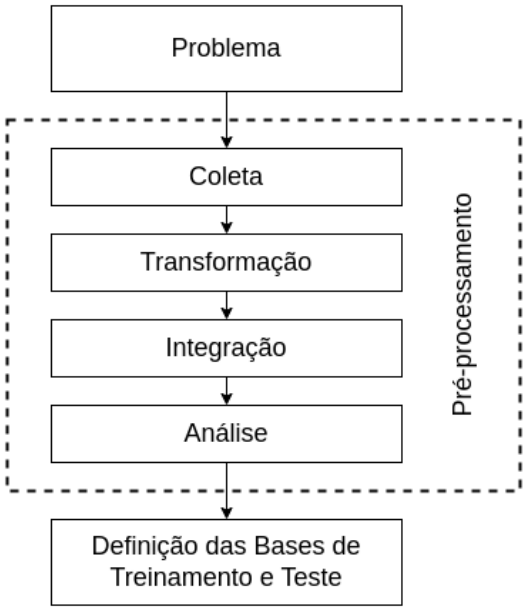


Figura 23 – *Fluxo de Pré-processamento dos Dados para Classificação*

4.3.1 Coleta de Dados

A coleta dos dados se dá através de consultas (do inglês *Queries*) SNMP aos *switchs*. Essas coletas são feitas pelo software Zabbix[80], que é um software livre, bastante completo, para monitoramento de ativos de rede que incluem, *switchs*, roteadores, *firewalls*, servidores ou qualquer outro tipo de equipamento que possa responder ao protocolo SNMP.

Para esta pesquisa, foram criados modelos (do inglês *templates*) para cada tipo diferente de *switch* disponível na instituição. Neste modelo são configurados os itens, que nada mais são do que a unidade básica de informação a ser coletada. Após a criação dos modelos, foram cadastrados os *switchs* que forneceram os dados, um dos modelos foi o Catalyst C9300L-24T-4X, que foi utilizado tanto na coleta dos dados normais como dos ataques realizados em laboratório. Este modelo possui 28 interfaces de conexão, sendo 24 de 1 Gigabit e 4 de 10 Gigabit. Trata-se de um equipamento de nível empresarial bastante robusto e tem a capacidade de fornecer os dados necessários.

Mesmo os *switchs* mais antigos, ou equipamentos de classe SOHO (*Small Office Home Office*) e ainda que sejam de outros fabricantes, que não a Cisco™, possuem a capacidade de fornecer esses dados, fazendo com que todos os equipamentos que compõe atualmente o parque da instituição, possam ser utilizados.

O sistema Zabbix, coleta as informações de maneira ininterrupta e os armazena em um banco de dados MySql, porém, estes dados estão disponíveis apenas via interface Web do sistema. Assim foi necessário a criação de uma *procedure* no banco de dados que fosse capaz de realizar a extração dos dados desejados.

Esta *procedure* é capaz de extrair os dados e exportar em um arquivo .CSV os dados referentes ao *switch*, que é chamado de *Host* pelo sistema Zabbix. Este *host* é identificado pelo seu endereço IP, como pode-se ver na Figura 24. Também são passados como parâmetros desta *procedure* a data, a hora inicial e a duração, bem como o nome que se deseja dar ao arquivo de saída.

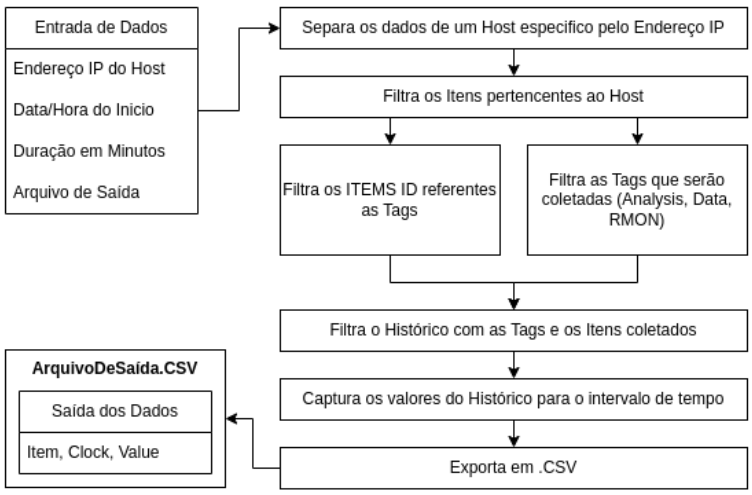


Figura 24 – *Fluxo de Captura dos Dados no Banco de Dados do Zabbix*

Os dados extraídos possuem três campos distintos:

- ***Item:*** Representa a identificação única do dado no sistema, é composta pelo nome da MIB (*Management Information Base*) e a interface do qual foi coletada;

- ***Clock:*** Trás a informação do tempo, do tipo *Unix Timestamp*, e representa a quantidade de segundos passados desde 01/01/1970 às 00:00:00. Esse valor indica o exato momento em que o dado foi coletado;

- ***Value:*** Trás o valor coletado ou o resultado de alguma transformação que possa ter passado antes de armazenado. Neste caso é sempre um valor inteiro que representa a diferença entre o valor coletado naquele instante e o valor coletado no instante anterior, ou seja, apenas o incremento ocorrido neste intervalo, que neste caso é de um minuto por padrão.

A quantidade de dados esperada como resposta à *procedure* é dada pela equação 4.1, onde o número de registros é igual ao produto do número de parâmetros (NP) pela duração (D), em minutos, e pela frequência (F) dos dados coletados por minuto. Porém, podem ocorrer algumas indisponibilidades durante a aquisição dos dados, tanto por parte do sistema Zabbix, quanto por parte do *switch*. Assim, a coleta dos dados pode não conter a quantidade esperada de registros, mas contém todos os dados que foram armazenados no intervalo de tempo passado à *procedure*.

$$Registros = NP \times (D \times F) \tag{4.1}$$

Os dados extraídos são armazenados em um arquivo no formato .CSV ou valores separados por vírgula (do inglês *Comma Separated Values*), no entanto esses dados ainda não estão em um formato adequado sendo necessário submetê-los a uma transformação.

4.3.2 Transformação dos Dados

Os dados obtidos na fase de coleta precisam passar por uma transformação para que estejam em um formato adequado e possam ser submetidos ao método de classificação. A transformação dos dados foi feita utilizando um *script* escrito em linguagem Python v3.8.16[81] com o auxílio das bibliotecas Pandas v1.3.5[82] e NumPy v1.21.6[83].

Essa transformação é composta por dois passos, o ajuste do tempo e o *pivoting* ou giro. O ajuste no tempo é necessário devido a um fenômeno que pode ocorrer no momento da captura dos dados, isso ocorre quando o sistema Zabbix recebe a resposta ao comando SNMPGET, enviado ao *switch*.

Quando a requisição é feita para uma grande quantidade de itens, durante o processo de resposta os mesmos podem ocorrer no limiar entre um segundo e outro causando uma defasagem no armazenamento em banco. Isso ocorre devido ao campo *clock* registrar o segundo exato da chegada da resposta. Para ajustar essa defasagem de tempo é feito um arredondamento do campo *clock* para um mesmo minuto, ignorando os segundos, dessa forma esse inconveniente é superado.

O segundo passo, conhecido como *pivoting*, consiste em colocar os itens coletados em um mesmo minuto, como um único registro, assim tem-se um registro que agrupa todos os valores para cada minuto. Esse processo de giro é exemplificado na Figura 25, salientando que os dados desta figura são fictícios.

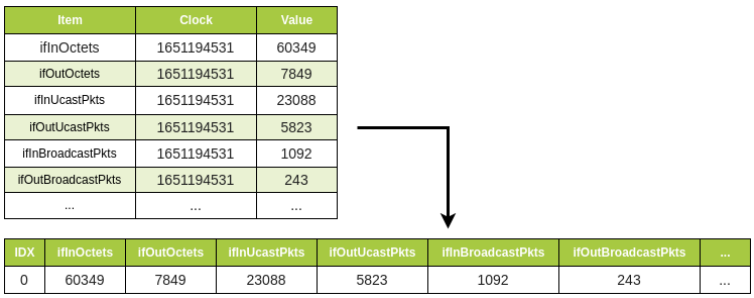


Figura 25 – *Exemplo de Pivoting de dados*

4.3.3 Integração

Os dados desta pesquisa estão divididos em duas partes, os dados "Normais" e os dados de "Ataques". Esses dados, apesar de terem sido obtidos de equipamentos idênticos, foram obtidos de maneira separada.

Para o caso dos dados normais, eles foram obtidos em um ambiente de produção e foram capturados em uma das interfaces do *switch* que conecta um computador e também um telefone VoIP.

No período de captura desses dados, a equipe técnica da instituição não detectou qualquer anormalidade advinda desse *switch*. Também foram analisados os registros de *log* do *firewall* afim de verificar se houve qualquer registro de anomalias advindas dos endereços IP, tanto do computador quanto do telefone. Também foi tomado o cuidado de que o computador estivesse com o software anti-vírus atualizado e foram realizadas inspeções diárias para verificar a existência de vírus.

O *dataset* de dados normais é composto pelos dados coletados entre às 00:00hr de 07/11/2022 e 00:00hr de 14/11/2022, assim espera-se uma quantidade máxima de 10080 registros. Foi adicionada uma coluna ao final deste *dataset* referente à classe dos dados, totalmente preenchida com a informação "normal".

Já os dados dos ataques foram coletados em um ambiente de laboratório, isolado, sem qualquer interferência de dados advindos da rede de produção. Participaram deste ambiente apenas o equipamento atacante, o equipamento alvo do ataque e o *switch* que faz a interligação. O equipamento responsável por capturar os dados da interface foi interligado ao *switch* através de uma interface exclusiva de forma a não interferir no tráfego entre o atacante e o alvo. Estes dados foram capturados à taxa de uma amostra

por minuto e feita por duas horas para cada ataque. Também foi adicionada uma coluna referente à classe preenchida com o nome da assinatura do ataque em questão.

