UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
SISTEMAS DE INFORMAÇÃO

Júlio Cesar Franke Fagundes

Abordagem baseada em Árvores de Decisão para detecção e identificação de intrusões em ambientes da Internet das Coisas baseados em Computação em Nevoeiro

Florianópolis
2022

Júlio Cesar Franke Fagundes

Abordagem baseada em Árvores de Decisão para detecção e identificação de intrusões em ambientes da Internet das Coisas baseados em Computação em Nevoeiro

Este Trabalho de Conclusão do Curso foi julgado adequado para obtenção do Título de Bacharel em Sistemas de Informação e aprovado em sua forma final pelo curso de Graduação em Sistemas de Informação.

Florianópolis, 24 de Fevereiro de 2022.

Prof. Cristian Kolliver, Dr
Coordenador do Curso

Banca Examinadora:

Prof. Carlos Westphall, Dr.
Orientador
Universidade Federal de Santa Catarina

Cristiano Antonio de Souza
Coorientador
Universidade Federal de Santa Catarina

Prof. Carla Merkle Westphall, Dr.
Avaliador
Universidade Federal de Santa Catarina

Júlio Cesar Franke Fagundes

Abordagem baseada em Árvores de Decisão para detecção e identificação de intrusões em ambientes da Internet das Coisas baseados em Computação em Nevoeiro

Trabalho de Conclusão do Curso do Curso de Graduação em Sistemas de Informação do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para obtenção do título de Bacharel em Sistemas de Informação.
Orientador: Prof. Carlos Westphall, Dr.
Coorientador: Cristiano Antonio de Souza

Florianópolis
2022

AGRADECIMENTOS

Primeiramente ao reino de Deus, por me guiar ao longo dessa jornada, sempre estando comigo em todos os momentos, teu poder nunca fez com que eu fraquejasse.

Ao Alessandro Pereira, meu eterno reconhecimento e inspiração, por me guiar e acreditar em mim. As oportunidades que você me deu, espero em breve retribuí-lo. Saiba que contei ao longo dessa trajetória os anos, semestres, meses e dias restantes para referenciá-lo aqui. Levarei e defenderei seu nome onde eu estiver.

A minha família, que me acompanhou em tempo integral nessa jornada, apoiando em momentos difíceis e dando o suporte que estava ao nosso alcance.

Aos meus amigos que fiz ao longo dessa jornada, em especial: João Mafra, Pablo Carminatti e Vinícius Dias. Vocês me apoiaram em um dos momentos mais delicados dessa jornada. Levo vocês no peito para sempre.

“Ele nunca dorme e nunca dormirá, o guardião de Israel, HaShem, Deus de Abraão, Isaque e Jacó.”

— Salmos 121

RESUMO

A *Internet of Things* é um paradigma que está em grande ascensão nos últimos anos. O grande numero de dispositivos e as suas limitações de recursos, desafiam pesquisadores e desenvolvedores na área de segurança da informação. Ataques a dispositivos conectados se tornaram comuns ultimamente. Desse modo, contramedidas devem ser tomadas para oferecer uma camada de segurança na comunicação. Os *Intrusion Detection Systems* são sistemas que buscam detectar entidades maliciosas que tentam controlar e/ou indisponibilizar uma rede de dispositivos. Muitos dos trabalhos atuais nessa área de pesquisa focam em métodos de anomalias para detecção binária, que simplesmente detecta se um tráfego específico é ataque ou não, mas não é capaz de identificar o tipo de ataque. Adicionalmente, os *datasets* de treinamento e teste comumente utilizados na área da pesquisa, são criados de forma sintética, tendo um conjunto de *features* próprios. Essa diferença de *features* entre os *datasets* limita a avaliação dos estudos realizados, pois conhecidamente a qualidade das *features* são importantes para o desempenho do modelo. Recentemente foi disponibilizado um *dataset* que possui *features* baseadas no protocolo NetFlow v9, amplamente utilizado por provedores de serviço. Esse *dataset* proporciona um conjunto de *features* padrão, ou seja, representando fluxos de pacotes de uma rede *Internet of Things* "real". Portanto, neste trabalho é proposto métodos de detecção multiclasse, utilizando algoritmos de aprendizado supervisionado, comumente utilizados por pesquisas do estado da arte em detecção de intrusão. A maior contribuição deste trabalho é a utilização de um *dataset* recente, gerado a partir de uma fusão de outros *datasets* bastante utilizados na área de pesquisa em detecção de intrusão para ambientes *Internet of Things*. Através dos experimentos verificou-se que todas as abordagens avaliadas apresentaram taxas de detecção satisfatórias, apresentando baixas taxas de falsos negativos e falsos positivos.

Palavras-chave: Internet of Things. Fog Computing. Intrusion Detection. Machine Learning.

ABSTRACT

The Internet of Things is a paradigm that has been on the rise in recent years. The large number of devices and their resource limitations challenge researchers and developers in the area of information security. Attacks on connected devices have become common lately. Thus, countermeasures must be taken to provide a layer of security in communication. Intrusion Detection Systems are systems that seek to detect malicious entities that try to control and/or make a network of devices unavailable. Much of the current work in this area of research focuses on anomaly methods for binary detection, which simply detect whether a specific traffic is an attack or not, but is not able to identify the type of attack. Additionally, the training and test datasets commonly used in the research area are synthetically created, having a set of their own features. This difference in features between the datasets limits the evaluation of the studies carried out, since the quality of the features is known to be important for the performance of the model. Recently, a dataset was made available that has features based on the NetFlow v9 protocol, widely used by service providers. This dataset provides a set of standard features, that is, representing packet flows from a "real" Internet of Things network. Therefore, in this work, multiclass detection methods are proposed, using supervised learning algorithms, commonly used by state-of-the-art research in intrusion detection. The main contribution of this work is the use of a recent dataset, generated from a fusion of other datasets widely used in the area of intrusion detection research for Internet of Things environments. Through the experiments it was found that all approaches evaluated presented satisfactory detection rates, with low rates of false negatives and false positives.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DT	*Decision Tree*
DDoS	*Distributed Denial of Service*
DoS	*Denial Of Service*
DT	*Decision Tree*
ET	*Extra Tree Classifier*
Fog	*Fog Computing*
HIDS	*Host-based Intrusion Detection System*
IaaS	*Infrastructure as a service*
IDS	*Intrusion Detection System*
IEEE	*Institute of Electrical and Electronics Engineers*
IG	*Information Gain*
IoT	*Internet of Things*
IPS	*Intrusion Prevention System*
kNN	*k-Nearest Neighbors*
MDPI	*Multidisciplinary Digital Publishing Institute*
IDS	*Intrusion Detection System*
IG	*Information Gain*
ISP	*Internet Service Provider*
NIDS	*Network-based Intrusion Detection System*
NF	*NetFlow Protocol*
PaaS	*Platform as a service*
RF	*Random Forest*
RFC	*Request for Comments*
RFE	*Recursive Feature Elimination*
ROC	*Operating Characteristic Curve*

SaaS	*Software as a service*
SFS	*Sequencial Feature Selection*
SFSDT	*Sequence Forward Selection Decision Tree*
SQL	*Structured Query Language*
SLA	*Service Level Agreement*
SVM	*Support Vector Machine*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
TNR	*True Negative Rate*
UDP	*User Datagram Protocol*
UQ	*University of Queensland*

SUMÁRIO

1 INTRODUÇÃO

A Internet das Coisas *(Internet of Things - IoT)* é um paradigma que está em grande ascensão nos últimos anos. O desenvolvimento e crescimento acelerado desses pequenos dispositivos conectados à Internet, fará com que essa grande rede seja de difícil gerenciamento e controle. Os dispositivos que compõem a IoT são provenientes de pequenos sensores de baixo poder computacional, onde trocam informações entre si, com funções limitadas.

A Cisco prevê que o número de dispositivos interconectados no planeta poderá chegar a 500 bilhões em 2025 (CAMHI, 2015). Devido ao baixo poder computacional destes pequenos dispositivos, atrelados a grande quantidade dos mesmos e a necessidade de estabelecer uma comunicação eficiente, os dados originados são enviados para um centro computacional de maior capacidade de armazenamento e poder de processamento, a *Cloud Computing* (MELL; GRANCE et al., 2011).

No entanto, a *Cloud Computing* tem o problema de latência causado pela distância entre os dispositivos IoT e os *data centers* (SATYANARAYANAN, 2015). Com esse gargalo mapeado, a Cisco criou um novo paradigma de comunicação eficiente, a *Fog Computing*, onde leva um dispositivo com maior poder computacional, para a borda da rede (BONOMI et al., 2012). Dessa forma, é possível processar e armazenar informações próximas aos nós finais, aliviando o tráfego enviado para a nuvem (TORDERA EVA et al., 2017). Com isso, viabiliza aplicações que necessitam de processamento em tempo real.

O ambiente da IoT não está livre de ameaças e vulnerabilidades de segurança. Com o crescimento mencionado anteriormente e a heterogeneidade dos dispositivos, há o aumento da probabilidade de surgirem vulnerabilidades.

Os mecanismos de detecção de intrusão são pontos críticos de segurança, eles são capazes de mapear as tentativas de ataques sobre essas entidades. Para detectar ataques IoT na camada de transporte, existem diferentes tipos de sistemas de detecção de intrusão, incluindo aqueles baseados na detecção de assinaturas e baseados em detecção por anomalias.

No ambiente de pesquisa, a maioria das abordagens de detecção de intrusão enfoca em métodos de anomalia para detecção binária (MIRANDA et al., 2020), (PRIYADARSHINI; BARIK, 2019), (SHAFI et al., 2018). Por outro lado, as poucas abordagens de detecção multiclasse existentes que visam classificar o ataque em categorias, apresentam taxas de acurácia inferiores aos métodos binários.

Nos últimos anos, muitos esforços na área de pesquisa para IDSs têm como objetivo alavancar modelos de *Machine Learning* (ML) cada vez mais poderosos. Para isso, foram gerados *datasets* contendo tráfego de rede de forma sintética para serem utilizados nos experimentos, ou seja, cada um contendo seus próprios registros e *features* pré determinadas em sua criação.

Com isso, a avaliação de modelos de ML geralmente não é confiável, pois diferentes estudos aplicados na área de NIDS utilizam diferentes *datasets* contendo registros de eventos de segurança variados, outro ponto crítico são as *features* utilizadas, muitas delas são distintas

para cada *dataset*.

Estudos recentes propuseram um novo *dataset*, através da mesclagem de registros de quatro *datasets* bastante explorados em pesquisas para NIDS. As *features* foram padronizadas seguindo o protocolo NetFlow versão 9. Este protocolo é comumente utilizado por Provedores de Acesso à Internet *(Internet Service Provider - ISP)* para a análise de pacotes que trafegam na rede, apoiando na mitigação de desvios de padrões.

No entanto, até onde sabemos, não há trabalho no estado da arte conduzindo uma investigação completa da capacidade dos principais algoritmos de ML para realizar tarefas de identificação de intrusão multiclasse utilizando o novo *dataset* NF-UQ-NIDS-v2. Portanto, neste trabalho, investigamos a habilidade dos classificadores baseados na família de Árvores de Decisão para detecção e categorização de intrusão.

1.1 MOTIVAÇÃO

A IoT está cada vez mais presente no dia a dia da sociedade. Esses dispositivos conectados estão presentes nas mais diversas áreas de aplicação, seja na indústria, automação residencial, cidades conectadas, monitoramento de ambientes agrícolas, *data centers* e afins. Devido a grande difusão dessa tecnologia, o mercado de desenvolvimento dessas soluções, ainda possuem grandes desafios quanto à segurança. Os dispositivos são de pequeno porte e com baixo poder computacional, pois designam funções simples, como sensores de captura. Essa limitação propicia um ambiente fértil para hackers entrarem em redes e sistemas. A principal motivação deste trabalho é propor uma avaliação de modelos para detecção e classificação de eventos de ataque em redes IoT, através dos métodos de classificação variantes da família de Arvores de Decisão, utilizando um novo *dataset* que contêm diversos registros de ataques e um conjunto de *features* padrão.

1.2 OBJETIVOS

Nesta seção são apresentados os objetivos deste trabalho.

1.2.1 Objetivo Geral

Contextualizar o atual estado da arte em relação à segurança em *Fog Computing* e IoT, de modo a identificar quais as questões em aberto a serem trabalhadas no futuro. Além disso, propor uma abordagem baseada nos algoritmos da família de Árvores de Decisão para detecção e identificação de intrusões em ambientes baseados no paradigma de *Fog Computing* e IoT. Esses algoritmos foram utilizados em diversos experimentos sobre um *dataset* criado recentemente, originado através da mesclagem de quatro *datasets* comumente utilizados em pesquisas de IDSs para ambientes IoT.

1.2.2 Objetivos Específicos

Como objetivos específicos foram listados os três principais:

- Realizar uma revisão bibliográfica do atual estado da arte relacionado a detecção e identificação de intrusão para ambientes baseados em *Fog Computing* e IoT.
- Propor abordagens com técnicas de ML, utilizando algoritmos da família de Árvores de Decisão para detecção e classificação de intrusões em ambientes baseados em *Fog Computing* e IoT.
- Utilizar seletores de *features* para otimização do desempenho de detecção e redução de custo computacional.
- Implementar, testar, avaliar e comparar as abordagens propostas com o *dataset* NF-UQ-NIDS-v2.

1.3 ORGANIZAÇÃO DESTE TRABALHO

O capítulo 2 explana conceitos fundamentais para o entendimento deste trabalho. No Capítulo 3 são apresentados alguns trabalhos correlatos. No Capítulo 4, são apresentados detalhes a respeito da abordagem proposta e seu funcionamento. No Capítulo 5, são descritos aspectos básicos para aplicação da metodologia experimental e são apresentados os fluxos de experimentos realizados. No Capítulo 6 são apresentados os resultados dos experimentos realizados neste trabalho, seguido de discussões sobre os resultados obtidos pelos modelos. Por fim, o Capítulo 7 apresenta a conclusão sobre o trabalho realizado, abordando reflexões a respeito dos objetivos deste trabalho e dos resultados obtidos nos experimentos, bem como de possíveis trabalhos futuros.

2 CONCEITOS FUNDAMENTAIS

Neste capítulo são apresentados conceitos fundamentais envolvidos na temática deste trabalho. São descritos os conceitos relacionados a *Internet of Things (IoT)* e de tecnologias relacionadas como *Cloud Computing* e *Fog Computing*, além da integração entre elas. Em seguida, apresenta-se de forma breve e objetiva um Sistema de Detecção de Intrusão, bem como suas técnicas de funcionamento. Posteriormente, são apresentados os aspectos fundamentais de ML, classificadores e seletores de *features* utilizados. Por fim, são explanados detalhes para o entendimento do *dataset* baseado no protocolo *NetFlow* que foi abordado nos experimentos.

2.1 INTERNET OF THINGS

A origem do nome *Internet of Things* foi atribuído a Kevin Ashton em 1999, em uma apresentação feita por ele na empresa Procter & Gamble (Ashton, 2009). Trata-se de uma tecnologia onde pequenos dispositivos comunicam-se entre si, ou com outros sistemas. As "coisas" podem derivar de um simples sensor de incêndio, uma lâmpada ou até mesmo um monitor cardíaco. Resumindo, tudo aquilo que é natural ou construído por um humano que possa transmitir ou receber dados, através de uma rede de comunicação, seja ela *wireless* ou cabeada. Os dispositivos IoT atuam como pequenos sensores, coletando informações e enviando para uma central de processamento de maior pode computacional, para que dados sejam processados e transformados em informação, já que os dispositivos não possuem capacidade de processamento para gerar informação, mas sim capturar e enviar dados brutos sem antes realizarem um tratamento sobre os dados. Uma das alternativas para realizar o processamentos das informações coletadas pelos dispositivos IoT é a *Cloud Computing*, a seguir, são apresentados maiores detalhes a respeito desse paradigma.

2.2 CLOUD COMPUTING

Atualmente, a computação em nuvem, em inglês *Cloud Computing*, é considerada uma grande visão do mundo da computação. Uma tecnologia versátil que conta com servidores conectados à internet e distribuídos ao longo do mundo. Essa tecnologia provê a alocação e uso de recursos de forma inteligente e programável. Os recursos disponíveis são para atender demandas de capacidade de armazenamento, processamento e disponibilidade de aplicações *(Service Level Agreement - SLA)*. O *National Institute of Standards and Technology (NIST)* definiu três tipos de serviço fornecidos pela tecnologia em Nuvem (MELL; GRANCE et al., 2011), que são:

- Software como Serviço *(Software as Service - SaaS)*: A capacidade de fornecer uma infraestrutura em Nuvem para executar aplicações do cliente. Tais aplicações são heterogêneas, atendendo serviços como webmails, e-commerces e intranets.

- Plataforma como Serviço *(Platform as Service - PaaS)*: Fornecer um ambiente de desenvolvimento de aplicações para clientes.
- Infraestrutura como Serviço *(Infrastructure as Service - IaaS)*: A capacidade de fornecer recursos de hardware de forma virtualizada. Instanciar e distribuir máquinas conforme a demanda, podendo utilizar políticas de escalonamento. Neste serviço é possível atender demandas de processamento e armazenamento para o cliente.

Devido a distribuição geográfica dos *datacenters* e a dependência direta por redes de alta velocidade e disponibilidade, existem alguns gargalos nessa tecnologia, principalmente quanto a aplicações que necessitam de baixo tempo de resposta.

2.3 FOG COMPUTING

A *Fog Computing - Fog* surgiu recentemente, sendo introduzida pela *Cisco Networks* em 2014 (CHEN; AZHARI; LEU, 2018). A *Fog Computing* é uma proposta que promete habilitar uma computação próxima a borda da rede, provendo menor tempo de resposta para dispositivos clientes. Quanto à conexão com a *Cloud Computing*, a *Fog* realiza uma etapa chamada de pré-processamento e simultaneamente envia os dados tratados e necessários para a *Cloud*.

A Figura 2.3 mostra como a *Fog* está posicionada na arquitetura padrão e que pode dar amplo suporte ao ecossistema distribuído baseado em *Cloud* para atender os dispositivos clientes. A utilização da *Cloud*, dependendo da aplicação, se torna opcional. A heterogeneidade dos dispositivos inteligentes permite diferentes arquiteturas possíveis, adaptando-se com os casos de uso para cada dispositivo.

Os autores Patwary et al. (2021), definem as características e recursos da tecnologia *Fog Computing* da seguinte maneira:

- Apoio à distribuição geográfica;
- Baixa latência;
- Heterogeneidade;
- Segurança de dados e proteção de privacidade;
- Maior escalabilidade;
- Suporte a espaço de armazenamento.

Os ambientes baseado em IoT e nas tecnologias relacionadas podem ser alvos de entidades maliciosas que buscam causar prejuízos ou obter vantagens através de ataques aos dispositivos que integram a rede. Desse modo, mecanismos de segurança são extremamente necessários, a seguir são apresentados detalhes a respeito dos *Intrusion Detection Systems*.

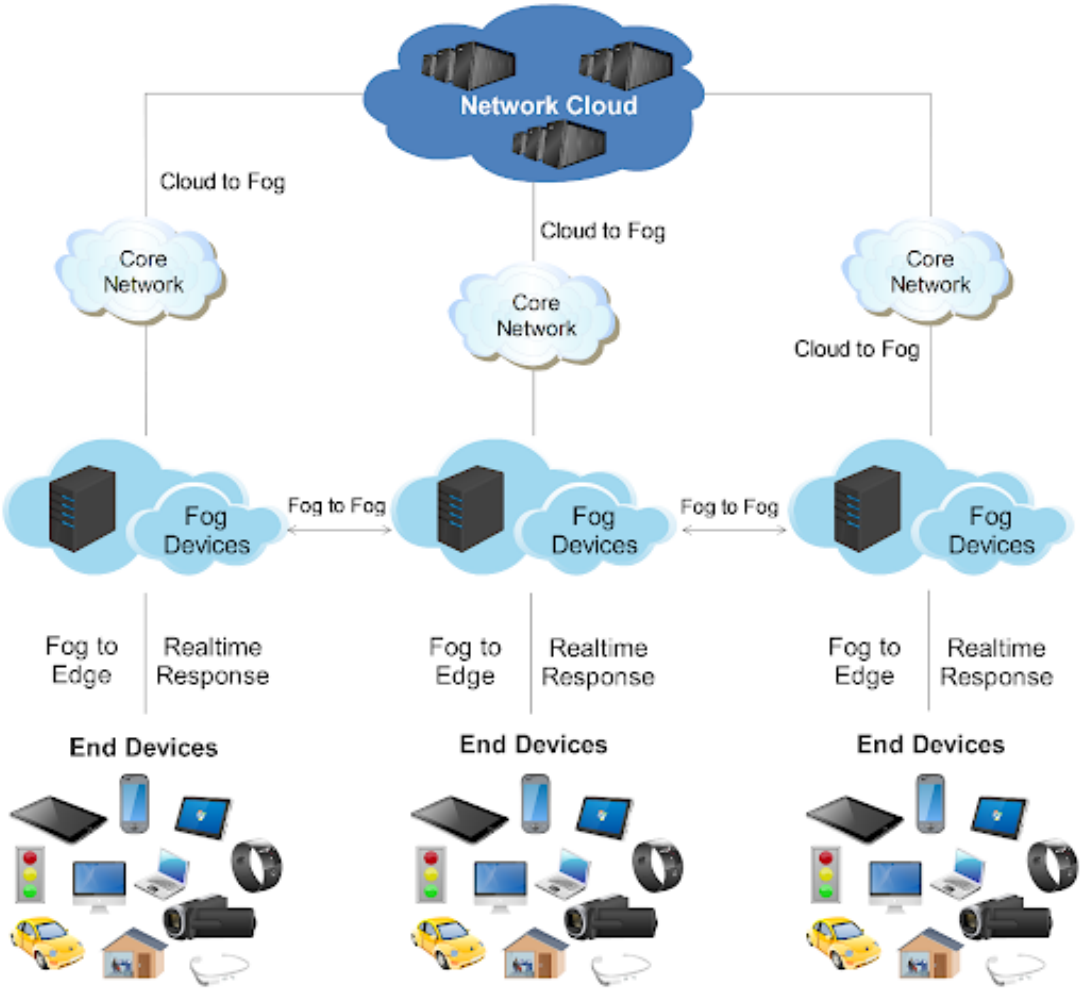


Figura 1 – Suporte e distribuição dos dispositivos Fog entre os dispositivos inteligentes e a Nuvem.

Fonte: Adaptado de (PATWARY et al., 2021)

.

