UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CAMPUS DE FOZ DO IGUAÇU
PROGRAMA DE PÓS-GRADUAÇÃO EM
ENGENHARIA ELÉTRICA E COMPUTAÇÃO

DISSERTAÇÃO DE MESTRADO

ABORDAGEM DE DETECÇÃO DE INTRUSÃO EM AMBIENTES *FOG COMPUTING* E *INTERNET OF THINGS*

JEAN DOUGLAS GOMES VALENCIO

FOZ DO IGUAÇU
2021

Jean Douglas Gomes Valencio

Abordagem de detecção de intrusão em ambientes *fog computing* e *internet of things*

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Computação da Universidade Estadual do Paraná como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica e Computação. Área de concentração: Sistemas Elétricos e Computação.

Orientador: Dr. Renato Bobsin Machado

Foz do Iguaçu
2021

Ficha de identificação da obra elaborada através do Formulário de Geração Automática do Sistema de Bibliotecas da Unioeste.

Valencio, Jean Douglas Gomes
ABORDAGEM DE DETECÇÃO DE INTRUSÃO EM AMBIENTES FOG COMPUTING E INTERNET OF THINGS / Jean Douglas Gomes Valencio; orientador Renato Bobsin Machado. -- Foz do Iguaçu, 2021.
114 p.

Dissertação (Mestrado Acadêmico Campus de Foz do Iguaçu) -- Universidade Estadual do Oeste do Paraná, Centro de Engenharias e Ciências Exatas, Programa de Pós-Graduação em Engenharia Elétrica e Computação, 2021.

1. Seleção de Atributos. 2. Métodos Ensemble. 3. Segurança de Redes. I. Machado, Renato Bobsin, orient. II. Título.

Abordagem de detecção de intrusão em ambientes *fog computing* e *internet of things*

Jean Douglas Gomes Valencio

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Computação e aprovada pela Banca Examinadora assim constituída:

Prof. Dr. **Dr. Renato Bobsin Machado** - (Orientador)
Universidade Estadual do Oeste do Paraná - UNIOESTE

Prof. Dr. **Narco Afonso Ravazzoli Maciejewski**
Centro Universitário Dinâmica das Cataratas - UDC

Prof. Dr. **Rômulo César Silva**
Universidade Estadual do Oeste do Paraná - UNIOESTE

Data da defesa: 21/12/2021.

Resumo

Diante de inúmeras inovações e avanços tecnológicos em nossa era, a conexão entre pessoas através de dispositivos conectados à Internet é intrínseca em nosso cotidiano, permitindo a troca e o compartilhamento de informações em tempo real, a emersão de variados modelos e cadeias de negócio cujo núcleo é a Internet e a conexão e controle dos dispositivos IoT que permeiam o ambiente criando uma interface como digital e físico. No entanto, algumas inovações se tornam catalisadores para atividades de atores maliciosos que buscam vulnerabilidades em sistemas para então explorá-los e causar danos ou realizar um ganho de recurso sobre a posse alheia. A fragilidade de sistemas vem sendo exposta constantemente através de incidentes computacionais crescentes. Nesse contexto, sistemas de detecção de intrusão são de grande valia para organizações que buscam uma maior resistência a ataques e ameaças externas, protegendo seus usuários e recursos. A quantidade de tráfego a ser analisada por sistemas de detecção de intrusão é muitas vezes proibitiva e consome uma grande quantidade de recursos computacionais, principalmente em dispositivos IoT que não dispõem de muitos recursos e são geralmente baseados em uma arquitetura de diversas camadas. Diante deste contexto, este trabalho consiste em uma abordagem de detecção de intrusão baseada em seleção de atributos e classificação de eventos. A fase de seleção de atributos *ensemble* é composta de duas etapas, na primeira é utilizado um método baseado em estatística e no ganho de informação, Information Gain (IG), reduzindo a quantidade de atributos e gerando um subconjunto que é então submetido à segunda etapa do método, composto por dois algoritmos, *Sequential Forward Feature Selection* (SFFS) e *Sequential Backward Feature Elimination* (SBFE), que realizam a avaliação baseado no desempenho da combinação de diversos subconjuntos, gerando um conjunto de atributos reduzidos combinados por um método. O conjunto resultante deste processamento é então utilizado para o treinamento do algoritmo classificador, *Extra-Tree* (ET). Para a realização dos experimentos utilizou-se a base de dados pública CICIDS2017, reduzida para 20% durante a fase de pré-processamento. Foram variados os arranjos dos algoritmos de seleção de atributos para então ser treinado o algoritmo de classificação e executar o mesmo, totalizando em 5 abordagens de seleção de atributos e mais uma abordagem utilizando a base completa com todos os atributos. A abordagem que utiliza IG + SFFS $\cap$ SBFE apresentou o melhor resultado em tempo de teste e tempo de treino mantendo os níveis de acurácia, acurácia balanceada e precisão das outras abordagens.

Palavras-chave: Seleção de Atributos, Métodos Ensemble, Segurança de Redes.

Abstract

Given the advantages of innovations and technological advances in our era, the connection between people through devices connected to the Internet is intrinsic in our daily lives, allowing the distribution and sharing of information in real time various business models and distribution chains are Internet based, that is also, useful to connect and control IoT devices that permeate the environment creating an interface from the digital world to the physical world. However, some innovations become catalysts for the activities of malicious actors that look for vulnerabilities in systems and then exploit them, causing damage or making a personal gain on possession of others resources. The fragility of systems has been constantly exposed through increasing computational incidents. In this context, intrusion detection systems add a great value to organizations that look for greater resistance to external solutions, protecting their users and resources. The amount of traffic to be analyzed by intrusion detection systems is often prohibitive and consumes a large amount of computing resources, especially on IoT devices that are resource weakly and are the architecture usually based on multiple layers. Given this context, this work consists of an intrusion detection approach based on attribute selection and event classification. The ensemble of attribute selection phase is composed of two steps, in the first one a method based on statistics and information gain is used, the Information gain (IG) method, reducing quantity of attributes and generating a subset which is then submitted to the second step of the method, that consist on two algorithms, the Sequential Forward Feature Selection (SFFS) and Sequential Backward Feature Elimination (SBFE), which perform the evaluation based on the performance of the combination of several subsets, generating a set of reduced attributes combined by a combination method. The resulting set of this processing is then used to train the classifier algorithm, Extra-Tree (ET). To carry out the experiments, a public database CICIDS2017 was used, reduced to 20% during a pre-processing phase. The arrangement of the attribute selection algorithms were varied in order to train the classification algorithm and execute it, totaling 5 attribute selection approaches and another approach using a complete base with all attributes. The approach using IG + SFFS cap SBFE presented the best result in testing time and training time maintaining the accuracy levels, balanced accuracy and precision of the other approaches.

Keywords: Feature Selection, Ensemble methods, Network Security.

Many are harmed by fear itself, and many may have come to their fate while dreading fate.
—SENECA, OEDIPUS, 992

Agradecimentos

Agradeço primeiramente a Deus e aos meus pais José Jorge Valencio e Deinize Geralda Amaral Gomes Valencio e ao meu irmão Felipe Gomes Valencio que sempre me apoiaram e me deram o suporte necessário para que eu alcançasse todos meus objetivos, sou eternamente grato por todo o esforço de vocês.

Agradeço ao professor, amigo e orientador deste trabalho Renato Bobsin Machado, por todo o apoio, paciência e ensinamentos que vão além do ambiente acadêmico, que com certeza foram muitos durante esses anos e a todos os professores da UNIOESTE que de alguma forma contribuíram em minha formação.

Ao Cristiano Souza que incansavelmente me apoiou e deu todo o suporte possível para que esse trabalho fosse concluído e sempre buscando me instruir com as melhores práticas e assim elevando o nível do trabalho para o patamar em que este se encontra.

Sou grato à minha família e meus amigos por sempre estarem presentes nos bons e nos maus momentos.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Abreviaturas

PGEEC	Programa de Pós-Graduação em Engenharia Elétrica e Computação, UNIOESTE, Campus de Foz do Iguaçu
UNIOESTE	Universidade Estadual do Oeste do Paraná
IoT	*Internet of Things*
ML	*Machine Learning*
SDI	Sistema de Detecção de Intrusão
NIST	*National Institute of Standards and Technology*
HIDS	*Host Based Intrusion Detection System*
NIDS	*Network Based Intrusion Detection System*
KDD	*Knowledge Discovery in Databases*
MD	Mineração de Dados
RNA	Redes Neurais Artificiais
k-NN	*k-Nearest Neighbor*
DT	*Decision Tree*
RF	*Random Forest*
ET	*Extra Tree*
SVM	*Support Vector Machine*
SU	*Simmetrical Uncertainty*
SFFS	*Sequential Forward Feature Selection*
SBFE	*Sequential Backward Feature Elimination*
IG	*Information Gain*
DDoS	*Distributed Denial of Service*
PCA	*Principal Component Analysis*
EFS	*Ensemble Feature Selection*
DNN	*Deep Feed-Forward Neural Network*
GBT	*Gradient Boosting Tree*
MSA	Método de Seleção de Atributos
MC	Método de Classificação
DM	Abordagem realiza Detecção Multiclasse
ESA	O trabalho é baseado em Ensemble Seleção de Atributos

CIC	*Canadian Institute for Cybersecurity*
pcap	*Packet Capture*
IP	*Internet Protocol*
IAT	*Inter Arrival Time*
Bwd	*Backward*
Fwd	*Forward*
Std	*Standard*
Avg	*Average*
GB	*GigaBytes*
Seg	*Segment*

Capítulo 1

Introdução

Com o exponencial crescimento e a evolução computacional e das mais diversas tecnologias, as redes de computadores ganham cada vez mais importância e atenção. Tais ferramentas são essenciais para a comunicação, entretenimento, informação e educação, tornando-se assim intrínseca em nosso cotidiano. Com a finalidade de penetrar e conectar ainda mais nossa realidade digital e física surge um paradigma, a Internet das Coisas (Internet of Things, ou IoT) (Atzori, Iera & Morabito, 2010).

Os dispositivos IoT estão se tornando onipresentes em nossa rotina. Dentre essas tecnologias, geladeira, aspirador de pó e ar condicionado são alguns exemplos de eletrodomésticos que podem ser controlados através de comandos de celular a quilômetros de distância. Tais produtos são considerados IoT devido a sua conexão com a Internet. O conceito principal deste paradigma é a interconexão de dispositivos chamados "coisas". São capazes de mensurar, inferir, entender e modificar o ambiente em que se encontram. O principal ponto a ser destacado é o impacto e o auxílio nas tarefas usuais do usuário, automatizando processos. Além disso, são esperadas grandes consequências para o mercado de maneira geral, como logística, automação industrial, segurança, agronomia, entre outros, podendo afetar de maneira positiva, diminuindo custos e maximizando a eficiência de processos (Atzori et al., 2010; Botta, De Donato, Persico & Pescapé, 2016).

Os dispositivos IoT, em geral, possuem um *hardware* de baixo custo e consumo de energia. Sendo assim, possuem diversas limitações e restrições de recursos. No entanto, são capazes de coletar e gerar dados capturados no ambiente em que se encontram. Com o intuito de centralizar e realizar o tratamento destes dados, começou-se a utilizar recursos em *cloud* para o processamento dos dados capturados, aliviando os dispositivos, que tem agora como principal função a coleta e o envio dos dados. Porém, o constante envio de dados pode gerar sobrecargas na rede, aumentando significativamente a latência da comunicação, culminando no retardamento e até mesmo em perdas de alguns pacotes de dados no tráfego da rede. Dentre os modelos de comunicação entre nós de toda a rede, podemos citar a abordagem de *fog* que busca a solução da latência entre os dispositivos IoT e a *cloud*, sendo uma camada intermediária entre ambas, fornecendo o processamento mais próximo da borda da rede (Bonomi, Milito, Zhu & Addepalli, 2012).

A arquitetura de rede em *fog* é uma abordagem de computação distribuída que é uma extensão da *cloud*. Esta busca centralizar todas as informações contidas da *fog*, visto que possui mais recursos para processamento e armazenamento que os dispositivos IoT, cuja maioria possui somente a capacidade de coletar e transmitir dados. A *fog* busca prover mobilidade, conhecimento de posição, heterogeneidade, ampla escalabilidade, baixa latência e distribuição geográfica. Sendo assim, a *fog* é uma camada cujo é objetivo diminuir o volume de dados e o tráfego entre dispositivos IoT e servidores na *cloud*, por exemplo (Alrawais, Alhothaily, Hu & Cheng, 2017).

Um tema que gera inúmeras preocupações é a segurança de computadores, mais especificamente dos dados que estão presentes nas máquinas e os que trafegam pela rede. Com a finalidade de sanar problemas de confidencialidade, disponibilidade e integridade dos dados (os três pilares da segurança da computação), as pesquisas na área de segurança da informação, ou segurança da computação, buscam formas de detectar e impedir atividades maliciosas que ferem os pilares supra mencionados.

Os métodos de segurança de computadores têm uma grande influência na adoção das tecnologias IoT ao redor do mundo devido a sua aceitação pelo mercado consumidor. Ainda é incerto quais seriam as consequências de um incidente cibernético em larga escala derivado de dispositivos IoT, isto se deve ao fato de ser uma tecnologia imatura no mercado, que ainda recebe mais atenção em desenvolvimento de usabilidade que em segurança de comunicações (Jalali, Kaiser, Siegel & Madnick, 2019).

A segurança de redes em IoT possui os mesmos problemas das redes de computadores, comunicação móvel e Internet. No entanto, além disso, possui problemas específicos, como tipos diferentes de autenticação e protocolos de comunicações mais vulneráveis a ataques e suscetíveis a falhas (Jing, Vasilakos, Wan, Lu & Qiu, 2014). A diversificação entre os dispositivos IoT de uma determinada aplicação e os diferentes serviços prestados também contribuem para agravar os problemas de segurança (Alrawais et al., 2017). Como forma de defender e tornar a rede um lugar mais confiável e com menor possibilidade de intromissão de agentes não confiáveis cita-se os sistemas de detecção de intrusão (Intrusion Detection System, ou IDS). Um IDS tem como objetivo detectar anomalias ou ações não usuais do usuário, como exemplo, detectar se algo malicioso está ocorrendo no fluxo da rede.

Os nós de dispositivos IoT em sua maioria dispõem de um limitado poder computacional e baixa capacidade de armazenamento, resultando em métodos mais leves de criptografia e comunicação, assim, expondo vulnerabilidades mais fáceis de serem exploradas e códigos mais fáceis de serem quebrados. Além disso, essas limitações dificultam ou inviabilizam a implantação de certos IDS nos nós de dispositivos IoT. O fato da arquitetura de uma aplicação IoT ser multi-camadas também dificulta a implantação de um IDS que opera apenas em uma única camada (Alaba, Othman, Hashem & Alotaibi, 2017).

Um IDS localizado na camada de *fog* tem a capacidade de analisar o tráfego da totalidade

da rede IoT antes que qualquer atividade maliciosa atinja a arquitetura em *cloud*, relativamente mais crítica e essencial na totalidade da rede IoT.

1.1 Motivações

A rede mundial de computadores é considerada uma ferramenta indispensável e insubstituível para os mais diversos segmentos da vida moderna. Esta tecnologia provê conhecimento, informação e entretenimento aos seus usuários sendo também o pilar fundamental de diversas empresas e modelos de negócio, que consequentemente geram empregos, renda e crescimento para a humanidade. Porém, o rápido crescimento da conectividade e acessibilidade a Internet ocasionou o surgimento de diversos problemas de segurança (Tsai, Hsu, Lin & Lin, 2009; Shon & Moon, 2007). Comumente ao realizar compras ou contratar serviços utilizando a Internet, armazenamos informações confidenciais e proprietárias, muitas vezes com grande importância e valor.

Ambientes expostos a ataques podem prejudicar a todos, visto que o roubo de informações pode ser danoso a qualquer ator. A exploração de vulnerabilidades pode ganhar uma expressão de terrorismo agravando o funcionamento de empresas, governos e organizações responsáveis por sistemas críticos, como aeroportos, distribuidoras de eletricidade ou oleodutos. Nos últimos anos os ataques a dispositivos conectados a Internet cresceram continuamente como demonstrado pela Figura 1.1, disponibilizada pelo Centro de Estudos, Respostas e Tratamento de Incidentes de Segurança no Brasil (CERT.br).

Os ambientes inteligentes estão se tornando reais por meio da IoT e da computação de névoa, mas eles não estão livres de ameaças e vulnerabilidades de segurança. As técnicas para explorar as vulnerabilidades da infraestrutura do computador estão sendo continuamente aprimoradas. Entre os principais objetivos está a aquisição de acesso aos sistemas, a obtenção e uso indevido de informações confidenciais e ocasionando indisponibilidade de recursos. Por exemplo, um incidente recente envolvendo dispositivos IoT em outubro de 2016, onde um ataque de *botnet* Mirai ao provedor de serviços Dyn derrubou centenas de sites, incluindo Twitter, Netflix, Reddit e GitHub, por várias horas (Kolias, Kambourakis, Stavrou & Voas, 2017; Tanaka & Yamaguchi, 2017). O dano significativo gerado por ataques neste ambiente cria a necessidade de concentrar esforços nesta área (Ni, Zhang, Lin & Shen, 2018). Técnicas especiais de segurança são indispensáveis em sistemas de computador modernos. A detecção de intrusão é um dos pontos críticos de segurança, visando identificar ocorrências de ataques.

Diversas pesquisas sobre segurança são conduzidas buscando desenvolver novas abordagens de detecção de intrusão e melhorias na segurança na rede de computadores.

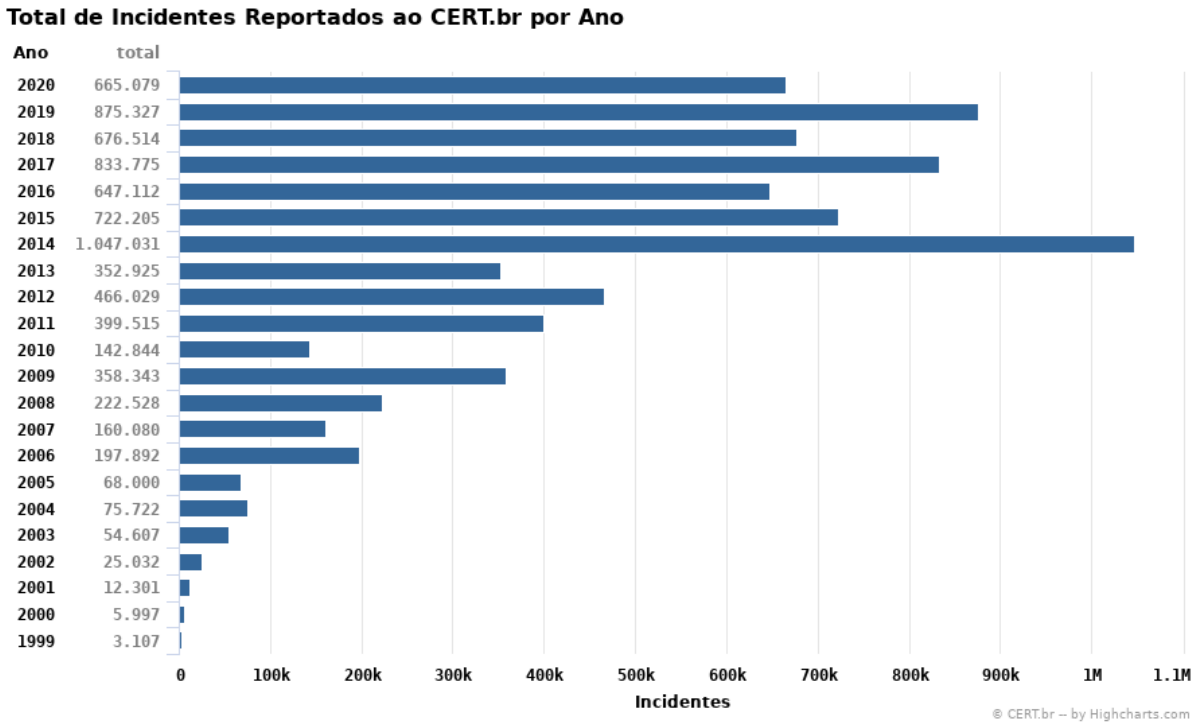

Figura 1.1: Índices de ocorrências de incidentes de segurança reportados ao CERT.br. Fonte: (CERT.br, 2020)

1.2 Contextualização do Problema

Muitos dos IDS tradicionais utilizados no mercado não possuem uma baixa restrição de recursos para operar, tornando a implementação destes em um ambiente IoT um desafio. Esta dificuldade na adequação de IDS consolidados em ambientes de grande diversidade de dispositivos, protocolos e serviços inovadores criou um nicho de pesquisas, buscando métodos de detecção eficientes para dispositivos de baixa capacidade de recursos.

A localização do IDS na arquitetura de IoT é proposta juntamente a camada de *fog*, para o armazenamento e pré-processamento mais perto da borda, já que possui maior capacidade de processamento e armazenamento, acelerando assim a análise e a tomada de decisão para o evento e diminuindo a latência que haveria entre os dispositivos e a *cloud* favorecendo aplicações em tempo real como as do contexto de Indústria 4.0 (Dastjerdi & Buyya, 2016).

A utilização de algoritmos de aprendizado de máquina (*Machine Learning*, ou ML) por IDS para a classificação dos eventos é crescente nas abordagens de detecção por anomalia que monitoram o sistema em busca de comportamentos anormais. Tais métodos são propostos em diversos trabalhos na literatura.

A utilização de duas abordagens diferentes na etapa de seleção de atributos se deve ao fato de suas especificidades e peculiaridades. Os algoritmos do tipo filtro são algoritmos que geralmente buscam uma relação matemática ao correlacionar os atributos com os resultados das classes não considerando a combinação dos atributos para inferir o resultado. No entanto, a

classe filtro pode ser utilizado para realizar uma pré-seleção de atributos visto que a execução de testes estatísticos sobre o conjunto inicial de atributos pode filtrar os recursos que possuem menor importância. Este corte pode ser realizado caso o atributo fique abaixo de um *threshold* previamente definido (Osanaiye, Cai, Choo, Dehghantanha, Xu & Dlodlo, 2016). Já as abordagens *Wrapper* utilizam um motor de indução para classificar subconjuntos de atributos, assim, podendo ter uma melhor *performance* na seleção de atributos. Outra influência importante a ser considerada nesta abordagem é a direção de busca, podendo ser para frente (*Forward*) ou para trás *(Backward)*, a primeira começa com um conjunto vazio e vai inserindo atributos conforme o resultado da última avaliação, a segunda estratégia de busca tem início com o conjunto completo de atributos e a remoção de atributos acontece conforme o resultado da última avaliação do conjunto. A utilização dessas duas abordagens em conjunto busca combinar o baixo custo da abordagem *filter* e da maior confiabilidade do *wrapper*.

A utilização de uma abordagem que diminua a dimensionalidade do fluxo de dados a serem considerados na classificação de eventos em tempo real tem grande importância visto que pode melhorar o processamento e consequentemente a *performance* dos classificadores. Além disso, os dispositivos IoT possuem poucos recursos computacionais para realizar a análise dos eventos por aprendizado de máquina, e o envio destes dados para uma camada superior como a *cloud* implica em uma alta latência impactando a classificação. O conjunto reduzido de características do pacote precisa ser selecionado para manter ou melhorar a *performance*, reduzindo o processamento e maximizando a precisão de detecção.

1.3 Proposta

É proposto um método de classificação de intrusões baseado em aprendizado de máquina, que utiliza como entrada atributos capturados de redes IoT, além disso, propõe-se um método *ensemble* de seleção de atributos para selecionar as características do tráfego que fornecem informações mais relevantes para o modelo de classificação.

O método seletor de atributos visa diminuir a quantidade de atributos a serem processados pelo método classificador. Reduzindo a dimensionalidade total do fluxo do tráfego de rede busca-se melhorar a performance de classificação e diminuir o custo computacional necessário para o classificador realizar a análise do tráfego, de modo a tornar mais plausível e possível a utilização em tempo real.

Os métodos *ensemble* buscam combinar as saídas de diferentes algoritmos de seleção de atributos através de um combinador, de modo a usufruir das vantagens e minimizar as desvantagens dos métodos. Oferecendo assim um conjunto mais preciso de atributos importantes ao IDS. Tal vantagem busca aumentar potencialmente a acurácia da classificação, também reduzindo a complexidade computacional e selecionando os atributos mais relevantes entre os presentes no

tráfego da rede (Osanaiye et al., 2016; Hoque, Singh & Bhattacharyya, 2018). Após a realização da seleção de atributos, o evento redimensionado utilizando estes atributos é submetido ao classificador para rotular o evento, como apresentado na Figura 1.2.

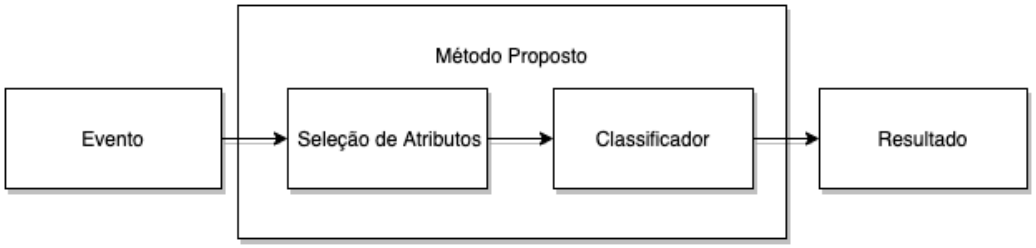

Figura 1.2: Arquitetura da Abordagem Proposta.
Fonte: O autor (2021).

Com a finalidade de tornar o ambiente IoT mais seguro, é proposto neste trabalho uma abordagem *ensemble* de seleção de atributos e um método de classificação treinado com este subconjunto de atributos. Permitindo assim o processamento dos atributos mais importantes do tráfego da rede de dispositivos IoT. Esta classificação de eventos pode melhorar a defesa necessária contra intrusos, assim como o dispêndio de recursos de rede e *hardware*.

1.4 Contribuições

A seguir são exploradas as contribuições que são importantes para a evolução da linha de pesquisa em segurança computacional desenvolvida em parceria entre os laboratórios LaPSeC da UNIOESTE e DMC-NS da UFSC.

- Proposta de uma abordagem *ensemble* de seleção de atributos, utilizando uma abordagem híbrida em série com duas abordagens *wrappers* combinadas;
- Proposta de uma arquitetura de seleção de atributos e classificador para a detecção de intrusão em ambientes IoT;
- Melhorias no tempo de teste e de treino em 16,7% e 66,7% respectivamente relação a utilização da base completa com todos os atributos. Também houve melhorias significativas em relação a base filtrada pelo método de seleção de atributos *Information Gain*.
- Melhorias das taxas de detecção de intrusão, como acurácia, acurácia balanceada e precisão em relação a base completa e a base filtrada pelo método de seleção de atributos *Information Gain*.

1.5 Estrutura do Trabalho

No Capítulo 2 é apresentado a fundamentação teórica do trabalho, explorando os conceitos de IoT, *cloud* e *fog*. Também é explorado alguns pontos de segurança de computação, como os pilares da segurança, algumas ameaças computacionais e sobre detecção de intrusão.

No Capítulo 3 é explorado sobre inteligência computacional, aprofundando em temas como a extração de conhecimento de dados, abordagens de métodos de seleção de atributos e métodos de aprendizado de máquina.

O Capítulo 4 apresenta diversos trabalhos científicos relacionados, considerando vários métodos, estratégias e abordagens aplicadas considerando o estado da arte.

No Capítulo 5 é explanado o método proposto, assim como suas características, envolvendo arquitetura, descrição dos métodos aplicados para a seleção de atributos e para a classificação.

O Capítulo 6 sobre os experimentos descreve como foram conduzidos estes. É realizado a descrição da base de eventos, a técnica de amostragem e as métricas de avaliação utilizada. A arquitetura e condução de cada experimento também é apresentado neste capítulo, e por fim os materiais utilizados.

No Capítulo 7 são apresentados os resultados de cada experimento, tal como a avaliação e discussão das métricas de avaliação calculadas como acurácia, acurácia balanceada e *F1-score*.

Por fim, no Capítulo 8 são apresentadas as conclusões, contribuições alcançadas e propostas para trabalhos futuros.

Capítulo 2

Segurança Computacional em ambientes de IoT e *Fog Computing*

Neste capítulo são apresentados os conceitos e fundamentos teóricos relacionados às áreas de *Internet of Things* (IoT), segurança de computadores, inteligência computacional e seleção de atributos, contextualizando de maneira sólida a base teórica necessária para melhor entendimento da proposta e problemática do presente trabalho.

2.1 *Internet of Things*

Com o exponencial crescimento e evolução computacional das mais diversas tecnologias, as redes de computadores ganham cada vez mais importância e atenção. Tais ferramentas são essenciais para a comunicação, entretenimento, informação e educação, tornando-se assim intrínseca em nosso cotidiano. Com a finalidade de penetrar e conectar ainda mais nossa realidade digital e física surge um paradigma, a Internet das Coisas (*Internet of Things*, ou IoT), levando ao surgimento de sistemas ciber físicos (Borgia, 2014).

Os dispositivos IoT estão se tornando onipresentes no cotidiano das pessoas. Devido ao tamanho pequeno e a capacidade de comunicação sem fio presente na maioria dos dispositivos, os mesmos vem sendo empregados de forma generalizada em diversas coisas e objetos, permitindo assim, a interação e a cooperação entre eles com um objetivo em comum. Os dispositivos podem fornecer informações importantes, além de serem remotamente ativados e controlados através de comandos de celular a quilômetros de distância, fornecendo maior comodidade e facilidade para o usuário. Dentre as aplicações baseada em dispositivos IoT pode-se citar, por exemplo, sistemas de climatização e de segurança de residências. Estes dispositivos possuem como uma das suas características principais a capacidade de se conectar a Internet, denotando fortemente seu conceito principal, que consiste na conexão e interconexão de dispositivos chamados “coisas”. Tais dispositivos são capazes de mensurar, inferir, entender e através do processamento dessas informações no mundo digital podem afetar e modificar o ambiente onde se encontram através de processos físicos por meio de atuadores. Assim podemos destacar

o impacto no auxílio das tarefas habituais do usuário na automatização de processos (Borgia, 2014).

São esperadas grandes consequências para o mercado global de maneira geral. Dentre os vários segmentos impactados estão os de logística, automação e manufatura industrial, segurança, agronomia, transporte inteligente de pessoas e mercadorias, entre outros. A IoT pode afetar de maneira muito positiva estes segmentos, diminuindo custos, ao passo que maximiza a eficiência de processos (Atzori et al., 2010; Botta et al., 2016). Na Figura 2.1 é apresentado de forma simplificada uma arquitetura que contempla a comunicação entre os dispositivos, a camada de *fog* e a camada de *cloud*.