A integração entre os dados normais e os dados dos ataques foi feita sobrepondo os dados dos ataques aos dados normais através de uma somatória entre eles, mas mantendo a etiqueta do ataque. O processo de integração pode ser visto na Figura 26.

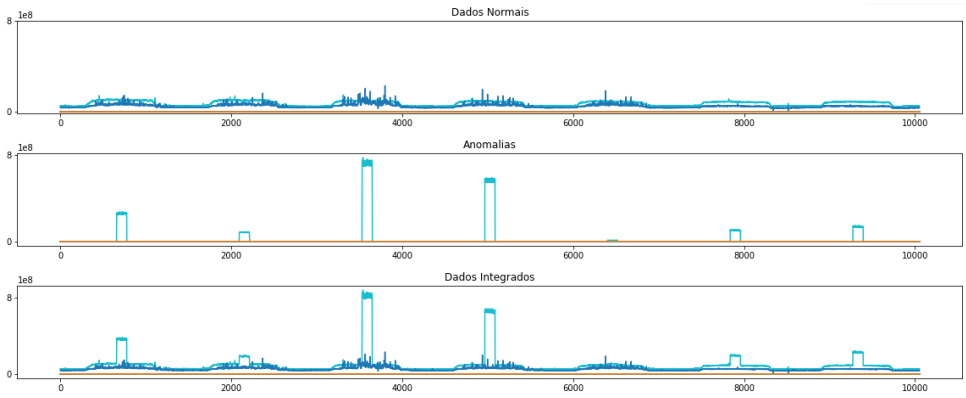


Figura 26 – *Integração entre Dados Normais e Assinaturas de Ataques*

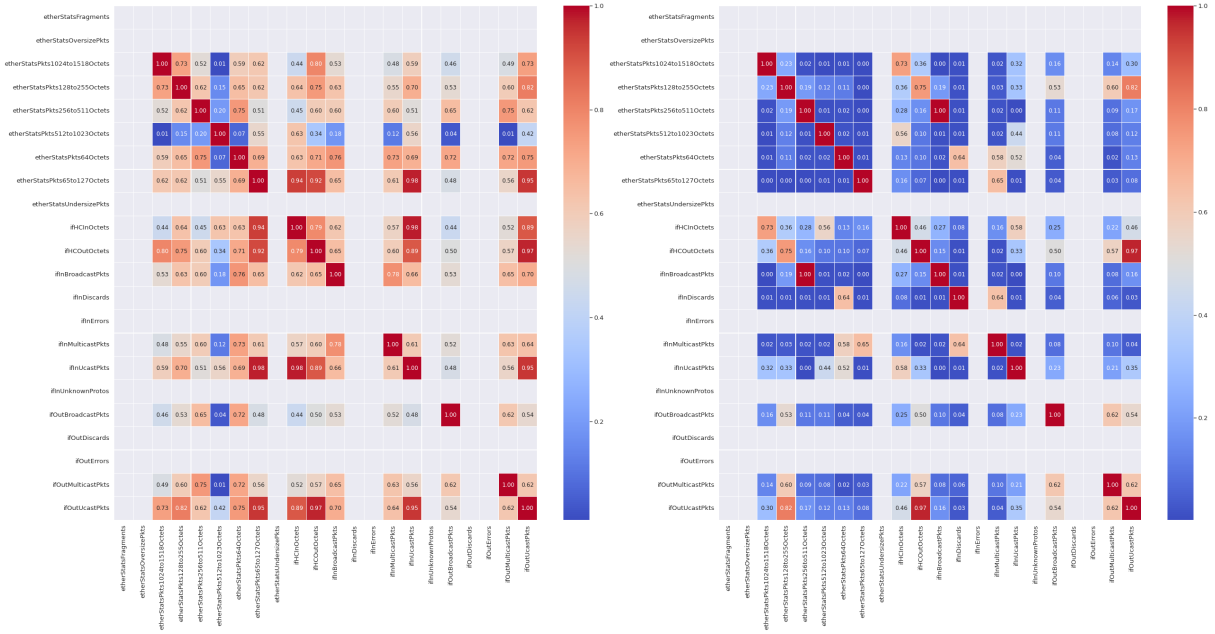
Os dados que compõem o *dataset*, contemplam dados das MIB's SNMP e RMON, diferentemente do trabalho de Boyar, O., M. E. Özen, and B. Metin.(2018)[16] cujos dados são apenas da MIB RMON, neste trabalho optou-se por incluir os dados utilizados pela ferramenta Cisco Storm Control[43] que pertencem a MIB SNMP, e diante da falta de um *dataset* com essas características que pudesse conter os dados referentes as anomalias estudadas, optou-se por criar um dataset próprio, cujo conteúdo está disponibilizado em um repositório público[84]. Desta forma, ao final tem-se um *dataset* bastante realista cujos dados são parcialmente gerados em laboratório e os dados classificados como normais, são oriundos de um ambiente real.

4.3.4 Análise dos Dados

Os dados das amostras para este tipo de ambiente podem trazer dados com as mais diversas características sem que sejam necessariamente anomalias. Um dado normal é aquele decorrente do uso de um equipamento, livre de vírus e outros softwares maliciosos, por um determinado usuário sob condições normais de uso.

As características de tráfego podem variar de acordo com o tipo de equipamento, por exemplo, um telefone VoIP tem uma característica de tráfego diferente de uma câmera IP, que por sua vez também é diferente de uma impressora e também de um computador. Além disso, é importante ter em mente que usuários diferentes podem ter características de tráfego diferentes, dependendo dos tipos de operações que realizam em seus computadores.

Dessa forma, quando utiliza-se dados com uma granularidade baixa, a nível de host, também evita-se que tenhamos que reajustar o modelo em função de variações nos



(a) Dados Normais (b) Dados Normais + Ataques

Figura 27 – Mapa de calor dos *Datasets* com dados normais e com dados integrados.

dados ao longo do tempo, como sugere o conceito de *Concept Drifting*[85].

Uma das principais preocupações quando se utiliza modelos de aprendizado de máquina é o problema da dimensionalidade. Isso ocorre quando dois ou mais parâmetros possuem uma alta correlação entre si, em decorrência disso, pode-se eliminar um deles e ainda assim obter o mesmo resultado.

Um ponto favorável à esta pesquisa é que os dados de ataques costumam ter características bastante diferentes de um tráfego convencional, assim quando um mapa de calor é analisado e as relações entre os diversos parâmetros coletados, pode-se perceber que quando ocorre alguma anomalia, apenas um conjunto de parâmetros é afetado, como pode ser visto na Figura 27.

Um dos *datasets* utilizados é resultado da integração dos dados coletados em condições normais com os *datasets* coletados em laboratório, contendo os dados dos ataques, além de um *dataset* somente com os dados normais.

Ao analisar os dados deste mapa de calor, percebe-se que alguns parâmetros não possuem qualquer relação com outros, além de parâmetros com valores constantes ou até mesmo totalmente preenchidos com zeros. Por outro lado, alguns parâmetros estão correlacionados, o que levaria a pensar que algum deles poderia ser dispensado, reduzindo assim a dimensionalidade, diminuindo a quantidade de parâmetros utilizados para o treinamento dos modelos, o que é benéfico para as árvores de decisão, por exemplo.

No entanto, não é possível afirmar que esses parâmetros nunca serão afetados por

algum tipo de ataque, pois observando o parâmetro **ifInDiscards** em um *dataset* de dados normais, ele é totalmente preenchido com zeros, já quando observa-se o *dataset* integrado ele possui uma relação de 64% com **etherStatsPkts64Octets** e **ifInUcastPkts**.

Assim, para fins desta pesquisa optou-se por não dispensar nenhum dos parâmetros presentes no *dataset*, sob o risco de perder alguma característica própria de um ataque, o que poderia ser determinante para que ele fosse identificado.

O modelo PCA, de acordo com o *Scikit Learn*[54] é uma técnica de aprendizado de máquina não supervisionada. A decomposição dos dados, via análise com componentes principais PCA (*Principal Component Analysis*)[86], permite a apresentação gráfica das distribuições dos casos em um espaço bidimensional, possibilitando visualizar os agrupamentos por tipo e previamente encontrando possíveis padrões a serem identificados com as técnicas de aprendizado de máquina supervisionada.

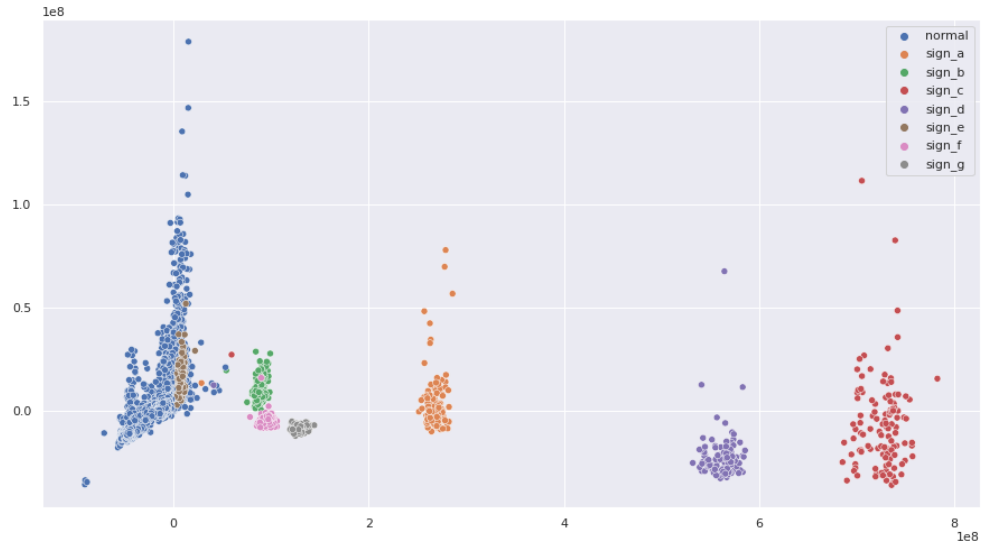


Figura 28 – *Análise com Componentes Principais PCA*

A análise utilizando PCA mostra que as assinaturas dos ataques podem ser visualmente identificadas, com certa facilidade, pois, mesmo utilizando uma técnica de aprendizado de máquina não supervisionada, as classes são claramente separadas, como pode-se ver na Figura 28.