2.4 INTRUSION DETECTION SYSTEMS

Em termos de segurança, as anomalias de rede na IoT são geralmente detectadas através da utilização de um Sistema de Detecção de Intrusão, em inglês *Intrusion Detection System - IDS*. Um IDS é um sistema (*hardware* ou *software*) que monitora e inspeciona o fluxo de tráfego de rede, em busca de possíveis violações dos princípios de segurança de computadores (LAWAL; SHAIKH; HASSAN, 2020).

O IDS tradicionalmente fica posicionado atrás da borda da rede. As operações podem ser divididas em três estágios. O primeiro é o de monitoramento, onde todo o tráfego de entrada da camada de rede é coletado. A seguir, os dados capturados por sensores baseados em rede ou host são armazenados em um banco de dados para posteriormente serem analisados. Os dados coletados são processados com métodos de extratos de assinaturas preditivos, extração de recursos ou métodos de identificação de padrões adequados para o estágio final de detecção de anomalias em nível de rede.

2.4.1 Técnicas de Detecção

O IDS através da sua coleta exaustiva de tráfego, disponibiliza dados para análise, que são submetidos a técnicas de detecção. Os dois principais tipos de detecção são apresentados na seções a seguir.

2.4.1.1 Signature-Based Intrusion Detection System (SIDS)

As técnicas baseadas em assinaturas realizam consultas à uma base de dados com extratos de ataques. As análises são feitas sobre esses extratos, caso seja detectada uma similaridade forte, o IDS realiza ações de bloqueio da rede ou isolamento de um nó diagnosticado como atacante. Segundo Lawal, Shaikh e Hassan (2020), a técnica baseada em assinaturas tem uma alta dependência da qualidade dos extratos armazenados e possui dificuldades para detectar novos ataques, conhecidos como ataques do dia zero.

2.4.1.2 Anomaly-Based Intrusion Detection System (AIDS)

As técnicas de detecção baseadas em anomalia utilizam o conceito de análise baseada em comportamento padrão. O IDS utiliza técnicas para determinar e estabelecer um comportamento padrão da rede. Todos os comportamentos que forem anômalos a este padrão modelado serão detectados como intrusões. Quando atacantes tentam invadir um sistema eles realizam diversas ações que podem ser anormais em relação ao funcionamento padrão da rede e, portanto, serão detectados como intrusões. Isso acaba sendo uma vantagem sobre a técnica baseada em assinatura, pois é capaz de detectar ataques do dia zero.

No entanto, nem todo comportamento anômalo é realmente intrusivo. Portanto, uma das desvantagens desse tipo de detecção é um número alto de falsos positivos que podem comprometer toda a rede, alertando um tráfego legítimo como se fosse um tráfego ilegal. Elrawy, Awad e Hamed (2018) apresentam uma pequena comparação entre os esquemas disponíveis para técnicas baseada em anomalias com um foco nas suas vantagens e desvantagens em cada técnica, conforme a Tabela 1 (ELRAWY; AWAD; HAMED, 2018).

Tabela 1 – Tabela comparativa entre técnicas de detecção de anomalias.

Técnica	Vantagens	Desvantagens
Data mining	1 - Modelos são criados automaticamente	1 - Baseado em dados históricos
	2 - Aplicável em diferentes ambientes	2 - Depende de algoritmos complexos
	3 - Adequada para *datasets online*	
Machine Learning	1 - Alta taxa de acurácia	1 - Requer dados de treinamento
	2 - Adequada para grande volume de dados	2 - Longo tempo de treinamento
Modelos estatísticos	1 - Adequado para *datasets online*	1 - Baseado em comportamento histórico
	2 - Simplicidade do sistema	2 - Taxa de detecção depende de operações estatísticas e matemáticas
Modelos de regras	1 - Adequado para *datasets online*	1 - Baseado em um conjunto de regras
	2 - Simplicidade do sistema	2 - Alta taxa de falsos positivos
Modelos de *payload*	1 - Alta taxa de detecção para ataques conhecidos	1 - Problemas de privacidade
		2 - Longo tempo de processamento
Modelos de protocolo	1 Alta taxa de detecção para um tipo especifico de ataque	1 - Projeto para apenas um tipo especifico de protocolo
Modelos de processamento de sinais	1 - Alta taxa de detecção	1 - Depende de métodos complexos de reconhecimento de padrões
	2 - Baixa taxa de falsos positivos	

2.4.2 Machine Learning

Aprendizado de máquina, em inglês *Machine Learning*, é um subconjunto da Inteligência Artificial que constrói um modelo matemático com base em dados e amostras a fim de fazer previsões para tomar decisões (CHO et al., 2020).

O aprendizado supervisionado é um tipo de técnica de aprendizado de máquina em que as características do *dataset* de treinamento são utilizados na fase de aprendizado para criar um modelo de classificação, que é então usado para classificar novas instâncias não vistas (NAMDEV; AGRAWAL; SILKARI, 2015).

As técnicas de ML são comumente aplicadas no contexto de segurança para construir modelos de classificação capazes de compor *Anomaly-Based Intrusion Detection Systems* e realizar a detecção de intrusões. Portanto, o uso de aprendizagem de máquina supervisionada para auxiliar na detecção de intrusão em sistemas IoT é uma técnica que depende de dois estágios: O estágio de treinamento ou aprendizado e o estágio de detecção (TSAI et al., 2009). O estágio de treinamento depende de algoritmos matemáticos ou funções que usam dados como uma entrada de referência para aprender as características do ambiente de computação. Na etapa de

detecção, essas características são utilizadas para detecção e classificação (NISHANI; BIBA, 2016). A seguir apresenta-se o método supervisionado conhecido como Árvores de Decisão.

2.4.2.1 Classificador Decision Tree

Árvores de decisão *(Decision Tree - DT)* é um dos algoritmos mais comumente usados em projetos de mineração de dados, na etapa de classificação (ROKACH, 2016). A DT utiliza formato em árvore, contendo nodos e arestas para tomar decisões com base nos recursos das instâncias.

Os nodos correspondem as *features*, as arestas representam um valor ou uma faixa de valores de uma determinada *feature*. Os nodos folha representam a classificação.

Começando com todo o *dataset*, o nodo raiz corresponde à primeira divisão que especifica como os dados devem ser divididos em partições separadas. Os nodos intermediários recursivos continuam a dividir os dados em partições menores até que nenhum particionamento adicional seja necessário. Desta forma os nós folha da estrutura representam as partições finais.

O algoritmo básico de indução de árvore de decisão constrói árvores de decisão recursivamente com base na divisão e conquista, de cima para baixo. Em cada iteração, o algoritmo procura a melhor *feature* do conjunto de *features* para realizar o split dos dados. Uma boa divisão em uma DT corresponde à escolha da *feature* com menor impureza e maior poder de separação dos dados.

A seleção da *feature* com maior poder de separação é feita de acordo com critérios de divisão, como Entropia ou Índice de Gini *(Gini Index)*.

O Índice de Gini *(Gini Index)* mede quão bem uma determinada *feature* 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 2.1 (BREIMAN et al., 1984). Onde P_i denota a probabilidade de que um elemento seja classificado para uma classe distinta.

$$Gini = 1 - \sum_{i=1}^{c} (P_i)^2 \tag{2.1}$$

A Entropia é comumente utilizada na teoria da informação para medir a pureza ou impureza de um determinado conjunto. Essa é uma medida de desordem ou incerteza e o objetivo dos modelos de ML em geral é reduzir a incerteza. A formula matemática para a entropia é H do negativo da probabilidade do evento i vezes o logaritmo na base dois da probabilidade do evento $p(x_i)$, onde p_i é a probabilidade da classe i e n é o número de classes.

$$Entropia = H(x) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \tag{2.2}$$

Ao construir a árvore, são escolhidos as *features* com os menores valores do Índice de Gini ou da Entropia. Com a ajuda dessas métricas, o algoritmo decidido qual *feature* deve ser dividida e qual é o ponto de divisão do recurso (BREIMAN et al., 1984).

2.4.2.2 Classificador Random Forest

O algoritmo de Florestas Aleatórias *(Random Forest - RF)* possui a mesma combinação e simplicidade de uma DT, com um processo adicional, que propicia uma maior flexibilidade e aleatoriedade para melhorar a precisão (BREIMAN, 2001). A RF foi criada para diminuir o Sobreajuste *(Overfitting)*, beneficiando modelos generalizáveis.

A RF 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. Uma de suas características principais é o emprego de um grau de aleatoriedade na seleção das *features* a serem consideradas para a divisão. Diferentemente da DT, que aplica métricas de impureza em todo o conjunto de *features* para descobrir a melhor, a RF aplica estas métricas somente em um subconjunto de *features* candidatas selecionadas aleatoriamente. Além disso, utiliza apenas um subconjunto de linhas do *dataset*, com reposição, para a construção da árvore.

O algoritmo, em cada nodo, procura pelas *features* que têm uma melhor separabilidade, produzindo uma melhor pureza nos dados. Em cada nodo, a RF seleciona aleatoriamente um conjunto de *K features* candidatas e aplica as medidas de impureza buscando encontrar o melhor ponto de corte de cada *feature* e a melhor *feature*. Essas *K features* são definidas para serem utilizadas na criação das árvores internas, contendo poucos níveis, também conhecidas como Aprendiz Fraco *(Week Learner)*. Esse processo garante que cada árvore gera um modelo diferente.

Após o treinamento da RF, a estrutura está apta para realizar a classificação de novos dados. Cada árvore gerada irá realizar a classificação do registro e os seus resultados serão combinados, através de votação, em uma classificação final.

2.4.2.3 Classificador Extra Tree

O algoritmo Árvores Extras *(Extremely Randomized Trees - Extra Trees)*, variante da família da DT, também cria muitas DT de maneira aleatória, para então utilizar da combinação dos resultados de cada árvore para encontrar a resposta final. O principal diferencial de uma *Extra Tree* está no processo de ser extremamente aleatório, contribuindo para modelos mais generalizáveis (GEURTS; ERNST; WEHENKEL, 2006).

O processo de construção das árvores é similar com o do algoritmo RF, que também cria muitas DT de forma aleatória, utilizando cada uma das árvores como definição do resultado. A principal diferença entre os dois está na quantidade de processos aleatórios empregados em cada algoritmo.

Em ET, a aleatoriedade vai um passo além na forma como as divisões são calculadas. Como em RF, em cada nodo um subconjunto aleatório de *K features* candidatas é utilizado, no entanto, em ET, em vez de procurar os pontos de corte mais discriminantes, eles são definidos de maneira aleatória para cada uma das *features* candidatas. A melhor das *K features* previamente selecionadas é escolhida então como regra da divisão do nodo. A Entropia é comumente utilizada na teoria da informação para medir a pureza ou impureza de um determinado conjunto. Essa é uma medida de desordem ou incerteza e o objetivo dos modelos de ML em geral é reduzir a incerteza.

A lógica por trás do método é que o corte explícito e a aleatorização de *features* 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 (GEURTS; ERNST; WEHENKEL, 2006).

2.5 SELEÇÃO DE *FEATURES*

A existência de *features* irrelevantes e redundantes em *datasets* com grande dimensionalidade pode interferir e prejudicar o desempenho de classificação. As abordagens de detecção por anomalia, baseadas em métodos de aprendizado de máquina, portanto, podem ser afetadas pela qualidade das *features* analisados.

O processo de seleção de *features* busca a otimização do *dataset*. O objetivo principal é encontrar o melhor subconjunto de *features*, partindo de um conjunto fixo de *features* originais. Desse modo, é possível melhorar o desempenho do modelo em relação a taxa de detecção e ao custo computacional.

Existem diversas técnicas de seleção de *features* na literatura e elas podem ser agrupadas nas categorias: *Filter*, *Embedded* e *Wrapper*. Os métodos *Filter* são baseados em métricas estatísticas. Eles são independentes do classificador. Tais métricas são calculadas aplicando o algoritmo e buscam remover *features* irrelevantes antes da indução ocorrer. A avaliação de cada *feature* é realizada individualmente baseada em sua correlação com a função alvo. Após isto então seleciona-se as *k features* com os maiores valores. Os métodos *Embedded* são caracterizados pela realização da seleção de *features* de forma dinâmica, selecionando um subconjunto de *features* no próprio processo de construção do modelo de classificação. Por fim, os métodos *Wrapper* também operam com a utilização de um modelo de classificação internamente, no entanto, eles trabalham com subconjuntos de *features* gerados a cada iteração do algoritmo. O classificador é executado com o subconjunto de *features* gerando uma precisão do classificador, que é utilizada para avaliar a qualidade do subconjunto de *features* selecionadas.

A seguir são apresentados algumas técnicas de seleção de *features* amplamente utilizadas em pesquisas com aprendizado de máquina.

2.5.1 Ganho de Informação

Uma das maneiras para se medir a qualidade de uma *feature* é avaliar o seu grau de associação com a classe através da medida do ganho de informação, em inglês *Info Gain*, um dos algoritmos *filter* mais conhecido e utilizado na literatura. Ele avalia o grau de associação da *feature* com a classe para encontrar os valores com o maior grau de utilização e importância através do calculo da redução da entropia de cada *features*. 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 processo de detecção.

Considerando um *dataset* $D(A_1, A_2, ..., A_n, C)$, com $n >= 1$, onde C é a *feature* classe e o seu domínio é $(c_1, c_2, ..., c_m)$, com $m >= 2$. Assume-se que os valores das *features* e da classe são discretos.

A medida de ganho de informação é baseado no conceito de entropia (QUINLAN, 1990), fórmula apresentada na Equação 2.2. A entropia da classe C, representada por $Entropia(C)$, pode ser calculada 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 uma *feature* é determinado pela redução da sua entropia (KAREGOWDA; MANJUNATH; JAYARAM, 2010). A Equação 2.3 calcula o ganho de informação de uma *feature A* em relação a classe C.

$$Ganho(C,A) = Entropia(C) - Entropia(C,A) \tag{2.3}$$

Onde $Entropia(C)$ é a entropia da classe C, e $Entropia(C,A)$ é entropia da classe relativa a *feature A*. Calculada pela Equação 2.4. Onde $Entropia(C|A = A_j)$ é a entropia relativa ao subconjunto de instâncias que tem um valor A_j para a *feature 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.

$$Entropia(C,A) = -\sum_{j=1}^{m} p(A,C)\log_2(p(A,C)) \tag{2.4}$$

Temos que o valor de ganho de informação consiste na variação da impureza, ou seja, *features* com menor entropia possuem maior ganho de informação. Esta medida tem um *bias* natural pois favorece *features* que possuem muitos valores.

A medida de razão de ganho de informação tenta corrigir o *bias* do Ganho de Informação (QUINLAN, 2014).

$$Razão\ \ de\ \ Ganho(C,A) = \frac{Ganho(C,A)}{Entropia(A)} = \frac{Entropia(C) - Entropia(C,A)}{Entropia(A)} \tag{2.5}$$

Após o cálculo do mérito de cada *feature* pela Equação 2.5, é gerado um ranking e seleciona-se as N melhores *features* desse ranking de acordo com algum critério.

2.5.2 Seletor Sequencial de *Features*

O método Seletor Sequencial de *Features (Sequential Feature Selector - SFS)* adiciona (seleção para frente) ou remove (seleção para trás) *features* para formar um subconjunto de *features* de maneira gananciosa com a utilização de um classificador. Em cada estágio, este estimador escolhe a melhor *feature* para adicionar ou remover com base na pontuação de validação cruzada. Ele consiste em uma abordagem *Wrapper* e trabalha com subconjunto de *features* gerados a cada iteração do algoritmo. O classificador é executado com essas *features* no conjunto de treinamento e tem como resultado uma precisão do classificador. Ela é utilizada para avaliar a qualidade do subconjunto de *features*. O processo é repetido para cada subconjunto de *features* até que o critério de parada seja satisfeito.

Os algoritmos 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* e a *backward*.

A estratégia de busca *forward selection* se refere a uma busca que tem o conjunto inicial vazio. A exploração é então iniciada a partir do estado pai onde são criadas variações do estado atual e gerando estados filhos, adicionando *features* a este subconjunto que antes eram ausentes no estado pai. 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 *features* nos subconjuntos se torna mais rápido a comparação e classificação. 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. (JOHN; KOHAVI; PFLEGER, 1994)

De maneira oposta, a estratégia de busca *backward elimination* tem como estado inicial o subconjunto total de *features* da base de eventos. Desse modo, captura as relações de interdependência entre *features* dentro de uma base com maior facilidade que a abordagem *forward*. A partir de um estado pai, uma *feature* é removida, 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 a *feature* removida não possui relevância no conjunto avaliado. (JOHN; KOHAVI; PFLEGER, 1994)

2.5.3 Eliminação Recursiva de *Features*

O método de Eliminação Recursiva de *Features (Recursive Feature Elimanation - RFE)* da família *Wrapper*, é um algoritmo de seleção de *features*. Sua composição têm duas camadas, onde o núcleo possuí um primeiro método de seleção, por exemplo uma DT, logo esse método é encapsulado pelo RFE.

A RFE procura por um subconjunto de *features*, começando com todo o conjunto de

features de treinamento e removendo de forma recursiva até que o número desejado de *features* seja alcançado. Uma facilidade que a RFE proporciona é a passagem de hyperparâmetros, como o algoritmo que será encapsulado e o número de *features* desejado.

O algoritmo de núcleo deve fornecer uma maneira de pontuar e calcular as *features* mais importantes. O núcleo não precisa ser o algoritmo que se ajusta as *features* selecionadas, ou seja, diferentes algorítimos podem ser utilizados.

2.6 PROTOCOLO NETFLOW

O NetFlow é um protocolo aberto, inicialmente desenvolvido pela empresa *Cisco Systems* em 1996 (KERR; BRUINS, 2001). O protocolo tem como objetivo permitir a análise de tráfego IP em tempo real, por meio da coleta e análise de fluxos de rede.

Ao longo dos anos, o protocolo foi recebendo atualizações. A versão 5 é a mais comum, porém há limitações na quantidade de campos monitorados nos pacotes e também não possuí suporte para IPv6. A versão 9, mais recente e descrita na RFC 3954, possuí diversas melhorias como suporte ao protocolo IPv6 e informações da camada de enlace de dados.

O NetFlow basicamente permite:

- Monitoramento da largura de banda;
- Detecção de anomalias de tráfego de rede;
- Análise Forense do tráfego;
- Contabilização de uso da rede;
- Identificação de aplicações e *hosts* que fazem uso da rede.

Para estabelecer o NetFlow na rede, é necessário três componentes básicos:

- Exportador de fluxos (*Probe*): Tem a função de agregar os pacotes em fluxos e exportar para um coletor de fluxos;
- Coletor de fluxos (*Collector*): Responsável pela recepção, armazenamento e pré-processamento dos dados dos fluxos;
- Analisador (*Analyzer*): Permite realizar análises dos fluxos, por meio de filtros, gráficos e geração de alertas.

2.6.1 Fluxo de rede

Um fluxo de rede é uma sequencia de pacotes enviados entre dois *hosts*, que compartilham as mesmas propriedades. Os componentes de um fluxo são:

- Endereço de IP de origem;
- Endereço de IP de destino;
- Porta de origem da camada de transporte;
- Porta de destino da camada de transporte;
- Classe - Marcação do campo DSCP/TOS no cabeçalho IP;
- Protocolo da camada de transporte;
- Interface de rede de origem.

Os fluxos normalmente são enviados em amostras, por exemplo, 1:100 (1 a cada 100 fluxos serão exportados para o coletor).

O Administrador de rede deve avaliar a relação de amostragem de fluxos, pois quanto maior a relação de amostragem, menor é a acurácia da análise. Deve-se levar em consideração o tamanho da rede e a criticidade dos serviços em produção para estabelecer uma boa relação de amostragem.

2.6.2 NF-UQ-NIDS

Netflow University of Queensland Network Intrusion Detection System é um *dataset* originado por pesquisadores da Universidade de Queensland na Austrália.

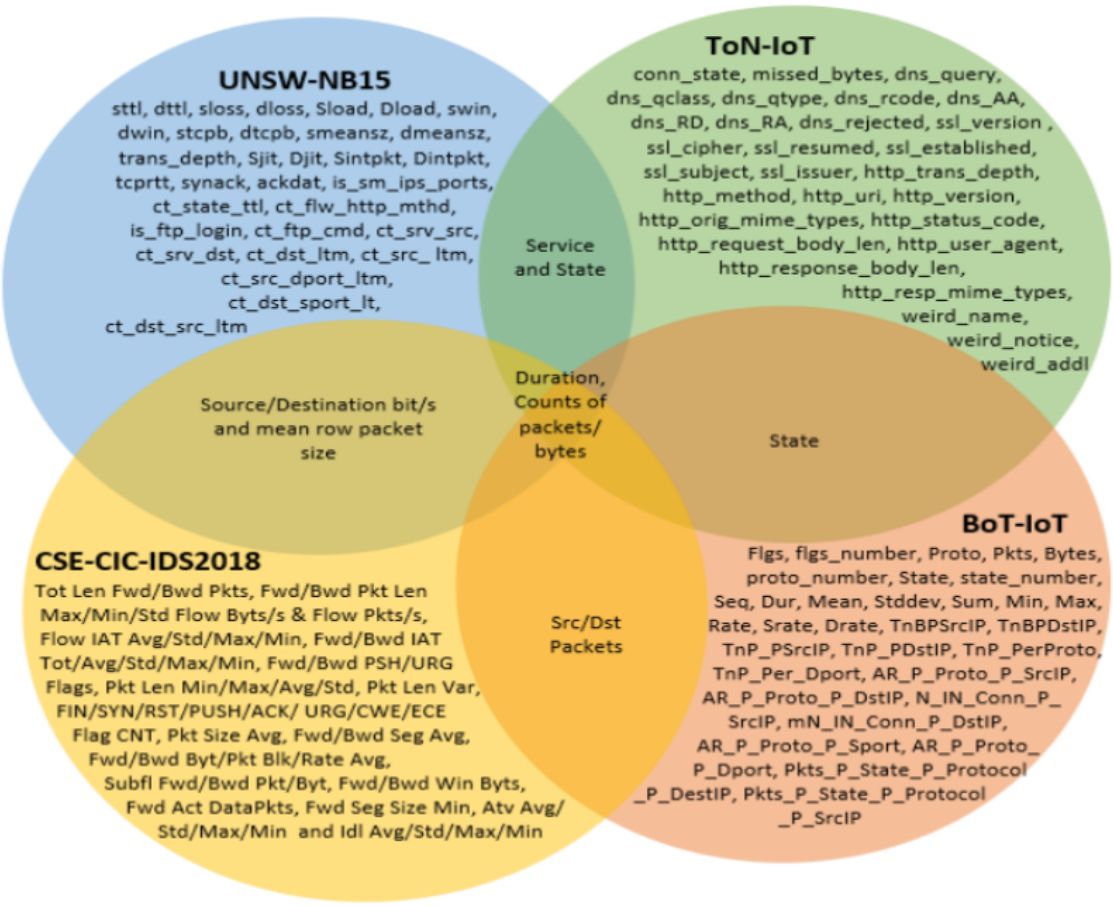
Os pesquisadores realizaram uma mesclagem com os principais *datasets* utilizados no campo da pesquisa em NIDS para IoT. Para isso, os *datasets* UNSW-BN15, BoT-IoT, ToN-IoT e CSE-CIC-IDS2018 tiveram seus arquivos de captura de pacotes traduzidos para *features* do NetFlow.