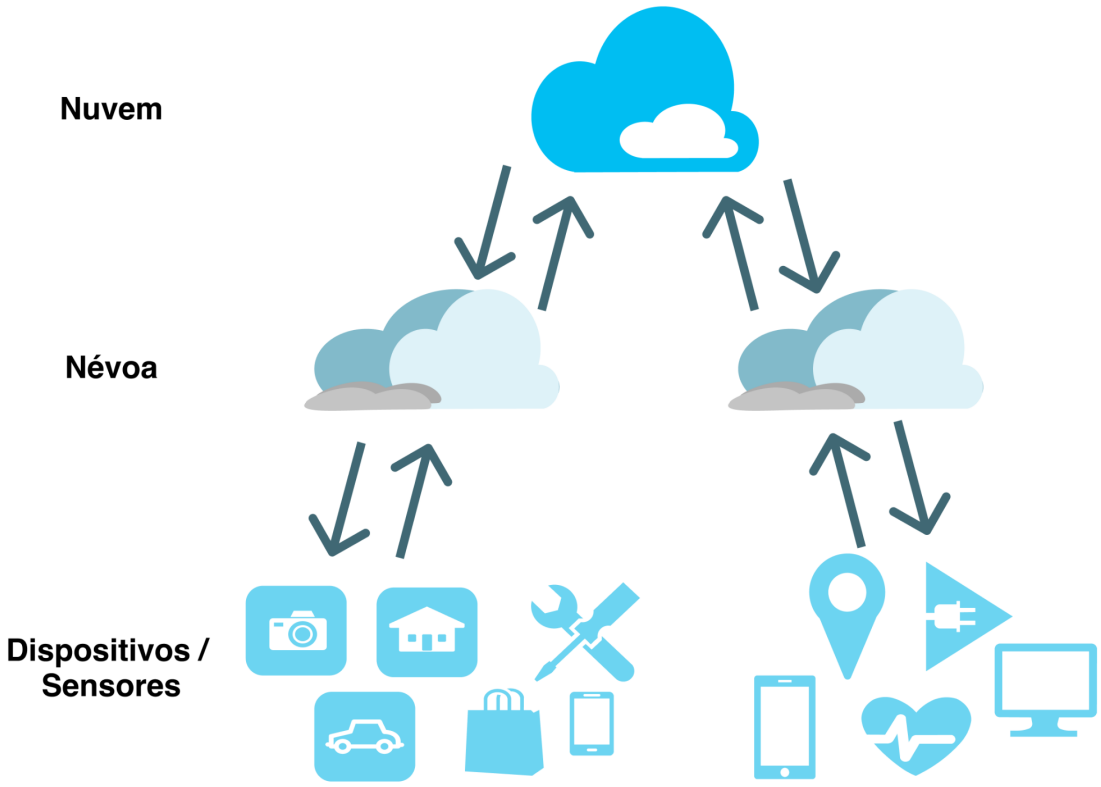

Figura 2.1: Arquitetura IoT.
Fonte: O Autor.

Segundo previsão de Camhi (2015), ex-CEO da Cisco, o número de dispositivos conectados a Internet em 2025 alcançará os 500 bilhões. Além disso, 40% das atuais empresas no mundo serão irrelevantes em até 10 anos devido à aceleração digital e a tecnologias como IoT, *cloud* e dispositivos móveis. As oportunidades de mercado, de acordo com a consultoria *McKinsey & Company*, indicam que o crescimento na área de IoT é variado em diversos segmentos, com ampla possibilidade de aplicações e significante participação no processo de crescimento de diversas empresas. A previsão de geração de receita de empresas que proveem plataformas de conectividade para IoT e TI em 2023 é de 30,2 bilhões de euros, um aumento de 24% em

relação ao ano de 2018 (Dahlqvist, Patel, Rajko & Shulman, 2019).

O barateamento e a miniaturização de *hardware*, assim como a melhor relação custo benefício tornou possível a rápida evolução dos dispositivos embarcados, de forma a permearem nossas vidas oferecendo funções inovadoras (Yoo, Henfridsson & Lyytinen, 2010). No entanto, os mesmos possuem *hardware* de baixo processamento e com limitações de energia, implicando em diversas restrições de recursos. Mesmo assim, estes são capazes de coletar e gerar um alto volume de dados capturados no ambiente em que se encontram. Desse modo, necessitando de maior poder computacional para processar as informações capturadas e atuar de maneira rápida e eficaz no ambiente em que se encontram.

Sendo assim podemos concluir ser necessário o envio dos dados capturados para um centro de maior capacidade para realização do processamento. A computação em *cloud* pode ser integrada com a finalidade de prover disponibilidade, recursos e padronização ao ambiente IoT.

2.2 *Cloud Computing*

A computação em *cloud* possui diversas características que oferecem mecanismos para administrar e processar uma grande quantidade de dados. Desse modo, é capaz de processar um alto volume de informações capturadas pelos sensores de uma rede IoT, por exemplo (Al-Fuqaha, Guizani, Mohammadi, Aledhari & Ayyash, 2015). Com o intuito de centralizar e realizar o tratamento destes dados, os recursos em *cloud* começaram a ser utilizados no ambiente IoT para o processamento das informações capturadas. Assim é possível aliviar os recursos dos dispositivos IoT, que possuem agora como principal função a coleta e o envio dos dados.

O NIST (*National Institute of Standards and Technology*) tem como objetivo desenvolver e prover padrões e diretrizes de tecnologia. Sendo assim, os pesquisadores do NIST, Mell, Grance et al. (2011), definem o paradigma de computação em *cloud* como um modelo que torna viável o acesso onipresente e sob-demanda a um agrupamento compartilhado de recursos na rede, como servidores, serviços e plataformas.

As características essenciais da computação em *cloud* são: o serviço sob-demanda, o amplo acesso à rede, o agrupamento de recursos, a elasticidade e as métricas de serviço. Em relação à característica de serviço sob-demanda, a máquina deve dispor do serviço requerido pelo usuário independentemente de intervenção do provedor deste, estando sempre disponível. O amplo acesso à rede indica que os recursos estão disponíveis na rede sendo acessados por diferentes categorias de dispositivos e protocolos de comunicação. A característica de agrupamento de recursos, esta relacionada com os *clusters* de provedores de serviços de computação para processamento e armazenamento. Os consumidores alocam seus espaços e possuem seus serviços dinamicamente realocados e reatribuídos de acordo com sua demanda tornando este controle abstrato, porém, possuindo conhecimento em relação à região de localização do *da-*

tacenter. A elasticidade e as métricas de serviço são as duas ultimas características essenciais na computação em *cloud*. A primeira busca mostrar a importância dos recursos escaláveis que com a disponibilidade transparece ao usuário a sensação de um serviço ilimitado a qualquer momento. Do mesmo modo, as métricas de serviço possuem grande importância no monitoramento e controle da utilização de recursos, esse sistema deve ser realizado de maneira automática e entregar de maneira transparente ao usuário suas métricas de uso, podendo assim otimizar sua ferramenta (Mell et al., 2011).

Na Tabela 2.1 é apresentada a comparação entre a computação em *cloud* e a Internet das Coisas, estes dois paradigmas possuem aspectos complementares e quando integrados podem resolver diversos problemas. Os dispositivos IoT podem se beneficiar principalmente de recursos virtualmente ilimitados da *cloud*. Por exemplo, a *cloud* pode colaborar no armazenamento e no processamento de dados coletados, economizando memória e energia dos mesmos. A *cloud* pode se beneficiar deste ecossistema ganhando maior permeabilidade no mundo real e ampliando seu escopo para lidar com outras categorias de problemas de forma distribuída e dinâmica aumentando seu ambiente de aplicações (Botta et al., 2016).

Tabela 2.1: Comparação de aspectos complementares da computação em *Cloud* e IoT.

Característica / Paradigma	**Iot**	***Cloud***
Modelo de computação	distribuído ou pervasivo	centralizado
Disponibilidade de acesso	local ou limitado	global ou ubíquo
Natureza dos componentes	objetos físicos	recursos virtuais
Capacidade de processamento	limitada	virtualmente ilimitada
Capacidade de armazenamento	limitada ou nenhuma	virtualmente ilimitada
Função da Internet	ponto de convergência	meio de prover serviços
Análise de dados	análise em tempo real	análise de *Big Data*

Fonte: (Coutinho, Carneiro & Greve, 2016).

Porém, a computação em *cloud* para ambientes IoT possui alguns desafios, como a sincronização em tempo real entre dispositivos e a *cloud*, a padronização entre os serviços de comunicação e estruturas de dados, o balanceamento entre os serviços presentes na *cloud* e os serviços requeridos pelos dispositivos, a confiabilidade na troca de mensagens entre os dispositivos e a *cloud* e, por fim, o complicado gerenciamento de recursos entre os dois ambientes devido as suas grandes diferenças entre componentes e recursos (Al-Fuqaha et al., 2015).

Além disso, o principal problema na integração entre IoT e a computação em *cloud* se deve ao constante envio de dados, que pode gerar sobrecargas na rede. Essa sobrecarga pode aumentar significativamente a latência da comunicação e culminando no retardamento na rede, possibilitando até mesmo perdas de alguns pacotes de dados no tráfego da mesma. A computação em *fog* busca solucionar o problema da latência entre os dispositivos IoT e a *cloud*, atuando como uma camada intermediária entre ambas, fornecendo o processamento mais próximo da borda da rede (Bonomi et al., 2012).

2.3 *Fog Computing*

Aplicações sensíveis à latência podem sofrer problemas quando a arquitetura é baseada na comunicação direta entre dispositivos e a *cloud*. O constante e grande envio de dados gerados pelos dispositivos IoT para a *cloud* pode gerar grande saturação na rede, congestionando o tráfego e inviabilizando certas aplicações (Dastjerdi & Buyya, 2016).

A arquitetura de rede em *fog* é uma abordagem de computação distribuída que estende recursos até os nós subjacentes, estendendo o paradigma de computação em *cloud* até a borda da rede. Desse modo, é capaz de prover serviços melhores de armazenamento, comunicação e computação entre os dispositivos IoT finais, além de contribuir para a diminuição do congestionamento da rede (Aazam & Huh, 2014).

Por estar situada mais próxima da borda da rede quando comparada com a *cloud* e possuir mais recursos computacionais em relação aos dispositivos IoT, a *fog* tem a capacidade de auxiliar na tomada de decisão em tempo real a partir do processamento realizado nos dados nesses dispositivos mais próximos. Além disso, permite aos sensores e atuadores focarem no que importa, a coleta de dados. Em comparação, a camada de *cloud* possui uma visão mais ampla e total do sistema, sendo uma camada mais robusta em recursos, além disso, centraliza os dados auxiliando na tomada de decisão total, como o *Business Intelligence* (Bonomi et al., 2012).

Segundo Al-Fuqaha et al. (2015) a *fog computing* possui vantagens para a aplicação no ambiente IoT devido a sua localização, fornecendo melhor desempenho em relação à velocidade de comunicação, pois está situada entre os dispositivos inteligentes e as máquinas na *cloud*. A escalabilidade é um grande aliado para o sistema IoT permitindo assim a maior quantidade de dispositivos finais no sistema com a *fog* distribuindo entre seus dispositivos este aumento de carga. A replicação de serviços, padronização de comunicação, suporte de mobilidade e análise imediata de dados são outros das muitas vantagens de uma arquitetura em *fog* atuando como uma camada intermediária entre dispositivos IoT e a *cloud*.

Na Figura 2.2 é ilustrada a separação e interação entre as camadas de *cloud computing*, *fog computing* e os dispositivos de borda. As camadas mais baixas possuem um maior número de dispositivos e menor latência.

Podemos assim, concluir que a *fog* é uma camada que tem como objetivo diminuir o volume de dados e o tráfego entre dispositivos IoT e servidores na *cloud* melhorando principalmente a latência da comunicação e a quantidade de recursos consumidos pelos dispositivos (Alrawais et al., 2017).

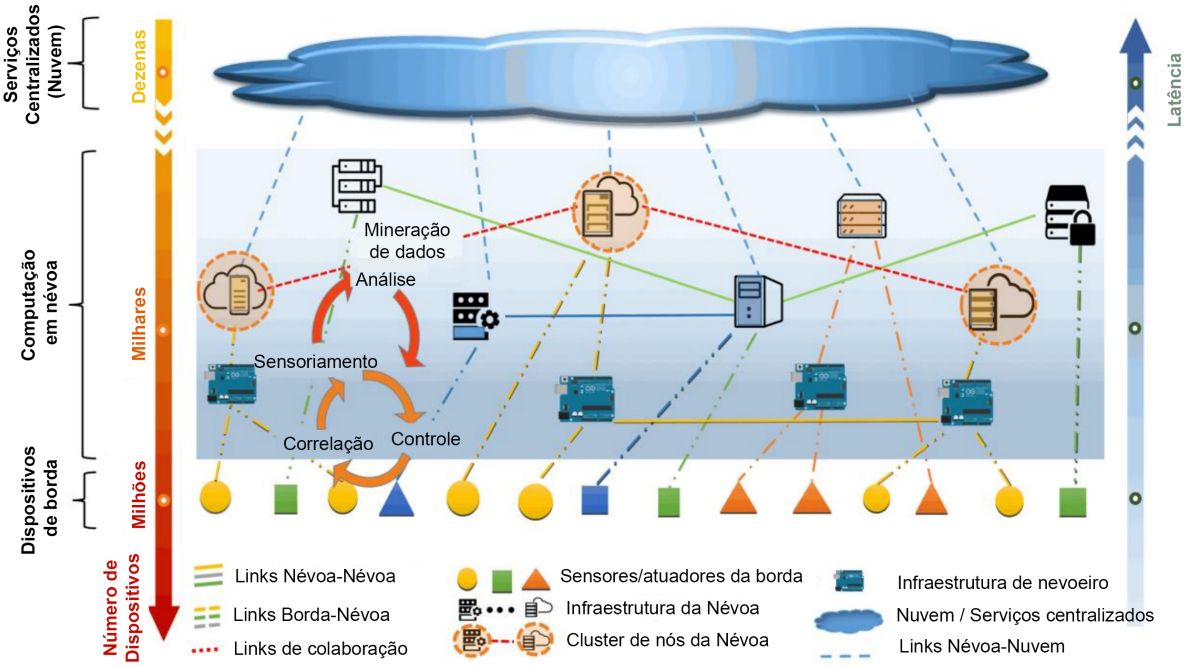

Figura 2.2: Modelo de *fog computing*.
Fonte: (Iorga, Feldman, Barton, Martin, Goren & Mahmoudi, 2017).

2.4 Segurança de Computadores

No início da utilização das redes de computadores a troca de mensagens ocorriam basicamente no meio acadêmico, sendo assim, a segurança de computadores não gerava grandes preocupações entre os envolvidos. No entanto, com a evolução, o passar dos anos e o surgimento de inúmeras aplicações de contexto sigilosos, como, por exemplo, as aplicações bancárias, começaram a circular pela rede informações e dados sensíveis. Desse modo, problemas de segurança foram expostos e ganharam grandes proporções (Tanenbaum & Wetherall, 2003).

Segundo Guttman & Roback (1995) a segurança de computadores pode ser definida como uma proteção a um sistema de informação automatizado com o objetivo de preservar a integridade, disponibilidade e a confidencialidade dos recursos do sistema (incluindo *hardware, software, firmware*, informações/dados e telecomunicações), estes sendo considerados os três pilares da segurança de computadores (Stallings, Brown, Bauer & Bhattacharjee, 2012).

2.4.1 Pilares da Segurança

Segundo Stallings et al. (2012) os objetivos considerados essenciais para a segurança da computação giram em torno de três conceitos:

- Confidencialidade: consiste na confidencialidade dos dados, garantindo que a informação confidencial ou privada não será disponível, ou acessível a pessoas não autorizadas. No

contexto da privacidade, o indivíduo controla ou influência quais informações relacionadas ao mesmo serão coletadas e armazenadas, assim como quem coleta e acessa estas informações. São utilizadas ferramentas com base em encriptação, controle de acesso, autenticação, autorização e segurança física para garantir a confidencialidade dos dados (Goodrich & Tamassia, 2013).

- Integridade: representa a integridade de dados e do sistema, garantido que ambos sejam operados somente por pessoas autorizadas tornando tais objetos controlados. Além disso, no contexto de transmissão de dados, eles devem chegar sempre ao destino sem sofrer nenhuma alteração indevida. Algumas das técnicas projetadas para manter a integridade da informação são as cópias de segurança, as somas de verificação *checksum* e os códigos de correção de dados (Goodrich & Tamassia, 2013).
- Disponibilidade: esta relacionada a garantia de que o sistema trabalhe prontamente e o serviço esteja sempre disponível para o usuário.

2.4.2 Ameaças Computacionais

Este trabalho considera o contexto de ambientes inteligentes baseados em computação em *fog* e IoT. A segurança nesses ambientes é fundamental, pois informações confidenciais geralmente são monitoradas, manipuladas, armazenadas e transmitidas por estes dispositivos. Além disso, muitos sistemas são críticos e precisam ter operação ininterrupta garantida.

No entanto, ambientes inteligentes baseados em computação em *fog* e IoT não estão livres de ameaças (Aversano, Bernardi, Cimitile & Pecori, 2021). Estes ambientes estão sujeitos a ataques externos da Internet (Cardoso, Sampaio, Souza & Westphall, 2019; Ahmad & Alsmadi, 2021). Um ataque perigoso presente neste contexto é a tentativa de obter acesso privilegiado a dispositivos IoT (Muhammad, Anjum & Mazhar, 2015). Através deste acesso, é possível realizar ataques de *Botnet*, onde dispositivos IoT comprometidos podem ser usados como *bots* ou *zumbis*, sendo controlados por um servidor de comando e controle (C&C) para realizar diversas tarefas maliciosas (Ahmad & Alsmadi, 2021). Além disso, os ataques de *Denial of Service* (DoS) são bastante comuns. Este tipo de ataque visa afetar a disponibilidade da vítima. Isso pode ser feito inundando com um grande volume de pedidos ou esgotando recursos como memória e capacidade de computação. No contexto, o dispositivo IoT pode fazer parte da rede sob ameaça ou ser usado como um zumbi para iniciar um *Distributed DoS* (DDoS) em outra rede. A IoT também é vulnerável a ataques tradicionais que visam sistemas de computador, como interceptação, fabricação e modificação de dados de comunicação.

Além disso, ataques internos também podem ocorrer. Nesse contexto, presume-se que o invasor seja uma entidade mal-intencionada que foi autenticada com sucesso no nevoeiro ou um dispositivo IoT legítimo que se tornou malicioso com o passar do tempo, ele pode então realizar vários ataques, incluindo negação de serviço, enviando uma alta taxa de pacotes de

dados para o nevoeiro. Isso permite sobrecarregar os dispositivos de *fog* e camadas superiores, prejudicando ou mesmo causando a interrupção dos serviços prestados pelos sistemas, serviços que em muitos casos são extremamente importantes.

O esforço no desenvolvimento de métodos de segurança são essenciais para estimular ainda mais a adoção das tecnologias IoT ao redor do mundo, fortalecendo a sua aceitação pelo mercado consumidor. Ainda é incerto quais seriam as reais consequências de um incidente cibernético em larga escala derivado de dispositivos IoT, isto se deve ao fato de ser uma tecnologia ainda imatura no mercado, e recebe maior atenção no desenvolvimento de usabilidade e experiência do usuário do que no desenvolvimento de protocolos de segurança de comunicações e armazenamentos (Jalali et al., 2019).

A segurança de redes em IoT possui os mesmos problemas das redes de computadores, comunicação móvel e de Internet. Além disso, possui problemas específicos, como tipos diferentes de autenticação e protocolos de comunicações mais vulneráveis a ataques (Jing et al., 2014). A diversificação entre os dispositivos IoT de uma determinada aplicação e os diferentes serviços prestados também contribuem para agravar os problemas de segurança (Alrawais et al., 2017).

2.4.3 Detecção de Intrusão

Para defender e tornar a rede um lugar mais confiável e com menor possibilidade de intromissão de agentes não confiáveis utilizam-se os Sistemas de Detecção de Intrusão (SDI, em inglês *Intrusion Detection System*, ou IDS).

Um SDI tem como objetivo identificar a ocorrência de ações intrusivas e atividades maliciosas em sistemas computacionais, responsável por alertar e em alguns casos executar ações de contramedidas para conter potenciais violações e ameaças computacionais (Axelsson, 2000).

Os nós de dispositivos IoT em sua maioria dispõem de um limitado poder computacional e baixa capacidade de armazenamento. Desse modo, são geralmente utilizados métodos mais leves de criptografia e comunicação. Isso, expõe vulnerabilidades mais fáceis de serem exploradas e códigos mais fáceis de serem quebrados. Além disso, essas limitações dificultam ou inviabilizam a implantação de certos SDI nos nós de dispositivos IoT. O fato da arquitetura de uma aplicação IoT ser multicamadas também dificulta a implantação de um SDI que opera apenas em uma única camada (Alaba et al., 2017).

Os SDI possuem principalmente três componentes lógicos segundo Stallings et al. (2012): Os sensores, responsáveis por coletar os dados e estão presentes nas partes do sistema onde podem ocorrer intrusões, eles avaliam entradas como pacotes de dados e arquivos de *logs* encaminhando para o analisador. Os analisadores são responsáveis por analisar os dados de um ou mais sensores e identificar ocorrências de intrusões. Caso a saída seja uma indicação da ocorrência de

intrusão, esta deve conter a evidência que confirme a conclusão tomada pelo analisador, assim como, providenciar suporte sobre quais ações devem ser tomadas sobre o resultado da intrusão. As entradas dos sensores também podem ser armazenadas em bases de dados para análises futuras. A interface do usuário é a última dos componentes lógicos e é responsável por apresentar as saídas e controlar o comportamento do sistema na totalidade, sendo um componente gerencial.

Um aspecto importante em relação ao SDI é o método de detecção responsável por analisar os dados e identificar as intrusões. A detecção por comportamento ou anomalia, identifica os comportamentos anômalos do *host* ou da rede assumindo que os ataques diferem das atividades normais. O perfil normal base do comportamento da rede ou *host* é construído, e são detectados os comportamentos que são anormais ao perfil base. Os sensores então coletam dados de eventos e o analisador verifica quando a atividade monitorada desvia da normalidade do histórico de atividade anterior Bace & Mell (2001). O analisador neste caso geralmente é baseado em métodos automatizados de *Machine Learning* (ML). Os ataques desconhecidos são melhores identificados por esta abordagem de detecção, pois se identifica como intrusão tudo o que for anômalo ao comportamento normal predefinido. No entanto, pode ocorrer uma taxa alta de falsos positivos (Vasilomanolakis, Karuppayah, Mühlhäuser & Fischer, 2015). Dado o constante crescimento da rede de computadores e a quantidade de dados a serem analisados em um período curto de tempo as técnicas de detecção de intrusão por anomalia tendem a melhorar conforme elas consigam colaborar entre si, aumentando sua eficiência. Entretanto, mesmo assim terá problemas para detectar ataques mais sofisticados e distribuídos(Ahmed, Mahmood & Hu, 2016).

Existe também a detecção por especificação, que consiste em uma variação da detecção baseada em anomalia, onde um especialista humano define previamente um escopo comportamental legítimo para os nodos, protocolos e tabelas de roteamento, por exemplo, e então o sistema detecta a intrusão quando o comportamento do evento desvia do mesmo. Uma vantagem é o baixo número de falso negativo já que somente os eventos não definidos serão acusados como eventos maliciosos e o sistema estará imediatamente disponível, pois não há fase de treinamento anterior, a principal desvantagem é o processo de especificação do comportamento do tráfego benigno (Mitchell & Chen, 2014).

Por fim, em uma estratégia diferente, a detecção baseada por assinatura se baseia em padrões de ataques, também chamadas assinaturas de comportamentos de eventos intrusivos anteriormente catalogados. A abordagem compara dados, como pacotes de rede e logs de eventos, com conjuntos de assinaturas de ataques conhecidos, para definir se os eventos presentes apresentam alguma ameaça ao sistema. Esta categoria de método de detecção é a técnica mais comumente utilizada em SDI comerciais (Vasilomanolakis et al., 2015; Axelsson, 2000).

No que concerne ao alvo que será monitorado pelo SDI, ou seja, o ambiente de monitoramento das informações que serão analisadas pelo método de detecção. Segundo Machado et al. (2005) e Lima et al. (2005), eles podem ser classificados em baseados em *host* ou baseados em rede:

- Detecção de intrusão baseada em *Host*: Também denominado *Host Based Intrusion Detection System* (HIDS), baseia-se principalmente nos componentes de máquina local onde está instalado, verificando através de captura de *logs* do Sistema Operacional, dados sobre o usuário, serviços, pacotes de rede relacionados a atividades locais. Tal abordagem torna a máquina e o SDI independentes da rede, são capazes de detectar ameaças internas e também gerar ações e contra-medidas contra as mesmas. Porém, há um grande dispêndio de recursos em uma análise contínua do sistema, ocasionando problemas de degradação do desempenho do *host* diminuindo a confiabilidade e aumentando os riscos de ataques afetares componentes do mecanismo de detecção.
- Detecção de intrusão baseada em rede: também conhecido como *Network Bases Intrusion Detection System* (NIDS), são sensores inseridos em vários pontos estratégicos da rede onde esse conjunto captura em modo promíscuo o tráfego da rede, assim, não monitorando somente a máquina como também todo o tráfego da rede. O custo computacional pode ser muito alto dependendo da velocidade da rede, já que uma análise precisa ser feita de um grande fluxo de dados, podendo assim, impactar o tempo de resposta do sistema, outro problema enfrentado em relação a esta arquitetura é o posicionamento dos sensores de detecção.

Um SDI localizado na camada de *fog* é uma alternativa para o ambiente IoT, considerando a proximidade da borda física da rede e o maior fornecimento de recursos de processamento e armazenamento. Possuindo assim, maior capacidade em analisar o tráfego da totalidade da rede, protegendo a rede de dispositivos contra ataques externos e evitando que os dispositivos sejam utilizados para atacar o mundo exterior.

A detecção por anomalia vem obtendo bastante destaque nos trabalhos do estado da arte por sua capacidade de detectar ataques desconhecidos. Essas abordagens geralmente são baseadas em métodos de *Machine Learning* e possuem custos computacionais relacionados a treinamento e predição. O ambiente IoT e de *fog computing* possui restrições de recursos, desse modo, é necessário desenvolver soluções leves para operar neste contexto. As abordagens de seleção de atributos tem como objetivo justamente reduzir a dimensionalidade dos dados analisados para construir modelos de detecção mais leves. Um método seletor de atributos pode ser capaz de diminuir a dimensionalidade total do fluxo do tráfego de rede, assim melhorando a *performance* e diminuindo a utilização de recursos utilizados pelo classificador de eventos, dentro do SDI. No próximo capítulo serão apresentados maiores detalhes a respeito de técnicas de *machine learning* que podem ser utilizadas em analisadores baseados em anomalia.

Capítulo 3

Inteligência Computacional

A Inteligência Computacional, ou Inteligência Artificial, possui diversas definições. Segundo Russell & Norvig (2002), elas estão geralmente relacionadas aos sistemas pensarem ou agirem como humanos ou os sistemas pensarem ou agirem racionalmente. Mas de maneira geral pode-se concluir que a inteligência computacional atua nos sistemas buscando tomar a melhor ação possível em uma situação, reproduzindo atividades inteligentes e capacidades encontradas em humanos.

Na área de segurança computacional, métodos baseados em inteligência artificial estão sendo empregados em diversas esferas. Em relação a de detecção de intrusão destaca-se a utilização desses métodos inteligentes no processo de seleção de atributos e na análise de eventos.

3.1 Extração de Conhecimento de Dados

Diante de grandes avanços tecnológicos um maior número de dispositivos estão conectados à Internet. Desse modo, maiores volumes de dados são gerados e transmitidos a todo momento. Estes dados também são geralmente armazenados em bancos de dados, os quais precisam ser conhecidos e entendidos pelos responsáveis, para assim, extraírem conhecimento para auxiliá-los nas tomadas de decisões. O desenvolvimento de técnicas para extração de informações de um conjunto de dados ainda desconhecido vem sendo realizado há séculos, desde a utilização de ferramentas e técnicas arcaicas e manuais. Segundo a Lei de Moore (Moore, 1965; Mollick, 2006), a evolução dos dispositivos tecnológicos acontece de diferentes maneiras, por exemplo, o aumento da capacidade de processamento de uma CPU, memória e *cache* e do aumento armazenamento de dados, sendo a última com maior velocidade de evolução na capacidade pelo mesmo valor ao passar dos anos. O resultado desta combinação nos mostra que somos mais capazes em gerar dados que em nossa habilidade de processar e obter informações destes.

Todos esses avanços propiciaram um grande aumento na complexidade dos dados a serem analisados tornando cada vez mais necessário o aperfeiçoamento de técnicas de análise e processamento de dados. A descoberta de conhecimento em bases de dados (do inglês, *Knowledge*

Discovery in Databases ou KDD), busca por meio de métodos de análise resolver problemas, buscar por percepções, situações anômalas, tendências, padrões e sequências nos dados armazenados e organizados no ambiente computacional que se encontra.

A realização de predições e classificações são possíveis a partir da obtenção do entendimento dos dados durante o processo de KDD através de técnicas de análise de dados. O processo de KDD se divide em grandes etapas essenciais, as quais são muitas vezes desempenhadas de forma iterativa buscando refinar o entendimento da área de aplicação. Embora existam opiniões divergentes acerca das fases que compõem o processo de KDD, a seguir são apresentadas as etapas consideradas básicas segundo Kantardzic (2011); Kurgan & Musilek (2006):

1. Definição do domínio da aplicação: é imprescindível o esforço na definição do escopo do domínio da aplicação e a especificação de como o conhecimento vai ser utilizado. Geralmente este conhecimento é empregado tendo como objetivo, por exemplo, gerar uma análise de risco, gerar um sistema de recomendação ou de identificação do perfil do usuário. Desse modo, definem-se as condições e metas do usuário final, de maneira ideal esta etapa conta com as duas figuras principais, um especialista em mineração de dados e um especialista na área de aplicação. O estudo de viabilidade e os custos da aplicação também podem ser inseridos nesta etapa, que pode impactar todo o processo de KDD.

2. Seleção da amostra e entendimento dos dados: uma amostra significativa é essencial para o processo de KDD, pois influenciará diretamente os resultados. No caso da amostra não ser representativa, os dados a serem classificados terão um baixo índice de assertividade. Desse modo, ao se criar a amostra deve-se considerar fatos como tamanho, homogeneidade e dinâmica. As estratégias para obtenção da amostra podem ser realizadas de dois modos gerais, modelando o experimento ou através de uma abordagem observacional. Na primeira o especialista gera os dados e tem o total controle do processo. Por outro lado, na segunda estratégia são capturados dados aleatórios e com distribuições geralmente desconhecidas, mais próximo ao mundo real.

3. Pré-processamento: após a seleção é realizada uma etapa de pré-processamento para preparação e redução da amostra. A amostra é geralmente disponibilizada em conjuntos de dados organizados em um formato digital, como bancos de dados, planilha ou tabelas de representação atributo-valor. São realizadas algumas tarefas necessárias, assegurando uma melhor qualidade da informação e colaborando com as próximas etapas do processo. Essas tarefas consistem principalmente na eliminação de dados duplicados e dados inválidos, no tratamento de ruídos nos dados, como os *outliers*, na manipulação de atributos faltantes ou duplicados, na normalização de dados, na redução do número de atributos através do processo de seleção de atributos, entre outras.

4. Mineração de dados (MD): a etapa de mineração de dados (do inglês *Data Mining*, ou DM) envolve a utilização de algoritmos capazes de, em modo automático ou semi automático, extraírem padrões válidos, compreensíveis e potencialmente úteis dos dados para

gerar uma estrutura que possa representar o conhecimento implícito deste conjunto de dados. Três componentes principais compõem esses algoritmos, o primeiro é o modelo, composto de função e representação, o segundo é o critério de preferência ou bias e por último o algoritmo de busca. Tais parâmetros variam conforme a natureza e as características do conjunto de dados ou do tipo da análise que pretende-se realizar com o modelo gerado.