4.4 Definição das Bases de Treinamento e Teste

Após a análise dos dados é necessário definir as bases de dados que serão utilizadas para realizar os treinamentos e os testes dos modelos de aprendizado de máquina.

Ao realizar os testes com os mesmos dados utilizados para treinamento é cometido um erro de metodologia, pois ao fazer testes com os mesmos dados se obtém uma predição perfeita, o que não é verdade pois ocorre um *overfitting* do modelo. Segundo Pedregosa et al.[54], uma alternativa para evitar essa situação é a utilização da validação cruzada (do inglês *Cross Validation*), conforme a Figura 29.

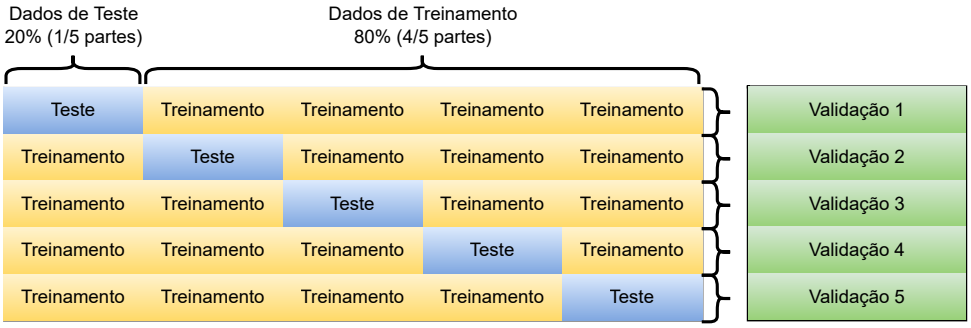


Figura 29 – Validação Cruzada
Adaptado de Pedregosa et al.(2011)[54]

A divisão da base de dados separando o conjunto de dados em dois subconjuntos de treinamento e teste pode ser feita utilizando o método K-Fold[87]. Esse método realiza a divisão do conjunto de dados em k partes, ou *folds*, e fornece k conjuntos mutuamente exclusivos com *k-1* partes para treinamento e uma parte para teste.

A ferramenta Scikit-Learn[54] dispõe de uma função chamada **cross_val_score**, que realiza o levantamento de um *score* utilizando uma validação cruzada do tipo *K-Fold*.

Segundo o estudo de Kohavi, Ron(1995)[87], foram feitos vários experimentos com dados de diversos tipos e conjuntos de dados de diferentes tamanhos e chegou-se a conclusão de que uma validação cruzada que divide o conjunto em 10 partes($K\text{-}Fold = 10$), foi suficientemente bom para se alcançar uma ótima acurácia em diversos modelos de aprendizado de máquina supervisionado.

Dessa forma, para fins de verificação das métricas dos modelos será utilizada uma validação cruzada com $K\text{-}Fold = 10$, estabelecendo assim as bases de treinamento e teste para esta pesquisa.

4.5 Considerações Finais

Neste capítulo foi descrito o problema apresentado nesta pesquisa, bem como os trabalhos relacionados, além da abordagem proposta para a metodologia de obtenção e tratamento dos dados que servem de base para a identificação das anomalias utilizando técnicas de aprendizado de máquina.

Ainda que tenha sido apresentado um estudo cujos dados são parcialmente comuns

à este estudo, não foi encontrado nenhum estudo voltado especificamente ao propósito de detecção de falhas contra ataques de negação de serviço em camada 2, do modelo OSI.

Também foi possível analisar as correlações entre os parâmetros de forma a entender quantos e quais são os mais relevantes para a diferenciação entre os dados considerados normais ou anomalias. Analisou-se também os dados dos *datasets* do ponto de vista de agrupamento entre os diversos ataques e dados normais.

No Capítulo 5 serão apresentados o planejamento e a execução dos experimentos, bem como os resultados e as discussões.

5 Experimentos e Discussões

5.1 Considerações Iniciais

Ultrapassados os desafios descritos no Capítulo 4, pode-se realizar experimentações utilizando as técnicas de aprendizado de máquina classificador supervisionado aplicados aos dados levantados para esta pesquisa.

O ambiente utilizado foi o Google Colaboratory™[88], que é um ambiente em nuvem oferecido em uma versão gratuita pela empresa Google™e permite a execução de código em linguagem Python, bem como todas as ferramentas necessárias para a realização dos experimentos utilizando aprendizado de máquina.

Como parte deste experimento foi avaliada a viabilidade de se utilizar as técnicas de balanceamento dos dados, em seguida é apresentado o planejamento dos experimentos e seus respectivos resultados. As discussões são apresentadas para cada técnica elencada e ao final uma comparação entre todas as técnicas.

5.2 Bases de Dados

5.2.1 Validação das bases de teste

Como parte essencial desta pesquisa, foi levantada na Subseção 3.3.2, a necessidade de realização de um balanceamento dos dados que compõem as diversas classes desse conjunto dos dados.

Dessa forma, realizou-se um levantamento comparativo das métricas definidas, de maneira a verificar a viabilidade de se utilizar um conjunto de dados balanceado através das técnicas de sub amostragem (*Undersampling*), sobre amostragem (*Oversampling - SMOTE*) ou ainda Neutro, sem realizar nenhum balanceamento.

Os conjuntos de dados submetidos a análise estão dispostos de acordo com a Tabela 7.

Tabela 7 – Técnicas de Balanceamento de Dados com número de amostras

Técnica/Classes	Normais	Assinatura A	Assinatura B	Assinatura C	Assinatura D	Assinatura E	Assinatura F	Assinatura G	Total
Undersampling	119	119	119	119	119	119	119	119	952
Neutro	9214	120	120	120	121	119	120	120	10054
Oversampling	9214	9214	9214	9214	9214	9214	9214	9214	73712

O conjunto de dados *Undersampling* é composto de 8 subconjuntos de 119 amostras. Nos casos em que a quantidade de amostras de uma classe é superior à quantidade

de amostras da menor classe, registros são subtraídos aleatoriamente de forma a fazer com que todas fiquem iguais.

Para o conjunto de dados *Oversampling*, quando os subconjuntos são menores que o maior subconjunto, então estes são complementados utilizando dados gerados sinteticamente utilizando a técnica *Oversampling*, ou *SMOTE*, já mencionada.

Afim de realizar uma comparação e avaliar os possíveis impactos, realizou-se as análises de duas técnicas de aprendizado de máquina, *Random Forest* e *Decision Tree*. Utilizou-se as técnicas com a métrica *f1_weighted* mais alta e mais baixa, entre as técnicas escolhidas, submetendo-as aos três conjuntos de dados avaliando alterações nas métricas *f1_weighted* e Tempo, como pode-se ver na Tabela 8.

Tabela 8 – Comparação entre as Técnicas de Balanceamento de Dados

Técnica	Conjunto	Tempo	f1_weighted	balanced_accuracy	precision_weighted	recall_weighted
RandomForestClassifier	smote (73712)	13.713611	0.999905	0.999905	0.999905	0.999905
DecisionTreeClassifier	smote (73712)	0.846222	0.999810	0.999810	0.999812	0.999810
RandomForestClassifier	neutro (10054)	1.694439	0.999488	0.994792	0.999506	0.999503
DecisionTreeClassifier	neutro (10054)	0.070706	0.998899	0.991626	0.998954	0.998906
RandomForestClassifier	undersampling (952)	0.463383	0.994728	0.994792	0.995291	0.994748
DecisionTreeClassifier	undersampling (952)	0.007302	0.992578	0.992614	0.993209	0.992643

Observando os dados obtidos na métrica *f1_weighted*, percebe-se que tanto a técnica *Random Forest* quanto a *Decision Tree*, obtiveram pioras ao serem submetidas ao conjunto de dados *Undersampling*.

Já a comparação entre os conjuntos Neutro e SMOTE, percebeu-se um ganho muito discreto. Para a técnica *Decision Tree* o ganho foi de 0,000911 ao se utilizar o conjunto *SMOTE* em detrimento do Neutro, porém, houve um aumento de 11,97 vezes no tempo de treinamento.

No caso da técnica *Random Forest* o ganho foi de 0,000417, também ao utilizar o conjunto *SMOTE* em detrimento do Neutro, havendo também um aumento no tempo de treinamento na ordem de 8,09 vezes.

Em face a esta análise, optou-se por não utilizar qualquer das técnicas de balanceamento de dados, por entender que o ganho obtido na métrica *f1_weighted* não se justifica em função do aumento expressivo do tempo de treinamento e também por incluir um processo a mais na preparação dos dados, cujo tempo não foi computado.

5.2.2 Definição das bases de teste

As bases de dados[84] que compuseram os experimentos foram coletadas em 4 locais distintos, sendo que uma das bases foi utilizada para os treinamentos e validação, enquanto as outras três foram utilizadas para os testes de predição.

Como as bases de testes de predição não são apresentadas ao processo de treinamento, espera-se que os resultados sejam bastante próximos, o que validaria o processo de treinamento. As características das bases são mostradas na Tabela 9.