A fusão dos *datasets* propicia que haja diversos fluxos de diferentes configurações de rede, contendo características de ataques distintas. Isso pode ser usado para comparar os mesmos cenários de ataque conduzidos em redes de teste diferentes.

Observa-se através da Figura 2 que os quatro *datasets* possuem poucas *features* em comum. Algumas destas representando informações da camada de aplicação como no *dataset* ToN-IoT em sua grande maioria, e outras apenas a nível de rede/transporte, como por exemplo o BoT-IoT e CSE-CIC-IDS2018.

Algumas categorias de ataque foram agrupadas para uma categoria pai. Ataques como DoS-Hulk, DoS-SlowHTTPTest, DoS-GoldenEye e DoS-Slowloris foram acopladas como DoS. Ataques DDoS, como DDoS-LOIC-UDP, DDoS-HOIC e DDoS-LOIC-HTTP foram acoplados em DDoS. Ataques de Força Bruta como FTP-BruteForce, SSH-BruteForce, BruteForce-Web e BruteForce-XSS foram agrupados como *BruteForce*. Por fim, os ataques de injeção SQL foram agrupados em (*SQL Injection*).

Figura 2 – Diagrama de Venn representando a heterogeneidade de cada *dataset*.



Fonte: Adaptado de (SARHAN et al., 2020)

.

Em um cenário do mundo real, os recursos do NetFlow são relativamente mais fáceis de serem extraídos do tráfego da rede em comparação com os *datasets* originais, já que são extraídos dos cabeçalhos dos pacotes.

2.6.3 NF-UQ-NIDS-v2

A versão 2 (v2) foi gerada devida à insuficientes informações de segurança sendo representadas pelo conjunto de *features* básicas do NetFlow. Com a versão padrão (v1), contendo 12 *features*, os modelos de ML levam a uma precisão de detecção limitada, principalmente em experimentos multiclasse.

Para solucionar essa lacuna, os pesquisadores propuseram um conjunto de *features* estendidos do NetFlow v9, contendo 43 *features*, para que amplie a capacidade de representação de informações.

O NF-UQ-NIDS e NF-UQ-NIDS-v2 *dataset*, contêm um total de 75,987,976 registros, sendo 25.165.295 (33.12%) fluxos benignos e 50.822.681 (66.88%) são fluxos contendo

ataques. A Tabela 2 lista a distribuição das classes de ataques.

Tabela 2 – Distribuição de Ataques - NF-UQ-NIDS.

Classe	**Quantidade**
Benigno	25165295
DDoS	21748351
Reconnaissance	2633778
Injection	684897
DoS	17875585
Brute Force	123982
Password	1153323
XSS	2455020
Infilteration	116361
Exploits	31551
Theft	2431
Scanning	3781419
Fuzzers	22310
Backdoor	18978
Bot	143097
Generic	165560
Analysis	2299
Shellcode	1427
MITM	7723
Worms	164
Ransomware	3425

3 ESTADO DA ARTE

Neste capítulo são apresentadas soluções propostas por outros autores, de modo a construir uma visão do atual estado da arte em detecção e identificação de intrusão em ambientes de redes IoT, utilizando de técnicas baseadas em aprendizado de máquina. Os artigos escolhidos apresentam aspectos relacionados com a proposta que será defendida neste trabalho.

3.1 INTRUSION DETECTION SYSTEM BASED ON DECISION TREE OVER BIG DATA IN FOG ENVIRONMENT

Os autores Peng et al. (2018) propuseram um IDS baseado em DT. Como primeiro processo do trabalho, foi proposto um algoritmo de pré-processamento para digitalizar as strings no conjunto dos dados (KDDCUP99) e para normalizar os dados, garantindo a qualidade dos dados com o objetivo de melhorar a eficiência da detecção. No segundo processo, foi utilizado o método de DT para o IDS, para posteriormente comparar com outros métodos, como *Naive Bayes* e *K-Nearest Neighbor* (KNN). O experimento foi implementado em *Python* em um sistema operacional *Windows* 10, com processador *Intel Core i7 2.7GHz*, 16GB de memória RAM sobre as plataformas *Eclipse* e *Anaconda 2.7 SCIkit-learn*. Foram realizados testes sobre 10% do *dataset* e outro com o *dataset* completo. O KNN devido ao seu alto tempo de processamento foi descartado dos estudos. Por fim, foi discutida a taxa de detecção entre o *Naive Bayes* e a DT, onde a DT mostrou maiores taxas de acerto na detecção e classificação, por contrapartida o *Naive Bayes* mostrou menor tempo de processamento. No cenário do *Big Data* os autores afirmam o método de DT mais adequado para um IDS sobre *Big Data* em ambiente da Fog.

3.2 IMPROVING THE PERFORMANCE OF MACHINE LEARNING-BASED NIDS ON THE UNSW-NB15 DATASET

Moualla, Khorzom e Jafar (2021) propuseram um novo IDS, utilizando a técnica de aprendizado de máquina supervisionado para detectar e classificar intrusões para redes IoT. Os autores dividem o aprendizado em diversas etapas. Ele começa com o método *Synthetic Minority Oversampling Technique (SMOTE)* para equilibrar as classes presentes no *dataset* UNSW-NB15, criando instâncias sintéticas e, em seguida seleciona as melhores *features* para cada classe do conjunto de dados, utilizando o algoritmo *ExtraTree* com critério de impureza por índice de Gini. Após a seleção das *features*, utilizaram o modelo *Extreme Learning Machine (ELM)* para detecção de ataques, utilizando a técnica de *"One-Versus-All"* como um classificador binário para cada ataque. As saídas do classificador ELM se tornam parâmetros de entrada para uma camada que chamaram de Camada Totalmente Conectada onde aplica uma agregação sobre os resultados dos treinamentos. Por fim, os autores aplicaram o método de regressão logística sobre as classes, alegando melhores tomadas de decisões. Os resultados do sistema proposto foram promissores, onde os autores realizaram comparações com trabalhos correla-

tos, mostrando melhores taxas de precisão para a maioria das classes de ataque, menores taxas de falsos positivos e taxas de *recall* superiores.

3.3 COMPARATIVE ANALYSIS OF INTRUSION DETECTION ATTACK BASED ON MACHINE LEARNING CLASSIFIERS

Os autores Atnafu e Acharya (2021) realizaram uma abordagem para escolher um modelo de ML que apresenta melhores resultados em termos de precisão para um IDS. Para este estudo foram analisados três classificadores de ML. Máquina de Vetor de Suporte *(Support Vector Machine - SVM)*, classificador de *(Naive Bayes - NB)* e KNN. Os respectivos algoritmos foram testados usando o *dataset* NSL-KDD, utilizando como método de extração de *features*, o algoritmo Qui-Quadrado *(Chi Square)* e o classificador *ExtraTree*, utilizando a linguagem *Python*. Para os experimentos, foram utilizadas duas variações na quantidade de *features* e tamanho do *dataset* de treinamento e teste. Foram testadas 41 e 27 *features* respectivamente. Para o *dataset* foram feitas divisões de 10% e 20% para teste. Os resultados apresentados mostraram que o KNN é o melhor classificador para o estudo, chegando a taxas de até 100% de precisão, utilizando 27 *features* sobre 10% do *dataset* de teste. No entanto, o KNN possui um alto custo computacional em tempo de predição. Além disso, nesta pesquisa não foi realizada a identificação do tipo do ataque, apenas uma abordagem de detecção binária.

3.4 NETFLOW DATASETS FOR MACHINE LEARNING-BASED NETWORK INTRUSION DETECTION SYSTEM

Sarhan et al. (2020) propuseram e avaliaram um novo *dataset* para trabalhos e pesquisas, com a justificativa que os diferentes trabalhos do estado da arte em IDS para IoT não são avaliados de forma confiável. Cada *dataset* possui *features* independentes, sendo que algumas dessas *features* receberam algum pré-processamento em sua definição inicial. Uma etapa fundamental em um projeto de NIDS baseado em ML é a seleção de *features*, usadas na etapa de classificação. Essas *features* devem ser viáveis em termos de complexidade de contagem e extração para armazenamento e coleta eficientes. Tais *features*, devem representar informações valiosas para o modelo de ML, para que tenha um desempenho de classificação eficaz. Atualmente, a variação das informações representadas em cada *dataset* tem causado limitações no campo de pesquisa. Os autores alegam que os estudos atuais não têm a capacidade de avaliar a generalização de desempenho de um modelo de ML em vários *datasets* NIDS, utilizando um conjunto de *features* comuns. Para preencher essa lacuna, os autores escolheram quatro *datasets* recentes, UNSW-NB15, BoT-IoT, ToN-IoT e CSE-CIC-IDS2018 usando seus arquivos de captura de pacotes disponíveis publicamente, através destes *datasets*, os autores realizaram uma conversão para formato baseado em *features* NetFlow. Os autores realizaram uma avaliação de modelo, utilizando os quatro *datasets* e mais o novo *dataset* criado através da junção dos quatro *datasets*. Para finalizar o trabalho, os autores realizaram experimentos sobre os *datasets*

convertidos para o padrão NetFlow, onde utilizaram a técnica de detecção binária e multiclasse. O modelo de classificação utilizado foi algoritmo *ExtraTree*, com especificamente 50 árvores pré-definidas. Não foi especificado o porque da escolha do algoritmo e nenhuma variação na quantidade de árvores no experimento. Os resultados foram satisfatórios para um primeiro trabalho, porém deixando um campo de pesquisa em aberto para novas aplicações de modelos.

3.5 TOWARDS A STANDARD FEATURE SET FOR NETWORK INTRUSION DETECTION SYSTEM

Os autores Sarhan et al. (2021) estenderam suas pesquisas para atingirem melhores resultados com o novo *dataset* baseado nas *features* do NetFlow. Foram avaliados e comparados duas variantes do conjunto de *features* com base no protocolo NetFlow. Uma versão com 12 *features* e outra com 43 *features*. Como avaliação, foi realizado uma conversão de quatro *datasets* NIDS amplamente usados em pesquisas (UNSW-NB15, BoT-IoT, ToN-IoT, CSE-CIC-IDS2018), gerando variantes destes *datasets* (v2). Como base para comparar o desempenho de um classificador sob os *datasets*, foi utilizado o algoritmo *ExtraTree*. Enquanto o *dataset* com menor quantidade de *features* não conseguiu corresponder a um bom desempenho de classificação dos conjuntos de *features* proprietários, o *dataset* com maior número de *features*, 43, versão 2 (v2) do NetFlow, surpreendentemente atinge um desempenho de classificação consistente, em comparação com os *datasets* nativos e o conjunto NetFlow contendo 12 *features* versão 1 (v1). O benefício deste trabalho é um novo *dataset* de referência, disponibilizado para a comunidade de pesquisa, permitindo uma comparação justa de classificadores de tráfego de rede baseados em ML em diferentes *datasets* NIDS. Ter um conjunto de *features* padrão é fundamental para permitir uma avaliação mais rigorosa e completa do NIDS baseados em ML e que pode ajudar a preencher a lacuna entre a pesquisa acadêmica e a implantação prática de tais sistemas.

3.6 DISCUSSÕES SOBRE OS TRABALHOS CORRELATOS

Ao longo desta seção foram descritos os trabalhos correlatos. São diferentes propostas que utilizaram métodos de aprendizado de máquina para buscar o estado da arte em detecção de intrusão em redes IoT. Dentre os trabalhos, foram utilizadas diferentes *datasets*, evidenciando uma lacuna entre os trabalhos que visam o estado da arte. As bases de dados disponíveis e comumente utilizadas, possuem um conjunto de *features* heterogêneos, dificultando a avaliação de modelos de aprendizado de máquina para detecção de intrusão.

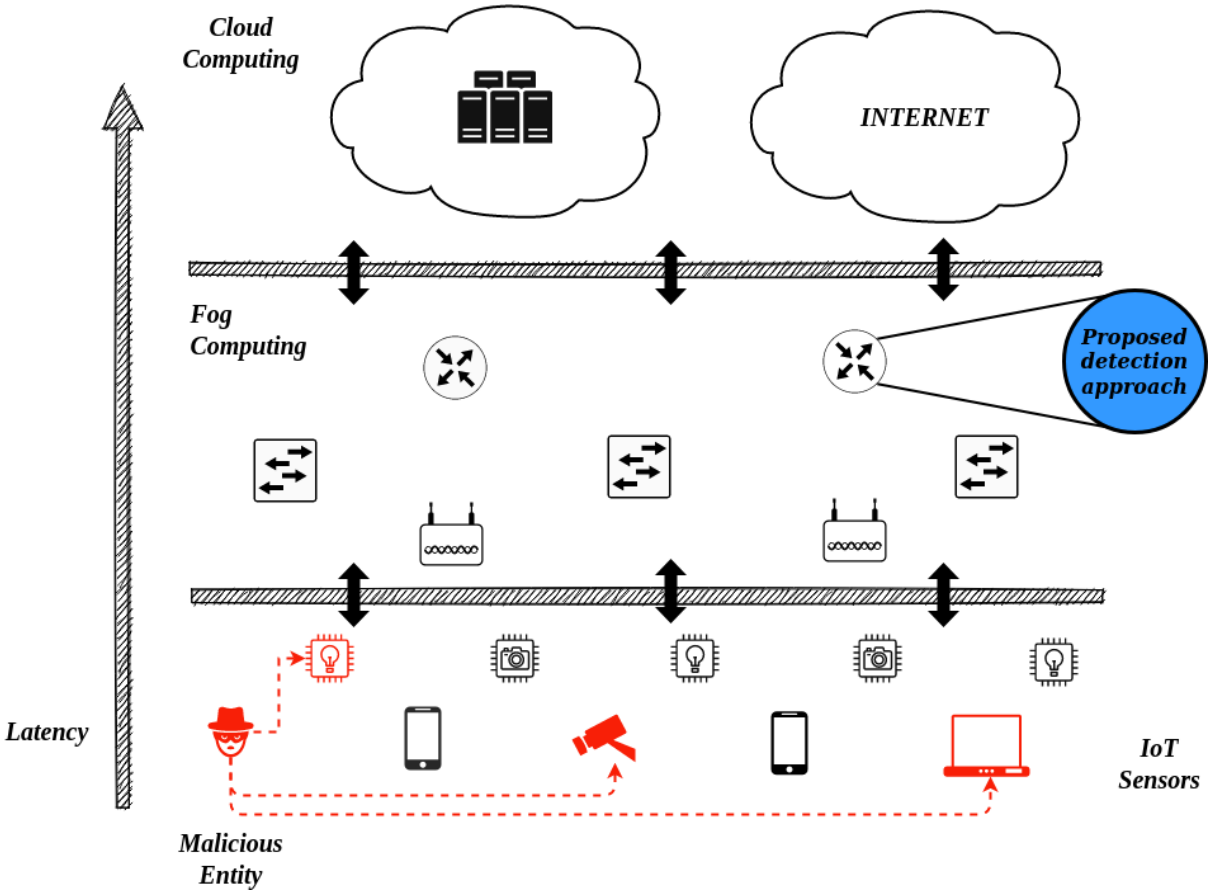
4 PROPOSTA

Considerando os conceitos, os trabalhos correlatos e a discussão apresentada, este trabalho propõe investigar a capacidade dos modelos de ML, variante da família de árvores, para realizar a detecção e classificação de intrusões, simulando um *ISP*, contendo tráfego de rede real de dispositivos *IoT* em ambientes baseados no conceito da *fog computing*.

4.1 CONTEXTUALIZAÇÃO DA PROPOSTA

O trabalho considera o contexto de um sistema *IoT*, onde utiliza-se dispositivos *fog* para prover os benefícios de pré-processamento e armazenamento para os usuários. Neste contexto, usuários maliciosos podem tentar comprometer esse sistema para roubar dados, indisponibilizar dispositivos e/ou sistemas. A Figura 3 mostra a arquitetura desse sistema, onde a abordagem proposta é posicionada no ambiente *IoT*. Este trabalho considera que o dispositivo possuí características dos atuais protocolos utilizados nas redes atuais, como o TCP/IP, para serem capturados e analisados, através do protocolo *NetFlow v9*.

Figura 3 – Arquitetura IoT.



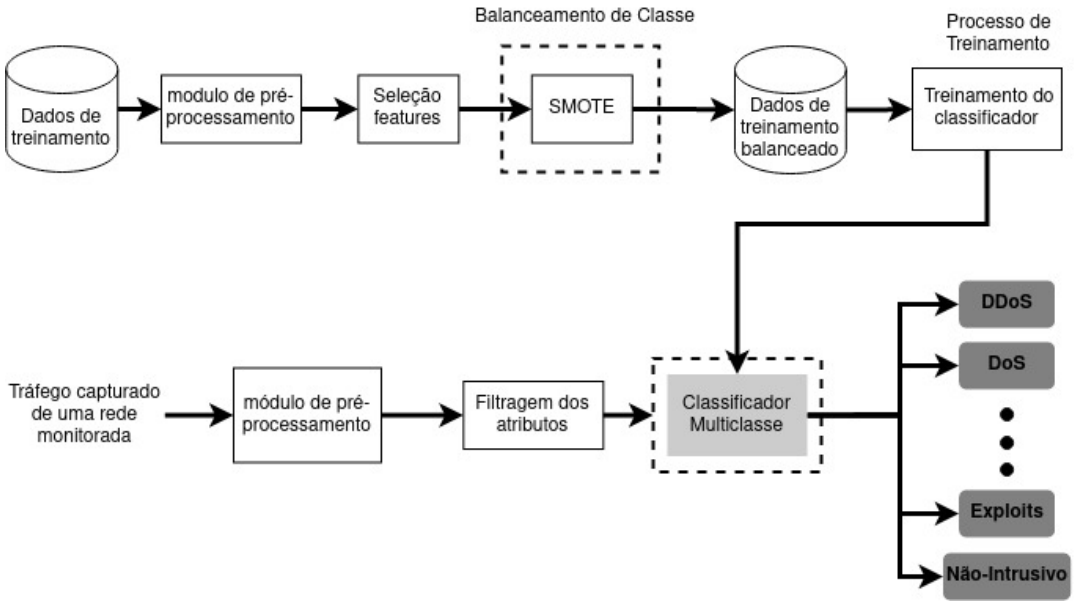
Fonte: O Autor.

A seguir, são apresentados os principais aspectos da abordagem proposta.

4.2 DESCRIÇÃO DA ABORDAGEM PROPOSTA

Nesta seção são fornecidos maiores detalhes sobre a abordagem proposta neste trabalho. A Figura 4 ilustra os processos envolvidos nas etapas de treinamento do modelo de classificação e de predição.

Figura 4 – Modelo de Classificação Proposto.



Fonte: O Autor.

Na etapa de treinamento, inicialmente os dados de treinamento passam por uma etapa de pré-processamento, onde são avaliados para verificar se há necessidade de serem formatados para o padrão do modelo de detecção e pré-processados para retirada de possíveis registros inválidos, que podem comprometer a qualidade do modelo.

A seleção de *features* é um processo opcional ao se trabalhar com métodos de aprendizado de máquina e é normalmente utilizada em situações onde se tem *datasets* com um conjunto grande de *features*. A seleção de *features* permite alcançar diversos benefícios, caso aplicável, tais como: modelo simples e explicativo, redução do *overfitting*, aumento da acurácia do modelo e por fim, redução de problemas envolvendo o custo dos modelos. O objetivo do processo é encontrar um subconjunto de *features* que seja capaz de maximizar o desempenho do modelo em relação a acertos de detecção e minimizar os custos computacionais. No entanto, o descarte de *features* importantes pode prejudicar o desempenho do modelo, desse modo, é necessário avaliar se o processo apenas reduziu o custo computacional ou se foi capaz de manter ou melhorar o desempenho de detecção. Caso os resultados de classificação com métodos de seleção de *features* sejam positivos, o processo pode ser considerado.

Para o processo descrito acima, foram considerados, nesta abordagem, três métodos de seleção de *features*. O primeiro é o algoritmo de Ganho de Informação, que consiste em um dos principais métodos da família *filter* de seleção de *features*. Os outros dois métodos

são da família *Wrapper*, o Seletor Sequencial de Features *(Sequential Feature Selector - SFS)* e o Eliminação Recursiva de *Features (Recursive Feature Elimanation - RFE)*. Eles selecionam um conjunto de *features* através de diferentes combinações para avaliação, onde possuem um modelo preditivo embutido para avaliar a combinação de *features* e atribuir um *score* baseando-se na precisão do modelo. Estes algoritmos são considerados nos experimentos realizados neste trabalho para verificar a opção ideal para compor está etapa da abordagem proposta.

Visando contornar a disparidade na proporção de tráfego entre os tipos de ataques, que podem causar problemas durante o treinamento, a abordagem possui uma etapa de balanceamento para que os modelos tenham dados suficientes para entender os padrões de ataques que têm pouca ocorrência. Uma das estratégias para trabalhar com conjuntos de dados desbalanceados é criar novas instâncias para classes minoritárias. Novas instâncias de dados podem ser geradas a partir de exemplos existentes. Seguindo esta estratégia, a abordagem conta com uma etapa de balanceamento de classes, onde é utilizado o método *Synthetic Minority Oversampling Technique* (SMOTE). O SMOTE funciona selecionando amostras juntas no espaço de recursos, desenhando uma linha entre as amostras no espaço de recursos e criando uma nova instância em um ponto ao longo dessa linha. Após aplicar o SMOTE aos dados de treinamento originais, um novo conjunto de dados com classes balanceadas é gerado. Portanto, esse processo permite que classes minoritárias consigam aumentar a quantidade de registros, consequentemente fornece maiores características para o modelo de aprendizado, permitindo realizar uma melhor classificação. O *dataset* com dados de treinamentos balanceados é então utilizado na realização do treinamento do classificador.