5. Avaliação e consolidação do conhecimento: os usuários do processo de KDD precisam entender e julgar a utilidade do conhecimento extraído pelo processo. Geralmente a complexidade do modelo está relacionada com a complexidade do domínio. Estes modelos precisam auxiliar em tarefas de grande relevância ou até mesmo problemas críticos. Sendo assim, a tarefa de avaliação dos modelos resultantes do processo de KDD é uma das suas fases mais relevantes. Esta avaliação é realizada utilizando-se métodos para fazer uma filtragem do conhecimento extraído e encontrar os melhores modelos e também utilizando-se métricas de desempenho e métodos analíticos para mensurar a capacidade do modelo. Além disso, deve-se levar em consideração técnicas de visualização.

As técnicas empregadas nas etapas de um processo de KDD podem ser utilizadas para avaliar dados e modelos para detecção de intrusão em redes de computadores. A seguir são apresentadas algumas técnicas de seleção de atributos e mineração de dados que podem ser aplicados no contexto de detecção de intrusão.

3.2 Métodos de Seleção de Atributos

A existência de dados irrelevantes e redundantes em conjuntos de dados de grande dimensionalidade pode interferir negativamente no desempenho dos classificadores (Zhou, Cheng, Jiang & Dai, 2020). A abordagem por anomalia, por exemplo, é afetada pela qualidade dos atributos analisados. Além disso, ruídos possuem a capacidade de interferir no resultado da análise.

O processo de seleção de atributos busca a otimização da base de dados. Busca-se encontrar o melhor subconjunto de atributos, de acordo com o critério estabelecido e partindo de um conjunto fixo de atributos originais. Desse modo, é possível tornar o SDI computacionalmente mais rápido e eficiente. Não necessariamente existirá apenas um subconjunto de dados ótimo. Diferentes subconjuntos de dados podem garantir desempenhos iguais, isto dependerá do conjunto original de dados, do método de processamento e do critério selecionado de parada que esta sendo utilizado (Hota & Shrivas, 2014).

Os dispositivos em geral lidam com uma grande quantidade de tráfego diariamente. Além do grande volume de dados, há também uma quantidade grande de características do tráfego que podem ser analisadas por SDIs. Visando uma detecção mais eficiente é muito interessante tratar

esta sobrecarga de dados, a seleção de atributos é uma ótima alternativa para melhoria nesse quesito. Características significativas sobre classes de dados, padrões intrusivos e combinações de atributos podem ser revelados. Além disso, padrões redundantes, atributos de falsa correlação ou atributos irrelevantes podem ser removidos. Desse modo, colabora-se com os algoritmos de classificação, permitindo que estes lidem com um menor e mais relevante conjunto de atributos (Zainal, Maarof, Shamsuddin et al., 2009).

É essencial, para o bom funcionamento do SDI, trabalhar com um subconjunto de atributos de boa qualidade e seja representativo o suficiente para o aprendizado do algoritmo de classificação (Zhou et al., 2020).

A seguir são apresentadas as principais abordagens para a seleção de atributos segundo Blum & Langley (1997).

Tabela 3.1: Técnicas de seleção de atributos.

Método	**Vantagens**	**Desvantagens**
Filter	Independente de classificador, menor custo computacional em comparação com algoritmos *wrapper*, rápido, boa habilidade de generalização	Não possui interação com classificador
Embutido	Possui interação com o classificador, custo computacional menor que algoritmos *wrapper*, captura a dependência entre atributos	Seleção é dependente do classificador
Wrapper	Possui interação com o classificador, captura a dependência entre atributos	Computacionalmente custoso, risco de super ajuste (*overfitting*), seleção é dependente do classificador

Fonte: (Bolón-Canedo, Sánchez-Maroño & Alonso-Betanzos, 2015).

3.2.1 Métodos *Embedded*

Os métodos *embedded* são caracterizados pela realização da seleção de atributos de forma dinâmica, selecionando um subconjunto de atributos no próprio processo de construção do modelo de classificação. Conforme ilustrado na Figura 3.1, isso ocorre durante a fase de treinamento, enquanto procuram pela sua hipótese, realizando as duas partes em conjunto (Baranauskas et al., 2002; Almeida et al., 2018).

As abordagens embutidas costumam ser computacionalmente menos intensivas comparados com os métodos *wrapper*. Pois, há uma interação entre o algoritmo de seleção de atributos com o processo de aprendizado mais incorporada, se tratando de um mesmo algoritmo (Zhou et al., 2020). Tais algoritmos possuem funções integradas de risco regularizado para otimizar os recursos que designam os parâmetros de *machine learning* e os parâmetros de seleção de

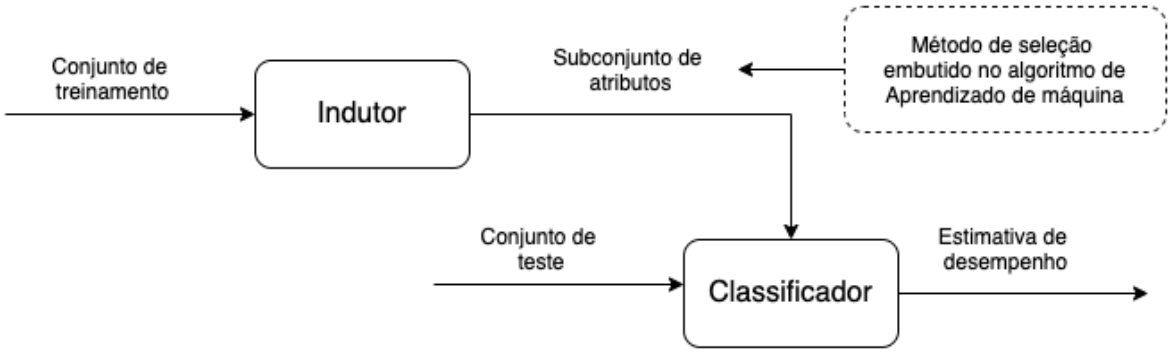

Figura 3.1: Abordagem embutida.
Fonte: Adaptado de Lee (2005).

atributos (Bolón-Canedo et al., 2015).

3.2.2 Métodos *filter*

Os métodos *filter* são baseados em métricas estatísticas. Eles são independentes do classificador. Tais métricas são calculadas aplicando o algoritmo e buscam remover atributos irrelevantes antes da indução ocorrer, conforme ilustrado na Figura 3.2. A avaliação de cada atributo é realizada individualmente baseada em sua correlação com a função alvo. Após isto então seleciona-se os k atributos com os maiores valores (Blum & Langley, 1997).

A seleção do mínimo subconjunto de atributos e da classificação de atributos são algumas das estratégias utilizadas na abordagem *filter*. A primeira pode causar problemas caso não considere a influência no resultado geral da aprendizagem se aplicado livremente. Já a segunda abordagem pode ser gerada a partir de métodos estatísticos para a avaliação de atributos, qualificando e ordenando uma lista de atributos por exatidão, consistência, distância e dependência (Galvão et al., 2007).

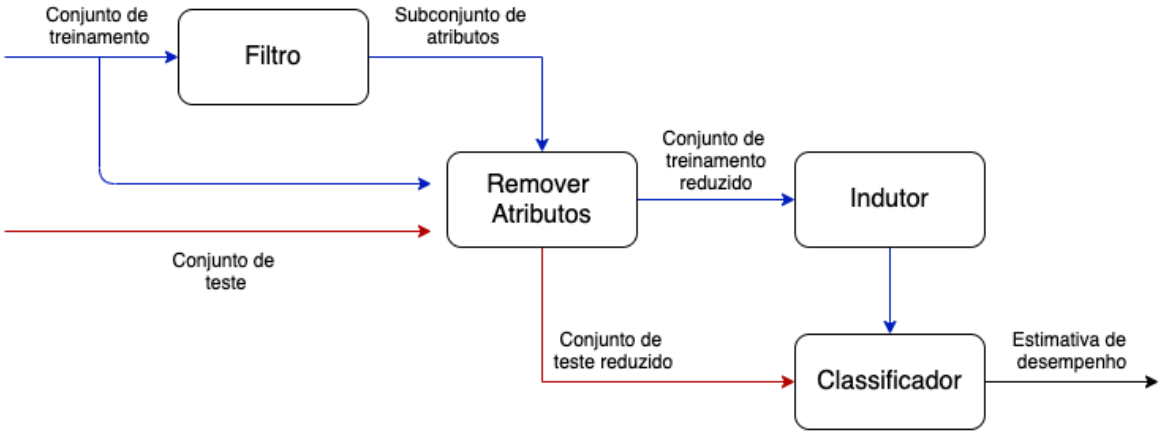

Figura 3.2: Abordagem *filter*.
Fonte: Adaptado de Baranauskas et al. (2002).

Dentre os métodos *filter*, o mais conhecido e utilizado na literatura é o ganho de informação (do inglês *Info Gain*). Ele avalia o grau de associação do atributo com a classe para encontrar os valores com o maior grau de utilização e importância calculando a redução da entropia de cada atributo. Quanto maior a entropia, maior o grau de impureza. O ganho de informação indica a redução da entropia. Desse modo, os atributos que possuírem o maior ganho de informação serão os mais úteis para o propósito de aprendizado futuro (Manzoor, Kumar et al., 2017; Hota & Shrivas, 2014).

Considerando-se uma base de dados $D(A_1, A_2, ..., A_n, C)$, com $n >= 1$, onde C é o atributo classe e o seu domínio é $(c_1, c_2, ..., c_m)$, com $m >= 2$. Assume-se que os valores dos atributos e da classe são discretos.

Segundo Quinlan (1986), a medida do ganho de informação é baseado no conceito de entropia. Entropia é uma medida de impureza e falta de homogeneidade de um atributo. A fórmula apresentada na Equação 3.1 realiza o cálculo da entropia para um atributo A, cujo domínio é $(a_1, a_2, ..., a_k)$, com $k >= 1$. Os valores p_i, com $1 <= i <= k$, correspondem a razão entre o número de instâncias da base em que ocorre o valor a_i para o atributo A e o número total de instâncias.

$$Entropia(A) = \sum_{i=1}^{m} p_i \log_2(p_i) \tag{3.1}$$

A entropia da classe C, representada por $Entropia(C)$, pode ser calculada da mesma forma, considerando p_j a razão entre o número de instâncias em que o valor c_j da classe, com $1 <= j <= m$, ocorre na base e o número total de instâncias.

O ganho de informação de um atributo é determinado pela redução da sua entropia (Karegowda, Manjunath & Jayaram, 2010). A Equação 3.2 calcula o ganho de informação de um atributo A em relação à classe C. Onde $Entropia(C)$ é a entropia da classe C, e $Entropia(C, A)$ é entropia da classe relativa ao atributo A, calculada pela Equação 3.3.

$$Ganho(C, A) = Entropia(C) - Entropia(C, A) \tag{3.2}$$

A $Entropia(C|A = A_j)$ é a entropia relativa ao subconjunto de instâncias que tem um valor A_j para o atributo A. Se A é um bom descritor para a classe, cada valor de A terá uma baixa entropia distribuída entre as classes, ou seja, cada valor deve estar predominantemente em uma classe.

$$Entropia(C, A) = -\sum_{j=1}^{m} p(A, C) \log_2(p(A, C)) \tag{3.3}$$

Tem-se que o valor do ganho de informação consiste na diminuição da impureza, ou seja, atributos com menor entropia possuem maior ganho de informação. Esta medida tem um *bias*

natural, pois favorece atributos que possuem muitos valores.

A medida de razão de ganho de informação tenta corrigir o *bias* do Ganho de Informação (Quinlan, 1993).

$$Razão\ de\ Ganho(C, A) = \frac{Ganho(C, A)}{Entropia(A)} = \frac{Entropia(C) - Entropia(C, A)}{Entropia(A)} \tag{3.4}$$

Após o cálculo do mérito de cada atributo pela Equação 3.4, é gerado um ranking e seleciona-se os N melhores atributos desse ranking de acordo o *threshold* definido para o corte.

3.2.3 Métodos *Wrapper*

Os métodos *wrapper* trabalham com subconjunto de atributos gerados a cada iteração do algoritmo. O indutor é executado com esses atributos no conjunto de treinamento e tem como resultado uma precisão do classificador. Ela é utilizada para avaliar a qualidade do subconjunto de atributos como apresentado na Figura 3.3. O processo é repetido para cada subconjunto de atributos até que o critério de parada seja satisfeito (Baranauskas et al., 2002; Lee, 2005).

Devido a grande quantidade de iterações e induções em caixa-preta realizada durando a seleção de atributos o custo computacional da abordagem *wrapper* é relativamente maior comparada as outras abordagens de seleção de atributos.

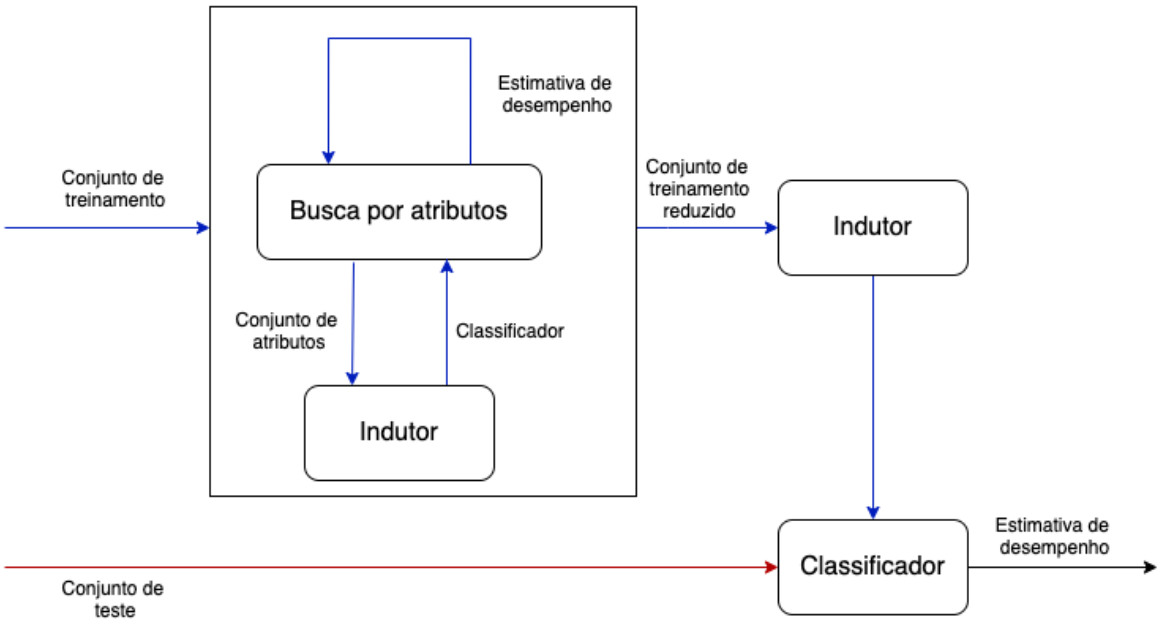

Figura 3.3: Abordagem *wrapper*.
Fonte: Adaptado de Lee (2005).

Os algoritmos *wrapper* geralmente trabalham de forma gulosa, com a utilização de uma

função heurística para conduzir a busca de um subconjunto com a melhor qualidade. Entre as abordagens mais utilizadas estão a *forward selection* e a *backward elimination*, que serão apresentadas a seguir (John, Kohavi & Pfleger, 1994).

- Estratégia de busca *Forward Selection*: A estratégia de busca *forward selection* se refere a uma busca que tem como vazio o conjunto inicial. A exploração é então iniciada a partir do estado pai onde são criadas variações do estado atual e gerando estados filhos, adicionando variáveis a este subconjunto que antes eram ausentes no estado pai, como apresentado na Figura 3.4. Cada componente criado dinamicamente é então avaliado pelo motor indutor de classificação. Caso sua avaliação seja melhor que o pai, este se torna o estado pai da próxima iteração. Um critério de parada comum é quando não é encontrado um estado filho melhor que o estado pai. A velocidade de construção de um classificador baseado em uma busca *forward* é mais rápida, pois quando se tem uma menor quantidade de atributos nos subconjuntos se torna mais rápido a comparação e classificação (John et al., 1994; Almeida et al., 2018). Esta estratégia de busca é uma forma gulosa de percorrer o espaço, sempre adicionando a variável que traz a maior contribuição para incrementar a *performance* do subconjunto.

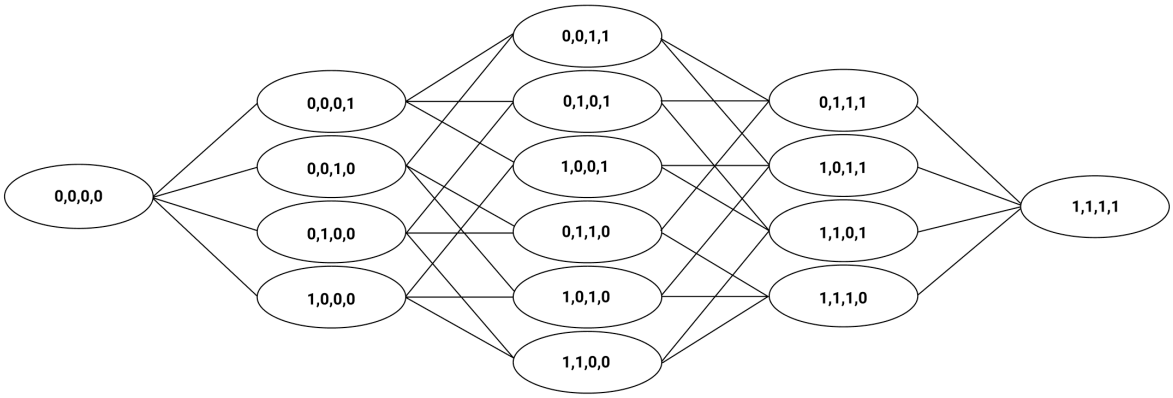

Figura 3.4: Espaço de busca com elipses representando os nodos e os bits representando o conjunto de atributos.
Fonte: Adaptado de John et al. (1994).

- Estratégia de busca *Backward Elimination*: De maneira oposta, a estratégia de busca *backward elimination* tem como estado inicial o subconjunto total de atributos da base de eventos. Desse modo, captura as relações de interdependência entre atributos dentro de uma base com maior facilidade que a abordagem *forward*. A partir de um estado pai, um atributo é removido, gerando um estado filho. O estado filho é classificado pelo motor de indução, caso o estado filho performe superior ao estado pai, é aferido que o atributo removido não possui relevância no conjunto avaliado (John et al., 1994).

3.2.4 Métodos Ensemble

Os métodos *ensemble* combinam múltiplas abordagens buscando uma melhor *performance* em relação a abordagens que atuam de maneira isolada (Tsai, Hsu & Yen, 2014).

Algoritmos que utilizam métodos *Ensemble* para combinar decisões de modelos de classificação estão sendo utilizados amplamente nas áreas de reconhecimento de padrões e *machine learning* (Tsai et al., 2014). Este aprendizado conjunto consiste em combinar vários modelos básicos de classificadores para melhorar o desempenho. Um classificador de conjunto tende a ser mais preciso, reduzindo o risco de selecionar o classificador errado. Porque combinando suas previsões, você pode estabilizar o resultado (Dietterich, 2000a).

Existem diversas técnicas ensemble que podem ser utilizadas, entre elas Bagging, Boosting e Stacking. Bagging (Breiman, 1996) é um método que gera várias versões dos classificadores básicos e as usa para obter um classificador agregado. As várias versões são formadas fazendo réplicas bootstrap do conjunto de aprendizado e usando-as como novos conjuntos de aprendizado para cada versão. O elemento vital é a instabilidade do classificador base. Se perturbar o conjunto de aprendizado pode causar mudanças significativas no classificador construído, então o Bagging pode melhorar a precisão (Breiman, 1996). O Boosting é uma abordagem de aprendizado conjunto para reduzir principalmente o *bias* e também a variação no aprendizado supervisionado. O objetivo é converter classificadores básicos, em um classificador agregado. No entanto, a combinação é realizada de forma diferente do Bagging, neste caso, cada classificador funciona dentro das limitações dos classificadores anteriores. Desta forma, os classificadores base posteriores só são treinados ou testados nos casos em que os classificadores anteriores não conseguiram atingir uma precisão satisfatória (Bartlett, Freund, Lee & Schapire, 1998). Por fim, a estratégia Stacking considera diversos modelos heterogêneos simples, ela os treina em paralelo e os combina para gerar um metamodelo para produzir uma previsão baseada nas previsões dos diferentes modelos fracos. Com base nestas estratégias foram criados vários algoritmos ensemble para classificação.

Neste contexto, algumas abordagens baseadas em *ensemble* de seletores de atributos tem sido propostas. A combinação de diversos seletores de atributos, podem resultar em uma seleção de atributos mais consistente de modo a alimentar um algoritmo de classificação final com uma menor dimensionalidade na quantidade de atributos, melhorando sua *performance* e seu custo computacional.

3.3 Classificadores baseados em *Machine Learning*

Existe uma grande variedade de abordagens de classificadores capazes de realizar análise sobre os dados gerando uma classificação. Estes métodos são comumente empregados na etapa

de mineração de dados do processo de KDD. Dentre elas estão as abordagens estatísticas, que analisam métricas uni variadas, multi-variadas ou séries temporais. Existe também a abordagem baseada em conhecimento, que utiliza de um conjunto de regras que modelam comportamentos anômalos e benignos.

Por fim, existem as abordagens baseadas em *machine learning*. A seguir são apresentados alguns métodos utilizados em problemas de classificação. Eles realizam um processo de treinamento a partir da extração de conhecimento de dados gerando um modelo para classificar eventos futuros (Stallings et al., 2012).

3.3.1 Redes Neurais

As Redes Neurais Artificiais (RNA) foram criadas inspiradas no cérebro humano, onde bilhões de neurônios interconectados são capazes de processar informações paralelamente em uma estrutura altamente complexa (Wang, 2003). Nessa estrutura os dados de entrada são inseridos em uma camada inicial de neurônios, que representam os dendritos. Os neurônios são responsáveis por realizar funções de soma e aplicar a função de ativação no neurônio. A estrutura pode possuir uma ou mais camadas de neurônios ocultos, dando maior profundidade para a rede. As ligações que ocorrem entre as camadas são realizadas e ponderadas com pesos sinápticos, representando o corpo celular simulando as sinapses. Por último, a estrutura possui uma camada de saída que também é composta por neurônios, conforme ilustrado na Figura 3.5.

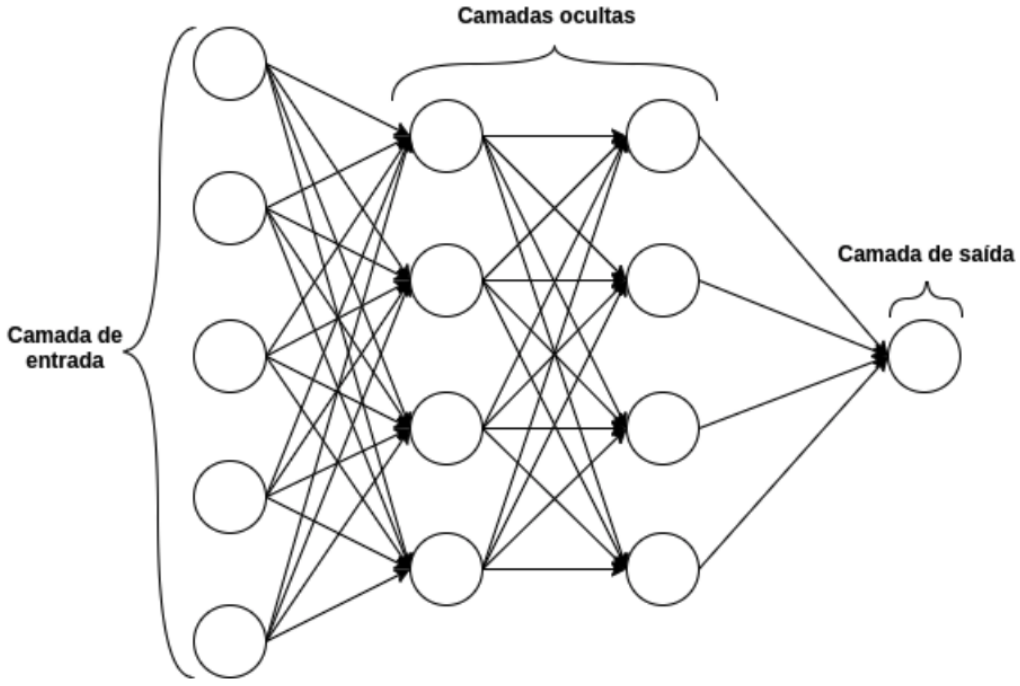

Figura 3.5: Arquitetura simplificada de uma rede neural.
Fonte: Adaptado de Souza et al. (2018).

3.3.2 K-Nearest Neighbor

O algoritmo kNN (K-*Nearest Neighbor*) é um algoritmo de aprendizagem supervisionado baseado em instâncias. Para classificar um novo dado é realizada uma comparação de similaridade entre este dado e os da base de exemplos. Os k exemplos rotulados mais próximos são encontrados e então a classe majoritária entre eles será atribuída ao novo dado (Ferrero, 2009).

Estes algoritmos necessitam de pouco esforço na fase de treinamento, porém o custo computacional para classificar um novo exemplo é alto, pois no pior caso, este deverá ser comparado com todos os exemplos presentes no modelo, tornando o processo de classificação custoso.

Além do conjunto de exemplos para realizar o treinamento, a medida de similaridade a ser utilizada também possui grande influência no algoritmo, este é responsável por medir o quão similar um novo exemplo é em relação aos exemplos presentes na base. As principais medidas são as de distância e de correlação, elas podem influenciar as taxas de aprendizado e os custos de processamento (Ferrero, 2009; Aha, Kibler & Albert, 1991).

A cardinalidade do conjunto de vizinhos mais próximos, ou tamanho do k, é um importante parâmetro do algoritmo. Ela define a quantidade de vizinhos que serão considerados para a classificação do novo dado, ou seja, caso o $k = 1$, então é atribuído ao dado a classe do exemplo mais próximo segundo a medida de similaridade utilizada. Se $k > 1$ então é levado em consideração a classe dos k exemplos mais próximos para realizar a classificação, onde a classe majoritária é comumente a escolhida para ser a atribuída ao exemplo.

Na Figura 3.6 dois casos são ilustrados, o primeiro na Figura 3.6a onde $k = 1$, o exemplo X_i é classificado de acordo com a classe do exemplo mais próximo, ou seja, negativo. Na Figura 3.6b podemos observar que a classe atribuída ao exemplo X_i será positiva, pois a classe majoritária entre os três vizinhos mais próximos é positiva. Assim podemos concluir que o número de vizinhos a ser utilizado para definir o resultado da classificação tem forte influência no processo de classificação, no entanto, o número de vizinhos a ser utilizado depende do escopo do problema, não existindo um valor ideal para o k (Ferrero, 2009).

3.3.3 Árvores de Decisão

A indução por árvores de decisão é o algoritmo de *machine learning* mais amplamente utilizado (Rokach, 2016). Possui objetos ou situações como entrada, descritas com um conjunto de propriedades. A saída pode ser representada em forma binária ou multi classe. A árvore de decisão é composta por nodos de decisão, que podem ser divididos em nodo raiz, um conjunto de nodos intermediários e um conjunto de nodos folhas. Tanto no nodo raiz, quanto nos nodos intermediários estão presentes internamente regras de decisão para definir o destino do resul-

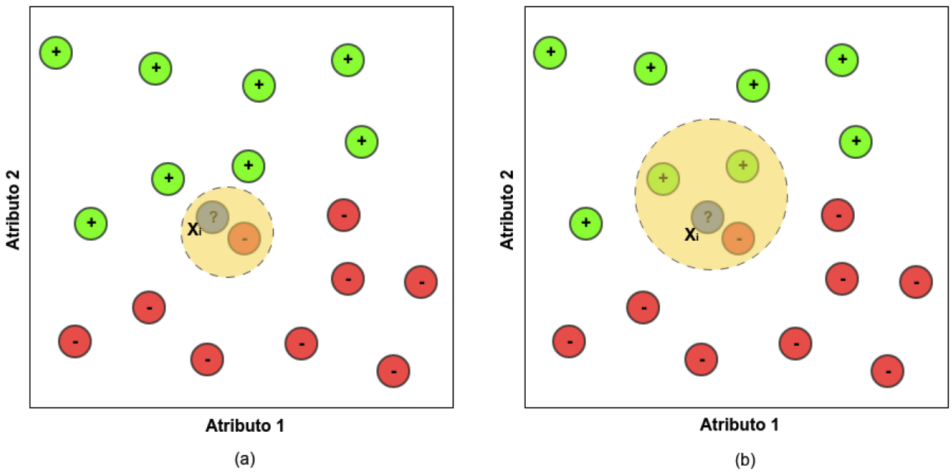

Figura 3.6: (a) Exemplo de k = 1. (b) Exemplo de k = 3.
Fonte: Souza et al. (2018).

tado. Com base nos valores recebidos, estes pontos de decisão são responsáveis por definir uma saída para o nodo. Ela pode ser o resultado da classificação, no caso de um nodo folha que corresponde a uma classe, ou uma entrada para o nodo subsequente a partir de uma aresta, onde este nodo subsequente é o nodo raiz de uma subárvore que possui a mesma estrutura da árvore (Russell & Norvig, 2002; Rokach, 2016).

Na Figura 3.7 é apresentado um exemplo de uma árvore de decisão para o diagnóstico de um paciente. As elipses representam os nodos de decisão, onde regras são aplicadas utilizando o atributo como entrada, e as arestas indicam as saídas, neste caso valores binários (sim e não). Cada retângulo, chamados nodos folhas, representam a classe, neste caso o diagnóstico do paciente. O fluxo inicia no nodo raiz, seguindo para uma das subárvores conectadas por arestas ou para um nodo folha.