Tabela 9 – Bases de Treinamento e Predição

Local/Classes	Normais	Assinatura A	Assinatura B	Assinatura C	Assinatura D	Assinatura E	Assinatura F	Assinatura G	Total	Propósito
Local A	9214	120	120	120	121	119	120	120	10054	Treinamento
Local B	9233	120	120	120	121	119	120	120	10073	Predição
Local C	9239	120	120	120	121	119	120	120	10079	Predição
Local D	9216	120	120	120	121	119	120	120	10056	Predição

Os dados dos diversos locais foram coletados entre 00:00hr de 07/11/2022 e 00:00hr de 14/11/2022 e possuem características ligeiramente diferentes devido ao uso que se faz dos computadores conectados em cada local. As pequenas diferenças na quantidade de amostras se explica devido à falhas na coleta, entre o *switch* e o software Zabbix, por motivos diversos, alheios a esta pesquisa.

Os dados dos ataques foram adicionados aos dados dos locais usados nos teste de predição da mesma forma como foram adicionados aos dados de treinamento.

5.3 Experimentos

Inicialmente os métodos de aprendizado de máquina foram submetidos à ferramenta *GridSearchCV* do *Scikit-Learn*, inspirada em Larochelle, Hugo, et al.(2007)[89] para que fosse encontrado de maneira empírica, porém automatizada, os parâmetros mais adequados a cada algoritmo. Já para o método *Gradient Boosting*, foi utilizada a ferramenta *RandomizedSearchCV*[90] devido ao longo tempo de treinamento.

Com base nos dados utilizados para treinamento, local A, são feitas recomendações dos parâmetros a serem utilizados para treinamento do método de forma a obter-se o melhor valor para a métrica informada como alvo.

Alguns parâmetros fixos foram definidos para todos os modelos, conforme as estratégias adotadas e alguns seguem recomendações da literatura, conforme descritas:

- **random_state** *Random State* é um valor que serve como semente (do inglês *seed*) para geração dos números aleatórios utilizados no algoritmo. Para que todos os algoritmos tivessem a aleatoriedade idêntica, definiu-se a semente com o valor 42.
- **cv** O parâmetro *Cross Validation* foi definido em 10 e informa a quantidade de vezes em que será realizada a validação cruzada. Uma explicação mais detalhada pode ser obtida na Seção 4.4.
- **n_jobs** *Number of Jobs* informa à ferramenta *GridSearchCV* quantos processadores podem ser utilizados para realizar o processamento paralelo de tarefas, quando pos-

sível. O valor utilizado foi -1 e indica que podem ser utilizados tantos processadores quanto estiverem disponíveis.

- **max_depth** Este parâmetro define a profundidade da árvore e foi definido em 20. Muitos métodos não definem um valor fixo por padrão, fazendo com que a árvore cresça indefinidamente até que outros critérios façam com que a árvore seja estabelecida.

  Para esta pesquisa um limite no crescimento da árvore é desejável, pois limita o tamanho do arquivo de salvamento e mantêm o tempo de treinamento, de certa maneira, estável.

Então realizou-se os experimentos para cada técnica de aprendizado de máquina, de maneira individual, permitindo verificar o comportamento de cada método aplicado aos locais de teste, apurando uma média entre eles.

5.3.1 Decision Tree

Para o método *Decision Tree* foram experimentados os seguintes parâmetros que podem ser modificados de maneira a ajustar o comportamento do algoritmo.

- **criterion** Este parâmetro é utilizado pelo algoritmo para decidir como serão as divisões dos ramos da árvore e pode ser definido entre *gini* ou *entropy*. O melhor valor obtido foi *entropy*, que mede a perda de log entre os rótulos verdadeiros e as previsões probabilísticas do modelo.

- **splitter** Define a estratégia para a divisão de um nó. Os valores possíveis são *best* e *random*. O melhor valor obtido foi *random*, onde a estratégia para divisão aleatória foi a mais bem sucedida.

Dessa forma, os experimentos utilizando os dados para testes de predição são apresentados na Tabela 10 e a matriz de confusão pode ser vista na Figura 30.

Tabela 10 – Predição utilizando *Decision Tree*

Conjunto (Tamanho)	Tempo Predição (s)	F1 Ponderada	Acurácia Balanceada	Precisão Ponderada	Rechamada Ponderada
Local B (10073)	0.00158	98.88 %	91.24 %	99.72 %	98.53 %
Local C (10079)	0.00145	99.18 %	92.82 %	99.97 %	98.78 %
Local D (10056)	0.00173	96.33 %	83.0 %	97.45 %	96.53 %
Média	0.00159	98.13 %	89.02 %	99.05 %	97.95 %

O método da Árvore de Decisão foi capaz de atingir uma média de 98.13% para a métrica F1 Ponderada, porém para a métrica de Acurácia Balanceada o índice obtido foi de apenas 89.02% que pode ser explicado pela dificuldade de predição da anomalia

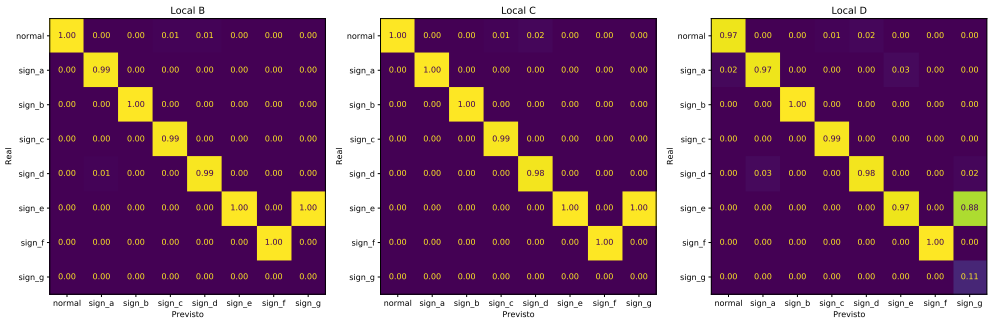


Figura 30 – Matriz de Confusão *Decision Tree*

representada pela classe **sign_g** em todos os cenários de predição, como pode ser visto na matriz de confusão.

Por outro lado houve pouca ou quase nenhuma confusão entre as anomalias e os dados normais.

5.3.2 Random Forest

O método *Random Forest* foi explorado com os seguintes parâmetros e os resultados das predições podem ser vistos na Tabela 11.

- **criterion** Este parâmetro é utilizado pelo algoritmo para decidir como serão as divisões dos ramos da árvore e pode ser definido entre *gini* ou *entropy*. O melhor valor obtido foi *gini*, que mede a impureza da divisão de um nó, como explicado na Subseção 3.3.4.1.
- **n_estimators** O número de estimadores representa a quantidade de árvores que irão compor a floresta. O melhor valor foi obtido com 22 árvores.
- **max_features** Representa o número de recursos que serão utilizados para se obter a melhor divisão de um nó na árvore. O melhor valor foi obtido com *sqrt*, ou seja, o valor máximo de recursos é igual a raiz quadrada do número de recursos disponíveis.
- **bootstrap** Indica se todos os dados serão utilizados na construção das árvores da floresta. O melhor valor foi *True* ou sim para o uso de *bootstrap*, que é o padrão para este método.

O método *Random Forest* foi capaz de atingir uma média de 99.87% para a métrica F1 Ponderada e para métrica de Acurácia Balanceada o índice obtido foi de 99.23%, sendo capaz de distinguir de maneira quase perfeita todos os cenários de anomalia, como pode ser visto na matriz de confusão representada na Figura 20.

Tabela 11 – Predição utilizando *Random Forest*

Conjunto (Tamanho)	Tempo Predição (s)	F1 Ponderada	Acurácia Balanceada	Precisão Ponderada	Rechamada Ponderada
Local B (10073)	0.02479	99.91 %	99.59 %	99.91 %	99.91 %
Local C (10079)	0.02289	99.95 %	99.99 %	99.95 %	99.95 %
Local D (10056)	0.02446	99.76 %	98.1 %	99.76 %	99.76 %
Média	0.02404	99.87 %	99.23 %	99.87 %	99.87 %

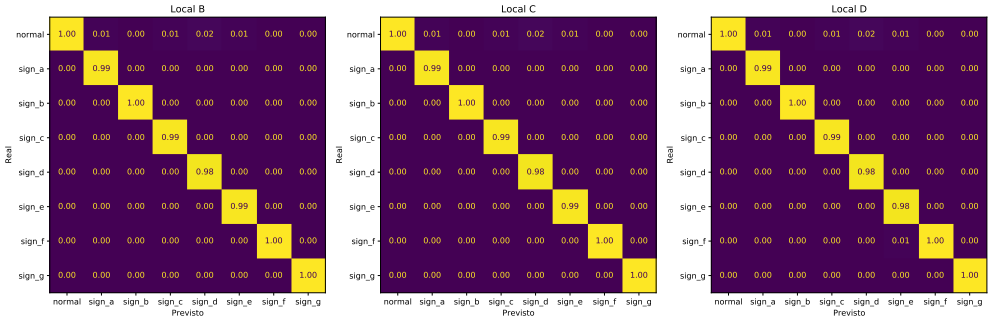


Figura 31 – Matriz de Confusão *Random Forest*

Os casos previstos como Normais foram assertivos, sendo que apenas algumas previsões de anomalias acabaram confundidas com dados normais. Esses dados representam falsos positivos e representam 2% ou menos das cerca de 120 amostras da anomalia. A matriz de confusão é normalizada pelas colunas de predição, ficando entre 2 a 3 amostras de um universo de mais de dez mil amostras.

5.3.3 Extra Trees Classifier

O método *Extra Trees* foi explorado com os seguintes parâmetros e os resultados das predições podem ser vistos na Tabela 12.

- **criterion** Este parâmetro é utilizado pelo algoritmo para decidir como serão as divisões dos ramos da árvore e pode ser definido entre *gini* ou *entropy*. O melhor valor obtido foi *gini*, que mede a impureza da divisão de um nó, como explicado na Subseção 3.3.4.1.

- **n_estimators** O número de estimadores representa a quantidade de árvores que irão compor a floresta. O melhor valor foi obtido com 22 árvores.