O classificador é responsável por utilizar os dados de treinamento para aprender as características da rede monitorada e então ser capaz de realizar detecções e classificações sobre os novos dados capturados da rede durante o monitoramento. Dessa maneira, diversos métodos de aprendizado de máquina podem ser empregados, devido as características do ambiente idealizou-se o emprego de métodos da família de Árvores de Decisão, por possuírem custo computacional relativamente menor que métodos mais complexos como *Artificial Neural Networks* e *Support Vector Machine*. Dentre os métodos de Árvores de Decisão destacam-se os a abordagem tradicional e os métodos *Random Forest* e *Extra Trees*, que são *Ensembles* de Árvores de Decisão. Estes métodos foram avaliados na abordagem proposta através dos diversos experimentos apresentados no Capítulo 5.

Com o modelo de treinamento completo e disponível a abordagem pode ser implantada em uma ambiente e iniciar o monitoramento da rede. Nesse contexto, o tráfego capturado da rede monitorada passa por uma etapa de pré-processamento para adaptar os registros para o padrão esperado pelo modelo de detecção. Além disso, os dados podem ser filtrados, de modo a submeter para o modelo de detecção apenas as *features* selecionadas durante o processo de treinamento pelo método de seleção de *features*. Os dados então são submetidos ao modelo de detecção que irá atribuir rótulos para o fluxo capturado.

5 EXPERIMENTOS

Nesta seção são descritos alguns aspectos da metodologia utilizada para avaliar a abordagem proposta. Primeiramente são apresentadas as características do *dataset* e a sua separação em conjunto de treino e teste. Em seguida, são apresentadas as métricas de classificação pertinentes para a avaliação do modelo. Por fim, são descritos os experimentos realizados.

5.1 DATASET

Neste trabalho foi utilizada a base de dados NF-UQ-NIDS-v2, previamente apresentada no Capítulo 2, para realizar os experimentos. A base que foi concebida por pesquisadores da *UQ*. O projeto tem como objetivo proporcionar uma melhor avaliação dos estudos sobre *NIDS*, utilizando diferentes *datasets*, contendo características de ataques distintas em um único *dataset*.

O *dataset* contém os seguintes ataques, conforme a tabela de distribuição 3:

- **DDoS**: O atacante captura diversas máquinas vulneráveis ao longo da internet (*bots*), onde realiza um ataque orquestrado sobre a vítima que recebe um despejo de tráfego/amplificação, deixando o servidor/aplicação indisponível.
- ***Reconnaissance***: O atacante coleta informações sobre a vítima para planejar seu ataque.
- ***Injection***: O atacante explora vulnerabilidades em bancos de dados através da injeção de parâmetros, ganhando acesso ao banco de dados.
- **DoS**: O atacante procura explorar protocolos que consumam excessivamente os recursos do servidor da vítima.
- ***Brute Force***: Ataque exaustivo que utiliza dicionários contendo diversas combinações de senhas comumentes utilizadas.
- ***Password***: O atacante utiliza uma senha de usuário comumente utilizada para diversas contas de usuário, evitando o bloqueio de contas.
- **XSS**: Ataque de injeção de código malicioso em aplicações do tipo Web.
- ***Infiltration***: A partir de um arquivo malicioso executado pela vítima, o atacante realiza uma varredura na rede, buscando outras vulnerabilidades.
- ***Exploits***: Injeção de códigos maliciosos, utilizados após os ataques de *Reconnaissance* e *Scanning*, para ganhar privilégios no sistema.
- ***Theft***: O atacante visa obter dados confidenciais, como roubo de dados e keylogging.

- ***Scanning***: O atacante procura por vulnerabilidades na rede, geralmente com o uso de ferramentas.
- ***Fuzzers***: O atacante pode utilizar essa ferramenta para testar um conjunto de parâmetros em aplicações, buscando estouros de *buffer*, *strings* invalidas e etc.
- ***Backdoor***: Um tipo de *Malware* que nega os procedimentos normais de autenticação para acessar um sistema.
- ***Bot***: O atacante utiliza *malwares* do tipo cavalo de troia para tentar controlar uma ou mais máquinas da vítima.
- ***Generic***: Ataque que serve como uma etapa dentro de um processo de infecção por *Ransomware*, através do protocolo *Remote Desktop Protocol* (RDP).
- ***Analysis***: O atacante realiza uma análise do *host*/rede da vítima. Analisando portas e serviços rodando no nó vítima.
- ***ShellCode***: O atacante insere trechos de código arbitrário, explorando uma vulnerabilidade no software.
- **MITM**: O atacante intercepta a comunicação entre dois *hosts*, forjando o endereço IP de uma das partes.
- ***Worms***: Um programa independente, da família *Malware*, que se espalha pela rede de forma rápida, com o objetivo de roubar dados.
- ***Ransomware***: Um vírus que codifica o sistema de arquivos da vítima, deixando-o inoperante.

Na Tabela 3 são listadas as *features* presentes no *dataset*.

Tabela 3 – Conjunto de *features* presentes no *dataset* NF-UQ-NIDS-v2.

Feature	Descrição
IPV4_SRC_ADDR	Endereço IPv4 de Origem
L4_SRC_PORT	Porta de Origem
IPV4_DST_ADDR	Endereço IPv4 de Destino
L4_DST_PORT	Porta de Destino
PROTOCOL	Identificador do Protocolo de Rede
L7_PROTO	Protocolo de aplicação
IN_BYTES	Número de bytes de Entrada
IN_PKTS	Número de Pacotes de Entrada
OUT_BYTES	Número de bytes de Saída

OUT_PKTS	Número de Pacotes de Saída
TCP_FLAGS	Cumulativo de todos os sinalizadores TCP
CLIENT_TCP_FLAGS	Cumulativo de todos os sinalizadores TCP Clientes
SERVER_TCP_FLAGS	Cumulativo de todos os sinalizadores TCP Servidor
FLOW_DURATION_MILLISECONDS	Duração do Fluxo em Milisegundos
DURATION_IN	Duração do fluxo do cliente para o servidor (Milisegundos)
DURATION_OUT	Duração do fluxo do cliente para o servidor (Milisegundos)
MIN_TTL	TTL Mínimo em Pacotes de Entrada do Fluxo
MAX_TTL	TTL Máximo em Pacotes de Entrada do Fluxo
LONGEST_FLOW_PKT	Maior Pacote em bytes do Fluxo
SHORTEST_FLOW_PKT	Menor Pacote em bytes do Fluxo
MIN_IP_PKT_LEN	Menor tamanho de Pacote Dentro do Fluxo
MAX_IP_PKT_LEN	Maior tamanho de Pacote Dentro do Fluxo
SRC_TO_DST_SECOND_BYTES	Quantidade de Bytes/seg (Origem e Destino)
DST_TO_SRC_SECOND_BYTES	Quantidade de Bytes/seg (Destino >Origem)
RETRANSMITTED_IN_BYTES	Quantidade de bytes de Fluxo TCP retransmitidos (Origem >Destino)
RETRANSMITTED_IN_PKTS	Quantidade de Pacotes de Fluxo TCP retransmitidos (Origem >Destino)
RETRANSMITTED_OUT_BYTES	Quantidade de bytes de Fluxo TCP retransmitidos (Destino >Origem)
RETRANSMITTED_OUT_PKTS	Quantidade de Pacotes de Fluxo TCP retransmitidos (Destino >Origem)
SRC_TO_DST_AVG_THROUGHPUT	Taxa de Transferência média de bytes/seg entre Origem >Destino
DST_TO_SRC_AVG_THROUGHPUT	Taxa de Transferência média de bytes/seg pelo Destino >Origem
NUM_PKTS_UP_TO_128_BYTES	Quantidade de Pacotes com Tamanho <= 128 bytes
NUM_PKTS_128_TO_256_BYTES	Quantidade de Pacotes com Tamanho >128 <= 256 bytes
NUM_PKTS_256_TO_512_BYTES	Quantidade de Pacotes com Tamanho >256 <= 512 bytes

NUM_PKTS_512_TO_1024_BYTES	Quantidade de Pacotes com Tamanho >512 <= 1024 bytes
NUM_PKTS_1024_TO_1514_BYTES	Quantidade de Pacotes com Tamanho >1024 <= 1514 bytes
TCP_WIN_MAX_IN	Maior Janela TCP do Fluxo (Origem >Destino)
TCP_WIN_MAX_OUT	Maior Janela TCP do Fluxo (Destino >Origem)
ICMP_TYPE	Tipo de Mensagem ICMP * 256 bits + Código ICMP
ICMP_IPV4_TYPE	Tipo de Mensagem ICMP
DNS_QUERY_ID	Identificação de Consulta DNS
DNS_QUERY_TYPE	Tipo de Consulta DNS
DNS_TTL_ANSWER	Tempo de Cache de Resposta DNS
FTP_COMMAND_RET_CODE	Código de Retorno de Comandos FTP

5.2 PRÉ-PROCESSAMENTO DO DATASET

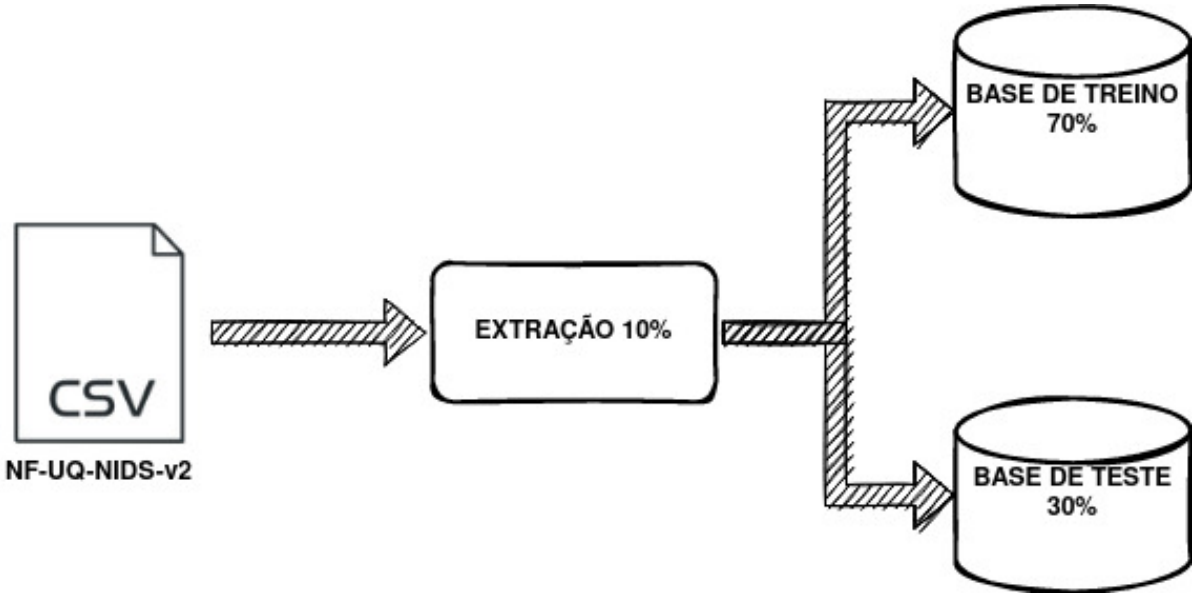
O *dataset* NF-UQ-NIDS-v2 possuí um único arquivo do tipo csv, contendo aproximadamente 12,8GB de dados. Devido ao grande tamanho do arquivo e a quantidade de registros, houve a necessidade de utilizar um subconjunto de 10% do *dataset* para realizar os experimentos. A plataforma utilizada nos experimentos possuí limitações quanto ao carregamento de grandes arquivos. A extração do subconjunto foi realizada de forma aleatória, porém mantendo a proporção na distribuição de registros por classes de ataque.

Foi necessário também realizar a remoção de duas *features* para evitar influencia externa no processo de detecção. Uma delas representava o rótulo binário (ataque ou não ataque) e a outra fazia referencia ao *dataset* original de onde a instância foi obtida.

O *dataset* reduzido e processado, com apenas 10% dos dados, é então utilizado nos experimentos. Para cada experimento optou-se por utilizar o método de amostragem *Hold-Out 70-30*, onde o *dataset* é dividido de forma aleatória e proporcional em 70% para treino e 30% para teste dos modelos. Essa divisão é realizada para tornar possível realizar a avaliação do modelo com dados diferentes dos utilizados no treinamento.

A Figura 5 apresenta um diagrama ilustrando o processo.

Figura 5 – Pré-processamento do *dataset*.



Fonte: O Autor

.

5.3 MÉTRICAS DE AVALIAÇÃO

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 5.1, apresentada a seguir:

$$ACC = \frac{VP + VN}{VP + VN + FP + FN} \tag{5.1}$$

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 5.2.

$$PRE = \frac{VP}{VP + FP} \tag{5.2}$$

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 5.3 demonstra como é calculada a sensibilidade de um modelo.

$$Recall = \frac{VP}{VP + FN} \tag{5.3}$$

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 5.4.

$$F1 - Score = 2 * \frac{Precisão * Recall}{Precisão + Recall} \tag{5.4}$$

5. **Acurácia Balanceada:** é uma métrica interessante para avaliar o desempenho da detecção em *datasets* não balanceados. É definido como a média do *recall* obtida em cada classe, conforme pode ser observado na Equação 5.5. Onde R_i corresponde a ao *recall* (R) obtido considerando a i-ésima (i) classe presente no *dataset* e n é a quantidade de classes existentes.

$$Acurcia\ Balanceada\ = \frac{\sum_{i=1}^{n} R_i}{n} \tag{5.5}$$

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 treinar o modelo com o conjunto de treinamento.

2. **Tempo de teste:** corresponde ao tempo, em segundos, que o algoritmo precisou para realizar a predição de todas as instâncias do conjunto de teste.

5.4 DESCRIÇÃO DOS EXPERIMENTOS REALIZADOS

Para fins de avaliação da abordagem proposta, foram realizados diversos experimentos considerando diferentes configurações de seletores de *features* e classificadores. Os métodos de seleções de *features* consideradas foram o Ganho de Informação, Seleção Sequencial de *Features* e Eliminação Recursiva de *Features*. Além disso, foram avaliados os classificadores *Decision Tree*, *Extra Tree* e *Random Forest*.

O primeiro classificador utilizado foi o método *Decision Tree*, ele possui diversos parâmetros que precisam ser definidos. O critério de Gini foi escolhido como função para medir a qualidade de uma divisão. Os critérios suportados são Gini para a impureza Gini e entropia para o ganho de informação. Foi escolhido o critério de Gini por ser o critério padrão da biblioteca. O parâmetro *max_depth* que indica a profundidade máxima permitida da árvore, foi definido como *None*, neste caso os nós são expandidos até que todas as folhas sejam puras ou até que todas as folhas contenham menos de *min_samples_split* amostras. O *min_samples_split* foi definido como 2, desse modo, são necessárias pelo menos duas amostras para dividir um

nó interno. Além disso, o número mínimo de amostras necessárias para formar um nó folha (*min_samples_leaf*) foi definido com 1, ou seja, um ponto de divisão em qualquer profundidade só será considerado se deixar pelo menos uma amostra em cada um dos ramos esquerdo e direito. Esses três parâmetros também foram definidos de acordo com o padrão da biblioteca e buscam formar uma árvore completa sem restrições de tamanho.

Também foi utilizado nos experimento o classificador *Random Forest*, ele consiste em um método *Ensemble* que agrega os resultados de várias *Decision Tree* decorrelacionadas acumuladas dentro de uma "floresta" para produzir os resultados da classificação. O principal parâmetro deste método é o *n_estimators*, ele indica o número de árvores na floresta. Neste trabalho o *n_estimators* foi definido como 100, sendo o valor padrão da biblioteca. O parâmetro *max_features* indica o número de atributos candidatos selecionados aleatoriamente em cada nó e foi definido como $sqrt(N)$, sendo N o numero de *features* do *dataset*. Além disso, por ser um método composto por diversas *Decision Tree*, neste trabalho optou-se por utilizar, para as 100 árvores da *Random Forest* , as mesmas configurações de parâmetros utilizadas no primeiro classificador *Decision Tree* único.

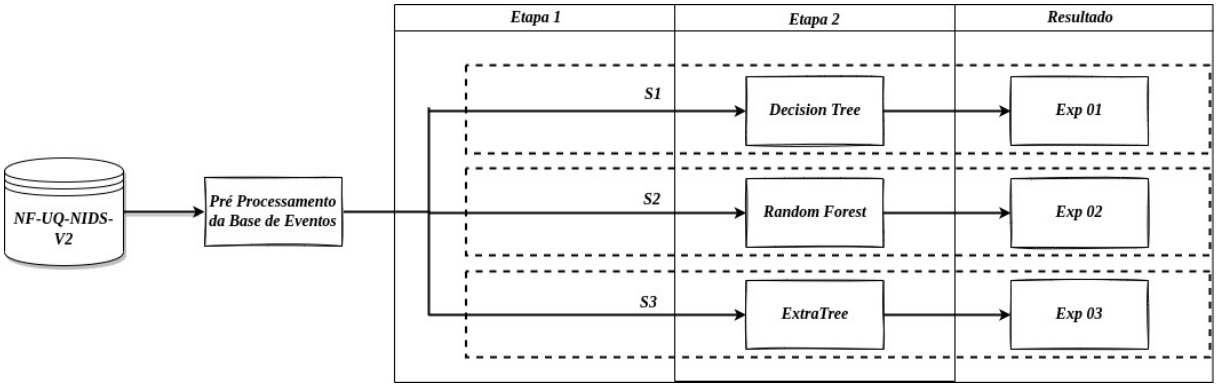
Por fim, o outro classificador utilizado foi a *Extra Tree*, ele também é um método *Ensemble* composto por diversas *Decision Trees* acumuladas dentro de uma "floresta". para produzir os resultados da classificação. Ela é bastante semelhante em operação a *Random Forest* e varia principalmente na forma de construir as árvores dentro da floresta. Assim como na *Random Forest*, um subconjunto aleatório de *max_features features* candidatas é usadas, no entanto, neste caso, em vez de procurar os pontos de corte mais discriminativos, os pontos de corte são sorteados aleatoriamente para cada *feature* candidata e o melhor desses pontos de corte gerados é escolhido aleatoriamente como o regra de divisão. Para a *Extra Tree* os parâmetros *n_estimators* e *max_features* também foram definidos como 100 e $sqrt(N)$, respectivamente. Do mesmo modo, utilizou-se, para as 100 árvores da *Extra Tree*, as mesmas configurações de parâmetros utilizadas no primeiro classificador *Decision Tree* único.

A seguir são apresentados os experimentos realizados considerando as diferentes combinações de seletores de *features* e classificadores.

5.4.1 Experimentos com todo o conjunto de *features*

Os primeiros experimentos foram realizados com *dataset* completo, contento todo o conjunto de *features*. O objetivo foi ter resultados de controle para comparação com os resultados obtidos utilizando métodos de seleção de *features*. A Figura 6 apresenta uma ilustração dos primeiros experimentos realizados. Observa-se que o *dataset* NF-UQ-NIDS-V2 passou por um processo de pré-processamento, etapa responsável por formatar e pré-processar para retirada de possíveis registros inválidos, que podem comprometer a qualidade do modelo. Posteriormente o *dataset* pré-processado foi utilizado nos experimentos Exp01, Exp02 e Exp03, para avaliar os classificadores. No experimento Exp01 foi empregado o classificador *Decision Tree*, no Exp02 a *Random Forest* e no Exp03 a *Extra Tree*.

Figura 6 – Experimentos realizados com o *dataset* NF-UQ-NIDS-V2 sem Seleção de *features*.

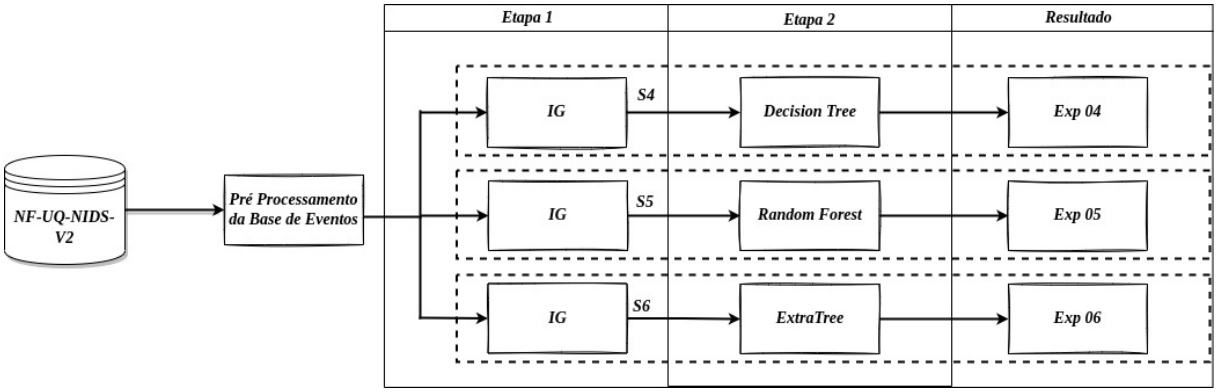


Fonte: O Autor.

5.4.2 Experimentos com o conjunto de *features* selecionadas pelo método de Ganho de Informação

Foram realizados 3 experimentos com *dataset* filtrado através do método de seleção de *features* por Ganho de Informação. A Figura 7 apresenta uma ilustração dos experimentos realizados. O *dataset* NF-UQ-NIDS-V2 também passou por um processo de pré-processamento e posteriormente foi utilizado nos experimentos Exp04, Exp05 e Exp06, para avaliar a seleção de *features* e os classificadores. Na primeira etapa dos experimentos foi aplicado a seleção por Ganho de Informação no *dataset* para a seleção de *features*. O método de seleção por Ganho de Informação gera um ranking de *features*, do maior ganho de informação para o menor. Utilizou-se como *threshold*, para descarte de *features* o valor de 0.10, desse modo, apenas as *features* com ganho de informação inferior a 0.10 foram descartados do *dataset*. Por ser um *dataset* criado com base nas *features* mais importantes extraídas do protocolo NetFlow, buscou-se não descartar muitas *features*. O *dataset* reduzido foi então utilizado na Etapa 2 dos experimentos, onde foram avaliados os classificadores *Decision Tree* no experimento Exp04, a *Random Forest* no experimento Exp05 e a *Extra Tree* no experimento Exp06.

Figura 7 – Experimentos realizados com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método de Ganho de Informação.