Baseado em Monard & Baranauskas (2003), a construção de uma árvore de decisão com um conjunto de treinamento *A* e um conjunto de classes $\{C_1, C_2, C_3, ..., C_k\}$ pode ser realizada com seguintes passos:

1. Se *A* contém um ou mais exemplos e todos pertencentes à mesma classe C_j então a árvore de decisão para *A* é um nodo folha representando a classe C_j;

2. Caso *A* não contenha exemplos, a árvore é uma folha, porém a classe associada à folha deve ser determinada a partir da informação além de *A*, por exemplo, a classe mais frequente do nodo pai pode ser a utilizada;

3. Se *A* contém exemplos que pertençam a mais de uma classe é necessário refinar *A* em subconjuntos de exemplos que sejam pertencentes a uma única classe ou um conjunto de classes com maiores similaridades. Um teste é escolhido para ser realizado sobre os exemplos do conjunto *A*, cada indutor possui sua própria maneira de escolher qual o atributo que será considerado no teste gerando resultados mutuamente exclusivos. Sendo

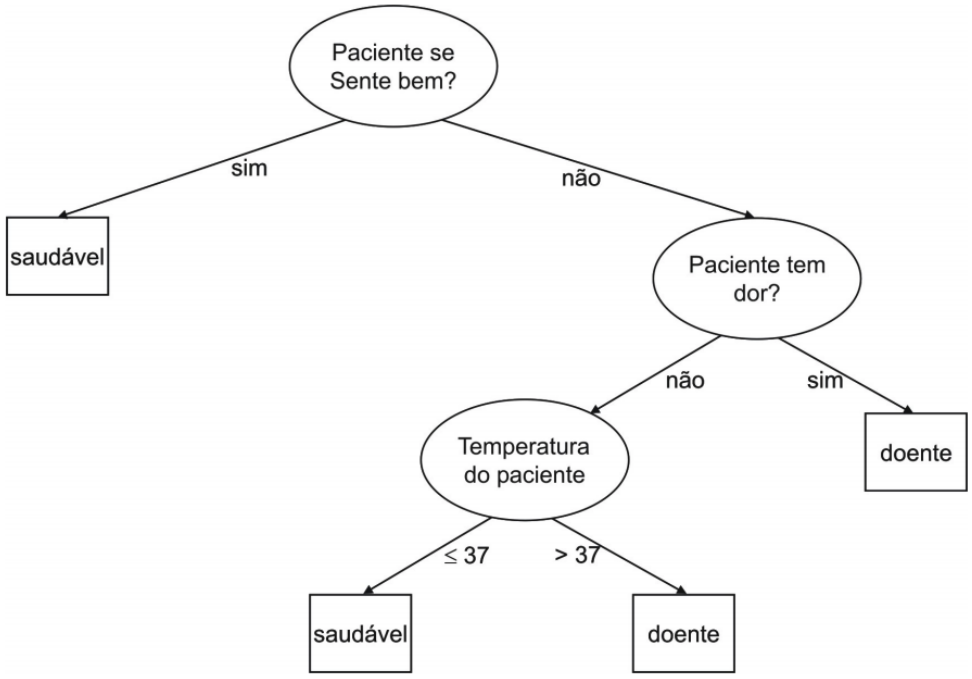

Figura 3.7: Árvore de decisão para diagnóstico de paciente.
Fonte: Monard & Baranauskas (2003)

estes resultados denotados por $\{R_1, R_2, R_3, ..., R_n\}$, o conjunto *A* é particionado em subconjuntos $A_1, A_2, A_3, ..., A_n$ onde cada A_i contém todos os exemplos de *A* que possuem como resultado daquele teste o valor R_i. A árvore de decisão para *A* consiste em um nodo de decisão, ou nodo interno, identificado pelo teste escolhido e uma aresta para cada um dos possíveis resultados;

4. Para cada subconjunto de exemplos de treinamento os passos 1, 2 e 3 são executados recursivamente de forma que, em cada nodo, as arestas levam para as subárvores construídas a partir do subconjunto de exemplos A_i.

No processo descrito acima, em cada iteração procura-se o atributo que melhor divide um conjunto de dados. Ele deve ser capaz separar ao máximo uma base de exemplos em duas ou mais bases, de modo que os dados de cada base possuam uma única classe predominante. A seleção do atributo mais adequado é feita de acordo com critérios de divisão específicos. Pode-se considerar que o sucesso de um algoritmo de *machine learning* por árvore de decisão está diretamente relacionado com o critério utilizado para escolher o melhor atributo para o particionamento do conjunto de dados em cada iteração (Rokach, 2016; Monard & Baranauskas, 2003).

As possibilidades para definir o particionamento são diversas, por exemplo, pode ser selecionado um atributo com base na menor ou a maior quantidade de valores possíveis. Outra maneira é selecionar um atributo que possua um ganho máximo informação, resultando no menor tamanho esperado das subárvores, assumindo que a raiz é o nodo atual. Além disso, ainda pode ser selecionado um atributo de forma aleatória. O índice de Gini e a razão de ganho

são métodos muito utilizados para o particionamento de uma Árvore de Decisão (Monard & Baranauskas, 2003).

O algoritmo básico de indução de árvore de decisão, como o apresentado na Figura 3.7 constrói árvores de decisão recursivamente com base na divisão e conquista com a abordagem de cima para baixo. Os atributos são avaliados e selecionados conforme o critério de divisão. O processo é recursivo, o conjunto de treinamento é subdividido em subconjuntos menores assim que uma divisão apropriada é encontrada. O algoritmo de indução busca o melhor ponto de corte avaliando o critério de divisão em cada ponto de corte possível para satisfazer o critério de parada do nodo. Caso todas as instâncias da partição atual pertençam à mesma classe ou nenhum atributo de divisão futuro pode ser determinado então a árvore de decisão termina o processo de divisão e o nodo é rotulado de acordo com a classe do conjunto de dados, sendo este um nodo folha (Rokach, 2016).

3.3.4 *Extra Tree*

Os classificadores baseados em árvores extremamente randomizadas, ou *Extra Trees (Extremely randomized trees), ET*, foram inicialmente propostas por Geurts, Ernst & Wehenkel (2006), ferramenta importante nas tarefas de classificação. A Extra Tree é um método *ensemble* que constrói uma "floresta" de árvores de decisão não correlacionadas e combina os resultados das mesmas produzindo os resultados finais de classificação.

Esta técnica adiciona aleatoriedade em relação à técnica padrão de árvore de decisão. Ela adiciona forte randomização na tarefa de escolher os atributos e na tarefa de selecionar o ponto de corte de cada atributo. A cada nodo e a cada iteração é definido um *threshold* aleatório para cada atributo, tornando o espaço de busca reduzido e o treinamento mais rápido (Maier, Wilms, von der Gablentz, Krämer, Münte & Handels, 2015).

Sendo assim, em cada nodo intermediário, uma amostra aleatória de K atributos do conjunto da base de eventos é selecionada. Cada árvore de decisão deve selecionar o melhor atributo para dividir os dados com base em alguns critérios matemáticos, geralmente o índice de Gini.

O Índice de Gini *(Gini Index)* mede quão bem uma determinado atributo consegue separar as classes contidas em um nodo. O intervalo de valores possíveis para o Índice de Gini variam entre 0 e 1, onde 0 expressa uma classificação pura, onde todos elementos pertencem a uma classe específica ou apenas existe uma classe nela. O valor 1 indica uma distribuição completamente desigual (Sundhari, 2011). O Índice de Gini é determinado subtraindo de 1 a soma dos quadrados das probabilidades de cada classe. É expresso matematicamente na Equação 3.5 (Breiman, Friedman, Olshen & Stone, 1984). Onde P_i denota a probabilidade de que um elemento seja classificado para uma classe distinta.

$$Gini = 1 - \sum_{i=1}^{c}(P_i)^2 \tag{3.5}$$

No caso extremo, são construídas árvores totalmente aleatórias, cujas estruturas são independentes dos valores de saída da amostra de aprendizagem (Geurts et al., 2006). As previsões da árvore são agregadas para produzir a previsão final, geralmente por maioria de votos em problemas de classificação e média aritmética em problemas de regressão (Geurts et al., 2006). É bastante semelhante em operação à Random Forest (RF) e varia principalmente na forma de construir as ADs dentro da floresta (Kaur & Mittal, 2020). Em ET, a aleatoriedade vai um passo além na forma como as divisões são calculadas. Como em Random Forest, um subconjunto aleatório de atributos candidatos é usado, no entanto, em ET, em vez de procurar os pontos de corte mais discriminantes, os pontos de corte são sorteados aleatoriamente para cada atributo candidato. O melhor atributo dos K previamente selecionados é escolhido como a regra de divisão.

A lógica por trás do método é que o corte explícito e a aleatorização de atributos combinados com a estrutura em forma de *ensemble* de árvores devem conseguir reduzir a variância mais fortemente do que os esquemas de randomização mais fracos usados por outros métodos. O uso de dados de treinamento originais e completos, ao invés de dados de treinamento artificiais, é motivado para minimizar o viés. A complexidade do procedimento do crescimento de árvores, assumindo árvores balanceadas, é da ordem de $N \log_n$, em relação ao tamanho da amostra de aprendizagem, como a maioria dos outros procedimentos do crescimento de árvores (Geurts et al., 2006). Além da precisão, a principal força do algoritmo resultante é a eficiência computacional, pois, dada a simplicidade do procedimento de divisão de nodos, o fator constante pode ser muito menor do que em outros métodos de *ensemble* que otimizam localmente os pontos de corte (Geurts et al., 2006).

O procedimento de divisão ET para recursos numéricos tem três parâmetros importantes. O primeiro parâmetro é K indica o número de atributos selecionados aleatoriamente em cada nodo. O segundo parâmetro é n_{min}, que representa o tamanho mínimo do conjunto de treinamento para dividir um nodo. O terceiro parâmetro é M, que indica o número de árvores na estrutura. Todos os três parâmetros desempenham um papel significativo. Onde K é responsável pela intensidade do procedimento de seleção de recursos, n_{min} a força da média do ruído de saída e M a força da redução da variância da agregação do modelo de *ensemble* (Geurts et al., 2006).

No capitulo seguinte são apresentados diversos trabalhos do estado da arte que tiveram por objetivo propor soluções baseadas em seleção de atributos e em classificadores de *machine learning* para detectar intrusões.

Capítulo 4

Trabalhos Relacionados

4.1 Considerações Iniciais

Neste capítulo são apresentados diversos trabalhos científicos desenvolvidos na área de segurança computacional, tendo como foco aqueles trabalhos que utilizam técnicas de seleção de atributos e inteligência artificial para realizar a classificação de eventos de redes de computadores. O objetivo é encontrar as principais abordagens já exploradas nesse contexto e identificar os problemas e lacunas que ainda precisam ser investigadas.

4.2 Estado da Arte

Com o objetivo de encontrar as principais abordagens existentes no estado da arte relacionadas ao tema de pesquisa deste trabalho, foi realizada uma revisão da literatura. Foram encontrados alguns trabalhos envolvendo métodos de seleção de atributos e classificadores para detecção de intrusão. Estes trabalhos foram analisados e são comparados nesta seção, de modo a construir uma perspectiva do atual panorama do tema. Além disso, para a realização da validação e experimentação do método proposto foram realizados estudos com a finalidade de definir a base de eventos a ser utilizada.

Para construir esta perspectiva foram pesquisados trabalhos em conceituadas base de artigos como a IEEE [1], ACM[2], Elsevier[3] e Springer[4]. Foram considerados trabalhos publicados em língua inglesa e disponíveis *on-line*. A pesquisa foi realizada utilizando uma string formada pela combinação das conjunções "ou" e "e" com as palavras chaves: "*Intrusion Detection*", "*Feature Selection*", "*Internet of Things*". O período de busca foi definido como a partir de 2015, no entanto um trabalho considerado importante e realizado no ano de 2008 também foi considerado no estado da arte.

[1] ieeexplore.ieee.org
[2] dl.acm.org
[3] sciencedirect.com
[4] link.springer.com

Os artigos buscados nas 4 bases de dados foram submetidos a alguns critérios de exclusão e inclusão de modo a obter um conjunto de estudos que realmente pertencem ao estado da arte do tema. O seguinte critério de inclusão foi utilizado:

- Serão incluídos os trabalhos que propõem abordagens de seleção de atributos para métodos de detecção de ataques no contexto de recursos restritos da *fog computing* e IoT;
- Serão incluídos trabalhos que propõem abordagens *ensemble* de seleção de atributos.

Além disso, visando selecionar apenas trabalhos relevantes, foram utilizados os seguintes critérios de exclusão:

- Serão excluídos trabalhos que não propõem abordagens de seleção de atributos;
- Serão excluídos trabalhos publicados como artigos curtos ou pôsteres;
- Serão excluídos trabalhos que apresentam avaliações sem propor o método utilizado;
- Serão excluídos trabalhos duplicados;
- Serão excluídos trabalhos que propõem abordagens de detecção e/ou seleção de atributos porém não realizam a implementação, validação e/ou experimentação com o método proposto.

O primeiro pilar do presente trabalho sustenta-se em métodos de seleção de atributos, desse modo, a seguir são apresentados diversas abordagens aplicadas em pesquisas sobre segurança de computadores. Tais métodos contribuem na melhoria do consumo de recursos e no tempo de processamento, devido à diminuição na dimensionalidade dos dados de entrada. Outro ponto positivo é a possível melhoria da acurácia na detecção da intrusão, já que alguns atributos podem causar ruídos e atrapalhar o processo de classificação. Diversas abordagens e técnicas vem sendo propostas com as finalidades supracitadas. No trabalho de Stiawan, Idris, Bamhdi, Budiarto et al. (2020) é aplicado o método *filter Information Gain* para a seleção de atributos e os algoritmos de classificação são *Random Forest (RF), Random Tree (RT)*, Naive Bayes (NB), rede de Bayes (*Bayes Net*, BN) e J48. A preparação da base de dados consistiu na união de todos os arquivos do formato CSV disponibilizados na CICIDS2017 e então foram selecionados 20% desses dados para o experimento. O alto desbalanceamento de classes foi resolvido através da criação de novas classes a partir da união de algumas classes presentes na base como sugerido por Panigrahi & Borah (2018). O estudo concluiu que o algoritmo de classificação RF teve uma melhor *performance* utilizando-se das sub-bases com 15, 22 e 35 atributos, onde os 22 atributos mais importantes atingiram uma acurácia de 99,86%, enquanto o algoritmo J48 obteve melhore resultado com 52, 57 e 77 atributos, onde os 52 melhores atributos atingiram 99,87% de acurácia, porém com um maior tempo de execução. O ponto de corte, ou *threshold* utilizado para gerar a sub-base foi pelo ranqueamento do algoritmo *Information Gain* e baseando-se nos

valores de seus pesos. Também concluiu-se que a quantidade de atributos selecionados afeta diretamente o tempo de execução.

Saeys, Abeel & Van de Peer (2008) enfatizam a importância da robustez e estabilidade e as consideram como principais características a serem avaliadas em um método de seleção de atributos. Segundo Dietterich (2000b), os métodos *ensemble* performam melhor que modelos utilizados de forma singular, os quais podem apresentar instabilidade. Já a combinação de modelos pode gerar hipóteses diferentes e ao combiná-las o risco de realizar a escolha da hipótese errada é diluído. O algoritmo de aprendizagem e a abordagem da busca podem terminar em ótimos locais diferentes, e quando combinados abrangem uma melhor aproximação. Por fim, o espaço de maior hipótese pode ser criado pela combinação dos métodos utilizados, já que cada um gera sua própria hipótese de saída. Neste trabalho o autor trabalha com bases de eventos da área da saúde, utilizando *ensemble* de métodos do tipo *filter* e embutidos. Os métodos utilizados baseados em *filter* foram o Relief e o *Symmetrical Uncertainty* (SU). Os métodos embutidos utilizaram *Random Forest* (RF) e Linear Support Vector Machine (SVM). Para a RF foram utilizadas dez árvores, já para o SVM foi utilizado a técnica de RFE para realizar o ranqueamento dos atributos. Como classificadores foram utilizadas as técnicas RF e kNN. Os autores concluíram que a utilização de métodos *ensemble* para a seleção de atributos aumentou a robustez dos métodos, principalmente quando utilizados com bases de eventos de grande volume.

Além da seleção de atributos, outro ponto extremamente importante do trabalho é a abordagem de análise e classificação dos eventos. Este, está diretamente relacionado com a seleção de atributos, sendo uma fase anterior da classificação e define quais os atributos que serão processados pelo motor de indução para a construção do modelo.

Muitos equipamentos inseguros podem ser ativados e controlados de forma remota por um atacante malicioso, criando assim, uma rede de bots (do inglês *Botnet*). Para a identificação de ataques lançados por *botnets*, Guerra-Manzanares, Bahsi & Nõmm (2019) em seu trabalho propuseram uma abordagem baseada na combinação de algoritmos do tipo *wrappers* combinados com algoritmos do tipo *filter* para realizar a seleção de atributos e aplicá-los a modelos de aprendizado de máquina. Os algoritmos do tipo *filter* utilizados foram a pontuação Fisher e o coeficiente de correlação Pearson. Já os algoritmos utilizados do tipo *wrapper* foram a seleção de atributos sequencial para frente (do inglês *Sequential Forward Feature Selection, ou SFFS*) e a eliminação de atributos sequencial para trás (do inglês *Sequential Backward Feature Elimination, ou SBFE*). Para a realização da classificação multi-classe foram utilizados os algoritmos de kNN e *Random Forest*. A partir deste estudo é concluído que a aplicação de uma técnica híbrida de seleção de atributos pode ser visto como um *trade-off* entre a simplicidade e velocidade de modelos baseados em *filter* com a demanda computacional e assertividade maior das técnicas *wrapper*. Demonstra-se assim, que ao utilizar-se de métodos *filter* anteriormente a métodos *wrapper* é possível reduzir significativamente o custo computacional exigido pelas técnicas *wrapper*. Além disso, sem perdas significantes nas taxas de detecção dos classificadores baseados em aprendizado de máquina. A base de eventos utilizada foi construída a partir de

tráfego gerado em laboratório e com os ataques BASHLITE e Mirai. A melhor acurácia atingida foi de 99,90% com testes utilizando *cross-classifier* e combinando os algoritmos de Fischer's com qualquer método *wrapper* utilizado neste trabalho quando classificado com o algoritmo de RF.

Com o objetivo de melhorar a defesa contra ataques DDoS, Osanaiye et al. (2016) elaboraram um seletor de atributos *ensemble*. Identificando os atributos mais importantes para os algoritmos de aprendizado de máquina é possível gerar melhores modelos e consequentemente melhorar a acurácia de classificação e reduzir a complexidade computacional. Eles propõem uma abordagem *ensemble* com múltiplos algoritmos de seleção de atributos do tipo *filter*. As saídas desses algoritmos são combinadas para alcançar uma seleção ótima. A base de eventos utilizada foi a NSL-KDD e o classificador foi a árvore de decisão J48. Os algoritmos de seleção de atributos utilizados foram: *Info Gain*, *Gain Ratio*, *Chi-squared*, *ReliefF* e para combiná-los foi utilizado o método de voto majoritário simples. O *threshold* foi definido como a ocorrência frequente do atributo entre os quatro métodos. Dos 41 atributos presentes na base, 13 atributos foram selecionados e apresentaram melhora na acurácia da classificação e taxa de detecção, além de um tempo para a construção do modelo de classificação extremamente reduzido.

Idhammad, Afdel & Belouch (2018) apresentaram uma abordagem de aprendizado de máquina para a detecção de ataques DDoS em tempo real de forma semi-supervisionada e sequencial. São cinco as principais etapas da abordagem proposta, são estas: o pré-processamento de dados, a estimativa da entropia do tráfego da rede, a realização da clusterização por blocos dos dados em tempo real, a computação do ganho de informação e por fim, a classificação do tráfego da rede. As etapas não supervisionadas são responsáveis por reduzir a dimensão dos dados, removendo dados irrelevantes do tráfico normal de dados para a identificação de ataques DDoS, reduzindo assim a taxa de falsos positivos e aumentando a acurácia. Com a finalidade de computar a taxa do ganho de informação foi utilizada o algoritmo de *Information Gain*, e para a classificação do tráfego da rede analisado foram utilizados *Ensemble* de *Extra-Tree*. Três bases de eventos públicas foram utilizadas para a execução dos experimentos, são essas a NSL-KDD, UNB ISCX 12 e UNSW-NB15 que obtiveram acurácias de 98,23%, 99,88% e 93,71%, respectivamente. Além disso, os índices de falsos positivos foram os seguintes: 0,33%, 0,35% e 0,46%, respectivamente. Os resultados indicam taxas de detecção satisfatórias em relação a trabalhos que buscam a detecção de DDoS nas bases supracitadas.

Em 2016 Indre & Lemnaru (2016) propuseram um sistema capaz de analisar, detectar e prevenir conexões maliciosas na rede aplicando conceitos de aprendizagem de máquina com uma grande ênfase na importância da seleção e extração de atributos que possam levar a uma detecção com maior acurácia. A solução proposta é a combinação de algoritmos de seleção de atributos que buscam a correlação entre atributos do histórico de pacotes de dados de iguais ou diferentes serviços e *hosts*, baseando-se no trabalho de (Olusola, Oladele & Abosede, 2010). A arquitetura possui três principais módulos em paralelo. O primeiro módulo utiliza regras estatísticas para filtrar os dados. O segundo utiliza uma classificação binária dos eventos e o

último a detecção por *malware*. A seleção de atributos foi realizada de forma analítica pelos pesquisadores, gerando amostras da base de evento principal e submetendo aos diversos classificadores. Posteriormente os ataques identificados são direcionados para um módulo de classificação multi-classe. A base de eventos utilizada foi a KDD 99 e para a classificação dos eventos os algoritmos kNN, Naive Bayes e árvores de decisão foram utilizados. A solução proposta atingiu uma precisão de 98,4& para a classificação binária com o classificador kNN e 89.5% de precisão utilizando a árvore de decisão 83.7% Naïve Bayes para a classificação multi classe com a utilização da base de eventos DARPA.

O trabalho dos autores Yulianto, Sukarno & Suwastika (2019) busca comparar a acurácia, precisão, o *recall* e o *F1 Score* entre a classificação de eventos utilizando como classificador o algoritmo AdaBoost. É utilizado como linha de base o trabalho de Sharafaldin, Lashkari & Ghorbani (2018), comparando com a abordagem proposta, utilizando variações na etapa de pré-processamento. A abordagem utiliza o algoritmo SMOTE para realizar *Oversampling*, com a finalidade de aumentar a quantidade de eventos das classes minoritárias, melhorando o balanceamento entre as classes. A outra etapa do pré-processamento é a seleção de atributos. A abordagem utilizou o algoritmo de Análise de Componente principal (do inglês *Principal Component Analysis*, ou PCA). Além disso, também foi avaliada a Seleção de Atributos por Ensemble (do inglês *Ensemble Feature Selection*, ou EFS), um pacote presente no R Studio. Após os experimentos a melhor configuração encontrada foi com a utilização do seletor de atributos EFS juntamente com o balanceamento por SMOTE. Essa configuração atingiu uma acurácia de 81,83% com a base de eventos CICIDS2017 focando somente nos ataques DDoS, portanto, sendo uma classificação binária, não sendo capaz de realizar a identificação das classes de ataques.

No ano de 2019, Faker & Dogdu (2019) propuseram uma abordagem integrada de *Big Data* e *Deep Learning* para melhorar a *performance* de SDI. Foram usados três classificadores, o *Deep Feed-Forward Neural Network* (DNN), *Random Forest* (RF) e o *Gradient Boosting Tree* (GBT). A proposta possui três grandes fases: o pré-processamento de dados, o ranqueamento de atributos e seleção, e por último, a criação e avaliação dos modelos de aprendizagem. A fase de pré-processamento foi realizada com a remoção e normalização de dados e com o tratamento e substituição de dados faltantes. Para o ranqueamento de atributos, o algoritmo de clusterização K*-means* é utilizado com o K igual ao número de classes a ser classificadas. As bases utilizadas foram a UNSW-NB15 e a CICDIDS2017. Os classificadores foram aplicados para a classificação multi classe e para classificação binária, o algoritmo DNN obteve uma acurácia de 97,04% de detecção na base UNSW-NB15 e 99,57% na detecção multi classe utilizando a base CICDIDS2017. No entanto, mesmo se tratando de uma classificação multi classe não foram apresentados os resultados individuais de cada classe. Sendo assim, classes com poucos exemplos como a classe *Infiltration* pode não ter sido detectada e mesmo assim a abordagem pode atingir uma acurácia total de 99%. A falta destas métricas para cada classe e de forma balanceada não permite tirar as conclusões necessárias deste trabalho.

A seguir, na Tabela 4.1, é apresentada uma comparação sobre as abordagens dos trabalhos do estado da arte e a abordagem proposta neste trabalho. São apresentadas as principais características e desvantagens encontradas. A seguir são apresentadas abreviações utilizadas na tabela.

- **MSA**: **M**étodo de **S**eleção de **A**tributos;
- **MC**: **M**étodo de **C**lassificação;
- **DM**: Abordagem realiza **D**etecção **M**ulticlasse;
- **ESA**: O trabalho é baseado em **E**nsemble **S**eleção de **A**tributos;

Conforme observado na Tabela 4.1 diversos métodos de seleção de atributos são utilizados em diversas abordagens, com diferentes arquiteturas e classificadores. A utilização de métodos *ensembles* vem sendo proposta devido à possibilidade de combinar as melhores características de cada algoritmo. Da mesma maneira, métodos híbridos vem sendo criados de modo a operar de forma sequencial. Onde o método inicial é capaz de realizar uma seleção prévia e entregar ao próximo método, no processo de seleção, um resultado otimizado e melhorado.

No entanto, são escassos na literatura, trabalhos a respeito de abordagens híbridas e *ensemble* para seleção de atributos. Nota-se também a falta de métodos *wrapper* organizados de forma *ensemble* possuindo o motor de indução como o mesmo método de classificação. Além disso, são necessárias alternativas para melhorar os métodos *wrapper*, pois eles possuem um alto custo computacional. Destaca-se ainda a lacuna de abordagens focadas em seleção de atributos com classificadores multiclasse. Esta categoria de classificação é extremamente importante para execução de contramedidas específicas e para a tomada de decisão do responsável pela rede. Além disso, há uma gama de trabalhos que utilizaram bases defasadas ou com baixa aplicação em trabalhos científicos e de domínio não público. É necessário a utilização de uma base de eventos com uma melhor representatividade de fluxos legítimos e intrusivos atuais para a validação de métodos de seleção de atributos.

A seguir são discutidos alguns detalhes a respeito dos principais conjuntos de dados existentes para a validação e avaliação de métodos de detecção de intrusão.

4.3 Conjuntos de dados para validação de métodos de detecção

A base de eventos KDD CUP 99 é uma base de eventos pública e é amplamente utilizada em trabalhos científicos para avaliar sistemas de detecção de intrusões baseado em anomalias. No entanto, estudos observaram que esta base possui alguns problemas, como o grande número de registros redundantes. Tal fato pode impactar diretamente o aprendizado dos algoritmos, criando uma tendência de classificação para uma maior frequência a classe majoritária e atributos

Tabela 4.1: Comparação entre os trabalhos do estado da arte.

Trabalhos	**MSA**	**MC**	**DM**	**ESA**	**Desvantagens**
Stiawan et al. (2020)	IG	RF, RT, NB, BN, J48	x	-	Apenas um método de seleção de atributos
Saeys et al. (2008)	Relief, SM, SVM, RF	RF, kNN	-	x	Não está relacionado com o contexto de detecção de intrusões.
Guerra-Manzanares et al. (2019)	Fs,ρ, SFFS, SBFE	kNN, RF	-	x	Não utiliza uma base de eventos relevante
Osanaiye et al. (2016)	IG+GR+Cs+RF	J48	-	x	Apenas uma classe de método de seleção de atributos
Idhammad et al. (2018)	IG	ET	-	-	Apenas um método de seleção de atributos
Indre & Lemnaru (2016)	Analítica	DT, kNN, NB	-	-	Método de seleção de atributos não evidenciado
Yulianto et al. (2019)	PCA, EFS	AdaBoost	-	-	Foca apenas em DDoS
Faker & Dogdu (2019)	*K-means*	DNN, RF, GBT	x	-	Não há profundidade na seleção de atributos
Este trabalho	**IG+(SFFS+SBFE)**	ET	**x**	**x**	

que possam conter informações que levam a falsas correlações. Sendo assim, a base de eventos KDD CUP 99 não se apresenta como uma base de eventos propícia a ser utilizada no presente trabalho (Tavallaee, Bagheri, Lu & Ghorbani, 2009; Staudemeyer & Omlin, 2014).

Visando sanar os problemas supracitados, Tavallaee et al. (2009) propôs uma nova base, a NSL-KDD. Esta base é amplamente utilizada e contém registros selecionados da KDD CUP 99, entretanto, a ausência de ataques e tráfegos que representem características atuais no tráfego de rede ainda não foram contemplados nesta nova versão (Staudemeyer & Omlin, 2014).

Em Azwar, Murtaz, Siddique & Rehman (2018) é apresentado uma análise das bases de eventos mais utilizadas no meio acadêmico, dentre elas a DARPA98, KDD99, ISC2012, e ADFA13. Estas bases de eventos foram levantadas e posteriormente descartadas pelos autores. Os critérios considerados para a avaliação foram: configuração completa, captura completa das atividades e do tráfego completo de uma rede, rotulação, anonimidade dos eventos, diversidade dos ataques, heterogeneidade de dispositivos, entre outros. Dentre as bases consideradas, diversas foram classificadas como obsoletas e com limitações que impossibilitariam uma melhor avaliação dos modelos construídos. A CICIDS2017 (Sharafaldin et al., 2018), porém, foi considerada relevante, pois contém diversas características das listadas acima.

Diante dessa exposição e devido aos ataques presentes, a base de eventos CICIDS2017 se apresenta como um conjunto de dados mais relevante. Ela apresenta eventos benignos e malignos sido construída com o objetivo de ser uma base de eventos mais atualizada e representativa para a validação de novos SDI, baseando-se nos ataques mais comuns presentes no relatório de 2016 da McAfee (Sharafaldin et al., 2018; Loganathan, 2018).

4.4 Considerações Finais

Neste capítulo foram analisados diversos trabalhos que apresentam abordagens relacionadas a seleção de atributos e a detecção de intrusão em redes de computadores. Através dessa revisão foi obtida uma compreensão sobre o estado da arte em métodos *ensemble* de seleção de atributos e métodos de classificação de eventos. Além disso, algumas lacunas foram observadas. A partir disso, propõe-se o método presente neste trabalho, uma abordagem híbrida, composta por métodos *ensemble* de seleção de atributos e por técnica de detecção multi classe. No próximo capítulo é apresentada a abordagem proposta neste trabalho, contemplando a arquitetura do modelo e os métodos empregados.

Capítulo 5

Abordagem Proposta

A detecção de atividades intrusivas é de extrema importância para prover um ambiente seguro, conforme pode ser observado no estado da arte. Além disso, observa-se que as instâncias de tráfego de rede vem progressivamente apresentando uma maior quantidades de atributos, resultando em um grande volume de dados a cada evento. Pesquisas no ambiente de Internet das Coisas (*Internet of Things* - IoT) e de computação em *fog* (*Fog Computing*) que propõem a redução de dimensionalidade, sem lesar a classificação, ainda são poucas. É possível destacar também a carência de abordagens focadas na detecção multi-classe, visando identificar a categoria de intrusão. À vista disso, neste trabalho é proposta uma abordagem *ensemble* de seleção de atributos em conjunto com um classificador multi-classe de ataques em ambientes de computação em *fog* e IoT.

Uma etapa de seleção de atributos ocorre anteriormente ao processo de classificação dos eventos. Através deste processo é possível aumentar a *performance* do classificador, diminuindo a quantidade de atributos sem correlação com a classe do evento (Garg, Kaur, Batra, Aujla, Morgan, Kumar, Zomaya & Ranjan, 2020). Na fase de classificação o evento com atributos selecionados será classificado pelo método *Extra Tree* (ET) de modo a categorizar o evento em uma determinada classe de intrusão.

5.1 Contextualização da abordagem proposta

Os métodos de seleção de atributos são amplamente utilizados em diversas aplicações relacionadas ao aprendizado de máquina, como, por exemplo, o reconhecimento de padrões e análises de grandes conjuntos de dados. Devido à escassez de recursos e em prol da otimização do tempo esta técnica é uma grande aliada, tratando e filtrando dados impuros e buscando diminuir a dimensionalidade das bases de dados de grandes proporções (Hoque et al., 2018).

A camada de computação em *fog* opera como uma camada intermediária entre os dispositivos IoT e a computação em *cloud*, conforme pode ser observado na Figura 5.1. Desse modo, pode ser um local ideal para a implantação da abordagem de detecção, monitorando todo o tráfego da rede de dispositivos IoT, e fornecendo uma resposta rápida graças a sua proximidade

em relação aos dispositivos IoT. Além disso, a implantação da abordagem nesta camada se deve a possibilidade de colaboração entre os nodos de detecção de intrusão da *fog*, criando um ambiente de informações compartilhadas sobre os atributos mais relevantes entre estes. Considera-se que o dispositivo da *fog* possui os mecanismos necessários para a captura do tráfego. Além disso, o dispositivo de *fog* contara com módulos de pré-processamento e também de seleção dos recursos. Por fim, as características principais de cada fluxo de dados são submetidas ao classificador proposto para a identificação da categoria da intrusão.