- **max_features** Representa o número de recursos que serão utilizados para se obter a melhor divisão de um nó na árvore. O melhor valor foi obtido com **none**, ou seja, o valor máximo de recursos é igual ao número de recursos disponíveis.

- **bootstrap** Indica se todos os dados serão utilizados na construção das árvores da floresta. O melhor valor foi *True* ou sim para o uso de *bootstrap*, o padrão para este

método é *False*.

Tabela 12 – Predição utilizando *Extra Trees*

Conjunto (Tamanho)	Tempo Predição (s)	F1 Ponderada	Acurácia Balanceada	Precisão Ponderada	Rechamada Ponderada
Local B (10073)	0.03133	99.97 %	99.89 %	99.97 %	99.97 %
Local C (10079)	0.02682	99.97 %	100.0 %	99.97 %	99.97 %
Local D (10056)	0.02381	99.69 %	97.58 %	99.7 %	99.7 %
Média	0.02732	99.88 %	99.16 %	99.88 %	99.88 %

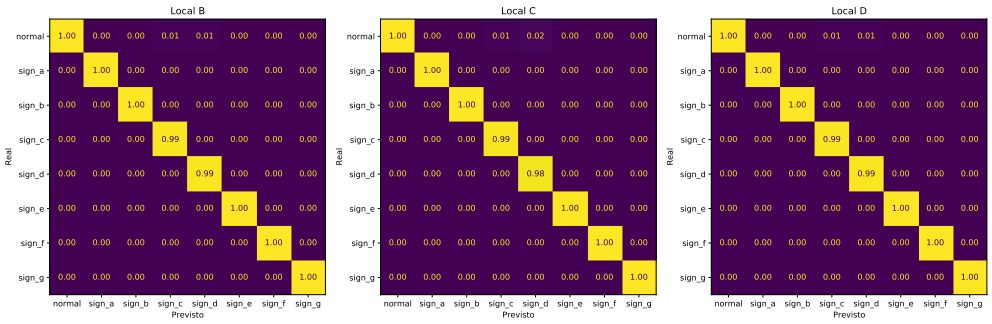


Figura 32 – Matriz de Confusão *Extra Trees*

O método *Extra Trees Classifier* obteve uma média de 99.88% para a métrica F1 Ponderada, e para métrica de Acurácia Balanceada obteve 99.16%, e assim como *Random Forest* também foi capaz de distinguir de maneira quase perfeita todos os cenários de anomalia, como pode ser visto na matriz de confusão representada na Figura 32.

Os casos previstos como Normais e anomalias ficaram muito próximos aos obtidos para *Random Forest*, sob a mesma justificativa.

5.3.4 Gradient Boosting

Para o método *Gradient Boosting* foi necessário utilizar uma ferramenta alternativa ao *GridSearchCV* devido ao número de parâmetros testados e ao elevado tempo do método. Assim utilizou-se a ferramenta *RandomizedSearchCV*, que faz uma busca de maneira aleatória.

Dessa forma, o método *Gradient Boosting* foi explorado com os seguintes parâmetros e os resultados das predições podem ser vistos na Tabela 13.

- **criterion** Este parâmetro é utilizado pelo algoritmo para decidir como serão as divisões dos ramos da árvore e pode ser definido entre *friedman_mse* ou *squared_error'*. O melhor valor obtido foi *squared_error'*, que mede o erro quadrático médio.

- **n_estimators** O número de estimadores representa a quantidade de árvores que irão compor a floresta. O melhor valor foi obtido com 100 árvores, que é o padrão para este método.

- **max_features** Representa o número de recursos que serão utilizados para se obter a melhor divisão de um nó na árvore. O melhor valor foi obtido com log_2, ou seja, o valor máximo de recursos é igual a log_2 do número de recursos disponíveis.

Tabela 13 – Predição utilizando *Gradient Boosting*

Conjunto (Tamanho)	Tempo Predição (s)	F1 Ponderada	Acurácia Balanceada	Precisão Ponderada	Rechamada Ponderada
Local B (10073)	0.09936	99.87 %	99.18 %	99.87 %	99.87 %
Local C (10079)	0.07979	99.33 %	96.61 %	99.9 %	98.79 %
Local D (10056)	0.08238	99.52 %	97.39 %	99.55 %	99.52 %
Média	0.08718	99.57 %	97.73 %	99.77 %	99.39 %

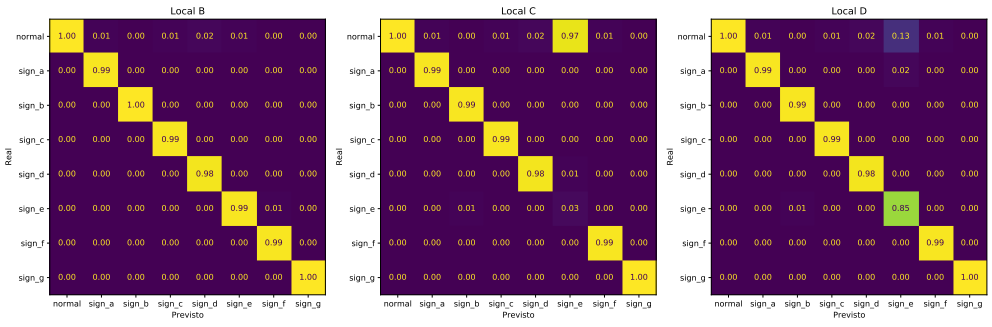


Figura 33 – Matriz de Confusão *Gradient Boosting*

As médias obtidas pelo método *Gradient Boosting* foram de 99.57% para a métrica F1 Ponderada e, 97.73% para a métrica de Acurácia Balanceada.

No entanto, como pode ser visto na matriz de confusão representada na Figura 33, o método só obteve um desempenho satisfatório com relação ao Local A, não sendo capaz de identificar a anomalia **sign_e** no Local C, identificando-a como Normal em 97% das 119 amostras. No Local D também houve problemas com a anomalia **sign_e** onde 13% foram identificados como Normal e 2% foram identificados como **sign_a**.

5.3.5 Extreme Gradient Boosting

O método *Extreme Gradient Boosting* ou *XGBoost* foi explorado com os seguintes parâmetros e os resultados das predições podem ser vistos na Tabela 14.

- **n_estimators** O número de estimadores representa a quantidade de árvores que irão compor a floresta. O melhor valor foi obtido com 50 árvores, o padrão para este método é 100.

- **learning_rate** Representa a taxa de aprendizado ou encolhimento. O ajuste dessa taxa é um fator de multiplicação para os valores das folhas. Os valores variam de 0 a 1 e o melhor valor foi obtido com 0.3, que é o padrão para este método. A taxa de aprendizagem determina o quão rápido um método aprende, sendo que quanto mais lentamente o método aprende mais robusto se torna o modelo.

Tabela 14 – Predição utilizando *Extreme Gradient Boosting*

Conjunto (Tamanho)	Tempo Predição (s)	F1 Ponderada	Acurácia Balanceada	Precisão Ponderada	Rechamada Ponderada
Local B (10073)	0.06351	99.48 %	95.69 %	99.49 %	99.49 %
Local C (10079)	0.05949	99.94 %	99.89 %	99.94 %	99.94 %
Local D (10056)	0.06227	97.16 %	89.37 %	97.65 %	97.62 %
Média	0.06176	98.86 %	94.98 %	99.03 %	99.02 %

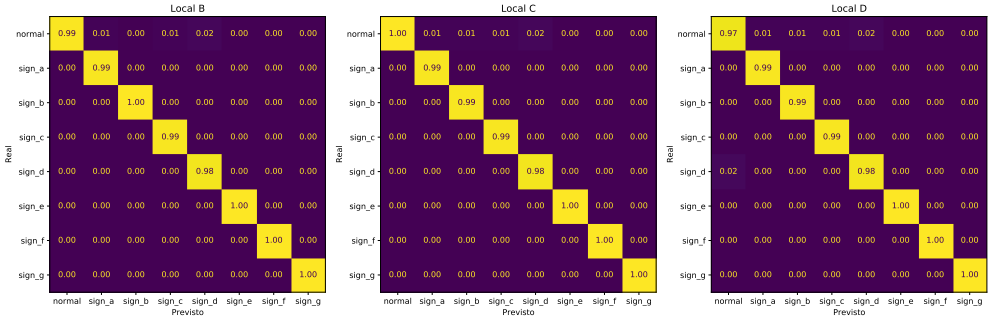


Figura 34 – Matriz de Confusão *Extreme Gradient Boosting*

O método *Extreme Gradient Boosting* também foi um dos métodos capazes de distinguir quase perfeitamente todos os cenários de anomalia, sendo que os índices obtidos tiveram uma média de 98.86% para a métrica F1 Ponderada e, para métrica de Acurácia Balanceada, o índice obtido foi de 94.98%, como pode ser visto na matriz de confusão representada na Figura 34.

Os casos previstos como Normais foram assertivos, sendo que apenas algumas previsões de anomalias acabaram confundidas com dados normais. Esses dados representam falsos positivos e representam 2% ou menos das cerca de 120 amostras da anomalia, como a matriz de confusão é normalizada pelas colunas de predição, ficando entre 2 a 3 amostras de um universo de mais de dez mil amostras.

5.3.6 Histogram Gradient Boosting

O método *Histogram Gradient Boosting* foi explorado com os seguintes parâmetros e os resultados das predições podem ser vistos na Tabela 15.

- **max_bins** O número máximo de *bins* estimadores representa a quantidade de árvores que irão compor a floresta. O melhor valor foi obtido com 50 árvores, o padrão para este método é 100.
- **learning_rate** Representa a taxa de aprendizado ou encolhimento. O ajuste dessa taxa é um fator de multiplicação para os valores das folhas. Os valores variam de 0 a 1 e o melhor valor foi obtido com 0.1, que é o padrão para este método.
- **max_iter** Define o número máximo de iterações, o melhor resultado foi obtido com o valor definido em 50.
- **min_samples_leaf** Define o número mínimo de amostras por folha. O melhor resultado foi obtido com o valor definido em 200.