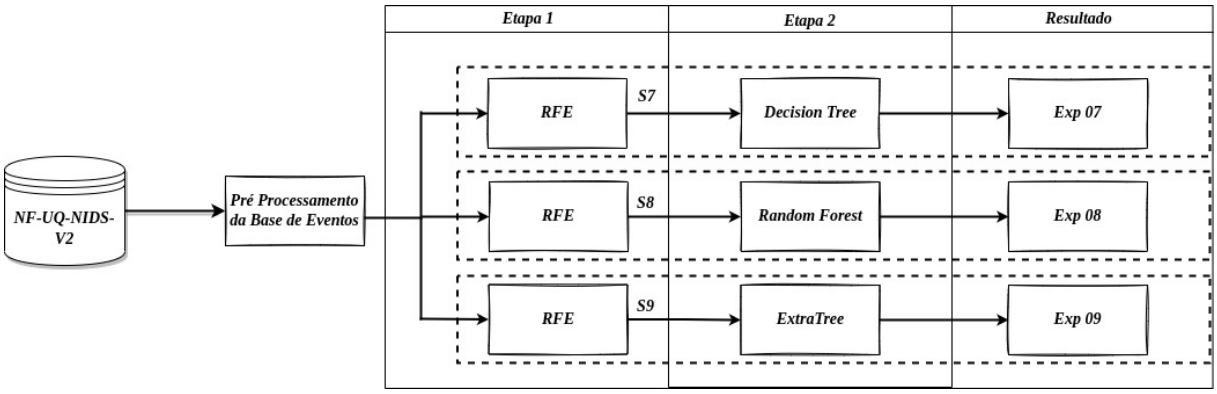


Fonte: O Autor.

5.4.3 Experimentos com o conjunto de *features* selecionadas pelo método de Eliminação Recursiva de *Features*

A Figura 8 ilustra os três experimentos realizados com o *dataset* filtrado através do método de Eliminação Recursiva de *Features*. O *dataset* NF-UQ-NIDS-V2 também passou por um processo de pré-processamento e posteriormente foi utilizado nos experimentos Exp07, Exp08 e Exp09, para avaliar a seleção de *features* e os classificadores, conforme pode ser observado na Figura 8.

Figura 8 – Experimentos realizados com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método Eliminação Recursiva de *Features*.



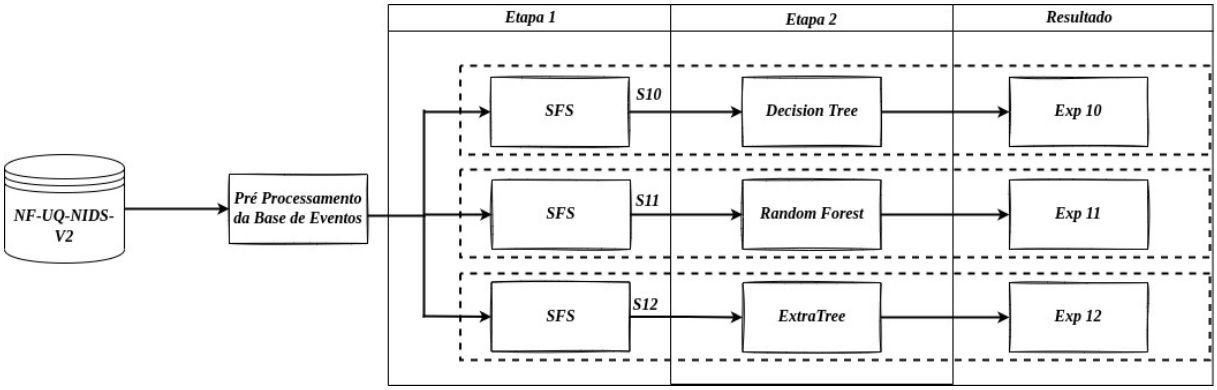
Fonte: O Autor.

O *dataset* foi reduzido na Etapa1 pelo método de Eliminação Recursiva de *Features* e então foi utilizado para avaliar os classificadores *Decision Tree* no experimento Exp07, a *Random Forest* no experimento Exp08 e a *Extra Tree* no experimento Exp09. O método de Eliminação Recursiva de *Features* também consiste em um processo extremamente custoso se o objetivo for encontrar a quantidade ideal de *features* para o subconjunto. Buscando um menor processamento, também utilizou-se como quantidade de *features* selecionadas o valor de $3/4$ da quantidade de *features* existentes, ou seja, 32 *features*. Sendo assim, a Eliminação Recursiva de *Features* também gera um subconjunto contendo as 32 melhores *features* do *dataset* e descarta 11 *features*.

5.4.4 Experimentos com o conjunto de *features* selecionadas pelo método de Seleção Sequencial de *Features*

Os últimos três experimentos realizados consideraram o método de Seleção Sequencial de *Features*, conforme apresentado na Figura 9. O *dataset* NF-UQ-NIDS-V2 também passou por um processo de pré-processamento e posteriormente foi utilizado nos experimentos Exp10, Exp11 e Exp12, para avaliar a seleção de *features* e os classificadores.

Figura 9 – Experimentos realizados com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método Seleção Sequencial de *Features*.



Fonte: O Autor.

Na primeira etapa dos experimentos foi aplicado a Seleção Sequencial de *Features* no *dataset*. A estratégia adotada com o método de Seleção Sequencial de *Features* não busca encontrar a quantidade ideal de *features* para o subconjunto, pois este seria um processo extremamente custoso. Desse modo, utilizou-se como quantidade de *features* selecionadas o valor de $3/4$ da quantidade de *features* existentes, neste caso foi definido como 32 *features*. Sendo assim, a Seleção Sequencial de *Features* gera um subconjunto contendo as 32 melhores *features* do *dataset* e descarta 11 *features*. O *dataset* reduzido foi então utilizado na Etapa 2 dos experimentos, onde foram avaliados os classificadores *Decision Tree* no experimento Exp10, a *Random Forest* no experimento Exp11 e a *Extra Tree* no experimento Exp12.

5.5 MATERIAIS UTILIZADOS

Neste trabalho foi utilizado a plataforma Microsoft Azure para realização dos experimentos. É uma plataforma em nuvem que disponibiliza um ambiente de execução em Python. A configuração da máquina utilizada contava com processador Intel® Xeon® Platinum 8370C de terceira geração (Ice Lake), 28 GB de memória de 56 GB de armazenamento. Para a criação e execução da abordagem proposta, foram baseadas em bibliotecas Python. A biblioteca *Pandas* foi utilizada no carregamento do *dataset* NF-UQ-NIDS-v2, armazenando em uma variável para manipulação. A biblioteca *Numpy* foi utilizada para manipulação de *arrays* e matrizes, onde serviu para agrupar os ataques em suas respectivas classes. Também foi utilizada a biblioteca *Scikit-Learn* para instanciar e manipular os classificadores baseados em árvores de decisão, onde também possibilitou o uso de métodos de avaliação de *features* como o Ganho de Informação.

6 RESULTADOS

Nesta seção são apresentados os resultados dos 12 experimentos realizados com o *dataset* NF-UQ-NIDS-V2. Os experimentos foram realizados com diferentes configurações em relação aos métodos de seleção de *features* e aos classificadores. Cada combinação representada nas figuras do capitulo anterior foram testadas e os resultados obtidos foram utilizados para calcular as métricas descritas na Seção 5.3.

A Tabela 4 apresenta os resultados dos experimentos Exp01, o qual utilizou uma DT para realizar a classificação do *dataset* NF-UQ-NIDS-V2 sem seleção de *features*. Os resultados obtidos na detecção de classe normal, *Bot*, *DDoS*, *DoS*, *Reconnaissance* se mostraram muito satisfatórios, alcançando taxas de *recall*, precisão e *F1-Score* próximas a 100%.

Tabela 4 – Resultados do Experimento Exp01 com o *dataset* NF-UQ-NIDS-V2 sem seleção de *features* e utilizando uma *DT*.

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,57%	99,40%	99,48%
Analysis	16,67%	27,27%	20,69%
Backdoor	98,64%	97,68%	98,16%
Bot	99,95%	100,00%	99,98%
Brute Force	95,99%	98,04%	97,00%
DDoS	99,75%	99,67%	99,71%
DoS	99,82%	99,44%	99,63%
Exploits	77,78%	85,60%	81,50%
Fuzzers	78,19%	87,66%	82,65%
Generic	43,04%	44,16%	43,59%
Infilteration	45,94%	53,96%	49,62%
Reconnaissance	99,62%	99,63%	99,62%
Shellcode	65,22%	71,43%	68,18%
Theft	96,97%	96,97%	96,97%
Worms	00,00%	00,00%	00,00%
Injection	87,52%	90,22%	88,85%
MITM	05,50%	56,19%	10,00%
Password	95,94%	96,16%	96,05%
Ransonware	91,51%	97,00%	94,17%
Scanning	97,61%	98,38%	97,99%
XSS	96,98%	96,28%	96,63%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
422,4359	0,6476	99,04%	80,72%	99,18%

Além disso, os resultados obtidos para as classes *Backdoor*, *Brute Force*, *Theft*, *Password*, *Ransonware*, *Scanning* e XSS também apresentaram resultados satisfatórios, com taxas de *recall* superiores a 96%. Por fim, classes como *Analysis*, *Generic*, *Worms* e MITM demonstraram uma deficiência da arquitetura na classificação destes ataques, levando em consideração os resultados ruins obtidos no *recall*. O *recall* de cada uma destas classes apresenta a taxa de eventos classificados corretamente como de determinada classe, dentre todos os eventos intru-

sivos existentes daquela classe no *dataset*. Portanto, o modelo de detecção foi capaz de detectar apenas 27% dos ataques *Analysis*, 44% dos ataques *Generic* e 56% dos ataques MITM. Além disso, é necessário destacar o desempenho ruim do modelo para detectar ataques *Worms*, nenhum ataque deste tipo foi detectado. Este desempenho ruim em algumas classes prejudicou a métrica de acurácia balanceada, que ficou na casa dos 80%. A métrica de acurácia não leva em consideração o desbalanceamento existente entre as classes, desse modo, traz uma falsa percepção de detecção ótima. Esta métrica é influenciada por classes que possuem muitas instâncias, como a classe *Benign*. Neste caso, a abordagem apresentou 99% de acurácia mesmo não detectando nenhum ataque do tipo *Worm*, pois, existem poucas instâncias desse tipo de ataque e muitas instâncias da classe *Benign*, que teve uma alta taxa de detecção. Portanto a métrica de acurácia balanceada é mais justa e realista, pois leva em consideração o desequilíbrio na quantidade de instancias existentes em cada classe, desse modo, uma taxa de detecção ruim em algumas classes tem influência sobre o desempenho geral do modelo.

Na Tabela 5 são apresentados os resultados obtidos pela abordagem RF com o *dataset* NF-UQ-NIDS-V2 sem seleção de *features*.

Tabela 5 – Resultados do Experimento Exp02 com o *dataset* NF-UQ-NIDS-V2 sem seleção de *features* e utilizando uma *RF*.

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,65%	99,33%	99,49%
Analysis	22,22%	36,36%	27,59%
Backdoor	99,80%	97,88%	98,83%
Bot	100,00%	100,00%	100,00%
Brute Force	97,72%	98,04%	97,88%
DDoS	99,90%	99,67%	99,79%
DoS	99,81%	99,44%	99,62%
Exploits	77,30%	92,23%	84,11%
Fuzzers	77,96%	94,16%	85,29%
Generic	60,87%	54,55%	57,53%
Infilteration	40,13%	51,59%	45,15%
Reconnaissance	99,74%	99,74%	99,62%
Shellcode	70,83%	80,95%	75,56%
Theft	98,44%	95,45%	96,92%
Worms	66,67%	40,00%	50,00%
Injection	92,72%	90,70%	91,70%
MITM	05,01%	62,83%	09,27%
Password	97,08%	96,89%	96,98%
Ransonware	98,97%	96,00%	97,46%
Scanning	98,10%	98,76%	98,43%
XSS	96,80%	98,44%	97,61%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
3954,4303	62,3194	99,14%	84,90%	99,32%

Esta abordagem apresentou resultados similares a abordagem DT para a grande maioria das classes de ataques. No entanto podemos destacar a melhora na detecção dos ataques

Analysis, *Generic*, *MITM* e principalmente *Worms*. O ataque *Worm* não foi detectado pelo modelo DT do Experimento Exp01, já a abordagem RF foi capaz de detectar 40% dos ataques desta classe. No entanto, estas classes de ataques ainda tiveram resultados inferiores a 70%. A melhora na detecção desses ataques também influenciou na métrica de acurácia balanceada que alcançou 84,9%.

Na Tabela 6 são apresentados os resultados obtidos pela abordagem ET com o *dataset* NF-UQ-NIDS-V2 sem seleção de *features*. Assim como as abordagens anteriores, este modelo também foi capaz de obter ótimas taxas de detecção para a maioria dos ataques e também apresentou dificuldades na detecção de ataques *Analysis*, *Generic*, *MITM* e *Worms*, no entanto, com uma leve melhora em suas taxas de detecção. Destaca-se a melhora na detecção de ataques *Worms* que alcançou 60% com 75% de precisão.

Tabela 6 – Resultados do Experimento Exp03 com o *dataset* NF-UQ-NIDS-V2 sem seleção de *features* e utilizando o classificador ExtraTree.

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,58%	99,41%	99,50%
Analysis	20,83%	45,45%	28,57%
Backdoor	99,80%	97,68%	98,73%
Bot	100,00%	100,00%	100,00%
Brute Force	97,56%	98,01%	97,78%
DDoS	99,88%	99,67%	99,77%
DoS	99,82%	99,44%	99,63%
Exploits	76,01%	92,34%	83,38%
Fuzzers	75,49%	92,64%	83,19%
Generic	52,54%	40,26%	45,59%
Infilteration	42,68%	42,48%	45,58%
Reconnaissance	99,71%	99,72%	99,71%
Shellcode	90,00%	85,71%	87,80%
Theft	98,46%	96,97%	97,71%
Worms	75,00%	60,00%	66,67%
Injection	92,40%	89,57%	90,97%
MITM	04,88%	59,29%	09,01%
Password	96,75%	96,52%	96,64%
Ransonware	97,94%	95,00%	96,45%
Scanning	97,60%	98,57%	98,08%
XSS	96,46%	98,32%	97,38%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
2204,8072	100,5811	99,12%	85,09%	99,26%

Buscando melhorias na detecção dos ataques e no consumo de recursos, foram aplicadas técnicas de seleção de *features* para selecionar um conjunto menor de *features* para serem utilizado pelo modelo de classificação. A primeira técnica empregada foi o Ganho de Informação, os resultados obtidos com essa técnica são apresentados nas Tabelas 7, 8 e 9. Observa-se na Tabela 7 que a abordagem DT apresentou resultados melhores nos experimentos com seleção de *features* de Ganho de informação em relação ao primeiro experimento sem seleção de *features*. A classe de ataques *Worms* que não havia sido detectada agora teve 20% dos seus ataques detectados. Os resultados superiores podem ser observados na métrica de acurácia balanceada que no experimento sem seleção de *features* foi de 80% e neste com seleção por Ganho de Informação foi de 83,7%.

Além disso, é importante destacar aqui que o classificador DT com a seleção de *features* por Ganho de Informação foi capaz de alcançar o menor tempo de treino e de teste dentre todas as abordagens. Para o treino foram necessários 240 segundos, uma redução significativa em relação aos 422 segundos que a DT sem seleção de *features* levou para realizar o seu treinamento.

Tabela 7 – Resultados do Experimento Exp04 com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método de Ganho de Informação e utilizando uma *DT*

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,56%	99,42%	99,49%
Analysis	20,00%	36,36%	25,81%
Backdoor	99,02%	97,49%	98,25%
Bot	100,00%	99,93%	99,97%
Brute Force	96,22%	98,01%	97,11%
DDoS	99,80%	99,65%	99,73%
DoS	99,84%	98,95%	99,39%
Exploits	78,34%	84,34%	81,23%
Fuzzers	77,95%	85,71%	81,65%
Generic	47,95%	45,45%	45,59%
Infilteration	46,99%	53,49%	50,03%
Reconnaissance	99,60%	99,57%	99,59%
Shellcode	72,73%	76,19%	74,42%
Theft	96,97%	96,97%	96,97%
Worms	100,00%	20,00%	33,33%
Injection	87,21%	90,67%	88,91%
MITM	03,64%	85,84%	06,98%
Password	95,68%	96,31%	95,99%
Ransonware	96,12%	99,00%	97,54%
Scanning	97,57%	98,35%	97,96%
XSS	97,28%	96,11%	96,69%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
240,1409	0,4395	98,90%	83,70%	99,20%

Os resultados dos experimentos da abordagem RF com o *dataset* NF-UQ-NIDS-V2 utilizando seleção de *features* por Ganho de Informação são apresentados na Tabela 8. Observa-se que esta seleção de *features* também contribuiu para melhorias nos resultados em relação a abordagem RF do Experimento Exp02 sem seleção de *features*. A acurácia balanceada alcançada no Experimento Exp02 foi de 84,9% e a do Experimento Exp05 com Ganho de informação foi de 85,3%. No entanto, destaca-se negativamente o tempo de teste, no experimento Exp02 a RF necessitou de apenas 62 segundos para realizar a classificação do conjunto de teste, já no experimento Exp05 foram necessários 73 segundos.

Tabela 8 – Resultados do Experimento Exp05 com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método de Ganho de Informação e utilizando *RF*

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,61%	99,31%	99,46%
Analysis	21,74%	45,45%	29,41%
Backdoor	100,00%	97,68%	99,46%
Bot	100,00%	99,93%	99,97%
Brute Force	97,56%	98,04%	97,80%
DDoS	99,85%	99,71%	99,78%
DoS	99,89%	98,96%	99,42%
Exploits	77,31%	92,69%	84,30%
Fuzzers	77,13%	94,16%	84,80%
Generic	56,52%	50,65%	53,42%
Infilteration	39,54%	54,34%	45,77%
Reconnaissance	99,72%	99,72%	99,72%
Shellcode	66,67%	76,19%	71,11%
Theft	96,46%	96,97%	96,71%
Worms	50,00%	20,00%	28,57%
Injection	89,01%	91,70%	90,34%
MITM	03,68%	88,50%	07,07%
Password	96,76%	96,98%	96,87%
Ransonware	98,97%	96,00%	97,46%
Scanning	97,62%	98,80%	98,21%
XSS	97,34%	96,63%	96,98%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
3795,8481	73,6415	98,95%	85,35%	99,28%

Na Tabela 9 são apresentados os resultados obtidos pela abordagem ET no experimento com seleção de *features* por Ganho de Informação. A abordagem ET foi capaz de alcançar uma acurácia balanceada de 85,09% no Experimento Exp03 sem seleção de *features* e de 85,4% neste Experimento Exp06 com seleção de *features* por Ganho de Informação. Não houve uma grande melhoria no desempenho de detecção geral, porém, o tempo necessário para o treinamento da abordagem foi reduzido de 2204 segundos para 1754 segundos. Apesar da melhora no tempo de treino, o tempo de teste aumentou de 100 segundos para 116 segundos.

Tabela 9 – Resultados do Experimento Exp06 com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método de Ganho de Informação e utilizando o classificador ExtraTrees

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,58%	99,16%	99,37%
Analysis	16,00%	36,36%	22,22%
Backdoor	99,61%	97,49%	98,54%
Bot	100,00%	100,00%	100,00%
Brute Force	96,84%	98,06%	97,45%
DDoS	99,84%	99,71%	99,77%
DoS	99,89%	98,96%	99,42%
Exploits	76,59%	92,00%	83,59%
Fuzzers	73,57%	91,56%	81,58%
Generic	50,85%	38,96%	44,12%
Infilteration	33,38%	52,47%	40,80%
Reconnaissance	99,71%	99,70%	99,71%
Shellcode	78,26%	85,71%	81,82%
Theft	96,97%	96,97%	96,97%
Worms	50,00%	40,00%	44,44%
Injection	88,27%	91,19%	89,71%
MITM	03,61%	86,28%	06,94%
Password	96,62%	96,61%	96,61%
Ransonware	98,00%	98,00%	98,00%
Scanning	97,20%	98,61%	97,90%
XSS	97,14%	96,44%	96,79%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
1754,5366	116,8267	98,86%	85,44%	99,23%

Na Tabela 10 são apresentados os resultados obtidos pela abordagem DT no experimento com a Eliminação Recursiva de *Features*. A abordagem DT não obteve boas taxas de detecção, levando em consideração a acurácia balanceada em comparação com outros seletores de *features*, onde alcançou apenas 81,18%. A RFE pode ter excluído *features* com papéis fundamentais na classificação, pois piorou comparado com a DT utilizando Ganho de Informação, que obteve 83,71% de Acurácia Balanceada.

Destaca-se como ponto interessante desta abordagem o fato de alcançar um dos menores tempos de teste dentre todas as abordagens avaliadas. Neste experimento, com seleção de *features* por RFE, o classificador DT foi capaz de classificar todo conjunto de teste em apenas 0,45 segundos.

Tabela 10 – Resultados do Experimento Exp07 com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método *RFE* utilizando uma *DT*

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,55%	99,42%	99,48%
Analysis	17,39%	36,36%	23,53%
Backdoor	98,44%	97,30%	97,86%
Bot	100,00%	100,00%	100,00%
Brute Force	96,35%	97,98%	97,16%
DDoS	99,76%	99,68%	99,72%
DoS	99,81%	99,42%	99,61%
Exploits	79,26%	85,14%	82,09%
Fuzzers	77,89%	84,63%	81,12%
Generic	53,52%	49,35%	51,35%
Infilteration	46,57%	52,26%	49,25%
Reconnaissance	99,63%	99,64%	99,64%
Shellcode	69,57%	76,19%	72,73%
Theft	95,52%	96,97%	96,24%
Worms	00,00%	00,00%	00,00%
Injection	87,63%	90,21%	88,90%
MITM	04,97%	53,54%	09,09%
Password	95,92%	96,14%	96,03%
Ransonware	94,12%	96,00%	95,05%
Scanning	97,46%	98,44%	97,95%
XSS	96,92%	96,14%	96,53%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
388,9889	0,4541	99,04%	81,18%	99,17%

Na Tabela 11 são apresentados os resultados do experimento Exp 08, obtidos pela abordagem RF com seleção de *features* com RFE. Neste experimento a RF teve resultados muito semelhantes ao experimento Exp02 em termos gerais. O Exp02 que utilizou RF com todo o conjunto *features* diferenciou-se com um tempo de treino maior, cerca de 3954 segundos. Portanto, nesta abordagem com RFE a RF apresentou uma redução no tempo para treinar o modelo, o treinamento nesta abordagem durou aproximadamente 3529 segundos.

No entanto, a RF do experimento Exp 08 obeteve 84,55% de Acurácia Balanceada, uma leve redução em relação ao experimento Exp05, onde utilizou-se RF com Ganho de informação e obteve-se 85,36%.