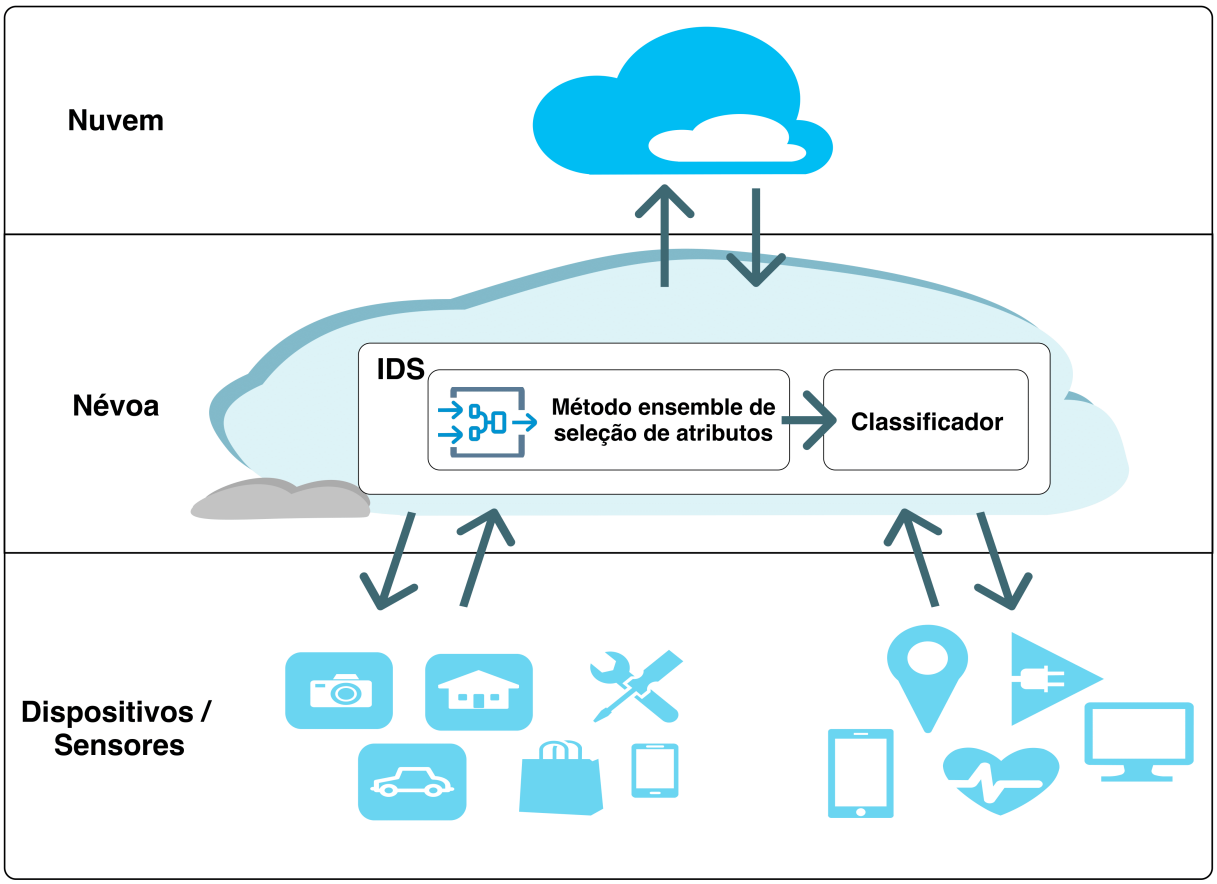

Figura 5.1: Localização do método proposto em uma rede IoT.
Fonte: O autor (2021).

Antes da abordagem de detecção iniciar a sua operação é necessário realizar um processo prévio, onde são definidos os atributos que serão filtrados do tráfego capturado. Estes atributos são também filtrados do conjunto de dados de treinamento sendo utilizados para o treinamento da abordagem de classificação. Portanto, o método de seleção de atributos é aplicado antes de a abordagem de detecção ser colocada para execução. Após o início da operação da abordagem, a captura de eventos ocorre de forma completa, desse modo, é necessário que dentre todas as informações existentes sejam filtrados somente aqueles atributos que foram definidos pelo algoritmo de seleção de atributos. As informações filtradas são então encaminhadas para o algoritmo de classificação, que com base nestas informações irá classificar o fluxo de dados em uma categoria de intrusão ou em comportamento normal.

5.2 Descrição da abordagem

O principal pilar deste trabalho consiste na definição de uma abordagem de detecção multi-classe para identificação de intrusões em ambientes de computação em *fog* e IoT. Desse modo, é proposto um método *ensemble* de seleção de atributos combinando técnicas *filter* e *wrapper*. Além disso, é proposta uma abordagem de detecção multi-classe com o classificador *Extra Tree*. A Figura 5.2 apresenta uma ilustração completa da abordagem proposta. Os métodos para compor esta arquitetura foram escolhidos baseados em estudos realizados e apresentados na Seção 3.2.

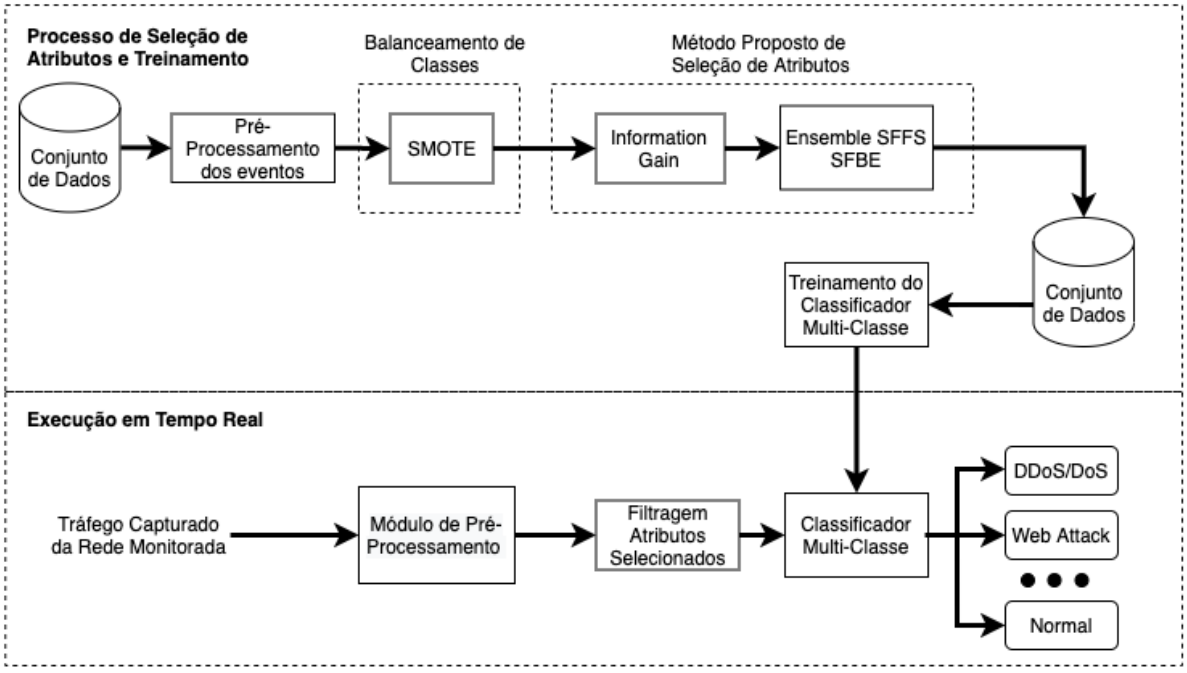

Figura 5.2: Etapas da abordagem proposta.
Fonte: O autor (2021).

A Figura 5.2 ilustra divide em dois blocos principais de execução da abordagem. O primeiro bloco é referente ao processo prévio de seleção de atributos e treinamento da abordagem de detecção. Primeiramente o conjunto de dados escolhido para treinamento é pré-processado. Essa etapa envolve a padronização dos dados, remoção de valores nulos e remoção de valores NaN, permitindo assim um formato compreensível pelo classificador. Em seguida, é realizado o balanceamento de classes utilizando o algoritmo SMOTE para realizar um *oversampling* nas classes minoritárias, de modo que todas as classes possuam pelo menos 20% da quantidade de dados da classe majoritária.

A técnica utilizada para realizar o balanceamento de classes SMOTE (do inglês, *Synthetic Minority Oversampling Technique*) consiste em sintetizar novos exemplos para as classes que possuem poucos exemplos. Para não gerar uma grande quantidade de novos dados, a abordagem

padrão de geração de exemplos que busca igualar o número de exemplos das classes, não foi utilizado. Diante disto, a estratégia de balanceamento na criação de dados foi para que as classes tenham ao menos 20% da quantidade de exemplos da classe majoritária (Bowyer, Chawla, Hall & Kegelmeyer, 2011).

Após a realização do balanceamento de classes é realizado a execução do método *ensemble* proposto para a seleção de atributos, que será explorado na Seção 5.2.1. De modo resumido, pode se observar que em uma primeira etapa é realizada a seleção de atributos utilizando um algoritmo do tipo *filter*, o *Information Gain*. Na segunda etapa é realizada a combinação entre dois algoritmos do tipo *wrapper*, o SFFS e o SBFE. O subconjunto de eventos resultante da seleção de atributos é então utilizado para realizar o treinamento e construção da *Extra Tree*, que será utilizada na classificação do tráfego.

O outro bloco apresentado na Figura 5.2 representa a execução em tempo real, onde é realizado a captura do tráfego na rede monitorada. O tráfego capturado passa por um pré-processamento e as principais informações são selecionadas através de uma filtragem com bases nos atributos definidos como mais relevantes pelo método de seleção de atributos, aplicado durante a fase de treinamento. Essas informações são então submetidas ao classificador *Extra Tree* treinado. Esta árvore foi gerada durante a etapa de treinamento, aplicada no tráfego para então classificar o evento em um tipo de ataque malicioso ou em um comportamento benigno.

Na Subseção 5.2.1 serão apresentados maiores detalhes sobre o método *ensemble* de seleção de atributos, já na Subseção 5.2.2 são expostas informações sobre o classificador utilizado.

5.2.1 Seleção de atributos

A abordagem de seleção de atributos é apresentada na Figura 5.3. Durante o processo de seleção de atributos os dados de treinamento são submetidos para a abordagem. Os algoritmos de seleção são empregados sobre os dados e geram como saída um conjunto dos melhores atributos de acordo com os dados de treinamento.

O método proposto é uma abordagem *ensemble* de seleção de atributos e possui duas etapas principais. Na Etapa 1 é aplicado um algoritmo de seleção de atributos da classe *Filter*, o *Information Gain*. Este algoritmo tem como resultado um *ranking* dos atributos mais relevantes encontrados de acordo com a característica de cada atributo em relação a classe. A Etapa 2 é composta por um *ensemble* de algoritmos *wrapper*. Por fim, o conjunto de melhores atributos gerado pela segunda etapa é tido como resultado da abordagem. A evolução do processo de seleção de atributos e dos atributos selecionados será abordado no Capítulo 6.

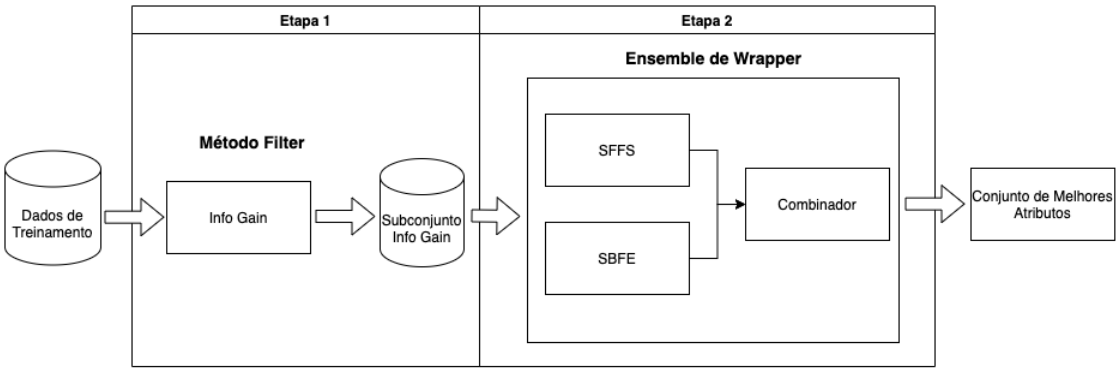

Figura 5.3: Arquitetura proposta.
Fonte: O autor (2021).

Os dados de treinamento são submetidos inicialmente para a etapa 1. O conjunto de atributos selecionado pela primeira etapa é então submetido para a etapa 2, conforme apresentado na Figura 5.3. A utilização de um método do tipo *filter* na Etapa 1 possui grande importância, pois, possui independência de um algoritmo de classificação. Diferentemente, os métodos *wrappers* possuem um *bias* em relação ao classificador devido ao motor indutivo utilizado neste. Outra vantagem da utilização de algoritmos *filter* anteriormente a uma abordagem *wrapper* é a redução de atributos a serem processados pela segunda abordagem. Isso agiliza o processo, pois, abordagens *wrapper* demandam maior processamento e geram maiores custos computacionais, fato este que pode ser proibitivo ao se lidar com grandes quantidades de dados. No entanto, as abordagens *wrapper* tendem a obter conjuntos de atributos com maior qualidade para a melhora do desempenho de detecção. Desse modo, propõe-se a utilização combinada de métodos *filters* com *wrappers* para alcançar melhoras na acurácia da detecção e diminuir o custo computacional(Das, 2001; Sharbaf, Mosafer & Moattar, 2016). A seguir são apresentados maiores detalhes a respeito de cada umas das etapas da abordagem de seleção de atributos.

- **Etapa 1:** Nesta etapa é aplicado inicialmente o algoritmo *Information Gain* (IG). Este algoritmo é da família *filter* de métodos de seleção de atributos, conforme apresentado na Seção 3.2.2. Na Etapa 1 do método proposto busca-se reduzir a dimensionalidade e reduzir a complexidade em um primeiro contato com o conjunto de dados, para esta função a utilização de métodos *filter* é de grande valia. Isto se deve a característica de se basearem em testes estatísticos intrínsecos sobre o conjunto de dados e por fim utilizando um esquema de classificação de atributos para selecionar os atributos por ordem (Osanaiye et al., 2016; Sharbaf et al., 2016).

  A utilização de métodos *filter* para a seleção de atributos provê uma abordagem computacionalmente mais leve em relação às abordagens *wrapper*. Diversos métodos do tipo *filter* são utilizados na literatura, e entre estes o IG é amplamente utilizado entre os pesquisadores para analisar quais os atributos mais relevantes e significantes (Stiawan et al., 2020). Segundo Osanaiye et al. (2016) a etapa de seleção de atributos é identificada como uma fase de pré-processamento de grande importância para a defesa contra ataques do tipo

DDoS em nuvens. Em seu trabalho é utilizado um *ensemble* de quatro algoritmos do tipo *filter*, buscando aumentar a acurácia da classificação e reduzindo a complexidade computacional ao identificar os atributos mais importantes da base de eventos original durante o aprendizado supervisionado.

O IG é um algoritmo baseado na teoria de informação, ele gera como saída um ranqueamento de modo a evidenciar os melhores atributos que mais geram ganho de informação. Conforme apresentado no Capítulo 3.2.2, a entropia indica o nível de impureza, desse modo, o ganho de informação de um atributo indica o quanto de impureza aquele atributo é capaz reduzir. A partir disso, com base nos valores de ganho de informação os atributos são ranqueados (Tesfahun & Bhaskari, 2013; Azhagusundari, Thanamani et al., 2013). Os atributos são descartados caso pontuarem abaixo de um *threshold* previamente estipulado.

Conforme supracitado, o algoritmo IG trabalha com um ponto de corte, ou *threshold*, que precisa ser estabelecido para descartar os atributos menos relevantes. Com o propósito de definir o ponto de corte, diversas abordagens foram avaliadas na prática para o algoritmo supracitado. Segundo Bolón-Canedo et al. (2015), o ponto de corte deve ser considerado para cada escopo. Belanche & González (2011) optou por descartar os atributos cujo peso variavam da média mais que duas vezes a variância. Bolón-Canedo et al. (2015) fixou uma porcentagem de atributos, de acordo com a quantidade total de atributos presentes na base de eventos, por exemplo, descartar 25% dos atributos quando $N < 10$. A razão discriminante de Fisher e o $log_2(N)$ foram propostos por Langley & Iba (1993) e Li, Zhang & Ogihara (2004) consecutivamente. A abordagem considerada neste trabalho foi a apresentada por Mejía-Lavalle, Sucar & Arroyo (2006). Esta abordagem é definida pela maior diferença entre dois atributos consecutivos no ranking, ou seja, busca-se descartar todos os atributos abaixo da maior redução de pontuação.

Por fim, o IG gera um subconjunto dos melhores atributos do conjunto de dados, ou seja, atributos que obtiveram pontuação superior ao *threshold*. Então esse subconjunto é submetido para a Etapa 2 da abordagem.

- **Etapa 2:** A Etapa 1 gera um subconjunto que é posteriormente submetido para Etapa 2 (como demonstrado na Figura 5.3). Na segunda etapa, é empregado uma abordagem ensemble com a combinação de dois métodos de seleção de atributos. Estes métodos, aplicados nesta segunda etapa, pertencem à classe *Wrapper*, e cada um deles gera um subconjunto de atributos. A utilização combinada de métodos *wrappers* com métodos *filter* potencializa as chances de ser encontrado melhor subconjunto, visto que os algoritmos de menor contribuição na avaliação foram removidos (Huang, Cai & Xu, 2007). Os métodos *wrappers* que formam a Etapa 2 são:

  - *Sequential Forward Feature Selection* (SFFS)[1]: algoritmo *wrapper* baseado na estratégia de busca *forward*.

[1] https://scikit-learn.org/stable/search.html?q=SequentialFeatureSelector

– *Sequential Backward Feature Elimination* (SBFE)[2]: algoritmo *wrapper* baseado em eliminação de atributos com estratégia de busca *backward*.

Os métodos *wrappers* de seleção de atributos geralmente são capazes de obter melhores conjuntos de atributos do que os métodos *filter*, por buscarem a melhor combinação de atributos em determinado conjunto com a utilização de um motor de inferência para avaliar o subconjunto de cada iteração (Guerra-Manzanares et al., 2019). Os algoritmos *wrapper* utilizam em sua maioria estratégias de buscas gulosas, inserindo ou removendo os atributos no subconjunto e avaliam este em relação ao conjunto pai. O algoritmo SFFS utiliza a estratégia de busca *Forward Selection* e o algoritmo SBFE utiliza a estratégia de busca *Backward Selection*, ambas apresentadas na Seção 3.2.3.

Para realizar a avaliação dos subconjuntos dentro de cada um dos métodos *wrappers*, foi usado o classificador Extra Tree com a mesma configuração utilizada na etapa de classificação. A utilização do algoritmo Extra Tree como motor de indução de ambos os métodos *wrapper* é baseada na abordagem de Guerra-Manzanares et al. (2019), onde são utilizados os classificadores k-NN e RF.

Cada um dos métodos *wrapper* gera um subconjunto com os melhores atributos escolhidos de acordo com seus respectivos processos. No entanto, a estratégia tradicional, aplicada a estes métodos, que busca encontrar o melhor conjunto de atributos de maneira automatizada tem um custo extremamente elevado, pois precisa realizar N iterações considerando em cada iteração encontrar os melhores K atributos, sendo N a quantidade de atributos existentes no conjunto de dados e K a quantidade de atributos selecionados. Em cada iteração é considerado um K diferente, de modo, que todas as quantidades possíveis de atributos sejam avaliadas com os seus melhores atributos. Após todo esse processo é possível obter os melhores atributos na quantidade ideal em relação a todas as combinações possíveis. Devido ao alto custo dessa abordagem tradicional, propomos uma nova estratégia que realiza apenas uma iteração e que seleciona uma quantidade K de atributos predefinida, portanto, uma estratégia extremamente mais leve. Esta quantidade foi definida como $1/3$ da quantidade de atributos existentes. De modo, a contornar o risco de selecionar um conjunto de atributos que não corresponde a quantidade ótima, buscou-se a utilização de duas estratégias *wrapper* distintas, cujos subconjuntos gerados são então combinados para obter um subconjunto de atributos otimizados. Para se combinar esses conjuntos de maneira efetiva, é aplicado um algoritmo combinador que busca uma relação entre os atributos de cada subconjuntos produzidos pelos métodos *wrapper*, para gerar um conjunto final de melhores e mais relevantes atributos.

Dentre as várias formas possíveis de combinar os subconjuntos de atributos gerados pelos métodos *wrappers*, inicialmente neste trabalho propõe-se avaliar a união e a intersecção entre estes subconjuntos, pois são técnicas consideradas interessantes por Bolón-Canedo & Alonso-Betanzos (2019). A união entre os conjuntos consistem em combinar todos

[2]https://scikit-learn.org/stable/search.html?q=SequentialFeatureSelector

os atributos selecionados pelo menos uma vez por algum dos métodos de seleção de atributos. Esta abordagem tende a gerar uma menor redução no número de atributos. A combinação por intersecção consiste em selecionar apenas os atributos selecionados por todos os métodos de seleção de atributos. No entanto, esta abordagem pode levar a um conjunto de atributos muito restritivo, produzindo maus resultados (Álvarez-Estévez, Sánchez-Maroño, Alonso-Betanzos & Moret-Bonillo, 2011). Outras abordagens utilizam a avaliação por classificação, porem o alto custo computacional dessa abordagem a inviabiliza no contexto do presente trabalho.

5.2.2 Classificador

Nessa Seção são apresentados detalhes a respeito da etapa de classificação da abordagem proposta ilustrada na Figura 5.2. Após todo o pré-processamento dos dados e filtro dos atributos selecionados pela abordagem de seleção de atributos, as informações do tráfego são submetidas para um classificador que irá identificar se o tráfego corresponde a alguma classe de intrusão ou corresponde a um comportamento benigno. O método que realiza esta classificação é o *Extra Tree*, uma floresta aleatória de árvores de decisão. Os principais conceitos envolvidos neste método são apresentados na Seção 3.3.4.

Conforme apresentado anteriormente, a *Extra Tree* possui três parâmetros principais. Os parâmetros utilizados neste trabalho são: 10 estimadores $(M = 10)$, tamanho mínimo da amostra para divisão é 2 $(n_{min} = 2)$, número de atributos considerados para melhor divisão é a raiz do número de atributos existentes $(K = \sqrt{N})$, e o Índice de Gini (Sundhari, 2011) é usado como critério ($criterion = \text{“}gini''$).

Capítulo 6

Experimentos

6.1 Considerações Iniciais

Neste capítulo serão apresentados detalhes a respeito dos experimentos realizados para avaliar a abordagem proposta, tal como as técnicas, ferramentas e métodos utilizados. Primeiramente o conjunto de dados utilizado na avaliação é descrito.

6.2 Base de Eventos

As principais características do conjunto de dados CICIDS2017 são apresentadas nesta seção com base no trabalho de Sharafaldin et al. (2018). Esta base foi construída a partir da captura de tráfego de uma rede controlada, criada pelo *Canadian Institute for Cybersecurity* (CIC). Durante cinco dias foram simuladas atividades benignas, malignas e diversos ataques. A CICIDS2017 é disponibilizada em dois formatos, no formato ".pcap" (*packet capture*) onde se encontra os eventos em estado bruto, como pacote, e no formato ".csv", com os atributos no formato atributo-valor e já rotulados.

Os eventos brutos em formato ".pcap" foram processados após toda a captura dos dados do período de tráfego ser realizada, com a finalidade de gerar uma base atributo-valor. A ferramenta CICFlowMeter foi utilizada para a extração e criação de atributos. A rotulação dos eventos foi realizada baseado no cronograma diário dos ataques realizados. Na Tabela A.1, localizada no Apêndice A, são apresentados e descritos os 84 atributos que compõem a base de eventos CICIDS2017.

Os ataques são agrupados conforme a Tabela 6.1. Este agrupamento foi utilizado nos experimentos realizados neste trabalho, utilizando-se como base os trabalhos de Sharafaldin et al. (2018); Bakhareva, Shukhman, Matveev, Polezhaev, Ushakov & Legashev (2019); Vinayakumar, Alazab, Soman, Poornachandran, Al-Nemrat & Venkatraman (2019).

Tabela 6.1: Classes de ataques computacionais gerado a partir da CICIDS 2017.

Classe	Ataques	Quantidade de Eventos
Benign	Benign Traffic	454261
DoS/DDoS	DDoS, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS Slowloris	75950
PortScan	PortScan	31761
Brute Force	SSH-Patator, FTP-Patator	2766
Web Attack	Web Attack Brute Force, Web Attack Sql Injection, Web Attack XSS	436
Botnet	Botnet	391
Infiltration	Infiltration	7

6.3 Amostragem

O método de amostragem *10 K-Fold Stratified Cross Validation* foi escolhido e aplicado na realização dos experimentos, considerando o valor de $k = 10$. Desta maneira, o conjunto de dados é subdividido em 10 *folds*, de forma aleatória. Estes *folds* possuem o mesmo tamanho e a mesma proporção de instâncias por classes, dado que foi utilizada uma estratégia estratificada da técnica de *Cross Validation* (Kohavi et al., 1995). A Figura 6.1 apresenta uma ilustração da estrutura de *folds* e iterações relativas ao funcionamento do *Cross Validation*.

Figura 6.1: Representação do processo de *cross-validation* com 10 folds.
Fonte: O autor (2021).

A cada iteração o algoritmo classificador utiliza nove conjuntos para treinar o modelo e um conjunto para testar. Este processo segue por 10 iterações, alternando o subconjunto de teste, como observado na Figura 6.1. O resultado consiste em uma média dos resultados das execuções dos 10 *folds*. O objetivo deste método é obter uma estimativa do quão preciso a

abordagem é na prática.

6.4 Métricas de avaliação

Nessa seção são apresentadas as métricas utilizadas para avaliar os resultados obtidos através dos experimentos. Estes resultados são inicialmente categorizados em duas categorias de erros de classificação (Goodrich & Tamassia, 2013).

- **Falso Positivo (FP):** Eventos classificados como **intrusivos** pelo método de detecção de intrusão e que são **normais**;
- **Falso Negativo (FN):** Eventos classificados como **normais** pelo método de detecção de intrusão e que são **intrusões**;

Dentre ambos os casos, o que possui maior gravidade é o falso negativo, pois se trata de uma atividade maliciosa não detectada. Esse tipo de erro pode resultar em sérias consequências para o sistema na totalidade e acarretar problemas graves de segurança. No que lhe concerne os falsos positivos desperdiçam tempo e recursos importantes do SDI em uma atividade que não é uma ameaça em potencial, mesmo sendo classificada como tal.

As classificações genuínas do SDI são classificadas em:

- **Verdadeiro Negativo (VN):** Eventos classificados corretamente pelo método de detecção de intrusão como **não intrusivos**;
- **Verdadeiro Positivo (VP):** Eventos classificados corretamente pelo método de detecção de intrusão como **intrusivos**;

Com base nas categorizações apresentadas acima, são calculadas diversas métricas de desempenho de classificação. As principais utilizadas neste trabalho são apresentadas a seguir (Liu & Lang, 2019).

1. **Acurácia:** esta métrica indica a proporção de eventos classificados corretamente em relação ao total de eventos existentes no conjunto de testes. Pode não ser um bom aspecto a ser considerado em casos de grande desbalanceamento de classes, sendo muito influenciada pela classe majoritária. A acurácia é calculada a partir da Equação 6.1, apresentada a seguir:

$$ACC = \frac{VP + VN}{VP + VN + FP + FN} \tag{6.1}$$

2. **Precisão:** essa taxa indica a proporção de eventos corretamente detectados como intrusivos dentre todos os detectados como intrusivos, como pode ser visto na Equação 6.2.

$$PRE = \frac{VP}{VP + FP} \tag{6.2}$$

3. ***Recall:*** ou *True Positive Rate* (TPR), também conhecida como sensibilidade, consiste no número de eventos corretamente classificados como intrusivos dentre todos os eventos intrusivos. A Equação 6.3 demonstra como é calculada a sensibilidade de um modelo.

$$Recall = \frac{VP}{VP + FN} \tag{6.3}$$

4. **F1-Score:** a pontuação F1 é a média harmônica de precisão e *recall*, onde uma *F1-Score* atinge seu melhor valor em 1 (precisão e *recall* perfeitas) e pior em 0. A fórmula para a *F1-Score* é apresentada na Equação 6.4.

$$F1 - Score = 2 * \frac{Precisão * Recall}{Precisão + Recall} \tag{6.4}$$

5. **Balanced Accuracy (BACC):** A Acurácia Balanceada é uma métrica interessante para avaliar o desempenho da detecção em conjuntos de dados não balanceados. É definido como a média do *recall* obtida em cada classe, conforme pode ser observado na Equação 6.5. Onde R_i corresponde a ao *recall* (R) obtido considerando a i-ésima (i) classe presente no conjunto de dados e n é a quantidade de classes existentes.

$$Acuracia\ Balanceada\ = \frac{\sum_{i=1}^{n} R_i}{n} \tag{6.5}$$

Além das métricas de classificação, também são consideradas neste trabalho, medidas a respeito do tempo necessário para realizar determinadas tarefas, como:

1. **Tempo de treino:** corresponde ao tempo, em segundos, que o algoritmo precisou para realizar o treinamento com um conjunto de treinamento correspondente a 9 *folds* do *10 K-Fold Stratified Cross Validation*.

2. **Tempo de teste:** corresponde ao tempo, em segundos, que o algoritmo precisou para realizar a predição das instâncias do *fold* de teste do *10 K-Fold Stratified Cross Validation*.

6.5 Avaliação estatística

Para realizar a avaliação foram aplicados métodos estatísticos com o objetivo de buscar a existência de diferenças estatísticas significativas entre os resultados dos experimentos após

compará-los. Sendo assim, foi definido a aplicação de testes de normalidade dos dados, testes de significância estatística, e o pós-teste.

Teste de Normalidade

Os testes de normalidade verificam se uma determinada amostra de dados pertence ou não a uma distribuição normal, podendo ser uma abordagem de análise gráfica ou estatística. Caso o conjunto de dados pertencer a uma distribuição normal ele deve ser submetido a uma análise estatística com métodos paramétricos, caso contrário, deve ser aplicado um método não paramétricos.

O teste de normalidade estatístico provê um valor estatístico que pode ser utilizado para realizar uma interpretação mais profunda dos resultados e um p-valor, utilizado para interpretar os testes de forma mais direta, indicando se a amostra faz parte de uma distribuição normal ou não.

Estes testes presumem inicialmente que a amostra faz parte de uma distribuição normal (hipótese nula ou H0). O nível de significância, também denotado como alfa ou α, é a probabilidade de rejeição da hipótese nula quando ela é verdadeira. Um nível de significância de $0,05$ indica um risco de 5% de concluir que existe uma diferença quando na realidade não há diferença. Desse modo, é definido um nível de significância (α) de 5% $(0,05)$, com intervalo de confiança de 95%. Caso $p <= \alpha$, H0 é rejeitada, caso $p > \alpha$, H0 não pode ser rejeitada. O teste de normalidade aplicado aos dados no processo de avaliação foi o *Shapiro-Wilk*. Um teste indicado para amostras pequenas e que retorna a W-statistic e o p-valor.