Tabela 15 – Predição utilizando *Histogram Gradient Boosting*

Conjunto (Tamanho)	Tempo Predição (s)	F1 Ponderada	Acurácia Balanceada	Precisão Ponderada	Rechamada Ponderada
Local B (10073)	0.26323	99.02 %	92.93 %	99.08 %	99.08 %
Local C (10079)	0.27293	99.89 %	99.57 %	99.89 %	99.89 %
Local D (10056)	0.28076	97.76 %	89.74 %	98.07 %	98.06 %
Média	0.2723	98.89 %	94.08 %	99.01 %	99.01 %

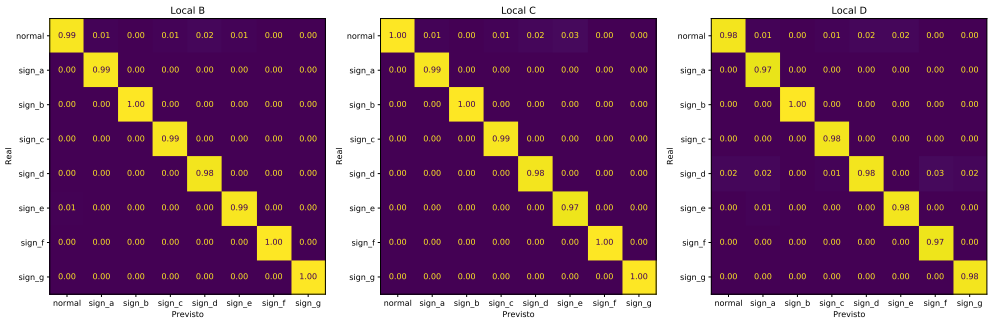


Figura 35 – Matriz de Confusão *Histogram Gradient Boosting*

O método *Histogram Gradient Boosting* atingiu uma média de 98.89% para a métrica F1 Ponderada e, para métrica de Acurácia Balanceada, o índice obtido foi de 94.08%, e foi capaz de distinguir quase todos os cenários de anomalia, como pode ser visto na matriz de confusão representada na Figura 35.

No Local B, cerca de 1% das anomalias do tipo **sign_b** foram confundidas, sendo previstas como Normais. O mesmo ocorreu no Local D, com cerca de 2% das anomalias do tipo **sign_d** sendo previstas como Normais, o que é caracterizado como falsos negativos.

No Local C, os casos previstos como Normais foram assertivos, sendo que apenas algumas previsões de anomalias acabaram confundidas com dados normais. Esses dados

caracterizam falsos positivos e representam 3% ou menos das cerca de 120 amostras de cada anomalia. Para este método, ainda ocorreram pequenas confusões entre as predições de anomalias, que é quando uma anomalia é prevista e na realidade se trata de outra anomalia, ocorrendo em 3% ou menos das amostras.

5.4 Discussões

Diante dos cenários vislumbrados nos resultados dos experimentos pode-se inferir que, em cada um dos métodos de aprendizado de máquina, os índices encontrados foram altos. Em uma comparação direta a métrica F1 Ponderada apresentou 99.97% para o método *Extra Trees Classifier* nos locais B e C, sendo que o pior índice para esta métrica foi de 96.33% para o método *Decision Tree* aplicado ao local D.

Os métodos *Extra Trees Classifier* e *Random Forest* obtiveram as melhores médias entre os locais utilizados para predição, com uma média de F1 Ponderada de 99.88%. O método *Extra Trees Classifier* se colocou como o melhor entre os analisados.

No entanto o método *Random Forest* ficou apenas 0.01% atrás de *Extra Trees Classifier*, como mostra a Tabela 16.

Tabela 16 – Médias dos modelos treinados nos locais B, C e D

Método	F1 Ponderada	Acurácia Balanceada	Precisão Ponderada	Rechamada Ponderada
Decision Tree	98.13 %	89.02 %	99.05 %	97.95 %
Random Forest	99.87 %	**99.23** %	99.87 %	99.87 %
Extra Trees Classifier	**99.88** %	99.16 %	**99.88** %	**99.88** %
Gradient Boosting	99.57 %	97.73 %	99.77 %	99.39 %
Extreme Gradient Boosting	98.86 %	94.98 %	99.03 %	99.02 %
Histogram Gradient Boosting	98.89 %	94.08 %	99.01 %	99.01 %

Dessa forma, esta pesquisa ainda deve levar em consideração alguns outros fatores relevantes, com relação ao tempo de treinamento e tempo de predição, além do tamanho do arquivo de salvamento do modelo. Assim, a Tabela 17 traz um ranking do tamanho dos arquivos de salvamento de cada método.

Tabela 17 – Arquivos de salvamento dos modelos treinados

Arquivo	Tamanho
DecisionTreeClassifier.sav	5.37 Kb
ExtraTreesClassifier.sav	91.09 Kb
XGBClassifier.sav	104.79 Kb
RandomForestClassifier.sav	108.9 Kb
HistGradientBoostingClassifier.sav	802.26 Kb
GradientBoostingClassifier.sav	2.47 Mb

O método *Extra Trees Classifier* apresenta uma leve vantagem sobre *Random Tree Classifier* da ordem de 17.81 Kb, que é muito pequena quando comparado aos métodos *Histogram Gradient Boosting* e *Gradient Boosting*.

No entanto, deve-se considerar que este modelo poderá ser executado em um dispositivo do tipo embarcado, alocado na memória *flash* de um *switch*, por exemplo. Assim a depender de quantos arquivos devam ser salvos, além do espaço disponível no dispositivo, qualquer que seja a economia de espaço pode significar uma enorme vantagem de um método sobre o outro.

Outro fator importante é o tempo de treinamento e predição, que pode ser fundamental quando essas informações precisem ser tratadas em tempo real.

Com relação aos tempos de treinamento, a Tabela 5 mostra que o tempo de treinamento de *Extra Tree Classifier* foi de aproximadamente 4,12 vezes mais rápida que *Random Forest*.

Já a Tabela 18 mostra os tempos médios de predição para todas as amostras de um local, aproximadamente 10.000 amostras por local.

Tabela 18 – Tempos médios de Predição

Técnica	Tempo Predição (s)
DecisionTreeClassifier	0.00171
RandomForestClassifier	0.02415
ExtraTreesClassifier	0.02600
XGBClassifier	0.06502
GradientBoostingClassifier	0.08966
HistGradientBoostingClassifier	0.27519

Percebe-se que a diferença entre *Extra Trees Classifier* e *Random Forest* é desprezível, ficando *Extra Trees Classifier* 0,00185s mais lento. No entanto, todos os métodos são satisfatoriamente rápidos, sendo que o mais lento ainda está abaixo de meio segundo.

Dessa forma, levando em consideração as métricas, os tempos de treinamento e predição, além do tamanho dos arquivos de salvamento dos modelos gerados, chegou-se a conclusão que o método *Extra Trees Classifier* é o melhor a ser adotado para responder a questão principal deste trabalho. A Tabela 19, mostra a compilação das métricas consideradas e que levaram à decisão pelo método *Extra Trees Classifier*.

Tabela 19 – Compilação das Técnicas de Classificação com todas as Métricas

Método	Tempo Treinamento	Tempo Predição	F1 Ponderada	Acurácia Balanceada	Precisão Ponderada	Rechamada Ponderada	Tamanho Arquivo
Decision Tree	0.146615	0.00171	98.13 %	89.02 %	99.05 %	97.95 %	5.37 Kb
Random Forest	2.208354	0.02415	99.87 %	99.23 %	99.87 %	99.87 %	108.9 Kb
Extra Trees Classifier	**0.536193**	**0.02600**	**99.88** %	**99.16** %	**99.88** %	**99.88** %	**91.09 Kb**
Gradient Boosting	29.942042	0.08966	99.57 %	97.73 %	99.77 %	99.39 %	2.47 Mb
Extreme Gradient Boosting	6.208331	0.06502	98.86 %	94.98 %	99.03 %	99.02 %	104.79 Kb
Histogram Gradient Boosting	2.659715	0.27519	98.89 %	94.08 %	99.01 %	99.01 %	802.26 Kb

Os altos índices obtidos nas métricas estão em consonância com outros trabalhos relacionados, citados na Subseção 3.4, que fizeram uso de métodos de aprendizado de máquina supervisionado semelhantes, aplicados a conjuntos de dados reais ou ainda gerados artificialmente como em Al-Naymat, Ghazi et al.(2018)[78].

5.5 Considerações Finais

Neste capítulo, foram feitas as análises comparativas entre os diversos métodos de aprendizado de máquina supervisionados, selecionados no Capítulo 3 e seguindo a metodologia definida no Capítulo 4.

No Capítulo 6 serão abordadas as conclusões e os trabalhos futuros.

6 Conclusão e Trabalhos Futuros

Neste trabalho foram aplicadas técnicas de aprendizado de máquina aos dados coletados em diferentes equipamentos da infraestrutura de redes de uma instituição de ensino superior.

Os *switchs* que foram utilizados na obtenção dos dados estão localizados em diferentes prédios e são utilizados por diferentes usuários, assim foi possível obter amostras variadas e diferentes entre si. À esses dados foram agregados os dados obtidos em laboratório, gerados a partir de ataques que reconhecidamente tem potencial de causar anomalias no funcionamento da rede.

Inicialmente foram aplicados diferentes métodos de aprendizado de máquina de maneira a verificar quais possuíam maior afinidade com o problema. Os 6 métodos escolhidos possuem uma característica em comum, todos empregam árvores de decisão ou variações delas, e são: *Decision Tree*, *Random Forest*, *Extra Trees Classifier*, *Gradient Boosting*, *Extreme Gradient Boosting* e *Histogram-based Gradient Boosting*.