Tabela 11 – Resultados do Experimento Exp08 com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método *RFE* utilizando *RF*

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,65%	99,33%	99,49%
Analysis	20,00%	36,36%	25,81%
Backdoor	99,80%	97,88%	98,83%
Bot	100,00%	100,00%	100,00%
Brute Force	97,64%	98,04%	97,84%
DDoS	99,90%	99,68%	99,79%
DoS	99,82%	99,40%	99,61%
Exploits	77,36%	92,57%	84,29%
Fuzzers	76,54%	93,94%	84,35%
Generic	56,72%	49,35%	52,78%
Infilteration	40,28%	52,15%	45,45%
Reconnaissance	99,75%	99,75%	99,75%
Shellcode	64,00%	76,19%	69,57%
Theft	98,46%	96,97%	97,71%
Worms	66,67%	40,00%	50,00%
Injection	91,77%	91,03%	91,40%
MITM	04,72%	63,27%	08,78%
Password	97,03%	96,91%	96,97%
Ransonware	98,97%	96,00%	97,46%
Scanning	98,00%	98,76%	98,38%
XSS	96,93%	98,07%	97,50%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
3529,0972	62,4198	99,12%	84,55%	99,31%

Na Tabela 12 é apresentado os resultados do experimento Exp09 com o seletor RFE e classificador ET. Nesse experimento obteve-se uma Acurácia Balanceada de 83,49%, inferior comparada ao experimento Exp03 e ao experimento Exp06, no Exp03 utilizou-se a ET sem seleção de *features* e obteve-se 85,10%. Porém, no restante dos valores e as taxas de Acurácia e Precisão ficaram bem próximas, como o tempo de treino e teste também.

Este experimento também apresentou taxa de Acurácia Balanceada inferior em comparação com o Exp06, onde foi utilizando ET com seletor por Ganho de Informação. Além disso, a abordagem necessitou de um maior tempo para treinar o modelo, cerca de 1820 segundos. No entanto, foi capaz de reduzir o tempo de teste de 116 segundos para 90 segundos.

Tabela 12 – Resultados do Experimento Exp09 com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método *RFE* utilizando o classificador ExtraTree

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,59%	99,34%	99,47%
Analysis	18,18%	36,36%	24,24%
Backdoor	99,80%	97,68%	98,73%
Bot	100,00%	100,00%	100,00%
Brute Force	97,28%	98,04%	97,65%
DDoS	99,87%	99,67%	99,77%
DoS	99,82%	99,41%	99,61%
Exploits	76,48%	92,91%	83,90%
Fuzzers	73,88%	93,07%	82,38%
Generic	48,21%	35,06%	40,60%
Infilteration	39,09%	44,85%	41,77%
Reconnaissance	99,73%	99,74%	99,73%
Shellcode	80,95%	80,95%	80,95%
Theft	96,97%	96,97%	96,97%
Worms	66,67%	40,00%	50,00%
Injection	91,04%	90,14%	90,59%
MITM	04,69%	60,18%	08,71%
Password	96,77%	96,60%	96,69%
Ransonware	97,96%	96,00%	96,97%
Scanning	97,52%	98,55%	98,03%
XSS	96,65%	97,80%	97,22%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
1820,5410	90,1163	99,08%	83,49%	99,25%

Na Tabela 13 são apresentados os resultados através do experimento do classificador DT utilizando o seletor de *features* SFS. A DT manteve o padrão dos demais experimentos, obtendo uma taxa de Acurácia Balanceada inferior aos demais classificadores experimentados neste trabalho. Portanto, a DT não foi capaz de obter um desempenho melhor de classificação mesmo com a seleção de *features*, ficando abaixo das abordagens RF e ET em todos os experimentos. Além disso, a taxa de Acurácia Balanceada neste experimento ficou abaixo da taxa obtida pela DT no experimento Exp04, onde utilizou-se seleção de *features* por Ganho de Informação.

Considerando apenas os experimentos com DT é possível observar que a abordagem que obteve os melhores resultados foi a do Exp04, onde aplicou-se a seleção de *features* por Ganho de Informação. Ela obteve a maior taxa de Acurácia Balanceada e os menores tempos de treino e teste.

Tabela 13 – Resultados do Experimento Exp10 com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método *SFS* utilizando uma *DT*

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,54%	99,36%	99,45%
Analysis	20,00%	27,27%	23,08%
Backdoor	99,83%	99,68%	98,25%
Bot	100,00%	100,00%	100,00%
Brute Force	96,22%	97,98%	97,09%
DDoS	99,75%	99,68%	99,71%
DoS	99,81%	99,45%	99,62%
Exploits	76,54%	86,51%	81,22%
Fuzzers	80,49%	85,71%	83,02%
Generic	43,75%	45,45%	44,59%
Infilteration	42,55%	52,09%	46,84%
Reconnaissance	99,62%	99,57%	99,60%
Shellcode	75,00%	85,71%	80,00%
Theft	98,46%	96,97%	97,71%
Worms	00,00%	00,00%	00,00%
Injection	87,76%	90,12%	88,93%
MITM	05,86%	58,41%	10,65%
Password	95,71%	96,18%	95,94%
Ransonware	94,12%	96,00%	95,05%
Scanning	97,46%	98,41%	97,93%
XSS	97,01%	96,15%	96,58%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
290,6010	0,5825	99,02%	81,36%	99,16%

O experimento Exp11 que utilizou uma RF com o seletor SFS, apresentou uma taxa de Acurácia Balanceada de 85,54%, portanto, superior as abordagens com seleção de *features* por Ganho de informação e por RFE, que apresentaram Acurácia Balanceada de 85,36% e 84,55%, respectivamente. Além disso, a abordagem do experimento Exp11 necessitou de 3146 segundos para treinar o modelo, portanto, um tempo de treino menor que a abordagem do experimento Exp08, que necessitou de 3529 segundos. Dentre todos os experimentos utilizando o classificador RF, o Exp11 foi o experimento que apresentou a maior taxa Acurácia Balanceada.

Tabela 14 – Resultados do Experimento Exp11 com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método *SFS* utilizando *RF*

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,63%	99,46%	99,55%
Analysis	19,05%	36,36%	25,00%
Backdoor	99,80%	97,88%	98,83%
Bot	100,00%	100,00%	100,00%
Brute Force	97,41%	97,98%	97,69%
DDoS	99,90%	99,67%	99,78%
DoS	99,81%	99,44%	99,62%
Exploits	76,56%	93,71%	84,28%
Fuzzers	77,86%	95,89%	85,94%
Generic	56,67%	44,16%	49,54%
Infilteration	47,10%	47,21%	47,16%
Reconnaissance	99,72%	99,69%	99,70%
Shellcode	66,67%	85,71%	75,00%
Theft	98,46%	96,97%	97,71%
Worms	100,00%	60,00%	75,00%
Injection	92,46%	90,96%	91,70%
MITM	04,97%	61,50%	09,20%
Password	96,94%	96,88%	96,91%
Ransonware	98,97%	96,00%	97,46%
Scanning	98,00%	98,79%	98,39%
XSS	96,85%	98,24%	97,54%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
3146,1072	63,1483	99,17%	85,54%	99,32%

Os resultados obtidos no experimento Exp12 são apresentados na Tabela 15. Este experimento contou com a seleção de *features* por SFS e classificador ET. Essa abordagem obteve Acurácia Balanceada de 83,49%, similar portanto, à abordagem ET com seleção de *features* por RFE. O principal destaque desta abordagem foi o tempo de teste que foi de 86 segundos, o que consiste no menor tempo de teste dentre as abordagens ET. Portanto, a partir destes resultados é possível inferir que houve um maior descarte de *features* neste experimento, no entanto, possivelmente *features* interessantes foram descartadas, o que acarretou em um pior desempenho de classificação do modelo.

Tabela 15 – Resultados do Experimento Exp12 com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método *SFS* utilizando o classificador ExtraTree

Classes	**Precisão**	***Recall***	***F1-Score***
Benign	99,56%	99,53%	99,55%
Analysis	16,00%	36,36%	22,22%
Backdoor	99,61%	97,68%	98,64%
Bot	100,00%	100,00%	100,00%
Brute Force	96,60%	97,98%	97,29%
DDoS	99,88%	99,66%	99,77%
DoS	99,82%	99,44%	99,63%
Exploits	76,34%	92,91%	83,81%
Fuzzers	72,93%	93,29%	81,86%
Generic	38,18%	27,27%	31,82%
Infilteration	52,68%	38,74%	44,65%
Reconnaissance	99,69%	99,71%	99,70%
Shellcode	80,00%	95,24%	86,96%
Theft	98,46%	96,97%	97,71%
Worms	66,67%	40,00%	50,00%
Injection	92,38%	89,94%	91,14%
MITM	04,89%	58,41%	09,03%
Password	96,53%	96,49%	96,51%
Ransonware	97,00%	97,00%	97,00%
Scanning	97,48%	98,57%	98,02%
XSS	96,54%	98,18%	97,36%

Treino (s)	**Teste (s)**	**Acurácia**	**Acurácia B.**	**Precisão**
2030,2043	86,4997	99,15%	83,49%	99,27%

6.1 DISCUSSÕES

Nesta seção, são apresentados e discutidos os resultados obtidos pelas diferentes configurações executadas em diversos experimentos. Para isso foi construída uma tabela comparativa entre as métricas gerais mais relevantes a serem discutidas e foi criado um gráfico sobre as taxas de Acurácia Balanceada obtidas pelas abordagens nos experimentos. Por fim, são discutidos outros aspectos que valem ser pontuados.

A Tabela 16 apresenta os resultados obtidos pelas diferentes configurações da abordagem nos 12 experimentos em relação a Acurácia, Acurácia Balanceada, Precisão, tempo de treino e tempo de teste. Os valores de tempo de treino nesta tabela foram arredondados para uma melhor visualização.

Tabela 16 – Tabela comparativa entre os experimentos realizados

Experimentos	Treino (seg)	Teste (seg)	Acurácia	Acurácia B.	Precisão
Exp01 (DT)	422	0,65	99,04%	80,72%	99,18%
Exp02 (RF)	3.954	62,32	99,14%	84,91%	99,32%
Exp03 (ET)	2.205	100,58	99,12%	85,09%	99,26%
Exp04 (IG+DT)	240	0,44	98,90%	83,70%	99,20%
Exp05 (IG+RF)	3.796	73,64	98,95%	85,35%	99,28%
Exp06 (IG+ET)	1.754	116,83	98,86%	85,44%	99,23%
Exp07 (RFE+DT)	389	0,45	99,04%	81,18%	99,17%
Exp08 (RFE+RF)	3.529	62,42	99,12%	84,55%	99,31%
Exp09 (RFE+ET)	1.820	90,12	99,08%	83,49%	99,25%
Exp10 (SFS+DT)	291	0,58	99,02%	81,36%	99,16%
Exp11 (SFS+RF)	3.146	63,15	99,17%	85,54%	99,32%
Exp12 (SFS+ET)	2.030	86,50	99,15%	83,49%	99,27%

Em todos os experimentos a Acurácia e a Precisão apresentaram altas taxas, próximas ou acima de 99%. Esses valores foram alcançados principalmente porque estas métricas são fortemente influenciadas pelo resultados das classes que possuem muitas instâncias e pouco influenciadas pelas classes que possuem poucas instâncias. Conforme a Tabela 2, o *dataset* NF-UQ-NIDS-v2 possui muitas instâncias de trafego benigno e de ataques de negação de serviço, estas classes tiveram elevadas taxas de detecção, desse modo, a acurácia e precisão foram extremamente altas, mesmo tendo classes que o modelo não conseguiu ter bom desempenho.

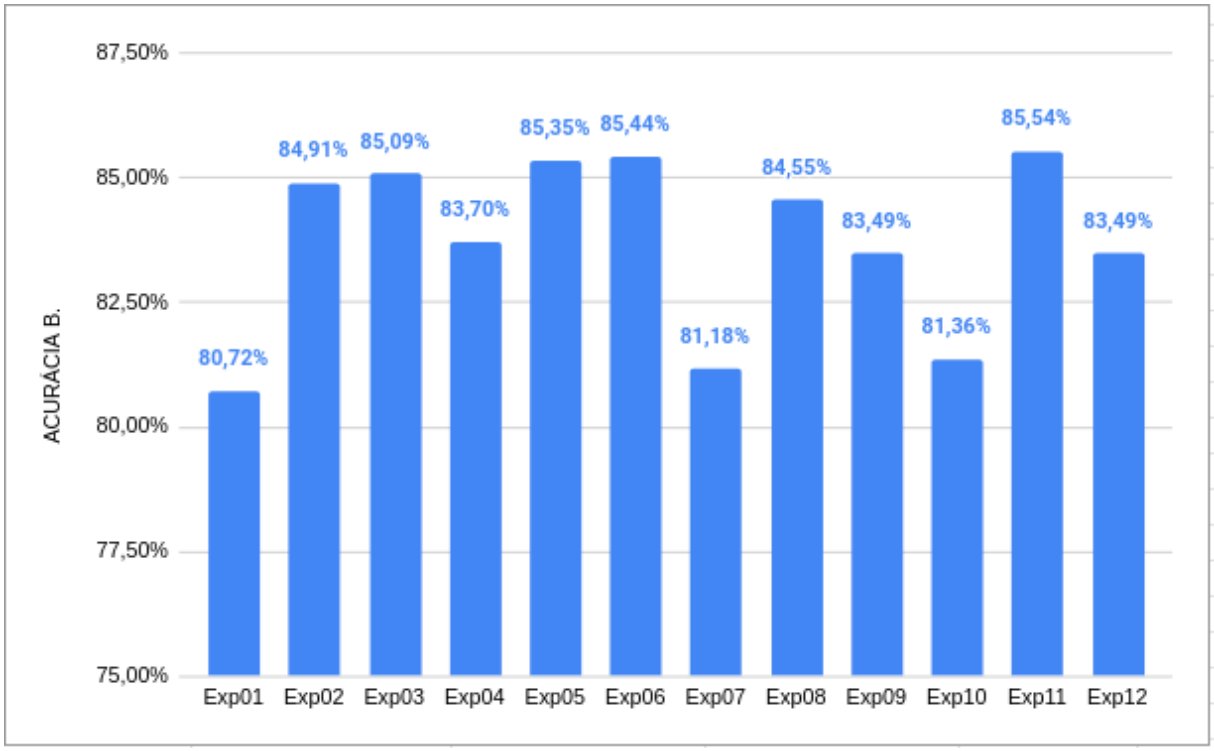
Já a métrica de Acurácia Balanceada contorna o problema anterior, pois realiza a média das taxas de detecção das classes. Os experimentos com DT apresentaram Acurácia Balanceada inferior aos demais métodos em todos os cenários, com ou sem seleção de *features*. Os experimentos utilizando DT (Exp01, Exp04, Exp07 e Exp10) não apresentaram boas taxas de detecção em comparação com os demais experimentos. Conforme visto na seção 6, a DT não conseguiu classificar com precisão classes como *Worms* e *MITM*.

Conforme pode ser observado no gráfico apresentado na Figura 10, a DT apresentou Acurácia Balanceada de 80,72% (Exp01), 83,71% (Exp04), 81,18% (Exp07) e 81,36% (Exp10). Portanto, o melhor desempenho da DT foi no Exp04, onde foi utilizado o Ganho de Informação para selecionar as *features*.

Os experimentos utilizando a RF como classificador apresentaram taxas de Acurácia Balanceada muito próximas em todos as abordagens, 83,91% (Exp02), 85,36% (Exp05), 84,55% (Exp08) e 85,54% (Exp11), conforme pode ser visualizado na Figura 10. A maior taxa de Acurácia Balanceada foi alcançada no Exp11, onde o *dataset* foi reduzido pela Seleção Sequential de *Features*.

O melhor resultado dentre todas as abordagens avaliadas sem seleção de *features* foi a ET, ela obteve 85,10% de Acurácia Balanceada, No experimento com seleção de *features* por Ganho de Informação (Exp06) essa métrica aumentou para 85,44%. No entanto, o desempenho da ET foi menor nos experimentos Exp09 e Exp12, onde *dataset* foi reduzido através da Seleção Sequencial de *Features* e Eliminação Recursiva de *Features*, respectivamente.

Figura 10 – Gráfico comparativo da Acurácia Balanceada dos diferentes modelos gerados.



Fonte: O Autor.

Com relação aos tempos, a DT mostrou-se a abordagem menos custosa. Sua estrutura e operação justificam seus baixos tempos de treino e teste. A iteração do algoritmo é mínima comparado com as demais abordagens. Realizando uma média do tempo de treino, considerando todos os experimentos entre a DT e RF, a DT é quase dez vezes mais rápida. Em questão de atualizações de modelos, conforme o tráfego de entrada é registrado e processado, a DT apresenta melhor desempenho. A mesma interpretação pode ser considerada para o tempo de teste que é muito inferior em relação as demais abordagens, logo a DT, caso tenha boas taxas de detecção, seria a mais recomendada para detectar ataques em ambientes com restrições de recursos e/ou com a necessidade de resposta à incidentes de forma imediata.

As abordagens com RF se mostraram as mais custosas em relação ao tempo de treino, em todos os experimentos precisou de mais de 3.000 segundos para completar a etapa. No processo de detecção de intrusão as atualizações de modelos possuem grande relevância. A dinamicidade e elasticidade presente em redes IoT contempla a inserção e remoção de diversos nós ou dispositivos. Essas mudanças podem alterar o comportamento da rede constantemente e tornar necessário atualizar os modelos de detecção com maior frequência. Portanto, quanto menor o tempo necessário para realizar o treinamento do modelo, menor será o tempo esperando um novo modelo ou utilizando um modelo desatualizado. No experimento Exp11, RF com SFS como abordagem de seleção de *Features*, o tempo de treino foi de 3.146 segundos, apresentando uma boa redução em relação ao tempo de treino com a base original Exp02 que levou 3.954 segundos.

O tempo de teste dos experimentos com RF, no geral foram maior que as DT, porém melhor do que os experimentos com a ET, que possui um tempo mais elevado. O tempo de teste indica o tempo necessário para a análise e classificação do tráfego, portanto é um atributo muito importante.

A ET foi menos custosa que a RF em tempo de treino, para alguns experimentos o tempo da ET foi a metade da RF. Nos experimentos utilizando a seleção de *features* por Ganho de Informação, a RF precisou de 3795 segundos para ser treinada no experimento Exp05 e a ET precisou de 1754 segundos no experimento Exp06.

De modo geral, destaca-se que as abordagens com seleção de *features* foram capazes de melhorar a Acurácia Balanceada para todos os classificadores em relação aos experimentos utilizando o *dataset* original sem utilização da técnica de seleção de *features*. Além disso, a DT e a ET tiveram reduções em seus tempos de treino e teste. A RF alcançou a redução apenas no tempo de treino. Desse modo, é possível concluir que a quantidade de *features* pode influenciar diretamente no tempo necessário para treinar e testar os classificadores baseados em árvores de decisão.

Para discutir a respeito da melhor abordagem, deve-se considerar na escolha o cenário e aplicação onde ela será implantada. Caso haja necessidade de resposta imediata de um incidente, a DT tende a ser a melhor abordagem, principalmente a configuração avaliada no experimento Exp04, que utilizou Ganho de Informação para redução de *features*, ela apresentou a melhor acurácia balanceada dentre as DTs e os menores tempos de treino e de predição. No entanto, a DT apresentou taxas de Acurácia Balanceada inferiores as abordagens *Ensemble*. Além disso, as Árvores de Decisão estão sujeitas a problemas de ajuste excessivo *(overfitting)* por problemas de ruído durante o treinamento (T.K.; ANNAVARAPU; BABLANI, 2021). A RF e a ET podem ser consideradas como alternativas robustas a ruídos de classificação e treinamento, os quais são comuns para aplicações de detecção de intrusão, além de apresentarem maior robustez a *overfitting*.

Em um cenário onde há necessidade de maior confiabilidade na classificação, a RF do experimento Exp11, utilizando a Seleção Sequencial de *Features*, classificou melhor por ter apresentado uma boa taxa de Acurácia Balanceada, com um tempo de teste relativamente baixo. Como a abordagem está inserida na fog, o treinamento que apresentou um alto tempo poderia ser designado para a nuvem, onde tem o recurso de processamento maior.

De modo geral, as abordagens avaliadas neste trabalho apresentaram excelentes resultados quanto a falsos positivos, as taxas de *recall* para a classe benigna ficaram acima de 99%, indicando que de todos os eventos benignos existentes no *dataset*, mais de 99% foram identificados como benignos, ou seja a taxa de falsos positivos foi baixa.

Do mesmo modo, as abordagens avaliadas neste trabalho apresentaram excelentes resultados quanto a falsos negativos, ou seja, todos os experimentos apresentaram taxas de precisão para o trafego benigno acima dos 99,16%, evidenciando que a quantidade de instancias classificadas erroneamente como benigna foi baixa. Isso evidencia que houve erros de classificação entre as classes de ataque.

Os erros entre os ataques podem ser justificados por diversas variáveis. Primeiro, devido a base de criação do *dataset* NF-UQ-NIDS-V2 ter sido concebida através de uma mistura de quatro *datasets* distintos, contendo registros de diferentes tipos de ataques, desde a nível de rede até de aplicações. A Figura 2 reforça a heterogeneidade das *features* presentes em cada *dataset*, onde agrupados em um conjunto padrão de *features* podem representar uma perda significativa de informação. Outro ponto importante foi o desbalanceamento de classes, por mais que utilizamos a técnica de SMOTE no processo de treinamento, quando passamos para a etapa de teste, existem poucos registros para serem testados de algumas classes minoritárias.

A grande quantidade de classes de ataques presentes no *dataset* explorado, também é um grande desafio para o classificador, onde essa grande quantidade torna mais complexa a tarefa de classificação, aumentando as chances de uma classificação errônea. Uma parcela significativa de trabalhos relacionados a NIDS baseados em ML para IoT, apresentam desafios de detecção multiclasse entre poucas classes. Se levarmos em consideração a detecção dos ataques mais comuns, como DDoS, DoS, BoT, Força Bruta e Ransonware, nossos experimentos também performaram com boas taxas de Recall, métrica na qual é a mais adequada para discussões de resultados sobre classes.

7 CONCLUSÃO E TRABALHOS FUTUROS

É evidente a consolidação da IoT em nosso dia a dia ao longo dos próximos anos. Com a demanda cada vez maior por soluções inteligentes nesta área, é possível inferir que esforços nos setores de desenvolvimento e pesquisa, para soluções seguras, serão imprescindíveis. De forma que consiga meios de segurança eficazes, muitas áreas de atuação sofrerão forte adesão desta tecnologia.

Neste trabalho foram abordados alguns pontos onde os dispositivos IoT possuem limitações em relação a segurança, especialmente devido sua baixa capacidade computacional, o que impossibilita a aplicabilidade de métodos tradicionais de IDS. O paradigma da *fog* foi relacionado aos recursos computacionais utilizados neste trabalho, como uma camada de agregação visando realizar um pré-processamento para a nuvem ou até mesmo suportando toda a carga de análise de fluxos de rede.