Teste de Significância

Os testes de significância podem ser utilizados para verificar se existem diferenças estatísticas significativas entre dois ou mais grupos. Podem ser aplicados utilizando conhecimento prévio do conjunto de dados, caso pertença a uma distribuição normal, aplicam-se testes paramétricos, caso não haja conhecimento prévio do conjunto de dados, deve ser utilizados testes não-paramétricos. Dentre os testes observados, considerando a natureza não pareada dos dados, e a existência de mais de dois conjuntos de amostras para comparação, as opções de testes considerados foram:

- **ANOVA:** Teste paramétrico que verifica se a média entre 2 ou mais grupos é significativamente diferente, utilizado para amostras independentes (Fisher, 1954);
- **Kruskal-Wallis:** Teste não-paramétrico, aplicado para comparar 2 ou mais amostras independentes (Kruskal & Wallis, 1952);

Para os dois casos considera-se que $p < \alpha$ indica que há diferença estatística significativa entre os grupos. Por outro lado, um $p > \alpha$ indica que não é possível dizer se há diferença estatística significativa.

Pós-Teste

A aplicação de um teste de significância como o ANOVA ou Kruskal-Wallis identifica se existem diferenças estatísticas significativas entre 2 ou mais amostras, no entanto, para verificar quais das amostras diferem é necessário aplicar uma técnica de pós-teste. Para realizar esta comparação foi selecionado o pós-teste de Dunn (Dunn, 1961), que realiza comparações em pares para cada grupo, e identifica quais pares apresentam diferenças estatísticas significativas.

6.6 Descrição dos experimentos

Nesta seção é apresentada a metodologia definida para a avaliação da abordagem proposta. A Figura 6.2 apresenta como foram conduzidos os experimentos. A base de eventos CICIDS2017 é primeiramente pré-processada, uma etapa que contempla a normalização de dados, a remoção de valores ou símbolos NaN, remoção de atributos repetidos e a criação de classes de ataques. Nos experimentos utilizamos um conjunto de 20% de eventos da base original, mantendo a proporção de classes.

Na Etapa 1 nenhuma ação é realizada no primeiro experimento. Nos demais experimentos o conjunto de dados CICIDS2017 é submetido para abordagens de seleção de atributos.

Na Etapa 2 os conjuntos de dados criados pela Etapa 1 são submetidos para a classificação de eventos. Conforme apresentados na proposta deste trabalho, propõe-se um classificador baseado no algoritmo Extra Tree. Em cada um desses experimentos foi utilizada a técnica *10 K-Fold Stratified Cross Validation* com 10 *folds*, com o objetivo de obter uma estimativa da precisão de cada abordagem em um cenário real.

A metodologia de experimentação conta com 6 experimentos. Desse modo, são gerados seis resultados, um com o conjunto de dados com a totalidade de atributos e cinco com subconjuntos reduzidos devido à seleção de atributos. Tais resultados são comparados conforme as métricas definidas na Seção 6.4. A seguir serão apresentados maiores detalhes a respeito de cada um dos experimentos realizados.

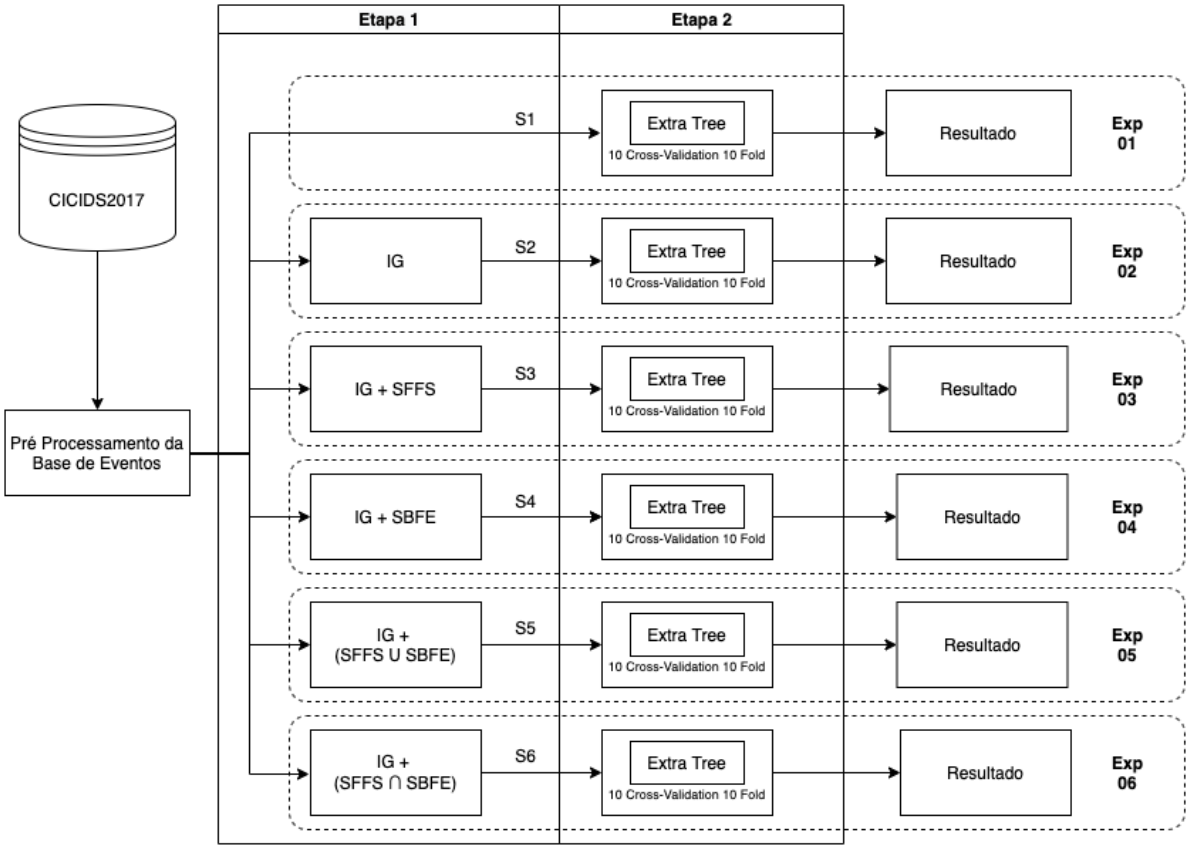

Figura 6.2: Arquitetura de avaliação proposta.
Fonte: O autor (2021).

6.6.1 Exp 01 - CICIDS2017 completa

Para efeitos de comparação e *benchmark* a base CICIDS2017 contendo todos os atributos foi utilizada no primeiro experimento. O conjunto completo foi submetido diretamente para o classificador *Extra Tree* na Etapa 2. O conjunto foi somente pré-processado. Este resultado será comparado com as variações do método de seleção de atributos proposto.

6.6.2 Exp 02 - CICIDS2017 selecionada por IG

No experimento 2, o conjunto de dados CICIDS2017 foi submetido inicialmente ao processo de seleção na Etapa 1. Neste experimento foi aplicado o algoritmo *Information Gain*. O código foi escrito em python, com suporte da biblioteca info_gain e segue a arquitetura apresentada na Figura 6.3. Este método gera um ranking de atributos, do maior ganho de informação para o menor. Após a aplicação do *threshold*, definido anteriormente como o maior gap no ganho de informação entre dois atributos em sequência, foram selecionados 56 atributos. Este subconjunto com 56 atributos é exibido na Tabela 6.2, e foi submetido para o classificador *Extra Tree* na Etapa 2 para o processo de classificação.

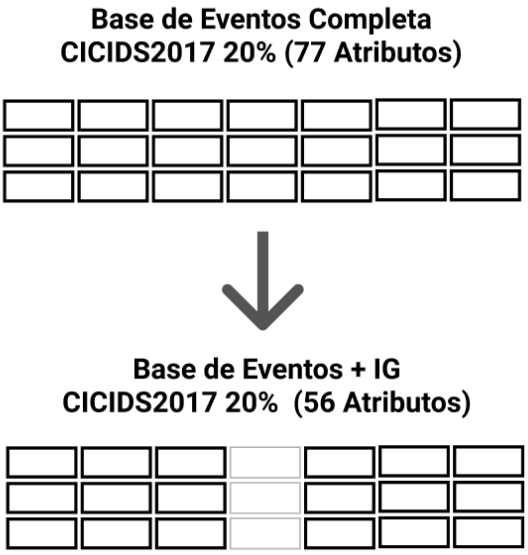

Figura 6.3: Fluxo da seleção de atributos por *Information Gain*.
Fonte: O autor (2021).

Tabela 6.2: Atributos selecionados pelo algoritmo Information Gain.

Atributos		
Destination Port	Flow Duration	Total Fwd Packets
Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets
Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean
Fwd Packet Length Std	Bwd Packet Length Max	Bwd Packet Length Min
Bwd Packet Length Mean	Bwd Packet Length Std	Flow Bytes/s
Flow Packets/s	Flow IAT Mean	Flow IAT Std
Flow IAT Max	Flow IAT Min	Fwd IAT Total
Fwd IAT Mean	Fwd IAT Std	Fwd IAT Max
Fwd IAT Min	Bwd IAT Total	Bwd IAT Mean
Bwd IAT Std	Bwd IAT Max	Bwd IAT Min
Fwd Header Length	Bwd Header Length	Fwd Packets/s
Bwd Packets/s	Min Packet Length	Max Packet Length
Packet Length Mean	Packet Length Std	Packet Length Variance
Average Packet Size	Avg Fwd Segment Size	Avg Bwd Segment Size
Subflow Fwd Packets	Subflow Fwd Bytes	Subflow Bwd Packets
Subflow Bwd Bytes	Init_Win_bytes_forward	Init_Win_bytes_backward
act_data_pkt_fwd	min_seg_size_forward	Active Mean
Active Max	Active Min	Idle Mean
Idle Max	Idle Min	

6.6.3 Exp 03 - CICIDS2017 selecionada por IG + SFFS

O experimento 3 é referente a seleção de atributos baseada em IG e na abordagem *wrapper* SFFS. Primeiramente o IG é aplicado, formando um subconjunto de 56 atributos. Este subconjunto é submetido para o algoritmo de seleção de atributos *wrapper* SFFS. O código foi escrito em python, com suporte da biblioteca SequentialFeatureSelector presente na sklearn e segue a arquitetura apresentada na Figura 6.4. Conforme citado no Capitulo 5, a estratégia adotada com os métodos *wrapper* não busca encontrar a quantidade ideal de atributos para o subconjunto. Ela utiliza um valor de K predefinido como $1/3$ da quantidade de atributos existentes. Neste caso foi definido um $K = 25$. Sendo assim, o resultado desta execução é um subconjunto contendo os 25 melhores atributos do conjunto de dados, eles são apresentados na Tabela 6.3. Como motor de indução foi utilizado o algoritmo ET, como apresentado na Subseção 5.2.1.

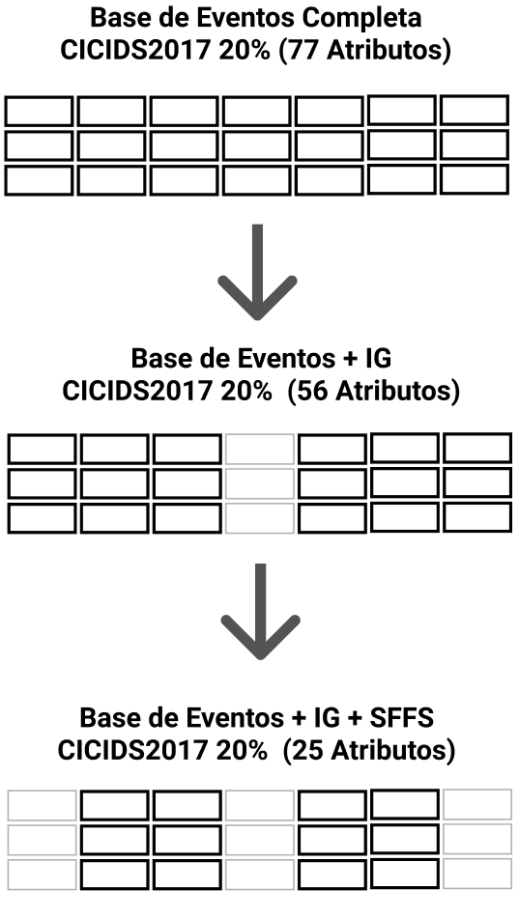

Figura 6.4: Fluxo da seleção de atributos por SFFS.
Fonte: O autor (2021).

Tabela 6.3: Atributos selecionados pelo algoritmo SFFS.

Atributos		
act_data_pkt_fwd	Active Max	Avg Fwd Segment Size
Bwd Header Length	Bwd IAT Max	Bwd IAT Mean
Bwd IAT Min	Bwd IAT Total	Bwd Packet Length Min
Destination Port	Fwd Header Length	Fwd IAT Max
Fwd IAT Min	Fwd Packet Length Min	Fwd Packet Length Std
Idle Mean	Idle Min	Init_Win_bytes_backward
Init_Win_bytes_forward	Min Packet Length	min_seg_size_forward
Packet Length Std	Subflow Bwd Bytes	Subflow Fwd Bytes
Total Length of Bwd Packets		

6.6.4 Exp 04 - CICIDS2017 selecionada por IG + SBFE

No experimento 4 sé avaliada a abordagem de seleção de atributos baseada em IG e na técnica *wrapper* SBFE. Do mesmo modo, inicialmente 56 atributos do conjunto de dados são selecionados pelo IG. Posteriormente é aplicado o SBFE sobre esse subconjunto com 56 atributos, e então o novo subconjunto gerado pelo SBFE é submetido para o classificador ET. O código foi escrito em python, com suporte da biblioteca SequentialFeatureSelector presente na sklearn e segue a arquitetura apresentada na Figura 6.5.

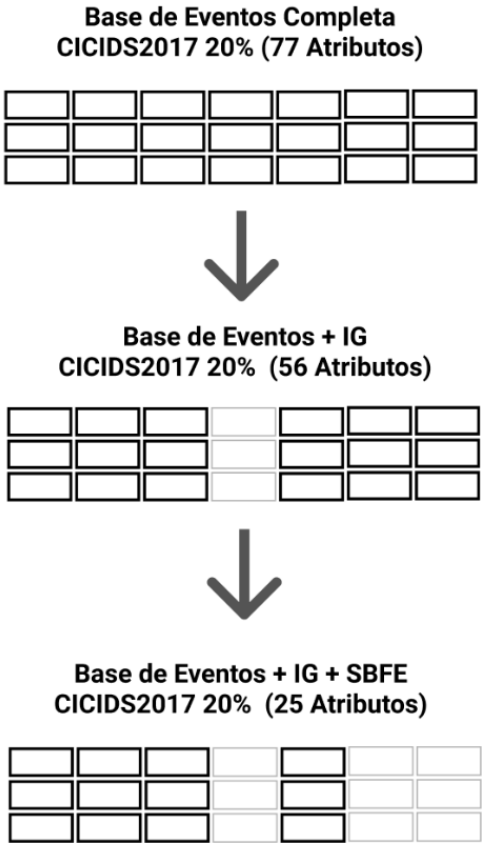

Figura 6.5: Fluxo da seleção de atributos por SBFE.
Fonte: O autor (2021).

Assim como o SFFS, o SBFE também utiliza uma estratégia baseada em um K pré definido para o número de atributos no subconjunto final, foi mantido o $K = 25$. Sendo assim, o resultado desta execução é um subconjunto contendo os 25 melhores atributos, os quais são indicados na Tabela 6.4. Como motor de indução foi utilizado o algoritmo ET, como apresentado na Subseção 5.2.1.

Tabela 6.4: Atributos selecionados pelo algoritmo SFBE.

Atributos		
act_data_pkt_fwd	Active Mean	Active Min
Avg Bwd Segment Size	Avg Fwd Segment Size	Bwd IAT Max
Bwd IAT Mean	Bwd IAT Total	Bwd Packet Length Mean
Bwd Packet Length Std	Destination Port	Fwd Header Length
Fwd IAT Min	Fwd IAT Std	Fwd Packet Length Max
Idle Max	Idle Min	Init_Win_bytes_backward
Init_Win_bytes_forward	Min Packet Length	min_seg_size_forward
Subflow Fwd Bytes	Total Backward Packets	Total Length of Bwd Packets
	Total Length of Fwd Packets	

6.6.5 Exp 05 - CICIDS2017 selecionada por IG + (SFFS ∪ SBFE)

O experimento 5 apresenta uma avaliação a respeito da abordagem de seleção de atributos proposta com IG e a combinação por união das técnicas SFFS e SBFE. A base é selecionada inicialmente pelo IG. Sobre o subconjunto de 56 atributos gerados pela primeira etapa, são aplicados o SFFS e o SBFE, conforme pode ser observado na Figura 6.6. Conforme apresentado na Seção 5.2.1, o método união é utilizado para realizar a combinação entre os atributos selecionados pelos algoritmos SFFS e SBFE. Desse modo, são selecionados todos os atributos que estão no conjunto gerado pelo SFFS ou no conjunto gerado pelo SBFE. Esta união acontece na Etapa 2 do método *ensemble* de seleção de atributos proposto. A união entre os dois subconjuntos de atributos resultaram em um subconjunto de 35 atributos, apresentados na Tabela 6.5. Este subconjunto com 35 atributos é então submetido para o método ET realizar a classificação dos eventos.

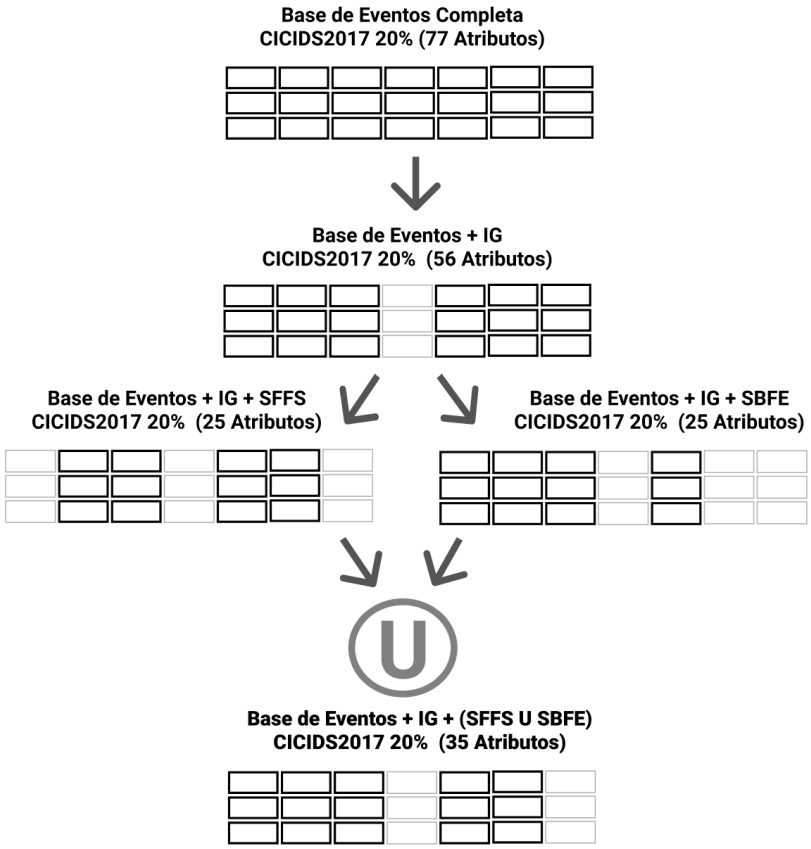

Figura 6.6: Fluxo da união dos atributos resultantes do método de seleção SFFS e SBFE.
Fonte: O autor (2021).

Tabela 6.5: Atributos resultantes da união dos atributos SFFS e SFBE.

Atributos		
act_data_pkt_fwd	Active Max	Active Mean
Active Min	Avg Bwd Segment Size	Avg Fwd Segment Size
Bwd Header Length	Bwd IAT Max	Bwd IAT Mean
Bwd IAT Min	Bwd IAT Total	Bwd Packet Length Mean
Bwd Packet Length Min	Bwd Packet Length Std	Destination Port
Fwd Header Length	Fwd IAT Max	Fwd IAT Min
Fwd IAT Std	Fwd Packet Length Max	Fwd Packet Length Min
Fwd Packet Length Std	Idle Max	Idle Mean
Idle Min	Init_Win_bytes_backward	Init_Win_bytes_forward
Min Packet Length	min_seg_size_forward	Packet Length Std
Subflow Bwd Bytes	Subflow Fwd Bytes	Total Backward Packets
Total Length of Bwd Packets	Total Length of Fwd Packets	

6.6.6 Exp 06 - CICIDS2017 selecionada por IG + (SFFS ∩ SBFE)

Por fim o experimento 6 apresenta a abordagem de seleção proposta com IG e com a combinação por intersecção das técnicas SFFS e SBFE. Inicialmente o IG é utilizado e gera um subconjunto com 56 atributos, sobre este subconjunto são aplicadas as técnicas SFFS e SBFE. Conforme apresentado na Seção 5.2.1, a combinação utilizada entre os atributos selecionados pelos algoritmos SFFS e SFBE é a interseção. Esta intersecção é realizada na Etapa 2 do método *ensemble* de seleção de atributos sendo apresentado na Figura 6.7. A intersecção entre os dois subconjuntos de atributos resultaram em um subconjunto de 15 atributos, apresentados na Tabela 6.6. O subconjunto com 15 atributos é então submetido para o método ET realizar a classificação dos eventos.

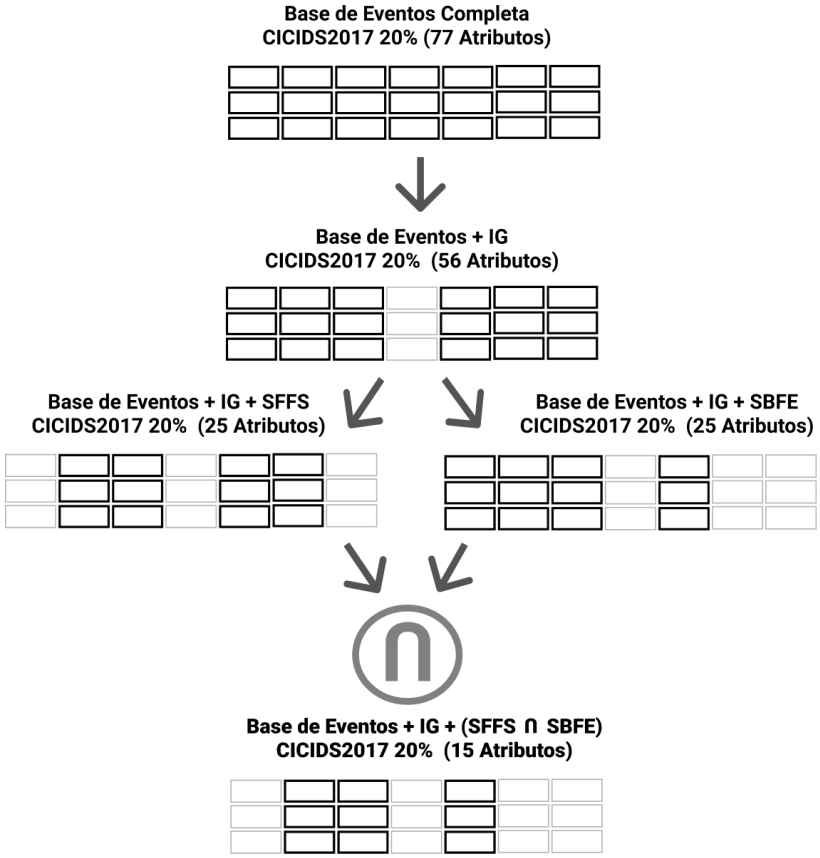

Figura 6.7: Fluxo da interseção dos atributos resultantes do método de seleção SFFS e SBFE. Fonte: O autor (2021).

Tabela 6.6: Atributos resultantes da interseção dos atributos SFFS e SFBE.

Atributos		
act_data_pkt_fwd	Avg Fwd Segment Size	Bwd IAT Max
Bwd IAT Mean	Bwd IAT Total	Destination Port
Fwd Header Length	Fwd IAT Min	Idle Min
Init_Win_bytes_backward	Init_Win_bytes_forward	Min Packet Length
min_seg_size_forward	Subflow Fwd Bytes	Total Length of Bwd Packets

6.7 Materiais Aplicados

A seguir são apresentadas os principais materiais e técnicas utilizados na metodologia e avaliação da abordagem proposta.

- Linguagem Python versão 3.6;
- Bibliotecas: Pandas, SciKitLearn, numpy, info_gain;
- Google Colabolatory Intel Xeon 2.0Ghz, 12,69 GB Ram, 68 GB disco;

- Azure Machine Learning 8 Núcleos, 32 GB Ram, 200 GB Disco.

Capítulo 7

Resultados

No capítulo anterior foi apresentada a metodologia para avaliação da abordagem proposta com diferentes variações do método *ensemble* de seleção de atributos. Os seis experimentos apresentados foram conduzidos de forma individual e isolada com o objetivo de avaliar os resultados experimentais, demonstrando assim as vantagens e diferenciais entre as variações. Para fins de comparação foi utilizado o mesmo classificador (*Extra Tree*, ou ET) em todos os experimentos, utilizando as mesmas parametrizações.

O número de atributos selecionados por cada abordagem de seleção de atributos é um fator importante que afeta diretamente o resultado do processo de classificação de cada experimento. A quantidade de atributos selecionados em cada experimento é apresentada na Tabela 7.1.

Tabela 7.1: Quantidade de atributos selecionados por abordagem.

Abordagem	Quantidade de Atributos
-	77
IG	56
IG + SFFS	25
IG + SBFE	25
IG+(SFFS $\cup$ SBFE)	35
IG+(SFFS $\cap$ SBFE)	15

A Tabela 7.2 apresenta os resultados obtidos no experimento 01. Este experimento contou com a classificação do conjunto de dados com o classificador ET diretamente, sem realizar o processo de filtragem de atributos. São apresentadas diversas métricas individuais das classes de ataques. Elas são calculadas conforme as métricas definidas na Seção 6.4 (Página 65).

Tabela 7.2: Resultados do Experimento 01 com conjunto de dados CICIDS2017 completa.

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,90%	99,88%	99,89%
DoS/DDoS	99,74%	99,73%	99,74%
PortScan	99,40%	99,51%	99,45%
Brute Force	99,82%	99,74%	99,78%
Web Attack	97,93%	97,66%	97,77%
Botnet	63,02%	71,98%	66,94%
Infiltration	18,89%	18,89%	18,89%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
23,3054	0,1294	99,81%	87,98%	99,82%

É possível observar que a abordagem ET considerando os 77 atributos do conjunto de dados conseguiu alcançar boas taxas de acertos. A métrica *recall* indica a quantidade de eventos detectados corretamente de uma determinada classe em relação à quantidade de eventos dessa classe existentes no conjunto, a fórmula dessa métrica é apresentada na Seção 6.4 (Página 65). Desse modo, pode-se afirmar que aproximadamente 99,88% do tráfego benigno foi identificado como benigno pela abordagem. Do mesmo modo, observa-se que a abordagem foi capaz de alcançar taxas de detecção muito próximas a 99,7% para os ataques *DoS/DDoS*, *PortScan* e *Brute Force*. Ao se analisar a precisão obtida nessas classes também observa-se um comportamento muito similar. Estas quatro classes também apresentaram precisão superior a 99,4%. Conforme definido na Seção 6.4 (Página 65), a precisão indica a proporção de eventos classificados corretamente como determinada classe em relação à quantidade de eventos daquela classe existentes no conjunto.

A métrica F1-Score apresentada na Tabela 7.2 corresponde a média harmônica de precisão e *recall*, onde uma *F1-Score* atinge seu melhor valor em 1 (precisão e *recall* perfeitas) e pior em 0. Desse modo, observa-se que as classes *Benign*, *DoS/DDoS*, *PortScan* e *Brute Force* que apresentaram *recall* e precisão superiores a 99% também apresentaram F1-Score superior a 99%.

A abordagem ET apresentou um *recall* de aproximadamente 97,66% para a classe *Web Attack*, indicando que cerca de apenas 2,34% dos eventos relacionados a esta categoria de ataque não foram detectados. Além disso, aproximadamente 97,93% dos eventos identificados como *Web Attack* eram realmente eventos relacionados a esta classe. Por alcançar *recall* e precisão similares, o F1-Score também ficou próximo a 98%.

Em relação aos eventos de ataques *Botnet*, foi possível alcançar uma taxa de *recall* de aproximadamente 72%, valor bem abaixo das classes anteriores. Além disso, apenas 63,02% dos eventos identificados como *Botnet* eram realmente ataques *botnet*, essa baixa precisão indica que a taxa de falsos positivos foi alta. Uma alta taxa de falsos positivos é um grande problema para abordagens de detecção baseadas em anomalia, pois, são eventos legítimos que

foram erroneamente identificados como intrusão. O tráfego detectado como intrusão é descartado ou bloqueado, interrompendo o fluxo de dados. Uma alta taxa de falsos positivos, portanto, pode inviabilizar todo o tráfego legítimo da rede.

No experimento 01 a abordagem apresentou uma precisão de 18,89% para a classe *Infiltration*, ou seja, apenas 18,89% dos eventos identificados como *Infiltration* era realmente tráfego desta categoria de ataque. O *recall* obtido também foi de apenas 18,89%, muito abaixo do ideal.

As métricas gerais dos resultados da abordagem avaliada no Experimento 01 são apresentadas na parte inferior da Tabela 7.2. Conforme pode ser observado, a abordagem apresentou um tempo de 23,3054 segundos para realizar o treinamento do modelo e precisou de 0,1294 segundos para realizar a classificação do conjunto de dados. Em relação à acurácia, a abordagem alcançou uma taxa de 99,81%, uma taxa bem alta. No entanto, conforme pode ser observado nas taxas de detecção por classe de ataque, houve classes que não obtiveram um desempenho satisfatório. A métrica geral de acurácia permaneceu com uma taxa tão alta porque as classes com maior número de instâncias tiveram ótimo desempenho. Desse modo, o desempenho ruim de algumas classes não influenciaram nessa métrica geral. Com o objetivo de avaliar de modo geral o real desempenho da abordagem, é apresentada a métrica de acurácia balanceada, para remover o *bias* que ocorre devido ao desbalanceamento de classes. Esta métrica calcula a acurácia considerando que todas as classes possuem o mesmo peso, ou influência. Neste caso a acurácia balanceada obtida foi de 87,98%, portanto, mais de 10% menor que a acurácia padrão.

Na Tabela 7.3 são expostos os resultados obtidos no experimento 02. Neste experimento o classificador ET foi aplicado sobre o conjunto de dados com 56 atributos, os quais foram selecionados pelo IG. Desse modo, 21 atributos não foram selecionados sendo descartados do experimento. Pode ser observado que as taxas de *recall*, precisão e F1-Score para as classes *Benign*, *DoS/DDoS*, *PortScan* e *Brute Force* mantiveram-se todas acima de 99% assim como no experimento 01. Em relação à classe *Web Attack* é possível observar uma diminuição miníma na precisão em relação ao experimento 01, o que ocasionou uma leve diminuição no *F1-Score*.

Tabela 7.3: Resultados do Experimento 02 com conjunto de dados CICIDS2017 com atributos selecionados por IG.