Para que fosse possível a aplicação desses modelos, os dados passaram por um pré-processamento antes de serem submetidos aos métodos de aprendizado de máquina. Diante dos resultados apresentados no Capítulo 5, foi possível verificar que a melhor performance foi obtida com *Extra Trees Classifier*, atingindo 99.88% na métrica F1 ponderada e 99.16% de acurácia balanceada.

A partir da análise deste trabalho, percebe-se que a aplicação de métodos de aprendizado de máquina classificador supervisionado podem ser utilizados para detecção das anomalias conhecidas. Porém cabe ressaltar que não foi verificado o comportamento do modelo ao ser submetido à anomalias não treinadas e, portanto, desconhecidas pelo modelo.

Entre as contribuições deste trabalho está a análise prévia de uma quantidade razoável de métodos de aprendizado de máquina supervisionado classificador, que expôs o comportamento dos mesmos quando aplicados a esse tipo de dado. Também foi possível perceber que a aplicação desse tipo de método foi adequado à identificação de anomalias em uma infraestrutura de redes, quando aplicado aos dados das MIB's SNMP/RMON coletados em *switchs*.

Este trabalho também apresentou de maneira mais detalhada o comportamento de seis métodos e realizou um comparativo entre eles, levando em consideração não só as métricas tradicionais, mas também o tempo de treinamento e predição, além do tamanho dos arquivos de salvamento dos modelos.

Dessa forma, como trabalho futuro, pode-se utilizar outros algoritmos, como os classificadores não supervisionados, que sejam capazes de detectar anomalias não mapeadas ou ainda não conhecidas como as que podem surgir com um ataque de dia zero. As técnicas de *bagging* e/ou *boosting* também podem ser utilizadas para agregar mais de um método, afim de melhorar a performance, diminuindo principalmente os falsos positivos e os falsos negativos. Outra possibilidade é a utilização de métodos de aprendizagem profunda (do inglês *Deep Learning*) do tipo GAN (do inglês *Generative Adversarial Networks*), para a detecção e/ou geração de anomalias.

Do ponto de vista de hardware, a medida que os equipamentos atuais se tornam obsoletos e são substituídos por modelos mais modernos, é possível mudar a abordagem de maneira a utilizar novos recursos, como SDN (do inglês *Software Defined Network*), por exemplo.

Já uma abordagem baseada nos dados históricos dos equipamentos também pode ser realizada de forma a aprimorar um sistema de detecção de anomalias ou o desvio de comportamento baseado no equipamento conectado à infraestrutura de rede, fornecendo alertas aos gestores de redes sobre possíveis mal usos.

Referências

1 POSTEL, J. *Transmission Control Protocol*. RFC Editor, 1981. RFC 793. (Request for Comments, 793). Disponível em: <https://www.rfc-editor.org/info/rfc793>. 10, 26

2 POSTEL, J. *Internet Control Message Protocol*. RFC Editor, 1981. RFC 792. (Request for Comments, 792). Disponível em: <https://www.rfc-editor.org/info/rfc792>. 10, 29

3 KUROSE, J.; ROSS, K. *Redes de Computadores: uma abordagem top-down*. 6a. ed. [S.l.]: Always Learning, Pearson, London, 2013. 10, 18, 26, 27, 30

4 BHATTARAI, D. *Cisco Discover Protocol*. 2020. <https://learningnetwork.cisco.com/s/article/cisco-discovery-protocol-cdp-x>. Acessado em: 25-05-2022. 10, 33

5 BRASIL, A. *Estudo mostra que pandemia intensificou uso das tecnologias digitais*. 2021. <https://agenciabrasil.ebc.com.br/geral/noticia/2021-11/estudo-mostra-que-pandemia-intensificou-uso-das-tecnologias-digitais>. Acessado em: 05-04-2022. 17

6 ADVISOR, C. *Vítimas de ransomware de dupla extorsão crescem 935%*. 2021. <https://www.cisoadvisor.com.br/vitimas-de-ransomware-de-dupla-extorsao-crescem-935/>. Acessado em: 04-04-2022. 17

7 BRASIL, R. F. do. *Lei Geral de Proteção de Dados Pessoais*. 2018. <https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>. Acessado em: 30-04-2022. 17

8 UNION, O. J. of the E. *General Data Protection Regulation*. 2016. <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679>. Acessado em: 30-04-2022. 17

9 CERT.BR. Cartilha de segurança para internet: Centro de estudos, resposta e tratamento de incidentes de segurança no brasil. v. 1, 2012. Acessado em 04-04-2022. 17, 18

10 CERT.BR. *Tentativas de Fraudes - Incidentes Reportados ao CERT.br – Janeiro a Junho de 2020*. 2022. <https://www.cert.br/stats/incidentes/2020-jan-jun/fraude.html>. Acessado em: 25-03-2022. 17

11 BRASIL, A. *PF investiga fraudes no pagamento do auxílio emergencial*. 2022. <https://agenciabrasil.ebc.com.br/geral/noticia/2021-11/pf-investiga-fraudes-no-pagamento-do-auxilio-emergencial>. Acessado em: 25-03-2022. 17

12 MORENO, D. *Introdução ao Pentest*. [S.l.]: Novatec Editora, 2019. 17

13 SILVEIRA, S. A. d. Ciberativismo, cultura hacker e o individualismo colaborativo. *Revista USP*, n. 86, p. 28–39, ago. 2010. Disponível em: <https://www.revistas.usp.br/revusp/article/view/13811>. 18

14 BRASIL, B. N. *Anonymous: como hackers estão tentando minar Putin*. 2022. <https://www.bbc.com/portuguese/internacional-60813283>. Acessado em: 25-03-2022. 18

15 FREDA, A. *O que é um ataque de dia zero?* 2021. <https://www.avast.com/pt-br/c-zero-day>. Acessado em: 04-04-2022. 19

16 BOYAR, O.; ÖZEN, M.; METIN, B. Detection of denial-of-service attacks with snmp/rmon. In: IEEE. *2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES)*. [S.l.], 2018. p. 000437–000440. 21, 66, 70, 74

17 VERSITRON. *Three Important Network Switching Parameters You Must Consider*. 2021. <https://www.versitron.com/whitepaper/network-switching-parameters>. Acessado em: 05-04-2022. 25

18 CERT.BR. *Recomendações para Melhorar o Cenário de Ataques Distribuídos de Negação de Serviço (DDoS)*. 2016. <https://www.cert.br/docs/whitepapers/ddos/>. Acessado em: 01-04-2022. 25

19 CLOUDFLARE. *Ataque de inundação SYN*. 2021. <https://www.cloudflare.com/pt-br/learning/ddos/syn-flood-ddos-attack/>. Acessado em: 04-04-2022. 27

20 POSTEL, J. et al. *User Datagram Protocol*. RFC Editor, 1980. RFC 768. (Request for Comments, 768). Disponível em: <https://www.rfc-editor.org/info/rfc768>. 27

21 TA14, C. *017A: UDP-Based Amplification Attacks*. 2014. Disponível em: <https://www.cisa.gov/uscert/ncas/alerts/TA14-017A>. 27, 28

22 WHITE, C. M. *Redes de Computadores - Tradução da 6ª ed. norte-americana*. [S.l.]: Cengage Learning Brasil, 2013. 29

23 CISA. *DDoS Quick Guide*. 2020. Disponível em: <https://www.cisa.gov/uscert/security-publications/DDoS-Quick-Guide>. 29

24 DROMS, R. *Dynamic Host Configuration Protocol*. RFC Editor, 1997. RFC 2131. (Request for Comments, 2131). Disponível em: <https://www.rfc-editor.org/info/rfc2131>. 30

25 CROFT, W.; GILMORE, J. *Bootstrap Protocol*. RFC Editor, 1985. RFC 951. (Request for Comments, 951). Disponível em: <https://www.rfc-editor.org/info/rfc951>. 30

26 WENDELL, O.; SEAN, W. Ccna routing and switching 200-120 network simulator. *Pearson IT Certification. Part of the Network Simulator series*, p. 500, 2014. 31

27 STANDARD for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges. *IEEE Std 802.1D-1990*, p. 1–176, 1991. 31

28 IEEE Standard for Information Technology -Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Common Specifications - Part 3: Media Access Control (MAC) Bridges: Amendment 2 - Rapid Reconfiguration. *IEEE Std 802.1w-2001*, p. 1–116, 2001. 31

29 CISCO. *Catalyst 2960 Switch - Software Configuration Guide*. 2007. <https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst2960/software/release/12-2_40_se/configuration/guide/scg.pdf>. Acessado em: 24-05-2022. 32

30 IEEE Standard for Local and metropolitan area networks - Station and Media Access Control Connectivity Discovery. *IEEE Std 802.1AB-2016 (Revision of IEEE Std 802.1AB-2009)*, p. 1–146, 2016. 32

31 CISCO. *LLDP-MED and Cisco Discovery Protocol [White-Paper]*. 2006. <https://www.cisco.com/en/US/technologies/tk652/tk701/technologies_white_paper0900aecd804cd46d.pdf>. Acessado em: 24-05-2022. 32

32 CISCO. *Cisco Discovery Protocol Configuration Guide, Cisco IOS Release 15M&T*. 2016. <https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/cdp/configuration/15-mt/cdp-15-mt-book.html>. Acessado em: 25-05-2022. 32

33 SANTOS, O.; STUPPI, J. *CCNA Security 210-260 Official Cert Guide: CCNA Sec 210-260 OCG*. [S.l.]: Cisco Press, 2015. 33

34 RIOREY. *Taxonomy of DDoS Attacks*. 2016. <https://www.riorey.com/types-of-ddos-attacks/>. Acessado em: 25-05-2022. 34, 35

35 CERTIFICATION, P. I. *7 Popular Layer 2 Attacks*. 2016. <https://www.pearsonitcertification.com/articles/article.aspx?p=2491767>. Acessado em: 25-05-2022. 34, 35