Este trabalho teve como objetivo propor uma abordagem para detectar e identificar intrusões em ambientes de computação fog e IoT. A abordagem proposta é baseada na detecção de anomalias, onde visa detectar e identificar desvios no comportamento padrão dos fluxos de rede. A partir disso, investigou-se a capacidade de três classificadores baseados em Árvores de Decisão para detecção multiclasse. Além disso, utilizou-se técnicas de seleção de *features* para minimizar tanto os tempos de treinamento e testagem do classificador, como maximizar as taxas de detecção e classificação dos fluxos.

De modo a encontrar os seletores de *features* e classificadores que alcançam melhor desempenho no cenário de *Fog* e IoT, foram realizados doze experimentos com diferentes configurações de classificadores e seletores de *features*. Os resultados obtidos através destes experimentos demonstram que o Exp11, que utilizou como seletor de *features* o SFS e o classificador RF, apresentou a melhor taxa de Acurácia Balanceada entre todos os experimentos deste trabalho, ou seja, em um cenário onde há a uma maior necessidade em classificar os ataques com confiabilidade no modelo. Por contrapartida, o seu tempo de treino, que representa o tempo necessário para atualizar e disponibilizar um modelo atualizado foi superior aos classificadores DT e ET. Em um contexto onde há necessidade de utilizar modelos atualizados com criticidade, as abordagens com DT parecem ser mais recomendáveis, devido ao seu custo inferior em relação aos demais classificadores considerados.

Trabalhos futuros podem ser realizados em cima de redes acadêmicas federadas, onde há uma gama de serviços de rede IoT em diversos setores, como *datacenters*, câmeras, sensores industriais e etc. Com essa rede agregada disponível, sugerimos a implantação de um serviço de captura de fluxos com NetFlow ou IPFIX em um ponto de agregação da rede, para que a partir desse processo, seja gerado um novo *dataset* disponibilizado para o campo de pesquisa da Universidade Federal de Santa Catarina (UFSC), como a âmbito nacional, em cooperação com as demais universidades interconectadas com a Rede Nacional de Ensino e Pesquisa (RNP).

REFERÊNCIAS

ATNAFU, S.; ACHARYA, A. Comparative analysis of intrusion detection attack based on machine learning classifiers. **Indian Journal of Artificial Intelligence and Neural Networking**, v. 1, p. 22–28, 04 2021.

BONOMI, F. et al. Fog computing and its role in the internet of things. In: **Proceedings of the first edition of the MCC workshop on Mobile cloud computing**. [S.l.: s.n.], 2012. p. 13–16.

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

BREIMAN, L. et al. Classification and regression trees–crc press. **Boca Raton, Florida**, 1984.

CAMHI, J. Former cisco ceo john chambers predicts 500 billion connected devices by 2025. **Business Insider**, 2015.

CHEN, Y.-D.; AZHARI, M. Z.; LEU, J.-S. Design and implementation of a power consumption management system for smart home over fog-cloud computing. In: IEEE. **2018 3rd International Conference on Intelligent Green Building and Smart Grid (IGBSG)**. [S.l.], 2018. p. 1–5.

CHO, S. et al. **Learning from machine learning in accounting and assurance**. [S.l.]: American Accounting Association, 2020.

ELRAWY, M. F.; AWAD, A. I.; HAMED, H. F. Intrusion detection systems for iot-based smart environments: a survey. **Journal of Cloud Computing**, SpringerOpen, v. 7, n. 1, p. 1–20, 2018.

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

JOHN, G. H.; KOHAVI, R.; PFLEGER, K. Irrelevant features and the subset selection problem. In: **MACHINE LEARNING: PROCEEDINGS OF THE ELEVENTH INTERNATIONAL**. [S.l.]: Morgan Kaufmann, 1994. p. 121–129.

KAREGOWDA, A. G.; MANJUNATH, A.; JAYARAM, M. Comparative study of attribute selection using gain ratio and correlation based feature selection. **International Journal of Information Technology and Knowledge Management**, v. 2, n. 2, p. 271–277, 2010.

KERR, D. R.; BRUINS, B. L. **Network flow switching and flow data export**. [S.l.]: Google Patents, 2001. US Patent 6,243,667.

LAWAL, M. A.; SHAIKH, R. A.; HASSAN, S. R. Security analysis of network anomalies mitigation schemes in iot networks. **IEEE Access**, IEEE, v. 8, p. 43355–43374, 2020.

LIU, H.; LANG, B. Machine learning and deep learning methods for intrusion detection systems: A survey. **Applied Sciences**, v. 9, n. 20, 2019. ISSN 2076-3417. Disponível em: https://www.mdpi.com/2076-3417/9/20/4396.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National . . . , 2011.

MIRANDA, C. et al. A collaborative security framework for software-defined wireless sensor networks. **IEEE Transactions on Information Forensics and Security**, IEEE, v. 15, p. 2602–2615, 2020.

MOUALLA, S.; KHORZOM, K.; JAFAR, A. Improving the performance of machine learning-based network intrusion detection systems on the unsw-nb15 dataset. **Computational Intelligence and Neuroscience**, Hindawi, v. 2021, 2021.

NAMDEV, N.; AGRAWAL, S.; SILKARI, S. Recent advancement in machine learning based internet traffic classification. **Procedia Computer Science**, Elsevier, v. 60, p. 784–791, 2015.

NISHANI, L.; BIBA, M. Machine learning for intrusion detection in manet: a state-of-the-art survey. **Journal of Intelligent Information Systems**, Springer, v. 46, n. 2, p. 391–407, 2016.

PATWARY, A. A.-N. et al. Towards secure fog computing: A survey on trust management, privacy, authentication, threats and access control. **Electronics**, Multidisciplinary Digital Publishing Institute, v. 10, n. 10, p. 1171, 2021.

PENG, K. et al. Intrusion detection system based on decision tree over big data in fog environment. **Wireless Communications and Mobile Computing**, Hindawi, v. 2018, 2018.

PRIYADARSHINI, R.; BARIK, R. K. A deep learning based intelligent framework to mitigate ddos attack in fog environment. **Journal of King Saud University-Computer and Information Sciences**, Elsevier, 2019.

QUINLAN, J. R. Induction of decision trees. In: SHAVLIK, J. W.; DIETTERICH, T. G. (Ed.). **Readings in Machine Learning**. [S.l.]: Morgan Kaufmann, 1990. Originally published in *Machine Learning* 1:81–106, 1986.

QUINLAN, J. R. **C4. 5: programs for machine learning**. [S.l.]: Elsevier, 2014.

ROKACH, L. Decision forest: Twenty years of research. **Information Fusion**, Elsevier, v. 27, p. 111–125, 2016.

SARHAN, M. et al. Netflow datasets for machine learning-based network intrusion detection systems. **arXiv preprint arXiv:2011.09144**, 2020.

SARHAN, M. et al. Towards a standard feature set of nids datasets. **arXiv preprint arXiv:2101.11315**, 2021.

SATYANARAYANAN, M. A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets. **GetMobile: Mobile Computing and Communications**, ACM New York, NY, USA, v. 18, n. 4, p. 19–23, 2015.

SHAFI, Q. et al. Fog-assisted sdn controlled framework for enduring anomaly detection in an iot network. **IEEE Access**, IEEE, v. 6, p. 73713–73723, 2018.

SUNDHARI, S. S. A knowledge discovery using decision tree by gini coefficient. In: IEEE. **2011 International Conference on Business, Engineering and Industrial Applications**. [S.l.], 2011. p. 232–235.

T.K., B.; ANNAVARAPU, C. S. R.; BABLANI, A. Machine learning algorithms for social media analysis: A survey. **Computer Science Review**, v. 40, p. 100395, 2021. ISSN 1574-0137. Disponível em: https://www.sciencedirect.com/science/article/pii/S1574013721000356.

TORDERA EVA, M.-B. X. M. et al. Do we all really know what a fog node is? current trends towards an open definition. **Computer Communications**, Elsevier, v. 109, p. 117–130, 2017.

TSAI, C.-F. et al. Intrusion detection by machine learning: A review. **expert systems with applications**, Elsevier, v. 36, n. 10, p. 11994–12000, 2009.

Apêndices

APÊNDICE A – ARTIGO

Este apêndice contém o artigo resultante deste trabalho no formato da SBC (Sociedade Brasileira de Computação).

Abordagem baseada em Árvores de Decisão para detecção e identificação de intrusões em ambientes da Internet das Coisas baseados em Computação em Nevoeiro

Júlio Cesar Franke Fagundes[1]

[1]Departamento de Informática – Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – CEP 88040-900 – Florianópolis – SC – Brazil

franke.julio@grad.ufsc.br, franke.julio@gmail.com

Abstract. *The Internet of Things is a paradigm that has been on the rise in recent years. The large number of devices and their resource limitations challenge researchers and developers in the area of information security. Attacks on connected devices have become common lately. Thus, countermeasures must be taken to provide a layer of security in communication. Intrusion Detection Systems are systems that seek to detect malicious entities that try to control and/or make a network of devices unavailable. Much of the current work in this area of research focuses on anomaly methods for binary detection, which simply detect whether a specific traffic is an attack or not, but is not able to identify the type of attack. Additionally, the training and test datasets commonly used in the research area are synthetically created, having a set of their own features. This difference in features between the datasets limits the evaluation of the studies carried out, since the quality of the features is known to be important for the performance of the model. Recently, a dataset was made available that has features based on the NetFlow v9 protocol, widely used by service providers. This dataset provides a set of standard features, that is, representing packet flows from a "real" Internet of Things network. Therefore, in this work, multiclass detection methods are proposed, using supervised learning algorithms, commonly used by state-of-the-art research in intrusion detection. The main contribution of this work is the use of a recent dataset, generated from a fusion of other datasets widely used in the area of intrusion detection research for Internet of Things environments. Through the experiments it was found that all approaches evaluated presented satisfactory detection rates, with low rates of false negatives and false positives.*

Resumo. *A Internet of Things é um paradigma que está em grande ascensão nos últimos anos. O grande numero de dispositivos e as suas limitações de recursos, desafiam pesquisadores e desenvolvedores na área de segurança da informação. Ataques a dispositivos conectados se tornaram comuns ultimamente. Desse modo, contramedidas devem ser tomadas para oferecer uma camada de segurança na comunicação. Os Intrusion Detection Systems são sistemas que buscam detectar entidades maliciosas que tentam controlar e/ou indisponibilizar uma rede de dispositivos. Muitos dos trabalhos atuais nessa área de pesquisa focam em métodos de anomalias para detecção binária, que simplesmente detecta se um tráfego específico é ataque ou não, mas não é capaz de identificar o tipo de ataque. Adicionalmente, os datasets de treinamento e teste*

comumente utilizados na área da pesquisa, são criados de forma sintética, tendo um conjunto de features próprios. Essa diferença de features entre os datasets limita a avaliação dos estudos realizados, pois conhecidamente a qualidade das features são importantes para o desempenho do modelo. Recentemente foi disponibilizado um dataset que possui features baseadas no protocolo NetFlow v9, amplamente utilizado por provedores de serviço. Esse dataset proporciona um conjunto de features padrão, ou seja, representando fluxos de pacotes de uma rede Internet of Things "real". Portanto, neste trabalho é proposto métodos de detecção multiclasse, utilizando algoritmos de aprendizado supervisionado, comumente utilizados por pesquisas do estado da arte em detecção de intrusão. A maior contribuição deste trabalho é a utilização de um dataset recente, gerado a partir de uma fusão de outros datasets bastante utilizados na área de pesquisa em detecção de intrusão para ambientes Internet of Things. Através dos experimentos verificou-se que todas as abordagens avaliadas apresentaram taxas de detecção satisfatórias, apresentando baixas taxas de falsos negativos e falsos positivos.

1. Introdução

A Internet das Coisas *(Internet of Things - IoT)* é um paradigma que está em grande ascensão nos últimos anos. O desenvolvimento e crescimento acelerado desses pequenos dispositivos conectados à Internet, fará com que essa grande rede seja de difícil gerenciamento e controle. Os dispositivos que compõem a IoT são provenientes de pequenos sensores de baixo poder computacional, onde trocam informações entre si, com funções limitadas.

Devido ao baixo poder computacional destes pequenos dispositivos, atrelados a grande quantidade dos mesmos e a necessidade de estabelecer uma comunicação eficiente, os dados originados são enviados para um centro computacional de maior capacidade de armazenamento e poder de processamento, a *Cloud Computing*.

No entanto, a *Cloud Computing* tem o problema de latência causado pela distância entre os dispositivos IoT e os *data centers* [Satyanarayanan 2015]. Com esse gargalo mapeado, a Cisco criou um novo paradigma de comunicação eficiente, a *Fog Computing*, onde leva um dispositivo com maior poder computacional, para a borda da rede [Bonomi et al. 2012]. Dessa forma, é possível processar e armazenar informações próximas aos nós finais, aliviando o tráfego enviado para a nuvem [Marin Tordera et al. 2017].

O ambiente da IoT não está livre de ameaças e vulnerabilidades de segurança. Os mecanismos de detecção de intrusão são pontos críticos de segurança, eles são capazes de mapear as tentativas de ataques sobre essas entidades. Para detectar ataques IoT na camada de transporte, existem diferentes tipos de sistemas de detecção de intrusão, incluindo aqueles baseados na detecção de assinaturas e baseados em detecção por anomalias.

No ambiente de pesquisa, a maioria das abordagens de detecção de intrusão enfoca em métodos de anomalia para detecção binária [Miranda et al. 2020], [Priyadarshini and Barik 2019], [Shafi et al. 2018]. Por outro lado, as poucas abordagens

de detecção multiclasse existentes que visam classificar o ataque em categorias, apresentam taxas de acurácia inferiores aos métodos binários.

Muitos esforços na área de pesquisa para IDSs têm como objetivo alavancar modelos de *Machine Learning* (ML) cada vez mais poderosos. Para isso, foram gerados *datasets* contendo tráfego de rede de forma sintética para serem utilizados nos experimentos. Com isso, a avaliação de modelos de ML geralmente não é confiável, pois diferentes estudos aplicados na área de NIDS utilizam diferentes *datasets* contendo registros de eventos de segurança variados, outro ponto crítico são as *features* utilizadas, muitas delas são distintas para cada *dataset*.

No entanto, até onde sabemos, não há trabalho no estado da arte conduzindo uma investigação completa da capacidade dos principais algoritmos de ML para realizar tarefas de identificação de intrusão multiclasse utilizando o novo *dataset* NF-UQ-NIDS-v2. Portanto, neste trabalho, investigamos a habilidade dos classificadores baseados na família de Árvores de Decisão para detecção e categorização de intrusão.

2. Trabalhos Correlatos

Neste capítulo são apresentadas soluções propostas por outros autores, de modo a construir uma visão do atual estado da arte em detecção e identificação de intrusão em ambientes de redes IoT, utilizando de técnicas baseadas em aprendizado de máquina. Os artigos escolhidos apresentam aspectos relacionados com a proposta que será defendida neste trabalho.

Os autores peng2018intrusion propuseram um IDS baseado em DT. Como primeiro processo do trabalho, foi proposto um algoritmo de pré-processamento para digitalizar as strings no conjunto dos dados (KDDCUP99) e para normalizar os dados, garantindo a qualidade dos dados com o objetivo de melhorar a eficiência da detecção. No segundo processo, foi utilizado o método de DT para o IDS, para posteriormente comparar com outros métodos, como *Naive Bayes* e *K-Nearest Neighbor* (KNN). . O KNN devido ao seu alto tempo de processamento foi descartado dos estudos. Por fim, foi discutida a taxa de detecção entre o *Naive Bayes* e a DT, onde a DT mostrou maiores taxas de acerto na detecção e classificação, por contrapartida o *Naive Bayes* mostrou menor tempo de processamento. No cenário do *Big Data* os autores afirmam o método de DT mais adequado para um IDS sobre *Big Data* em ambiente da Fog.

[Moualla et al. 2021] propuseram um novo IDS, utilizando a técnica de aprendizado de máquina supervisionado para detectar e classificar intrusões para redes IoT. Os autores dividem o aprendizado em diversas etapas. Ele começa com o método *Synthetic Minority Oversampling Technique (SMOTE)* para equilibrar as classes presentes no *dataset* UNSW-NB15, criando instâncias sintéticas e, em seguida seleciona as melhores *features* para cada classe do conjunto de dados, utilizando o algoritmo *ExtraTree* com critério de impureza por índice de Gini. Os resultados do sistema proposto foram promissores, onde os autores realizaram comparações com trabalhos correlatos, mostrando melhores taxas de precisão para a maioria das classes de ataque, menores taxas de falsos positivos e taxas de *recall* superiores.

Os autores [Atnafu and Acharya 2021] realizaram uma abordagem para escolher um modelo de ML que apresenta melhores resultados em termos de precisão para um

IDS. Para este estudo foram analisados três classificadores de ML. Máquina de Vetor de Suporte *(Support Vector Machine - SVM)*, classificador de *(Naive Bayes - NB)* e KNN. foram utilizadas duas variações na quantidade de *features* e tamanho do *dataset* de treinamento e teste. Foram testadas 41 e 27 *features* respectivamente. Para o *dataset* foram feitas divisões de 10% e 20% para teste. Os resultados apresentados mostraram que o KNN é o melhor classificador para o estudo, chegando a taxas de até 100% de precisão, utilizando 27 *features* sobre 10% do *dataset* de teste. No entanto, o KNN possui um alto custo computacional em tempo de predição. Além disso, nesta pesquisa não foi realizada a identificação do tipo do ataque, apenas uma abordagem de detecção binária.

[Sarhan et al. 2020] propuseram e avaliaram um novo *dataset* para trabalhos e pesquisas, com a justificativa que os diferentes trabalhos do estado da arte em IDS para IoT não são avaliados de forma confiável. Atualmente, a variação das informações representadas em cada *dataset* tem causado limitações no campo de pesquisa. Os autores alegam que os estudos atuais não têm a capacidade de avaliar a generalização de desempenho de um modelo de ML em vários *datasets* NIDS, utilizando um conjunto de *features* comuns. Para preencher essa lacuna, os autores escolheram quatro *datasets* recentes, UNSW-NB15, BoT-IoT, ToN-IoT e CSE-CIC-IDS2018 usando seus arquivos de captura de pacotes disponíveis publicamente, através destes *datasets*, os autores realizaram uma conversão para formato baseado em *features* NetFlow. O modelo de classificação utilizado foi algoritmo *ExtraTree*, com especificamente 50 árvores pré-definidas com abordagens binaria e multiclasse. Não foi especificado o porque da escolha do algoritmo e nenhuma variação na quantidade de árvores no experimento. Os resultados foram satisfatórios para um primeiro trabalho, porém deixando um campo de pesquisa em aberto para novas aplicações de modelos.

Os autores [Sarhan et al. 2021] estenderam suas pesquisas para atingirem melhores resultados com o novo *dataset* baseado nas *features* do NetFlow. Foram avaliados e comparados duas variantes do conjunto de *features* com base no protocolo NetFlow. Uma versão com 12 *features* e outra com 43 *features*. Como base para comparar o desempenho de um classificador sob os *datasets*, foi utilizado o algoritmo *ExtraTree*. Enquanto o *dataset* com menor quantidade de *features* não conseguiu corresponder a um bom desempenho de classificação dos conjuntos de *features* proprietários, o *dataset* com maior número de *features*, 43, versão 2 (v2) do NetFlow, surpreendentemente atinge um desempenho de classificação consistente, em comparação com os *datasets* nativos e o conjunto NetFlow contendo 12 *features* versão 1 (v1).

Ao longo desta seção foram descritos os trabalhos correlatos. São diferentes propostas que utilizaram métodos de aprendizado de máquina para buscar o estado da arte em detecção de intrusão em redes IoT. Dentre os trabalhos, foram utilizadas diferentes *datasets*, evidenciando uma lacuna entre os trabalhos que visam o estado da arte. As bases de dados disponíveis e comumente utilizadas, possuem um conjunto de *features* heterogêneos, dificultando a avaliação de modelos de aprendizado de máquina para detecção de intrusão.

3. Abordagem proposta

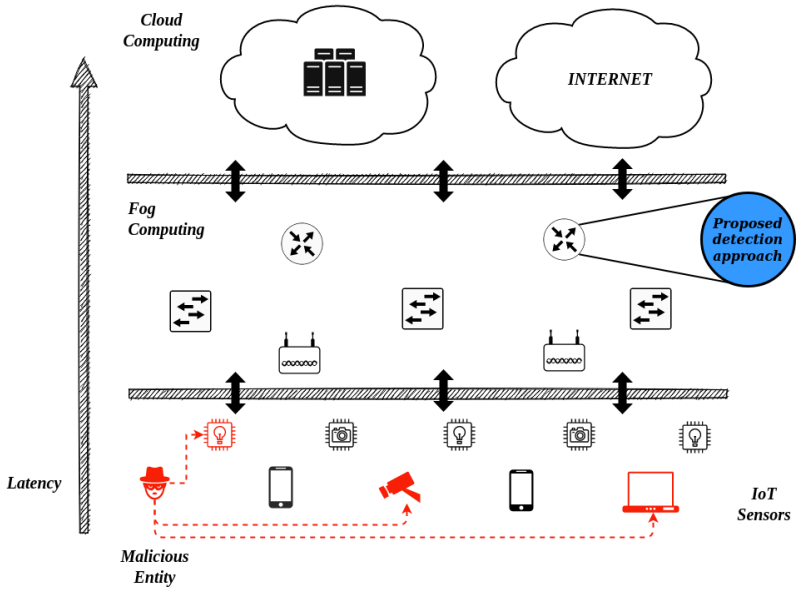
Considerando os conceitos, os trabalhos correlatos e a discussão apresentada, este trabalho propõe investigar a capacidade dos modelos de ML, variante da família de árvores,

para realizar a detecção e classificação de intrusões, simulando um *ISP*, contendo tráfego de rede real de dispositivos *IoT* em ambientes baseados no conceito da *fog computing*.

3.1. Contextualização da proposta

O trabalho considera o contexto de um sistema *IoT*, onde utiliza-se dispositivos *fog* para prover os benefícios de pré-processamento e armazenamento para os usuários. Neste contexto, usuários maliciosos podem tentar comprometer esse sistema para roubar dados, indisponibilizar dispositivos e/ou sistemas. A Figura 1 mostra a arquitetura desse sistema, onde a abordagem proposta é posicionada no ambiente *IoT*. Este trabalho considera que o dispositivo possuí características dos atuais protocolos utilizados nas redes atuais, como o TCP/IP, para serem capturados e analisados, através do protocolo *NetFlow v9*.