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,89%	99,87%	99,88%
DoS/DDoS	99,71%	99,68%	99,69%
PortScan	99,40%	99,51%	99,45%
Brute Force	99,80%	99,75%	99,78%
Web Attack	96,37%	97,30%	96,80%
Botnet	63,56%	72,84%	67,68%
Infiltration	21,11%	21,11%	21,11%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
20,6564	0,1207	99,80%	88,50%	99,81%

Assim como no experimento 01 com a base completa, as taxas de precisão, *recall* e *F1-Score* obtidos no experimento 02 para as classes *Botnet* e *Infiltration* se mantiveram menores em relação as demais classes. O resultado dos dois primeiros experimentos foram muito similares, tanto em relação as métricas gerais quanto em relação as métricas específicas de cada classe. Isso indica que os 21 atributos descartados praticamente não influenciaram, nem positivamente, nem negativamente na tarefa de classificação.

No experimento 03, o classificador ET foi aplicado sobre o conjunto de dados com 25 atributos. Neste caso, os 77 atributos originais do conjunto foram reduzidos para 56 pelo IG e, após isso, o método *wrapper* SFFS foi responsável por selecionar apenas 25 atributos, dos 56 existentes. Os resultados obtidos neste experimento são apresentados na Tabela 7.4.

Para os eventos das classes *Benign*, *DoS/DDoS*, *PortScan* e *Brute Force*, as métricas se mantiveram sem expressivas variações, mas com uma constante melhora no *Recall* de todas as classes. Já para a classe *Web Attack* a precisão foi menor, de 96,37% no Experimento 02 para 91,71% no Experimento 03, o que indica um leve aumento no número de falsos positivos. O *recall* desta classe apresentou uma pequena melhora em relação aos experimentos anteriores, indicando que mais *Web Attacks* foram detectados.

Tabela 7.4: Resultados do Experimento 03 com conjunto de dados CICIDS2017 com atributos selecionados por IG + SFFS.

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,96%	99,90%	99,93%
DoS/DDoS	99,89%	99,88%	99,89%
PortScan	99,39%	99,89%	99,64%
Brute Force	99,98%	99,84%	99,91%
Web Attack	91,71%	98,42%	94,90%
Botnet	71,63%	92,05%	80,42%
Infiltration	18,89%	18,89%	18,89%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
14,2545	0,1127	99,89%	91,20%	99,90%

Cabe destacar aqui que ocorreu um aumento significativo relacionado as métricas obtidas na classificação dos eventos da classe *Botnet*. A abordagem do experimento 02 obteve *recall* de apenas 72,84%, já a abordagem do experimento 03 foi capaz de obter um aumento extremamente significativo, chegando a detectar aproximadamente 92,05% do tráfego *Botnet*. Além disso, no experimento 02 a precisão foi de aproximadamente 63,56%, enquanto que a abordagem do experimento 03 conseguiu obter precisão de aproximadamente 71,63%, indicando uma redução do número de falsos positivos. Devido à melhora observada nas métricas precisão e principalmente *recall*, a métrica *F1-Score* também foi elevada, de 67,68% para 80,42%.

A melhora, observada nas taxas de detecção de alguns categorias de ataque, refletiu na métrica geral de acurácia balanceada. A abordagem do Experimento 04, com atributos selecionados por IG + SFFS, obteve acurácia balanceada de 91,20%, superior às abordagens anteriores que alcançaram aproximadamente 88,5%. Em relação ao tempo de treino houve uma melhora expressiva em relação ao experimento anterior, no Experimento 04 o tempo necessário para o treinamento foi de 14,2545 segundos.

No experimento 04, após a seleção dos 56 atributos pelo IG, o método *wrapper* SBFE foi responsável por descartar 31 atributos. Desse modo, o classificador foi aplicado no subconjunto de apenas 25 atributos que não foram descartados. Os resultados obtidos neste experimento são apresentados na Tabela 7.5. Ao analisar os resultados do Experimento 04 e compará-lo com os experimentos anteriores percebe-se que foi possível obter melhoras em relação aos dois primeiros experimentos. Além disso, percebe-se que em relação ao Experimento 03 não houve mudanças significativas nos resultados, nem em relação aos resultados gerais, nem em relação aos resultados específicos de cada classe.

Tabela 7.5: Resultados do Experimento 04 com conjunto de dados CICIDS2017 com atributos selecionados por IG + SBFE.

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,97%	99,90%	99,94%
DoS/DDoS	99,90%	99,90%	99,90%
PortScan	99,39%	99,89%	99,64%
Brute Force	99,98%	99,87%	99,92%
Web Attack	90,68%	98,52%	94,39%
Botnet	71,66%	91,56%	80,27%
Infiltration	22,22%	22,22%	22,22%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
15,4417	0,1117	99,89%	91,59%	99,90%

Há uma relação entre ambos experimentos, pois ambos tiveram como conjunto de atributos os inicialmente selecionados pelo IG e posteriormente selecionados por uma abordagem *wrapper* de seleção de atributos. Apesar de no Experimento 03 ter sido utilizada a abordagem SFFS e no Experimento 04 a abordagem SBFE, o classificador utilizado como motor de indução foi o mesmo, assim como a quantidade de atributos selecionados em cada abordagem de seleção.

Os conjuntos de 25 atributos selecionados por cada abordagem são distintos, possuem atributos em comum e atributos diferentes. Desse modo, estas diferenças e similaridades foram exploradas nos próximos experimentos através de combinações de união e intersecção.

A Tabela 7.6 apresenta os resultados obtidos no experimento 05. Neste experimento foi realizada a classificação do conjunto de dados com o classificador ET considerando apenas 35 atributos. Conforme apresentado no capítulo anterior, neste experimento foi empregada a abordagem completa de seleção de atributos contando com combinador de união. Na primeira etapa da abordagem o IG foi responsável por selecionar 56 atributos do conjunto original. Na segunda etapa o conjunto com 56 atributos foi aplicado de maneira paralela aos métodos *wrappers* SFFS e SBFE. Cada um deles selecionou 25 atributos, os quais foram então relacionados de modo a realizar uma combinação de união entre os dois conjuntos. Desse modo, o conjunto de dados submetido ao classificador possuía 35 atributos, os quais haviam sido selecionados pelo SFFS ou pelo SBFE. A análise dos resultados revela que não houve uma melhoria expressiva na detecção de intrusão quando comparado com os Experimentos 03 e 04, mesmo unindo os atributos que foram utilizados em ambas abordagens. Inclusive, tanto os resultados gerais quanto os resultados específicos de cada classe se mantiveram similares. Houve apenas um leve aumento no tempo de treinamento devido à quantidade de atributos superior.

Tabela 7.6: Resultados do Experimento 05 com conjunto de dados CICIDS2017 com atributos selecionados por IG+(SFFS ∪ SBFE).

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,96%	99,90%	99,93%
DoS/DDoS	99,89%	99,87%	99,88%
PortScan	99,39%	99,89%	99,64%
Brute Force	99,98%	99,84%	99,91%
Web Attack	91,38%	98,45%	94,73%
Botnet	71,37%	91,53%	80,09%
Infiltration	18,89%	18,89%	18,89%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
16,1894	0,1135	99,89%	91,29%	99,89%

Por fim, o experimento 06 contou com a classificação com o classificador ET no conjunto de dados com 15 atributos gerados pela abordagem completa de seleção de atributos contando com combinador de intersecção. Desse modo, o conjunto de dados de 25 atributos gerados pelas abordagens SFFS e SBFE, foram relacionados de modo a realizar uma combinação de intersecção entre os dois conjuntos. Desse modo, o conjunto de dados submetido ao classificador possuía 15 atributos, os quais haviam sido selecionados tanto pelo SFFS quanto pelo SBFE. Ao compararmos os resultados do Experimento 06 com os Experimentos 03 e 04 não há diferenças expressivas entre a Precisão o *Recall* e o *F1-Score*. Essa constância nos resultados mesmo com o Experimento 06 tendo utilizado uma quantidade de atributos reduzida em relação as demais, indica que a intersecção entre ambos os conjuntos permitem atingir um resultado similar com um menor esforço de treinamento do classificador. O tempo de treinamento atingido no Experimento 06 foi de 7,7558 segundos, sendo o treinamento mais rápido realizado entre os experimentos e diretamente relacionado com a quantidade de atributos utilizados, sendo o conjunto de atributos mais enxuto. A acurácia balanceada também tem destaque entre as métricas desta abordagem, visto que entre os experimentos obteve uma das melhores pontuações.

Tabela 7.7: Resultados do Experimento 06 com conjunto de dados CICIDS2017 com atributos selecionados por IG+(SFFS ∩ SBFE).

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,97%	99,90%	99,93%
DoS/DDoS	99,89%	99,89%	99,89%
PortScan	99,37%	99,87%	99,62%
Brute Force	99,94%	99,84%	99,89%
Web Attack	90,84%	98,34%	94,39%
Botnet	71,01%	91,87%	79,97%
Infiltration	21,11%	21,11%	21,11%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
7,7558	0,1078	99,89%	91,47%	99,90%

Os resultados dos experimentos apresentados nas tabelas anteriores são analisados e discutidos a seguir. Na Tabela 7.8 é apresentada uma nova visualização comparativa dos resultados dos experimentos em relação a tempo de treino (s), tempo de teste (s), acurácia, acurácia balanceada e precisão entre as abordagens propostas.

Tabela 7.8: Métricas de avaliação para os métodos de seleção de atributos.

	-	**IG**	**IG + SFFS**	**IG + SBFE**	**IG + (SFFS∪SBFE)**	**IG + (SFFS∩SBFE)**
Treino (s)	23,3054	20,6564	14,2545	15,4417	16,1894	7,7558
Teste (s)	0,1294	0,1207	0,1127	0,1117	0,1135	0,1078
Acc	99,81%	99,80%	99,89%	99,89%	99,89%	99,89%
Acc B.	87,98%	88,50%	91,20%	91,59%	91,29%	91,47%
Precisão	99,82%	99,81%	99,90%	99,90%	99,89%	99,90%

Além disso, as métricas gerais de classificação também são apresentadas no gráfico da Figura 7.1. Podemos observar que em relação a acurácia não é possível identificar grande variação, tem-se uma média de 99,96%. Com o objetivo de avaliar as métricas de classificação extraídas, foram aplicados testes estatísticos, visando verificar a existência de diferenças estatísticas significativas entre as medidas. Nesta análise foram levadas em consideração as 10 médias de acurácia para cada um dos 6 experimentos. As informações referentes a esta análise são apresentadas na Tabela 7.9. É possível observar que os dados não apresentam uma distribuição normal, e devido a essa não normalidade, foi aplicado o método de Kruskal Wallis para verificar a existência de diferenças estatísticas significativas entre os grupos.

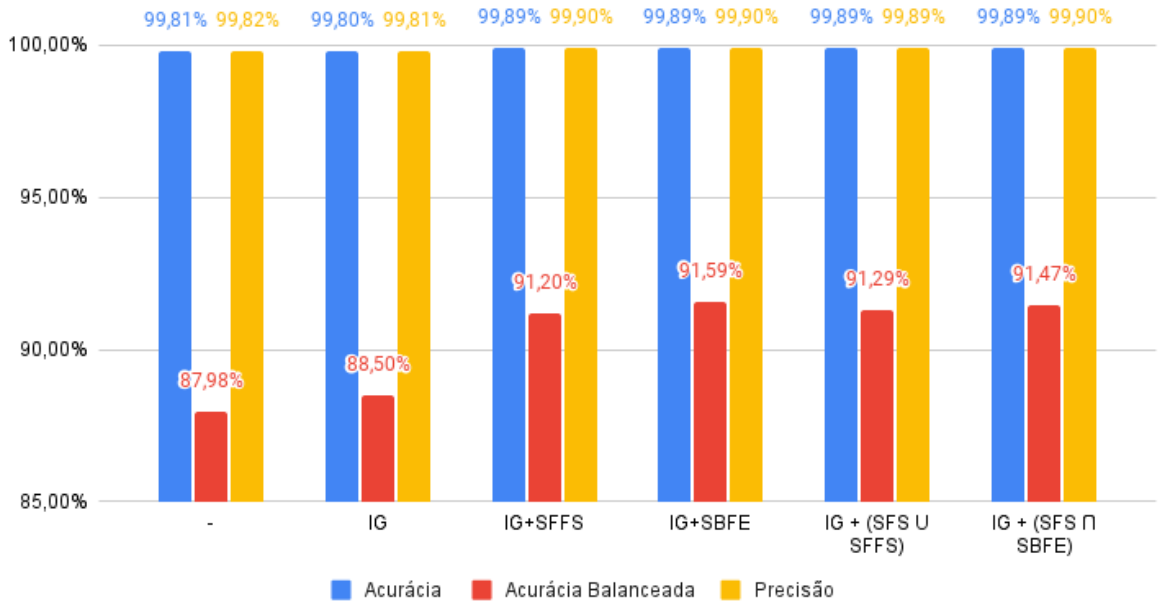

Figura 7.1: Gráfico da acurácia, acurácia balanceada e precisão versus abordagem de seleção de atributos.

Fonte: O autor (2021).

Tabela 7.9: Testes estatísticos para Acurácia.

Shapiro Wilk (Normalidade)		**Kruskal Wallis (Significância)**	
W	0.6378495	statistic	62.764662
$p-valor$	1.741809e-11	$p-valor$	3.256914e-12

Dunn	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	1.000000	1.000000	1.145e-03	3.291e-04	1.145e-03	1.145e-03
Exp 2	1.000000	1.000000	8.697e-07	1.659e-07	8.697e-07	8.697e-07
Exp 3	0.001145	8.697e-07	1.000000	1.000000	1.000000	1.000000
Exp 4	0.000329	1.659e-07	1.000000	1.000000	1.000000	1.000000
Exp 5	0.001145	8.697e-07	1.000000	1.000000	1.000000	1.000000
Exp 6	0.001145	8.697e-07	1.000000	1.000000	1.000000	1.000000

Considerou-se um intervalo de confiança de 95% (com nível de confiança de $\alpha = 0,05$), portanto, o valor observado do $p-valor = 3.256914e-12$, que é menor do que $0,05$, indica que há diferenças estatisticamente significativas entre os grupos. Para verificar quais amostras específicas apresentavam diferenças estatisticamente significativas, foi aplicado o Pós-Teste de Dunn, o teste recomendado para comparações par a par após a aplicação de Kruskal-Wallis. Este foi aplicado também com um intervalo de confiança de 95% ($\alpha = 0,05$), portanto, nas comparações onde observa-se um $p-valor > 0,05$, provavelmente as a mostras não apresentam diferenças estatísticas significativas, enquanto, onde observa-se um $p-valor <= 0,05$, as amostras provavelmente apresentam diferenças estatísticas significativas. A correção de Bonferroni foi utilizada para controlar a taxa de erro familiar.

Com a aplicação do pós-teste foram observadas diferenças estatísticas significativas para a acurácia entre o experimento 1 e os experimentos 3, 4, 5 e 6, assim como observado entre o experimento 2 os experimentos 3, 4, 5 e 6. A maior diferença significativa foi apontada entre o experimento 2 e 4. Assim demonstra-se que as quatro últimas abordagens, que possuem os algoritmos da abordagem *wrapper*, possuem melhor acurácia comparado a uma abordagem que não utiliza métodos de atributos para redução de dimensionalidade ou uma abordagem somente baseada em *Information Gain* (IG).

No caso da métrica de precisão, observa-se que os resultados obtidos pelas abordagens também apresentaram pouca variação, tendo uma maior oscilação nos dois primeiro experimentos e mantendo uma oscilação irrelevante entre os quatro últimos experimentos. Na análise estatística realizada os dados não apresentaram normalidade, conforme pode ser observado na Tabela 7.10. Devido a essa não normalidade, foi aplicado o método de Kruskal Wallis para verificar a existência de diferenças estatísticas significativas entre os grupos.

Tabela 7.10: Testes estatísticos para Precisão.

Shapiro Wilk (Normalidade)			**Kruskal Wallis (Significância)**			
W	0.673132		statistic	55.956568		
$p-valor$	7.649408e-11		$p-valor$	8.295643e-11		
Dunn	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	1.000000	1.000000	0.000652	2.0026e-04	0.087314	0.001985
Exp 2	1.000000	1.000000	0.000001	2.7945e-07	0.000845	0.000005
Exp 3	0.000652	1.2610e-06	1.000000	1.000000	1.000000	1.000000
Exp 4	0.000200	2.7945e-07	1.000000	1.000000	1.000000	1.000000
Exp 5	0.087314	8.4488e-04	1.000000	1.000000	1.000000	1.000000
Exp 6	0.001985	5.3125e-06	1.000000	1.000000	1.000000	1.000000

Considerou-se um intervalo de confiança de 95% ($\alpha = 0,05$), portanto, o valor observado do $p-valor = 8.295643e-11$ é menor do que $0,05$, indicando que há diferenças estatísticas significativas entre os grupos. Para verificar quais amostras específicas apresentavam diferenças estatísticas significativas, foi aplicado o Pós-Teste de Dunn com um intervalo de confiança de 95% ($\alpha = 0,05$).

Com a aplicação do pós-teste foram observadas diferenças estatísticas significativas para a precisão entre o experimento 1 e os experimentos 3, 4 e 6, assim como observado entre o experimento 2 os experimentos 3, 4, 5 e 6. A maior diferença significativa foi apontada entre o experimento 2 e 4. No geral, as quatro últimas abordagens, que possuem os algoritmos da abordagem *wrapper*, possuem melhor precisão comparada a uma abordagem que não utiliza métodos de atributos para redução de dimensionalidade ou uma abordagem somente baseada em *Information Gain* (IG).

Ao comparar a menor acurácia balanceada, de 87,98% do Experimento 01 utilizando

os atributos originais, e da segunda maior acurácia balanceada, de 91,47% do experimento 06 (IG+(SFFS ∩ SBFE)), há um ganho importante de acurácia balanceada, ou seja, há um desempenho melhor de detecção em algumas classes de ataques. As classes *Botnet* e *Web Attack* foram as duas classes que performaram melhor no último experimento em relação aos primeiros. Estas diferenças foram comprovadas através dos testes estatísticos, as informações sobre os resultados da análise são apresentadas na Tabela 7.11. Observa-se que os dados não apresentaram normalidade, desse modo, foi aplicado o método de Kruskal Wallis para verificar a existência de diferenças estatísticas significativas entre os grupos.

Tabela 7.11: Testes estatísticos para Acurácia Balanceada.

Shapiro Wilk (Normalidade)		**Kruskal Wallis (Significância)**	
W	0.884342	statistic	44.413485
$p-valor$	1.657468e-05	$p-valor$	1.909050e-08

Dunn	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	1.000000	1.000000	0.001126	0.000040	0.000195	0.000059
Exp 2	1.000000	1.000000	0.021096	0.001293	0.004900	0.001781
Exp 3	0.001126	0.021096	1.000000	1.000000	1.000000	1.000000
Exp 4	0.000040	0.001293	1.000000	1.000000	1.000000	1.000000
Exp 5	0.000195	0.004900	1.000000	1.000000	1.000000	1.000000
Exp 6	0.000059	0.001781	1.000000	1.000000	1.000000	1.000000

Considerou-se um intervalo de confiança de 95% ($\alpha = 0,05$), portanto, o valor observado do $p-valor = 1.909050e-08$ é menor do que $0,05$, indicando que há diferenças estatísticas significativas entre os grupos. Para verificar quais amostras específicas apresentavam diferenças estatísticas significativas, foi aplicado o Pós-Teste de Dunn com um intervalo de confiança de 95% ($\alpha = 0,05$). Com a aplicação do pós-teste foram observadas diferenças estatísticas significativas para a acurácia balanceada entre o experimento 1 e os experimentos 3, 4, 5 e 6, assim como observado entre o experimento 2 os experimentos 3, 4, 5 e 6. A maior diferença significativa foi apontada entre o experimento 2 e 4.

Desse modo, concluímos que as quatro últimas abordagens, que possuem atributos selecionados com algoritmos *wrapper*, alcançaram melhores resultados em todas as métricas de classificação em relação à abordagem que não utiliza métodos de seleção de atributos ou à abordagem com atributos selecionados somente pelo *Information Gain* (IG). Entre os experimentos com os algoritmos *wrapper*, mesmo com quantidade de atributos diferentes as taxas de detecção não foram significativamente alteradas. No entanto, a redução de atributos também reduziu o custo computacional, tornando mais plausível e possível aplicações em tempo real. A seguir as abordagens são avaliadas em relação ao tempo de treino e de teste para entender o impacto dessa redução de atributos no custo computacional.

Na Figura 7.2 é apresentado o gráfico a respeito do tempo de treino em segundos de cada

abordagem avaliada nos experimentos. Observa-se que os tempos de treinamento reduziram-se a medida que a quantidade de atributos era reduzida. Desta maneira, evidenciando a correlação entre a quantidade de atributos e tempo de treinamento. Nota-se que houve diferença entre tempo de treinamento para o experimento que utilizou a base completa com 77 atributos, e a base com os atributos selecionados pelo IG com 56 atributos, a melhoria foi de 11,4%. Quando o primeiro experimento é comparado com aqueles que utilizaram o método ensemble de seleção de atributos, IG + SBFE e IG + SFFS, ambos conjuntos com 25 atributos, vemos que existe uma melhora de 33,7% e 38,9%, respectivamente. O menor tempo para realização do treino do algoritmo classificador foi utilizando o método de seleção de atributos IG + (SFFS ∩ SBFE), com um subconjunto de 15 atributos, diminuindo o tempo de treinamento em mais de 45% em relação aos métodos que utilizam os algoritmos *wrappers* de forma separada e em 66,7% menos tempo em relação ao treinamento utilizando a base completa.

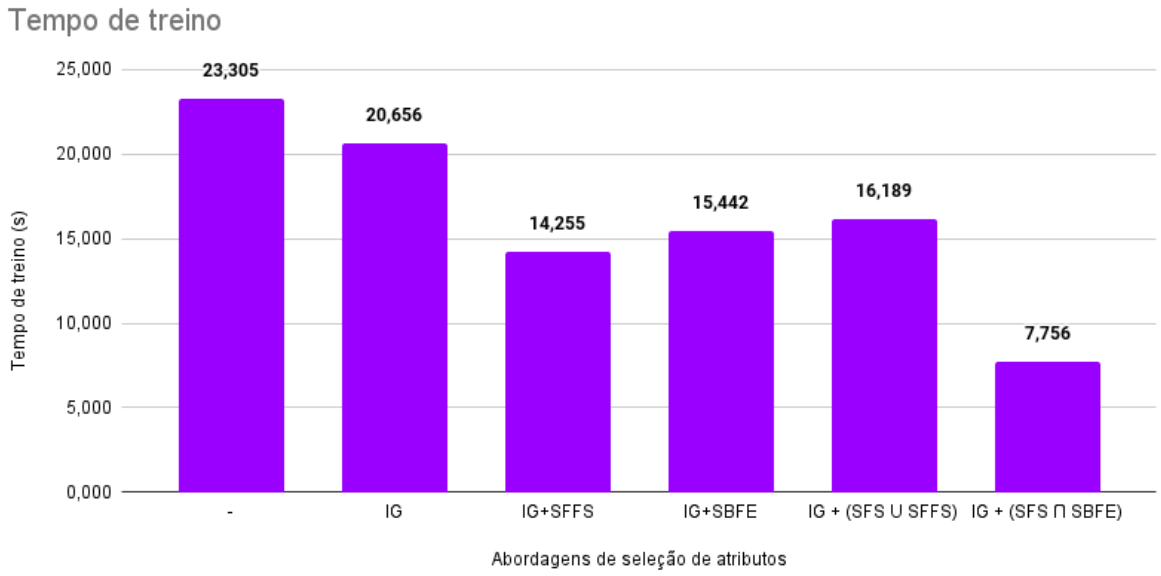

Figura 7.2: Gráfico do tempo de treino (s) versus abordagem de seleção de atributos.
Fonte: O autor (2021).

Desse modo, é possível concluir que a quantidade de atributos pode influenciar diretamente no tempo necessário para construção e treino do classificador ET.

No processo de detecção de intrusão o tempo de treino possui uma grande relevância. A etapa de treinamento é onde os modelos de detecção são construídos para então serem utilizados na classificação. Sendo assim, uma etapa de treino com um tempo reduzido tem vantagens em relação a uma etapa de treino de maior duração, principalmente durante as atualizações dos modelos de detecção. O processo de treinamento ocorre de forma separada do processo de análise da rede e classificação dos eventos. Devido as alterações que ocorrem nas redes, é necessário atualizar o modelo periodicamente, de forma a sempre manter o classificador do método SDI atualizado conforme o fluxo de eventos.

A dinamicidade e elasticidade presente em redes IoT contempla a inserção e remoção de diversos nós, ou dispositivos. Consequentemente, o comportamento da rede monitorada pode mudar e, desse modo, o SDI precisa acompanhar tais variações. É necessário, portanto, definir o momento no qual o algoritmo classificador precisa ser atualizado, realizando um novo treinamento e gerando uma nova árvore. Portanto, quanto menor o tempo necessário para realizar o treinamento do modelo, menor o tempo que o SDI ficará aguardando o novo modelo ou utilizando um modelo desatualizado.

O tempo de teste, por outro lado, indica o tempo necessário para a análise e classificação do tráfego, desse modo é bastante importante. O tempo de teste, em segundos, é apresentado na Tabela 7.8 e ilustrada no gráfico da Figura 7.3. Destaca-se que o tempo apresentado indica o tempo necessário para a classificação de todo o conjunto de dados de teste, não apenas uma instância.

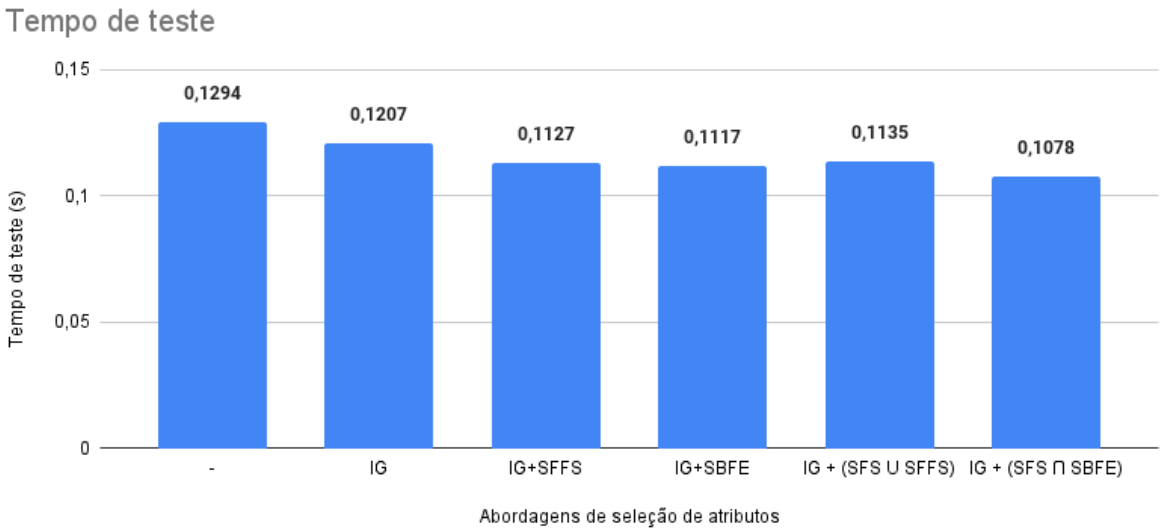

Figura 7.3: Gráfico do tempo de teste (s) versus abordagem de seleção de atributos.
Fonte: O autor (2021).

Neste caso, observa-se que o tempo de teste é reduzido à medida que ocorre a redução de atributos. É interessante observar que o pior tempo de teste foi da abordagem utilizando a base completa, que precisou de 1,1294 segundos para classificar o conjunto de dados. Já o melhor tempo de teste foi da abordagem IG + (SFFS ∩ SBFE) com 15 atributos, diminuindo o tempo de teste para 0,1078 segundos, reduzindo, portanto em 16,7% o tempo para teste, desse modo, tornando mais plausível e possível aplicações em tempo real.

Portanto, concluímos que a abordagem proposta IG + (SFFS ∩ SBFE), que considera a seleção de atributos com IG e a intersecção das seleções SFFS e SBFE, apresentou, juntamente com as outras abordagens *wrapper*, os melhores resultados em relação as métricas de detecção, conforme discutido nos testes estatísticos. Além disso, a abordagem proposta IG + (SFFS ∩ SBFE) foi capaz de alcançar os melhores tempos de treino e teste dentre todas as abordagens,

indicando desse modo, gerar um modelo de classificação baseado em Extra Tree com um menor custo computacional em relação as demais.

Capítulo 8

Conclusão

Com a exponencial evolução e expansão da rede mundial de computadores e das informações manipuladas em sistemas, diversas dificuldades para manter a segurança destes surgiram. Com a finalidade de proteger os sistemas, os Sistemas de Detecção de Intrusão (SDI) são de grande valia para as organizações, evitando assim que ocorram intrusões na rede.

Abordagens de detecção de intrusão baseados em Inteligência Artificial possuem maior adaptabilidade em relação aos dados analisados. No entanto, o custo computacional pode ser considerado proibitivo em arquiteturas que possuem poucos recursos, como o caso de arquiteturas *Internet of Things* (IoT). Este trabalho busca uma abordagem de classificação baseada em aprendizado de máquina e um método *ensemble* de seleção de atributos.

No presente trabalho a partir da definição da base CICIDS2017 foi realizado o pré-processamento, contemplando a etapa de normalização e limpeza de dados. Em seguida a base de eventos foi submetida a diversos experimentos, variando a arquitetura do método *ensemble* de seleção de atributos e o combinador dos métodos. O subconjunto resultante de cada experimento seletor de atributos foi de grande importância visto que cada um deste foi submetido ao classificador *Extra Tree*, denotando assim qual a contribuição para as métricas de classificação e para o tempo de treinamento e teste. Este processo resultou em seis subconjuntos, o primeiro sem seleção de atributos (77 atributos), o segundo selecionado pelo método *Filter* IG (*Information Gain*) (56 atributos), o terceiro e quarto subconjunto utilizaram a base resultante da seleção realizada por IG e foram selecionados pelos métodos *Wrapper* SFFS (*Sequential Forward Feature Selection*) e SBFE (*Sequential Backward Feature Elimination*) (25 atributos). Já o quinto e sexto subconjunto de atributos são resultantes da combinação entre a união e intersecção dos subconjuntos gerados pelos algoritmos SFFS e SFBE (com 35 e 15 atributos, consecutivamente).

Ao realizar a comparação das métricas obtidas nos experimentos observa-se que a abordagem proposta IG + (SFFS $\cap$ SBFE), que considera a seleção de atributos com IG e a intersecção das seleções SFFS e SBFE, apresentou, juntamente com as outras abordagens *wrapper*, os melhores resultados em relação as métricas de detecção, conforme discutido nos testes estatísticos. Esta abordagem também foi capaz de alcançar os melhores resultados relacionados ao tempo de treino e de teste dentre todas as abordagens. Ela foi capaz de diminuir o tempo de treinamento

em mais de 45% em relação aos métodos que utilizam os algoritmos *wrappers* de forma separada e em 66,7% menos tempo em relação ao treinamento utilizando a base completa. Também alcançou uma redução de 16,7% no tempo de teste em relação a base completa, indicando gerar um modelo de classificação baseado em Extra Tree com um menor custo computacional em relação as demais. Concluindo assim que a utilização de um menor número de atributos devidamente selecionados pode melhorar a *performance* e a quantidade de recursos utilizados pelos dispositivos, sem detrair as métricas de classificação das classes.