36 MUKHTAR, H.; SALAH, K.; IRAQI, Y. Mitigation of dhcp starvation attack. *Computers & Electrical Engineering*, v. 38, n. 5, p. 1115–1128, 2012. ISSN 0045-7906. Special issue on Recent Advances in Security and Privacy in Distributed Communications and Image processing. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0045790612001140>. 35

37 EDDY, W. *TCP SYN Flooding Attacks and Common Mitigations*. RFC Editor, 2007. RFC 4987. (Request for Comments, 4987). Disponível em: <https://www.rfc-editor.org/info/rfc4987>. 36, 37

38 SEGINFO. *Mirai Botnet – Visão geral de suas Ameaças e Mitigações*. 2021. <https://seginfo.com.br/2021/08/23/mirai-botnet-visao-geral-de-suas-ameacas-e-mitigacoes>. Acessado em: 05-04-2022. 37

39 CLOUDFLARE. *O que é um ataque de inundação UDP?* 2021. <https://www.cloudflare.com/pt-br/learning/ddos/udp-flood-ddos-attack/>. Acessado em: 04-04-2022. 37

40 CLOUDFLARE. *O que é um ataque de inundação Ping (ICMP)?* 2021. <https://www.cloudflare.com/pt-br/learning/ddos/ping-icmp-flood-ddos-attack/>. Acessado em: 04-04-2022. 38

41 KASPERSKY. *Kaspersky IT Encyclopedia - MAC flooding*. 2021. <https://encyclopedia.kaspersky.com/glossary/mac-flooding/>. Acessado em: 04-04-2022. 39

42 BHAIJI, Y. *Network security technologies and solutions (CCIE professional development series)*. [S.l.]: Pearson Education, 2008. 40

43 CISCO. *Cisco Storm Control*. 2018. <https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4500/XE3-9-0E/15-25E/configuration/guide/xe-390-configuration/bcastsup.html>. Acessado em: 18-03-2022. 41, 42, 70, 74

44 CISCO. *Cisco 500 Series Stackable Managed Switch Administration Guide*. 2015. <https://www.cisco.com/c/dam/en/us/td/docs/switches/lan/csbms/Sx500/administration_guide/Cisco_500Sx_v1_4_AG.pdf>. Acessado em: 18-03-2022. 43

45 FORTINET. *FortiGate 1500D Series Data Sheet*. 2021. <https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/FortiGate_1500D.pdf>. Acessado em: 30-03-2022. 45

46 FORTINET. *FortiOS 6.0 - Handbook*. 2020. <https://docs.fortinet.com/document/fortigate/6.0.0/handbook/556201/whats-new>. Acessado em: 30-03-2022. 46

47 MICHAELIS, D. *Dicionário Brasileiro da Língua Portuguesa*. 2022. <https://michaelis.uol.com.br/moderno-portugues/busca/portugues-brasileiro/inteligencia>. Acessado em: 20-09-2022. 49

48 ONLINE, D. *Dicionário Online de Português*. 2022. <https://www.dicio.com.br/inteligencia/>. Acessado em: 20-09-2022. 49

49 TURING, A. M.; HAUGELAND, J. Computing machinery and intelligence. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, p. 29–56, 1950. 49

50 COPELAND, M. *What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning?* 2016. <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>. Acessado em: 20-09-2022. 50

51 MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. 50, 51

52 SCHRÖER, C.; KRUSE, F.; GÓMEZ, J. M. A systematic literature review on applying crisp-dm process model. *Procedia Computer Science*, Elsevier, v. 181, p. 526–534, 2021. 51

53 KUMAR, A. et al. Classification of faults in web applications using machine learning. In: *Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*. [S.l.: s.n.], 2017. p. 62–67. 51, 52

54 PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, JMLR. org, v. 12, p. 2825–2830, 2011. Disponível em: <"https://scikit-learn.org/stable/index.html">. 52, 54, 56, 60, 76, 77

55 SUN, Y.; WONG, A. K.; KAMEL, M. S. Classification of imbalanced data: A review. *International journal of pattern recognition and artificial intelligence*, World Scientific, v. 23, n. 04, p. 687–719, 2009. 53

56 INFORMATION; CALIFORNIA, C. S. U. of. *UCI KDD Archive, KDD Cup 1999 data, 1999*. 1999. Disponível em: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>. 53, 64

57 NGUYEN, H. A.; CHOI, D. Application of data mining to network intrusion detection: classifier selection model. In: SPRINGER. *Asia-Pacific Network Operations and Management Symposium*. [S.l.], 2008. p. 399–408. 53

58 SMITH, D. S. et al. Robustness of quantitative compressive sensing mri: the effect of random undersampling patterns on derived parameters for dce-and dsc-mri. *IEEE transactions on medical imaging*, IEEE, v. 31, n. 2, p. 504–511, 2011. 54

59 CHAWLA, N. V. et al. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, v. 16, p. 321–357, 2002. 54

60 GÉRON, A. *Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow*. [S.l.]: Alta Books, 2019. 57, 58, 59

61 BREIMAN, L. et al. Classification and regression trees. Wadsworth, 1984. 59

62 OSHIRO, T. M. *Uma abordagem para a construção de uma única árvore a partir de uma Random Forest para classificação de bases de expressão gênica*. Tese (Doutorado) — Universidade de São Paulo, 2013. 60

63 BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996. 60

64 BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. 60

65 XUAN, S.; LIU, G.; LI, Z. Refined weighted random forest and its application to credit card fraud detection. In: SPRINGER. *International Conference on Computational Social Networks*. [S.l.], 2018. p. 343–355. 61

66 GEURTS, P.; ERNST, D.; WEHENKEL, L. Extremely randomized trees. *Machine learning*, Springer, v. 63, n. 1, p. 3–42, 2006. 61

67 BHATI, B. S.; RAI, C. Ensemble based approach for intrusion detection using extra tree classifier. In: *Intelligent computing in engineering*. [S.l.]: Springer, 2020. p. 213–220. 62

68 FRIEDMAN, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, JSTOR, p. 1189–1232, 2001. 62

69 KE, G. et al. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, v. 30, 2017. 62, 63

70 CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. [S.l.: s.n.], 2016. p. 785–794. 63

71 DEVELOPERS, X. *dmlc XGBoost Documentation*. 2022. Disponível em: <https://xgboost.readthedocs.io/en/stable/index.html>. 63

72 YU, J. et al. Traffic flooding attack detection with snmp mib using svm. *Computer Communications*, Elsevier, v. 31, n. 17, p. 4212–4219, 2008. 64

73 FENG, W. et al. Mining network data for intrusion detection through combining svms with ant colony networks. *Future Generation Computer Systems*, Elsevier, v. 37, p. 127–140, 2014. 64

74 TAO, P.; SUN, Z.; SUN, Z. An improved intrusion detection algorithm based on ga and svm. *Ieee Access*, IEEE, v. 6, p. 13624–13631, 2018. 64

75 DEVI, B. K. et al. A prediction model for flooded packets in snmp networks. In: IEEE. *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*. [S.l.], 2018. p. 82–86. 65

76 SHARAFALDIN, I.; LASHKARI, A. H.; GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, v. 1, p. 108–116, 2018. 65

77 LASHKARI, A. H. et al. Characterization of tor traffic using time based features. In: *ICISSp*. [S.l.: s.n.], 2017. p. 253–262. 65

78 AL-NAYMAT, G.; AL-KASASSBEH, M.; AL-HARWARI, E. Using machine learning methods for detecting network anomalies within snmp-mib dataset. *International Journal of Wireless and Mobile Computing*, Inderscience Publishers (IEL), v. 15, n. 1, p. 67–76, 2018. 65, 91

79 AL-KASASSBEH, M.; AL-NAYMAT, G.; AL-HAWARI, E. Towards generating realistic snmp-mib dataset for network anomaly detection. *International Journal of Computer Science and Information Security*, LJS Publishing, v. 14, n. 9, p. 1162, 2016. 66

80 SIA, Z. *Zabbix 5.4 Manual*. 2021. <https://www.zabbix.com/documentation/5.4/en/manual/introduction/about>. Acessado em: 30-03-2022. 69, 70

81 FOUNDATION, P. S. *Python Documentation*. 2022. <https://www.python.org/about/>. Acessado em: 30-03-2022. 72

82 MCKINNEY, W. *Pandas - Python for Data Analisys*. 2022. <https://pandas.pydata.org/docs/>. Acessado em: 30-03-2022. 72

83 COMMUNITY, N. D. *NumPy Documentation*. 2022. <https://numpy.org/doc/stable/>. Acessado em: 30-03-2022. 72

84 PAES, D. S. F.; MORAES, C. H. V.; BATISTA, B. G. *Dataset with SNMP/RMON MIB from Managed Switchs*. [S.l.]: GitHub, 2023. <https://github.com/domingospaes/Dataset-with-SNMP-RMON-MIBs-and-7-Anomalies-of-DoS>. 74, 80

85 ŽLIOBAITĖ, I.; PECHENIZKIY, M.; GAMA, J. An overview of concept drift applications. *Big data analysis: new algorithms for a new society*, Springer, p. 91–114, 2016. 75

86 SCHÖLKOPF, B.; SMOLA, A.; MÜLLER, K.-R. Kernel principal component analysis. In: SPRINGER. *International conference on artificial neural networks*. [S.l.], 1997. p. 583–588. 76

87 KOHAVI, R. et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: MONTREAL, CANADA. *Ijcai*. [S.l.], 1995. v. 14, n. 2, p. 1137–1145. 77

88 INC., G. *Google Colaboratory*. 2022. <https://colab.research.google.com/>. Acessado em: 30-04-2022. 79

89 LAROCHELLE, H. et al. An empirical evaluation of deep architectures on problems with many factors of variation. In: *Proceedings of the 24th international conference on Machine learning*. [S.l.: s.n.], 2007. p. 473–480. 81

90 BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, v. 13, n. 2, 2012. 81