Utilizando a arquitetura e os métodos propostos neste trabalho, assim como suas contribuições, novos trabalhos podem ser executados com a finalidade de evoluir e explorar o presente estudo, citam-se algumas sugestões:

- Investigar e propor métodos de contramedidas de forma automatizada e específica para cada classe intrusiva detectada;
- Realizar a execução de cada abordagem, classificando em tempo real os eventos com a dimensionalidade diminuída de acordo com cada subconjunto selecionado pelas abordagens, para verificar o desempenho em relação à classificação de eventos totais em seus atributos.
- Propor novas abordagens do método de seleção de atributos, variando a combinação entre os métodos da abordagem *filter* e *wrapper*.
- Realizar a execução da abordagem proposta em ambiente real, colhendo os eventos de dispositivos IoT e os processando na camada de *fog*.
- Realizar a análise da robustez dos sistemas, utilizando os mesmos dados de entrada, mas corrompendo-os com ruídos, como o branco, e verificar as métricas de avaliação.

Referências Bibliográficas

Aazam, M. & Huh, E.-N. (2014). Fog computing and smart gateway based communication for cloud of things, *2014 International Conference on Future Internet of Things and Cloud*, IEEE, pp. 464–470. Citado na página 26.

Aha, D. W., Kibler, D. & Albert, M. K. (1991). Instance-based learning algorithms, *Machine learning* **6**(1): 37–66. Citado na página 42.

Ahmad, R. & Alsmadi, I. (2021). Machine learning approaches to iot security: A systematic literature review, *Internet of Things* **14**: 100365.
URL: *https://www.sciencedirect.com/science/article/pii/S2542660521000093* Citado na página 28.

Ahmed, M., Mahmood, A. N. & Hu, J. (2016). A survey of network anomaly detection techniques, *Journal of Network and Computer Applications* **60**: 19–31. Citado na página 30.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications, *IEEE communications surveys & tutorials* **17**(4): 2347–2376. Citado 3 vezes nas páginas 24, 25 e 26.

Alaba, F. A., Othman, M., Hashem, I. A. T. & Alotaibi, F. (2017). Internet of things security: A survey, *Journal of Network and Computer Applications* **88**: 10–28. Citado 2 vezes nas páginas 16 e 29.

Almeida, T. B. et al. (2018). *Seleção de atributos usando abordagem wrapper para classificação hierárquica multirrótulo*, Master's thesis, Universidade Tecnológica Federal do Paraná. Citado 2 vezes nas páginas 35 e 39.

Alrawais, A., Alhothaily, A., Hu, C. & Cheng, X. (2017). Fog computing for the internet of things: Security and privacy issues, *IEEE Internet Computing* **21**(2): 34–42. Citado 3 vezes nas páginas 16, 26 e 29.

Álvarez-Estévez, D., Sánchez-Maroño, N., Alonso-Betanzos, A. & Moret-Bonillo, V. (2011). Reducing dimensionality in a database of sleep eeg arousals, *Expert Systems with Applications* **38**(6): 7746–7754. Citado na página 62.

Atzori, L., Iera, A. & Morabito, G. (2010). The internet of things: A survey, *Computer networks* **54**(15): 2787–2805. Citado 2 vezes nas páginas 15 e 23.

Aversano, L., Bernardi, M. L., Cimitile, M. & Pecori, R. (2021). A systematic review on deep learning approaches for iot security, *Computer Science Review* **40**: 100389.
URL: *https://www.sciencedirect.com/science/article/pii/S1574013721000290* Citado na página 28.

Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy, *Technical report*, Technical report. Citado 2 vezes nas páginas 29 e 30.

Azhagusundari, B., Thanamani, A. S. et al. (2013). Feature selection based on information

gain, *International Journal of Innovative Technology and Exploring Engineering (IJITEE)* **2**(2): 18–21. Citado na página 60.

Azwar, H., Murtaz, M., Siddique, M. & Rehman, S. (2018). Intrusion detection in secure network for cybersecurity systems using machine learning and data mining, *2018 IEEE 5th international conference on engineering technologies and applied sciences (ICETAS)*, IEEE, pp. 1–9. Citado na página 54.

Bace, R. & Mell, P. (2001). Nist special publication on intrusion detection systems, *Technical report*, BOOZ-ALLEN AND HAMILTON INC MCLEAN VA. Citado na página 30.

Bakhareva, N., Shukhman, A., Matveev, A., Polezhaev, P., Ushakov, Y. & Legashev, L. (2019). Attack detection in enterprise networks by machine learning methods, *2019 international Russian automation conference (RusAutoCon)*, IEEE, pp. 1–6. Citado na página 63.

Baranauskas, J. A. et al. (2002). Extração automática de conhecimento por múltiplos indutores. Citado 3 vezes nas páginas 35, 36 e 38.

Bartlett, P., Freund, Y., Lee, W. S. & Schapire, R. E. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods, *The annals of statistics* **26**(5): 1651–1686. Citado na página 40.

Belanche, L. A. & González, F. F. (2011). Review and evaluation of feature selection algorithms in synthetic problems, *arXiv preprint arXiv:1101.2320* . Citado na página 60.

Blum, A. L. & Langley, P. (1997). Selection of relevant features and examples in machine learning, *Artificial intelligence* **97**(1-2): 245–271. Citado 2 vezes nas páginas 35 e 36.

Bolón-Canedo, V. & Alonso-Betanzos, A. (2019). Ensembles for feature selection: A review and future trends, *Information Fusion* **52**: 1–12. Citado na página 61.

Bolón-Canedo, V., Sánchez-Maroño, N. & Alonso-Betanzos, A. (2015). *Feature selection for high-dimensional data*, Springer. Citado 3 vezes nas páginas 35, 36 e 60.

Bonomi, F., Milito, R., Zhu, J. & Addepalli, S. (2012). Fog computing and its role in the internet of things, *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16. Citado 3 vezes nas páginas 15, 25 e 26.

Borgia, E. (2014). The internet of things vision: Key features, applications and open issues, *Computer Communications* **54**: 1–31. Citado 2 vezes nas páginas 22 e 23.

Botta, A., De Donato, W., Persico, V. & Pescapé, A. (2016). Integration of cloud computing and internet of things: a survey, *Future generation computer systems* **56**: 684–700. Citado 3 vezes nas páginas 15, 23 e 25.

Bowyer, K. W., Chawla, N. V., Hall, L. O. & Kegelmeyer, W. P. (2011). SMOTE: synthetic minority over-sampling technique, *CoRR* **abs/1106.1813**.
URL: *http://arxiv.org/abs/1106.1813* Citado na página 58.

Breiman, L. (1996). Bagging predictors, *Machine learning* **24**(2): 123–140. Citado na página 40.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984). Classification and regression trees–crc press, *Boca Raton, Florida* . Citado na página 45.

Camhi, J. (2015). Former cisco ceo john chambers predicts 500 billion connected devices by 2025, *Business Insider* . Citado na página 23.

Cardoso, J. V., Sampaio, H. V., Souza, C. A. & Westphall, C. B. (2019). DoS attack detection and prevention in fog-based intelligent environments, *Brazilian Journal of Development* **5**(11): 23934–23956. Citado na página 28.

CERT.br (2020). Incidentes reportados ao cert.br.
URL: *https://cert.br/stats/incidentes/* Citado na página 18.

Coutinho, A., Carneiro, E. O. & Greve, F. G. P. (2016). Computação em névoa: Conceitos, aplicações e desafios, *Minicursos do XXXIV SBRC* pp. 266–315. Citado na página 25.

Dahlqvist, F., Patel, M., Rajko, A. & Shulman, J. (2019). Growing opportunities in the internet of things, *McKinsey & Company* . Citado na página 24.

Das, S. (2001). Filters, wrappers and a boosting-based hybrid for feature selection, *Icml*, Vol. 1, Citeseer, pp. 74–81. Citado na página 59.

Dastjerdi, A. V. & Buyya, R. (2016). Fog computing: Helping the internet of things realize its potential, *Computer* **49**(8): 112–116. Citado 2 vezes nas páginas 18 e 26.

Dietterich, T. G. (2000a). Ensemble methods in machine learning, *Multiple Classifier Systems*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–15. Citado na página 40.

Dietterich, T. G. (2000b). Ensemble methods in machine learning, *International workshop on multiple classifier systems*, Springer, pp. 1–15. Citado na página 49.

Dunn, O. J. (1961). Multiple comparisons among means, *Journal of the American statistical association* **56**(293): 52–64. Citado na página 68.

Faker, O. & Dogdu, E. (2019). Intrusion detection using big data and deep learning techniques, *Proceedings of the 2019 ACM Southeast Conference*, pp. 86–93. Citado 2 vezes nas páginas 51 e 53.

Ferrero, C. A. (2009). *Algoritmo kNN para previsão de dados temporais: funções de previsão e critérios de seleção de vizinhos próximos aplicados a variáveis ambientais em limnologia*, PhD thesis, Universidade de São Paulo. Citado na página 42.

Fisher, R. (1954). The analysis of variance with various binomial transformations, *Biometrics* **10**(1): 130–139. Citado na página 67.

Galvão, S. D. C. d. O. et al. (2007). A seleção de atributos e o aprendizado supervisionado de redes bayesianas no contexto da mineração de dados. Citado na página 36.

Garg, S., Kaur, K., Batra, S., Aujla, G. S., Morgan, G., Kumar, N., Zomaya, A. Y. & Ranjan, R. (2020). En-abc: An ensemble artificial bee colony based anomaly detection scheme for cloud environment, *Journal of Parallel and Distributed Computing* **135**: 219–233. Citado na página 55.

Geurts, P., Ernst, D. & Wehenkel, L. (2006). Extremely randomized trees, *Machine learning* **63**(1): 3–42. Citado 2 vezes nas páginas 45 e 46.

Goodrich, M. T. & Tamassia, R. (2013). *Introdução à segurança de computadores*, Bookman. Citado 2 vezes nas páginas 28 e 65.

Guerra-Manzanares, A., Bahsi, H. & Nõmm, S. (2019). Hybrid feature selection models for machine learning based botnet detection in iot networks, *2019 International Conference on Cyberworlds (CW)*, IEEE, pp. 324–327. Citado 3 vezes nas páginas 49, 53 e 61.

Guttman, B. & Roback, E. A. (1995). *An introduction to computer security: the NIST handbook*,

Diane Publishing. Citado na página 27.

Hoque, N., Singh, M. & Bhattacharyya, D. K. (2018). Efs-mi: an ensemble feature selection method for classification, *Complex & Intelligent Systems* **4**(2): 105–118. Citado 2 vezes nas páginas 20 e 55.

Hota, H. & Shrivas, A. K. (2014). Decision tree techniques applied on nsl-kdd data and its comparison with various feature selection techniques, *Advanced Computing, Networking and Informatics-Volume 1*, Springer, pp. 205–211. Citado 2 vezes nas páginas 34 e 37.

Huang, J., Cai, Y. & Xu, X. (2007). A hybrid genetic algorithm for feature selection wrapper based on mutual information, *Pattern recognition letters* **28**(13): 1825–1844. Citado na página 60.

Idhammad, M., Afdel, K. & Belouch, M. (2018). Semi-supervised machine learning approach for ddos detection, *Applied Intelligence* **48**(10): 3193–3208. Citado 2 vezes nas páginas 50 e 53.

Indre, I. & Lemnaru, C. (2016). Detection and prevention system against cyber attacks and botnet malware for information systems and internet of things, *2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP)*, IEEE, pp. 175–182. Citado 2 vezes nas páginas 50 e 53.

Iorga, M., Feldman, L., Barton, R., Martin, M., Goren, N. & Mahmoudi, C. (2017). The nist definition of fog computing, *Technical report*, National Institute of Standards and Technology. Citado na página 27.

Jalali, M. S., Kaiser, J. P., Siegel, M. & Madnick, S. (2019). The internet of things promises new benefits and risks: A systematic analysis of adoption dynamics of iot products, *IEEE Security & Privacy* **17**(2): 39–48. Citado 2 vezes nas páginas 16 e 29.

Jing, Q., Vasilakos, A. V., Wan, J., Lu, J. & Qiu, D. (2014). Security of the internet of things: perspectives and challenges, *Wireless Networks* **20**(8): 2481–2501. Citado 2 vezes nas páginas 16 e 29.

John, G. H., Kohavi, R. & Pfleger, K. (1994). Irrelevant features and the subset selection problem, *Machine Learning Proceedings 1994*, Elsevier, pp. 121–129. Citado na página 39.

Kantardzic, M. (2011). *Data mining: concepts, models, methods, and algorithms*, John Wiley & Sons. Citado na página 33.

Karegowda, A. G., Manjunath, A. & Jayaram, M. (2010). Comparative study of attribute selection using gain ratio and correlation based feature selection, *International Journal of Information Technology and Knowledge Management* **2**(2): 271–277. Citado na página 37.

Kaur, K. & Mittal, S. (2020). Classification of mammography image with cnn-rnn based semantic features and extra tree classifier approach using lstm, *Materials Today: Proceedings* . Citado na página 46.

Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection, *Ijcai*, Vol. 14, Montreal, Canada, pp. 1137–1145. Citado na página 64.

Kolias, C., Kambourakis, G., Stavrou, A. & Voas, J. (2017). Ddos in the iot: Mirai and other botnets, *Computer* **50**(7): 80–84. Citado na página 17.

Kruskal, W. H. & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis, *Journal of the American statistical Association* **47**(260): 583–621. Citado na página 67.

Kurgan, L. A. & Musilek, P. (2006). A survey of knowledge discovery and data mining process models, *Knowledge Engineering Review* **21**(1): 1–24. Citado na página 33.

Langley, P. & Iba, W. (1993). Average-case analysis of a nearest neighbor algorithm, *IJCAI*, Vol. 93, Citeseer, pp. 889–889. Citado na página 60.

Lee, H. D. (2005). *Seleção de atributos importantes para a extração de conhecimento de bases de dados*, PhD thesis, Universidade de São Paulo. Citado 2 vezes nas páginas 36 e 38.

Li, T., Zhang, C. & Ogihara, M. (2004). A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression, *Bioinformatics* **20**(15): 2429–2437. Citado na página 60.

Lima, I. V. M. d. et al. (2005). Uma abordagem simplificada de detecção de intrusão baseada em redes neurais artificiais. Citado na página 30.

Liu, H. & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey, *Applied Sciences* **9**(20).
URL: *https://www.mdpi.com/2076-3417/9/20/4396* Citado na página 65.

Loganathan, G. (2018). Real-time intrusion detection using multidimensional sequence-to-sequence machine learning and adaptive stream processing. Citado na página 54.

Machado, R. B. et al. (2005). Uma abordagem de detecção de intrusão baseada em sistemas imunológicos artificiais e agentes móveis. Citado na página 30.

Maier, O., Wilms, M., von der Gablentz, J., Krämer, U. M., Münte, T. F. & Handels, H. (2015). Extra tree forests for sub-acute ischemic stroke lesion segmentation in mr sequences, *Journal of neuroscience methods* **240**: 89–100. Citado na página 45.

Manzoor, I., Kumar, N. et al. (2017). A feature reduced intrusion detection system using ann classifier, *Expert Systems with Applications* **88**: 249–257. Citado na página 37.

Mejía-Lavalle, M., Sucar, E. & Arroyo, G. (2006). Feature selection with a perceptron neural net, *Proceedings of the international workshop on feature selection for data mining*, pp. 131–135. Citado na página 60.

Mell, P., Grance, T. et al. (2011). The nist definition of cloud computing. Citado 2 vezes nas páginas 24 e 25.

Mitchell, R. & Chen, I.-R. (2014). A survey of intrusion detection techniques for cyber-physical systems, *ACM Computing Surveys (CSUR)* **46**(4): 1–29. Citado na página 30.

Mollick, E. (2006). Establishing moore's law, *IEEE Annals of the History of Computing* **28**(3): 62–75. Citado na página 32.

Monard, M. C. & Baranauskas, J. A. (2003). Indução de regras e árvores de decisão, *Sistemas Inteligentes-Fundamentos e Aplicações* **1**: 115–139. Citado 3 vezes nas páginas 43, 44 e 45.

Moore, G. (1965). Moore's law, *Electronics Magazine* **38**(8): 114. Citado na página 32.

Muhammad, F., Anjum, W. & Mazhar, K. S. (2015). A critical analysis on the security concerns of internet of things (iot), *International Journal of Computer Applications* **111**(7). Citado na página 28.

Ni, J., Zhang, K., Lin, X. & Shen, X. (2018). Securing fog computing for internet of things applications: Challenges and solutions, *IEEE Communications Surveys & Tutorials* . Citado na página 17.

Olusola, A. A., Oladele, A. S. & Abosede, D. O. (2010). Analysis of kdd'99 intrusion detection dataset for selection of relevance features, *Proceedings of the world congress on engineering and computer science*, Vol. 1, WCECS, pp. 20–22. Citado na página 50.

Osanaiye, O., Cai, H., Choo, K.-K. R., Dehghantanha, A., Xu, Z. & Dlodlo, M. (2016). Ensemble-based multi-filter feature selection method for ddos detection in cloud computing, *EURASIP Journal on Wireless Communications and Networking* **2016**(1): 1–10. Citado 5 vezes nas páginas 19, 20, 50, 53 e 59.

Panigrahi, R. & Borah, S. (2018). A detailed analysis of cicids2017 dataset for designing intrusion detection systems, *International Journal of Engineering & Technology* **7**(3.24): 479–482. Citado na página 48.

Quinlan, J. R. (1986). Induction of decision trees, *Machine learning* **1**(1): 81–106. Citado na página 37.

Quinlan, J. R. (1993). C4. 5: Programming for machine learning, *Morgan Kauffmann* **38**(48): 49. Citado na página 38.

Rokach, L. (2016). Decision forest: Twenty years of research, *Information Fusion* **27**: 111–125. Citado 4 vezes nas páginas 42, 43, 44 e 45.

Russell, S. & Norvig, P. (2002). Artificial intelligence: a modern approach. Citado 2 vezes nas páginas 32 e 43.

Saeys, Y., Abeel, T. & Van de Peer, Y. (2008). Robust feature selection using ensemble feature selection techniques, *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, pp. 313–325. Citado 2 vezes nas páginas 49 e 53.

Sharafaldin, I., Lashkari, A. H. & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization., *ICISSP*, pp. 108–116. Citado 3 vezes nas páginas 51, 54 e 63.

Sharbaf, F. V., Mosafer, S. & Moattar, M. H. (2016). A hybrid gene selection approach for microarray data classification using cellular learning automata and ant colony optimization, *Genomics* **107**(6): 231–238. Citado na página 59.

Shon, T. & Moon, J. (2007). A hybrid machine learning approach to network anomaly detection, *Information Sciences* **177**(18): 3799–3821. Citado na página 17.

Souza, C. A. d. et al. (2018). Método híbrido de detecção de intrusão aplicando inteligência artificial. Citado 2 vezes nas páginas 41 e 43.

Stallings, W., Brown, L., Bauer, M. D. & Bhattacharjee, A. K. (2012). *Computer security: principles and practice*, Pearson Education Upper Saddle River, NJ, USA. Citado 3 vezes nas páginas 27, 29 e 41.

Staudemeyer, R. C. & Omlin, C. W. (2014). Extracting salient features for network intrusion detection using machine learning methods, *South African computer journal* **52**(1): 82–96. Citado na página 54.

Stiawan, D., Idris, M. Y. B., Bamhdi, A. M., Budiarto, R. et al. (2020). Cicids-2017 dataset feature analysis with information gain for anomaly detection, *IEEE Access* **8**: 132911–

132921. Citado 3 vezes nas páginas 48, 53 e 59.

Sundhari, S. S. (2011). A knowledge discovery using decision tree by gini coefficient, *2011 International Conference on Business, Engineering and Industrial Applications*, IEEE, pp. 232–235. Citado 2 vezes nas páginas 45 e 62.

Tanaka, H. & Yamaguchi, S. (2017). On modeling and simulation of the behavior of iot malwares mirai and hajime, *2017 IEEE International Symposium on Consumer Electronics (ISCE)*, pp. 56–60. Citado na página 17.

Tanenbaum, A. S. & Wetherall, D. (2003). Redes de computadores. [sl]. Citado na página 27.

Tavallaee, M., Bagheri, E., Lu, W. & Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set, *2009 IEEE symposium on computational intelligence for security and defense applications*, IEEE, pp. 1–6. Citado na página 54.

Tesfahun, A. & Bhaskari, D. L. (2013). Intrusion detection using random forests classifier with smote and feature reduction, *2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies*, IEEE, pp. 127–132. Citado na página 60.

Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y. & Lin, W.-Y. (2009). Intrusion detection by machine learning: A review, *expert systems with applications* **36**(10): 11994–12000. Citado na página 17.

Tsai, C.-F., Hsu, Y.-F. & Yen, D. C. (2014). A comparative study of classifier ensembles for bankruptcy prediction, *Applied Soft Computing* **24**: 977–984. Citado na página 40.

Vasilomanolakis, E., Karuppayah, S., Mühlhäuser, M. & Fischer, M. (2015). Taxonomy and survey of collaborative intrusion detection, *ACM Computing Surveys (CSUR)* **47**(4): 1–33. Citado na página 30.

Vinayakumar, R., Alazab, M., Soman, K., Poornachandran, P., Al-Nemrat, A. & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system, *IEEE Access* **7**: 41525–41550. Citado na página 63.

Wang, S.-C. (2003). Artificial neural network, *Interdisciplinary computing in java programming*, Springer, pp. 81–100. Citado na página 41.

Yoo, Y., Henfridsson, O. & Lyytinen, K. (2010). Research commentary—the new organizing logic of digital innovation: an agenda for information systems research, *Information systems research* **21**(4): 724–735. Citado na página 24.

Yulianto, A., Sukarno, P. & Suwastika, N. A. (2019). Improving adaboost-based intrusion detection system (ids) performance on cic ids 2017 dataset, *Journal of Physics: Conference Series*, Vol. 1192, IOP Publishing, p. 012018. Citado 2 vezes nas páginas 51 e 53.

Zainal, A., Maarof, M. A., Shamsuddin, S. M. et al. (2009). Ensemble classifiers for network intrusion detection system, *Journal of Information Assurance and Security* **4**(3): 217–225. Citado na página 35.

Zhou, Y., Cheng, G., Jiang, S. & Dai, M. (2020). Building an efficient intrusion detection system based on feature selection and ensemble classifier, *Computer Networks* p. 107247. Citado 2 vezes nas páginas 34 e 35.

Apêndice A

Tabela CICIDS 2017

Tabela A.1: Descrição dos atributos existentes na base CICIDS 2017.

Atributo	Descrição
Flow ID	Identificador do fluxo
Source IP	IP origem
Source Port	Porta origem
Destination IP	IP destino
Destination Port	Porta destino
Protocol	Tipo do protocolo do fluxo
Timestamp	Data e hora do início do fluxo
Flow Duration	Duração do fluxo em milissegundos
Total Fwd Packets	Total de pacotes *forward*
Total backward Packets	Total de pacotes *backward*
Total Length of Fwd Packets	Tamanho total dos pacotes *forward*
Total Length of Bwd Packets	Tamanho total dos pacotes *backward*
Fwd Packet Length Max	Tamanho do maior pacote *forward*
Fwd Packet Length Min	Tamanho do menor pacote *forward*
Fwd Packet Length Mean	Média de tamanho dos pacotes *forward*
Fwd Packet Length Std	Desvio padrão do tamanho dos pacotes no sentido *forward*
Bwd Packet Length Max	Tamanho do maior pacote *backward*
Bwd Packet Length Min	Tamanho do menor pacote *backward*
Bwd Packet Length Mean	Média de tamanho dos pacotes *backward*
Bwd Packet Length Std	Desvio padrão do tamanho dos pacotes no sentido *backward*
Flow Bytes s	Número de *bytes* por segundo no fluxo
Flow Packets s	Número de pacotes por segundo no fluxo
Flow IAT Mean	Média do IAT no fluxo
Flow IAT Std	Desvio padrão do IAT de todos os pacotes do fluxo
Flow IAT Max	Valor do maior IAT do fluxo
Flow IAT Min	Valor do menor IAT do fluxo
Fwd IAT Total	IAT total sentido *forward*
Fwd IAT Mean	Média do IAT no sentido *forward*

Fwd IAT Std	Desvio padrão do IAT de todos os pacotes do fluxo
Fwd IAT Max	Maior valor de IAT no sentido *forward*
Fwd IAT Min	Menor valor de IAT no sentido *forward*
Bwd IAT Total	IAT total no sentido *backward*
Bwd IAT Mean	Média do IAT no sentido *backward*
Bwd IAT Std	Desvio padrão do IAT dos pacotes *backward*
Bwd IAT Max	Maior valor de IAT no sentido *backward*
Bwd IAT Min	Menor valor de IAT no sentido *backward*
Fwd PSH Flags	Contador da *flag* PSH em pacotes no sentido *forward*
Bwd PSH Flags	Contador da *flag* PSH em pacotes no sentido *backward*
Fwd URG Flags	Contador da *flag* URG em pacotes no sentido *forward*
Bwd URG Flags	Contador da *flag* URG em pacotes no sentido *backward*
Fwd Header Length	Total de *bytes* usados para o *header* no sentido *forward*
Bwd Header Length	Total de *bytes* usados para o *header* no sentido *backward*
Fwd Packets s	Número de pacotes por segundo no sentido *forward*
Bwd Packets s	Número de pacotes por segundo no sentido *backward*
Min Packet Length	Tamanho do menor pacote
Max Packet Length	Tamanho do maior pacote
Packet Length Mean	Média dos tamanhos dos pacotes no fluxo
Packet Length Std	Desvio padrão dos tamanhos dos pacotes no fluxo
Packet Length Variance	Variância dos tamanhos dos pacotes no fluxo
FIN Flag Count	Contador da *flag* FIN nos pacotes do fluxo
SYN Flag Count	Contador da *flag* SYN nos pacotes do fluxo
RST Flag Count	Contador da *flag* RST nos pacotes do fluxo
PSH Flag Count	Contador da *flag* PSH nos pacotes do fluxo
ACK Flag Count	Contador da *flag* ACK nos pacotes do fluxo
URG Flag Count	Contador da *flag* URG nos pacotes do fluxo
CWR Flag Count	Contador da *flag* CWR nos pacotes do fluxo
ECE Flag Count	Contador da *flag* ECE nos pacotes do fluxo
Down Up Ratio	Proporção entre *download* a *upload*
Average Packet Size	Média do tamanho dos pacotes
Avg Fwd Segment Size	Média de tamanho de segmentos no sentido *forward*
Avg Bwd Segment Size	Média de tamanho de segmentos no sentido *backward*
Fwd Avg Bytes Bulk	Média de *bytes* de transmissão em massa no sentido *forward*
Fwd Avg Packets Bulk	Média de pacotes de transmissão em massa no sentido *forward*
Fwd Avg Bulk Rate	Média de taxa de transmissão em massa no sentido *forward*
Bwd Avg Bytes Bulk	Média de *bytes* de transmissão em massa no sentido *backward*
Bwd Avg Packets Bulk	Média de pacotes de transmissão em massa no sentido *backward*
Bwd Avg Bulk Rate	Média de taxa de transmissão em massa no sentido *backward*

Subflow Fwd Packets	Número de pacotes em um *subflow* no sentido *forward*
Subflow Fwd Bytes	Número de *bytes* em um *subflow* no sentido *forward*
Subflow Bwd Packets	Número de pacotes em um *subflow* no sentido *backward*
Subflow Bwd Bytes	Número de *bytes* em um *subflow* no sentido *backward*
Init_Win_bytes_forward	Total de *bytes* enviados na janela inicial no sentido *forward*
Init_Win_bytes_backward	Total de *bytes* enviados na janela inicial no sentido *backward*
Act_Data_Pkt_Fwd	Pacotes com ao menos 1 byte de *payload* no sentido *forward*
Min_Seg_Size_Forward	Tamanho mínimo de segmento observado no sentido *forward*
Active Mean	Tempo médio de atividade dos fluxos
Active Std	Devio padrão do tempo de atividade dos fluxos
Active Max	Tempo máximo de atividade dos fluxos
Active Min	Tempo mínimo de atividade dos fluxos
Idle Mean	Tempo médio de ociosidade dos fluxos
Idle Std	Devio padrão do tempo de ociosidade dos fluxos
Idle Max	Tempo máximo de ociosidade dos fluxos
Idle Min	Tempo mínimo de ociosidade dos fluxos
Label	Classe

Apêndice B

Resultados individuais por classe

Tabela B.1: Métricas de avaliação para a classe Benigna.

	Precisão	Recall	F1-Score
-	99,90%	99,88%	99,89%
IG	99,89%	99,87%	99,88%
IG + SFFS	99,96%	99,90%	99,93%
IG + SBFE	99,97%	99,90%	99,94%
IG+(SFFS ∪ SBFE)	99,96%	99,90%	99,93%
IG+(SFFS ∩ SBFE)	99,97%	99,90%	99,93%

Tabela B.2: Métricas de avaliação para a classe de *DoS/DDoS*.

	Precisão	Recall	F1-Score
-	99,74%	99,73%	99,74%
IG	99,71%	99,68%	99,69%
IG + SFFS	99,89%	99,88%	99,89%
IG + SBFE	99,90%	99,90%	99,90%
IG+(SFFS ∪ SBFE)	99,89%	99,87%	99,88%
IG+(SFFS ∩ SBFE)	99,89%	99,89%	99,89%

Tabela B.3: Métricas de avaliação para a classe de *PortScan*.

	Precisão	Recall	F1-Score
-	99,40%	99,51%	99,45%
IG	99,40%	99,51%	99,45%
IG + SFFS	99,39%	99,89%	99,64%
IG + SBFE	99,39%	99,89%	99,64%
IG+(SFFS ∪ SBFE)	99,39%	99,89%	99,64%
IG+(SFFS ∩ SBFE)	99,37%	99,87%	99,62%

Tabela B.4: Métricas de avaliação para a classe de *Brute Force*.

	Precisão	Recall	F1-Score
-	99,82%	99,74%	99,78%
IG	99,80%	99,75%	99,78%
IG + SFFS	99,98%	99,84%	99,91%
IG + SBFE	99,98%	99,87%	99,92%
IG+(SFFS ∪ SBFE)	99,98%	99,84%	99,91%
IG+(SFFS ∩ SBFE)	99,94%	99,84%	99,89%

Tabela B.5: Métricas de avaliação para a classe de WebAttack.

	Precisão	Recall	F1-Score
-	97,93%	97,66%	97,77%
IG	96,37%	97,30%	96,80%
IG + SFFS	91,71%	98,42%	94,90%
IG + SBFE	90,68%	98,52%	94,39%
IG+(SFFS ∪ SBFE)	91,38%	98,45%	94,73%
IG+(SFFS ∩ SBFE)	90,84%	98,34%	94,39%

Tabela B.6: Métricas de avaliação para a classe de BotNet.

	Precisão	Recall	F1-Score
-	63,02%	71,98%	66,94%
IG	63,56%	72,84%	67,68%
IG + SFFS	71,63%	92,05%	80,42%
IG + SBFE	71,66%	91,56%	80,27%
IG+(SFFS ∪ SBFE)	71,37%	91,53%	80,09%
IG+(SFFS ∩ SBFE)	71,01%	91,87%	79,97%

Tabela B.7: Métricas de avaliação para a classe de Infiltration.

	Precisão	Recall	F1-Score
-	18,89%	18,89%	18,89%
IG	21,11%	21,11%	21,11%
IG + SFFS	18,89%	18,89%	18,89%
IG + SBFE	22,22%	22,22%	22,22%
IG+(SFFS ∪ SBFE)	18,89%	18,89%	18,89%
IG+(SFFS ∩ SBFE)	21,11%	21,11%	21,11%