Figure 1. Arquitetura IoT.

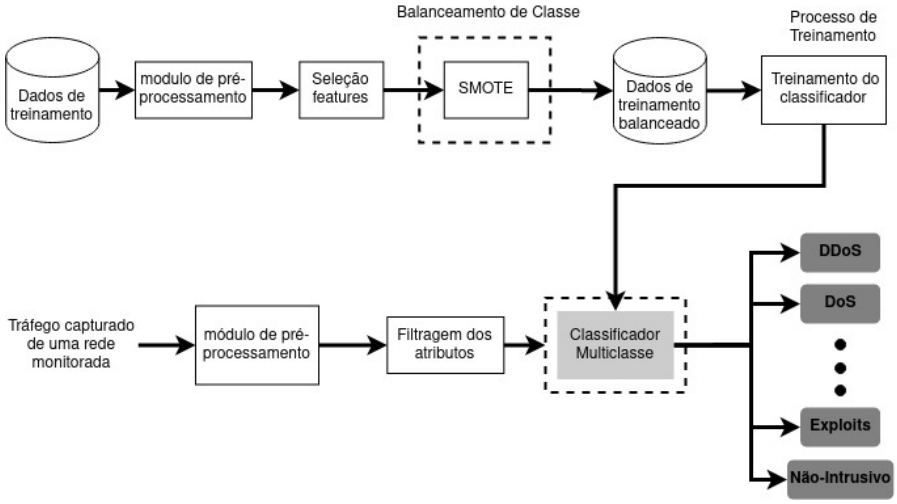


A seguir, são apresentados os principais aspectos da abordagem proposta.

3.2. Descrição da Abordagem Proposta

Nesta seção são fornecidos maiores detalhes sobre a abordagem proposta neste trabalho. A Figura 2 ilustra os processos envolvidos nas etapas de treinamento do modelo de classificação e de predição.

Figure 2. Modelo de Classificação Proposto.



Inicialmente o conjunto de dados de treinamento passa por um módulo de pré-processamento, onde procuramos retirar fluxos inválidos, ou seja, que apresentaram alguma falha, falta de informação em sua captura.

Logo após, opcionalmente, podemos utilizamos um seletor de *features* para selecionar as *features* com maior relevância para predizer nossa variável alvo.

O processo de SMOTE, realiza o balanceamento de classes, tem a função de criar novos dados para as classes minoritárias, ou seja, aquelas com um número menor de registros em comparação com outras classes presentes no *dataset*. Por fim, os dados são submetidos para o processo de treinamento do algoritmo de classificação. Consideramos para compor este classificador alguns métodos da família de árvores de decisão (DT, ou RF, ou ET), os quais serão avaliados através de experimentos.

Na linha inferior, temos modelado o processo em tempo real de execução, onde os fluxos exportados pelo *NetFlow* entram em um módulo de pré-processamento, idêntico ao explanado na fase de treinamento, em seguida, realizamos a filtragem das *features* de forma idêntica ao selecionados. Por fim, os dados são classificados através do modelo de classificação gerado durante o treinamento.

4. Experimentos

Nesta seção são descritos alguns aspectos da metodologia utilizada para avaliar a abordagem proposta. Primeiramente são apresentadas as características do *dataset* e a sua separação em conjunto de treino e teste. Em seguida, são apresentadas as métricas de classificação pertinentes para a avaliação do modelo. Por fim, são descritos os experimentos realizados.

4.1. Dataset

Neste trabalho foi utilizada a base de dados NF-UQ-NIDS-v2 para realizar os experimentos. A base que foi concebida por pesquisadores da *UQ*. O projeto tem como objetivo proporcionar uma melhor avaliação dos estudos sobre *NIDS*, utilizando diferentes *datasets*, contendo características de ataques distintas em um único *dataset*.

O *dataset* contém as seguintes classes de ataques:

- **DDoS**: O atacante captura diversas máquinas vulneráveis ao longo da internet (*bots*), onde realiza um ataque orquestrado sobre a vítima que recebe um despejo de tráfego/amplificação, deixando o servidor/aplicação indisponível.
- ***Reconnaissance***: O atacante coleta informações sobre a vítima para planejar seu ataque.
- ***Injection***: O atacante explora vulnerabilidades em bancos de dados através da injeção de parâmetros, ganhando acesso ao banco de dados.
- **DoS**: O atacante procura explorar protocolos que consumam excessivamente os recursos do servidor da vítima.
- ***Brute Force***: Ataque exaustivo que utiliza dicionários contendo diversas combinações de senhas comumentes utilizadas.
- ***Password***: O atacante utiliza uma senha de usuário comumente utilizada para diversas contas de usuário, evitando o bloqueio de contas.
- **XSS**: Ataque de injeção de código malicioso em aplicações do tipo Web.
- ***Infiltration***: A partir de um arquivo malicioso executado pela vítima, o atacante realiza uma varredura na rede, buscando outras vulnerabilidades.
- ***Exploits***: Injeção de códigos maliciosos, utilizados após os ataques de *Reconnaissance* e *Scanning*, para ganhar privilégios no sistema.
- ***Theft***: O atacante visa obter dados confidenciais, como roubo de dados e keylogging.
- ***Scanning***: O atacante procura por vulnerabilidades na rede, geralmente com o uso de ferramentas.
- ***Fuzzers***: O atacante pode utilizar essa ferramenta para testar um conjunto de parâmetros em aplicações, buscando estouros de *buffer*, *strings* invalidas e etc.
- ***Backdoor***: Um tipo de *Malware* que nega os procedimentos normais de autenticação para acessar um sistema.
- ***Bot***: O atacante utiliza *malwares* do tipo cavalo de troia para tentar controlar uma ou mais máquinas da vítima.
- ***Generic***: Ataque que serve como uma etapa dentro de um processo de infecção por *Ransomware*, através do protocolo *Remote Desktop Protocol* (RDP).
- ***Analysis***: O atacante realiza uma análise do *host*/rede da vítima. Analisando portas e serviços rodando no nó vítima.
- ***ShellCode***: O atacante insere trechos de código arbitrário, explorando uma vulnerabilidade no software.
- **MITM**: O atacante intercepta a comunicação entre dois *hosts*, forjando o endereço IP de uma das partes.
- ***Worms***: Um programa independente, da família *Malware*, que se espalha pela rede de forma rápida, com o objetivo de roubar dados.
- ***Ransomware***: Um vírus que codifica o sistema de arquivos da vítima, deixando-o inoperante.

4.2. Pré-processamento do dataset

O *dataset* NF-UQ-NIDS-v2 possuí um único arquivo do tipo csv, contendo aproximadamente 12,8GB de dados. Devido ao grande tamanho do arquivo e a quantidade de registros, houve a necessidade de utilizar um subconjunto de 10% do *dataset* para realizar

os experimentos. A plataforma utilizada nos experimentos possuí limitações quanto ao carregamento de grandes arquivos. A extração do subconjunto foi realizada de forma aleatória, porém mantendo a proporção na distribuição de registros por classes de ataque.

O *dataset* reduzido e processado, com apenas 10% dos dados, é então utilizado nos experimentos. Para cada experimento optou-se por utilizar o método de amostragem *Hold-Out 70-30*, onde o *dataset* é dividido de forma aleatória e proporcional em 70% para treino e 30% para teste dos modelos. Essa divisão é realizada para tornar possível realizar a avaliação do modelo com dados diferentes dos utilizados no treinamento.

4.3. Métricas de Avaliação

Com base nas categorizações apresentadas acima, são calculadas diversas métricas de desempenho de classificação. As principais utilizadas neste trabalho são apresentadas a seguir [Liu and 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 1, apresentada a seguir:

$$ACC = \frac{VP + VN}{VP + VN + FP + FN} \tag{1}$$

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 2.

$$PRE = \frac{VP}{VP + FP} \tag{2}$$

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 3 demonstra como é calculada a sensibilidade de um modelo.

$$Recall = \frac{VP}{VP + FN} \tag{3}$$

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 4.

$$F1 - Score = 2 * \frac{Precisão * Recall}{Precisão + Recall} \tag{4}$$

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

$$Acurácia\ Balanceada\ = \frac{\sum_{i=1}^{n} R_i}{n} \tag{5}$$

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 treinar o modelo com o conjunto de treinamento.
2. **Tempo de teste:** corresponde ao tempo, em segundos, que o algoritmo precisou para realizar a predição de todas as instâncias do conjunto de teste.

4.4. Descrição dos experimentos realizados

Para fins de avaliação da abordagem proposta, foram realizados diversos experimentos considerando diferentes configurações de seletores de *features* e classificadores. Os métodos de seleções de *features* consideradas foram o Ganho de Informação, Seleção Sequencial de *Features* e Eliminação Recursiva de *Features*. Além disso, foram avaliados os classificadores *Decision Tree*, *Extra Tree* e *Random Forest*.

O primeiro classificador utilizado foi o método *Decision Tree*, ele é um dos algoritmos mais comumente usados em projetos de mineração de dados, na etapa de classificação [Rokach 2016]. A DT utiliza formato em árvore, contendo nodos e arestas para tomar decisões com base nos recursos das instâncias. Os nodos correspondem as *features*, as arestas representam um valor ou uma faixa de valores de uma determinada *feature*. Os nodos folha representam a classificação. A DT possui diversos parâmetros que precisam ser definidos. O critério de Gini foi escolhido como função para medir a qualidade de uma divisão. Os critérios suportados são Gini para a impureza Gini e entropia para o ganho de informação. Foi escolhido o critério de Gini por ser o critério padrão da biblioteca. O parâmetro max_depth que indica a profundidade máxima permitida da árvore, foi definido como $None$, neste caso os nós são expandidos até que todas as folhas sejam puras ou até que todas as folhas contenham menos de $min\_samples\_split$ amostras. O $min\_samples\_split$ foi definido como 2, desse modo, são necessárias pelo menos duas amostras para dividir um nó interno. Além disso, o número mínimo de amostras necessárias para formar um nó folha ($min\_samples\_leaf$) foi definido com 1, ou seja, um ponto de divisão em qualquer profundidade só será considerado se deixar pelo menos uma amostra em cada um dos ramos esquerdo e direito. Esses três parâmetros também foram definidos de acordo com o padrão da biblioteca e buscam formar uma árvore completa sem restrições de tamanho.

Também foi utilizado nos experimento o classificador *Random Forest*, ele consiste em um método *Ensemble* que agrega os resultados de várias *Decision Tree* decorrelacionadas acumuladas dentro de uma "floresta" para produzir os resultados da classificação. O principal parâmetro deste método é o $n\_estimators$, ele indica o número de árvores na floresta. Neste trabalho o $n\_estimators$ foi definido como 100, sendo o valor padrão da biblioteca. O parâmetro $max\_features$ indica o número de atributos candidatos selecionados aleatoriamente em cada nó e foi definido como $sqrt(N)$, sendo N o numero de *features* do *dataset*. Além disso, por ser um método composto por diversas *Decision Tree*, neste trabalho optou-se por utilizar, para as 100 árvores da *Random Forest* , as mesmas configurações de parâmetros utilizadas no primeiro classificador *Decision Tree* único.

Por fim, o outro classificador utilizado foi a *Extra Tree*, ele também é um método *Ensemble* composto por diversas *Decision Trees* acumuladas dentro de uma "floresta".

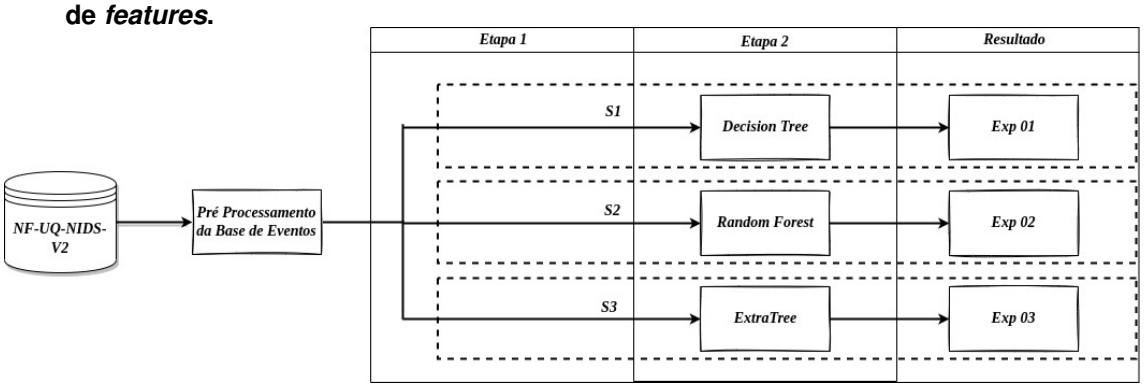
para produzir os resultados da classificação. Ela é bastante semelhante em operação a *Random Forest* e varia principalmente na forma de construir as árvores dentro da floresta. Assim como na *Random Forest*, um subconjunto aleatório de $max\_features$ *features* candidatas é usadas, no entanto, neste caso, em vez de procurar os pontos de corte mais discriminativos, os pontos de corte são sorteados aleatoriamente para cada *feature* candidata e o melhor desses pontos de corte gerados é escolhido aleatoriamente como o regra de divisão. Para a *Extra Tree* os parâmetros $n\_estimators$ e $max\_features$ também foram definidos como 100 e $sqrt(N)$, respectivamente. Do mesmo modo, utilizou-se, para as 100 árvores da *Extra Tree*, as mesmas configurações de parâmetros utilizadas no primeiro classificador *Decision Tree* único.

A seguir são apresentados os experimentos realizados considerando as diferentes combinações de seletores de *features* e classificadores.

4.4.1. Experimentos com todo o conjunto de *features*

Os primeiros experimentos foram realizados com *dataset* completo, contento todo o conjunto de *features*. O objetivo foi ter resultados de controle para comparação com os resultados obtidos utilizando métodos de seleção de *features*. A Figura 3 apresenta uma ilustração dos primeiros experimentos realizados.

Figure 3. Experimentos realizados com o *dataset* NF-UQ-NIDS-V2 sem Seleção de *features*.



4.4.2. Experimentos com o conjunto de *features* selecionadas pelo método de Ganho de Informação

Foram realizados três experimentos com *dataset* filtrado através do método de seleção de *features* por Ganho de Informação. Ele avalia o grau de associação da *feature* com a classe para encontrar os valores com o maior grau de utilização e importância através do calculo da redução da entropia de cada *features*. 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 processo de detecção.

O ganho de informação de uma *feature* é determinado pela redução da sua entropia [Karegowda et al. 2010]. A Equação 6 calcula o ganho de informação de uma *feature* A

em relação a classe C.

$$Ganho(C, A) = Entropia(C) - Entropia(C, A) \tag{6}$$

Onde $Entropia(C)$ é a entropia da classe C, e $Entropia(C, A)$ é entropia da classe relativa a *feature* A. Calculada pela Equação 7. Onde $Entropia(C|A = A_j)$ é a entropia relativa ao subconjunto de instâncias que tem um valor A_j para a *feature* 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.

$$Entropia(C, A) = -\sum_{j=1}^{m} p(A, C) \log_2(p(A, C)) \tag{7}$$

Temos que o valor de ganho de informação consiste na variação da impureza, ou seja, *features* com menor entropia possuem maior ganho de informação. Esta medida tem um *bias* natural pois favorece *features* que possuem muitos valores.

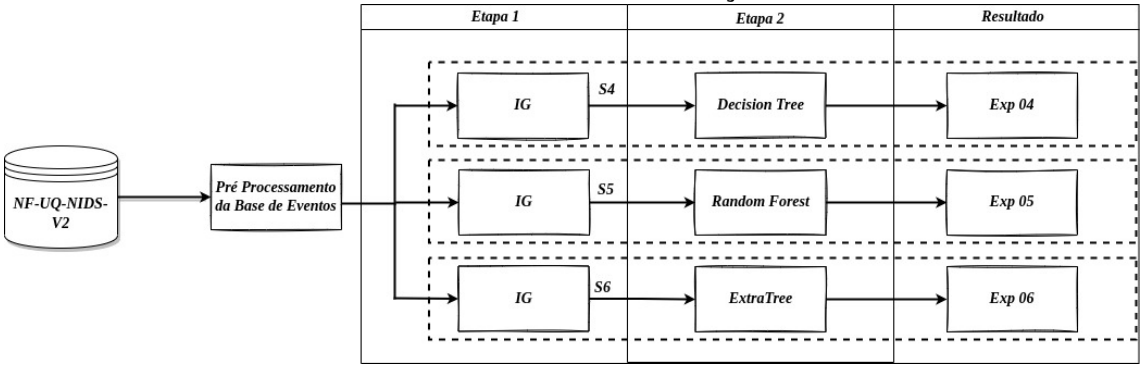
A medida de razão de ganho de informação tenta corrigir o *bias* do Ganho de Informação [Quinlan 2014].

$$Razão\ de\ Ganho(C, A) = \frac{Ganho(C, A)}{Entropia(A)} = \frac{Entropia(C) - Entropia(C, A)}{Entropia(A)} \tag{8}$$

Após o cálculo do mérito de cada *feature* pela Equação 8, é gerado um ranking e seleciona-se as N melhores *features* desse ranking de acordo com algum critério.

A Figura 4 apresenta uma ilustração dos experimentos realizados. O *dataset* NF-UQ-NIDS-V2 também passou por um processo de pré-processamento e posteriormente foi utilizado nos experimentos Exp04, Exp05 e Exp06, para avaliar a seleção de *features* e os classificadores.

Figure 4. Experimentos realizados com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método de Ganho de Informação.



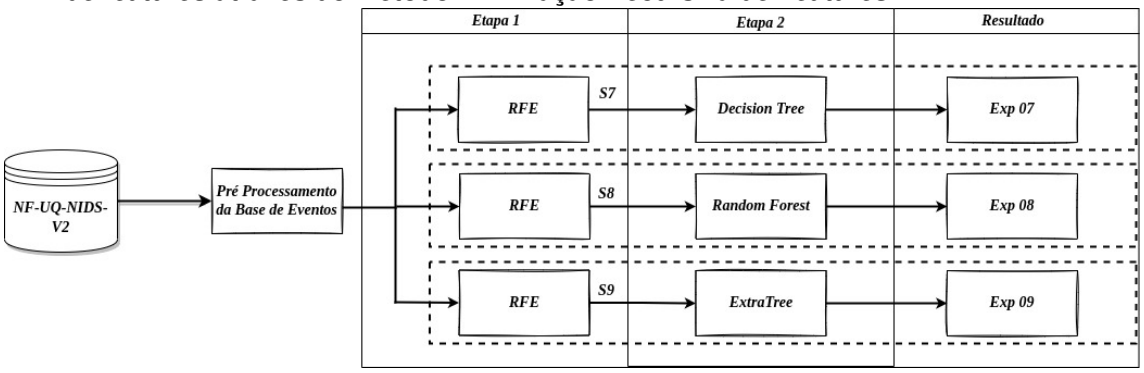
4.4.3. Experimentos com o conjunto de *features* selecionadas pelo método de Eliminação Recursiva de *Features*

A Figura 5 ilustra os três experimentos realizados com o *dataset* filtrado através do método de Eliminação Recursiva de *Features*.

O método de Eliminação Recursiva de *Features (Recursive Feature Elimanation - RFE)* da família *Wrapper*, é um algoritmo de seleção de *features*. Sua composição têm duas camadas, onde o núcleo possuí um primeiro método de seleção, por exemplo uma DT, logo esse método é encapsulado pelo RFE. A RFE procura por um subconjunto de *features*, começando com todo o conjunto de *features* de treinamento e removendo de forma recursiva até que o número desejado de *features* seja alcançado. Uma facilidade que a RFE proporciona é a passagem de hyperparâmetros, como o algoritmo que será encapsulado e o número de *features* desejado. O algoritmo de núcleo deve fornecer uma maneira de pontuar e calcular as *features* mais importantes. O núcleo não precisa ser o algoritmo que se ajusta as *features* selecionadas, ou seja, diferentes algorítimos podem ser utilizados.

O *dataset* NF-UQ-NIDS-V2 também passou por um processo de pré-processamento e posteriormente foi utilizado nos experimentos Exp07, Exp08 e Exp09, para avaliar a seleção de *features* e os classificadores, conforme pode ser observado na Figura 5.

Figure 5. Experimentos realizados com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método Eliminação Recursiva de *Features*.



4.4.4. Experimentos com o conjunto de *features* selecionadas pelo método de Seleção Sequencial de *Features*

Os últimos três experimentos realizados consideraram o método de Seleção Sequencial de *Features*, conforme apresentado na Figura 6.

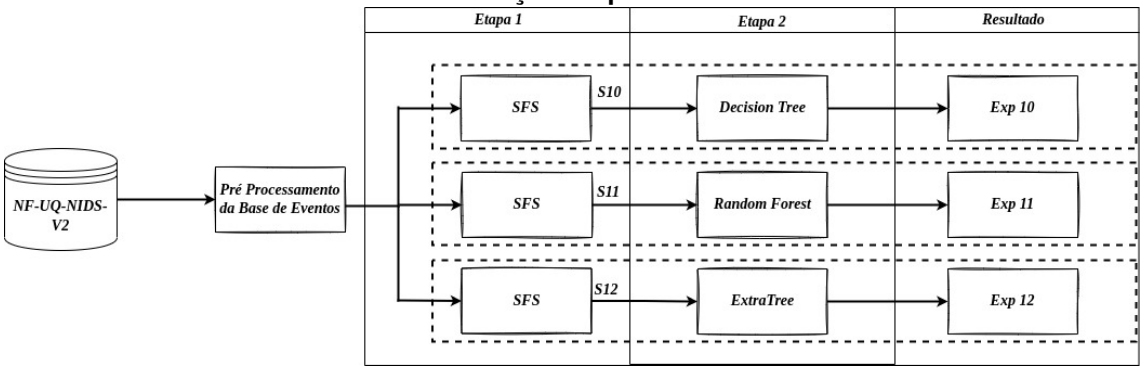
O método Seletor Sequencial de *Features (Sequential Feature Selector - SFS)* adiciona (seleção para frente) ou remove (seleção para trás) *features* para formar um subconjunto de *features* de maneira gananciosa com a utilização de um classificador. Em cada estágio, este estimador escolhe a melhor *feature* para adicionar ou remover com base na pontuação de validação cruzada. Ele consiste em uma abordagem *Wrapper* e trabalha com subconjunto de *features* gerados a cada iteração do algoritmo. O classificador é exe-

cutado com essas *features* no conjunto de treinamento e tem como resultado uma precisão do classificador. Ela é utilizada para avaliar a qualidade do subconjunto de *features*. O processo é repetido para cada subconjunto de *features* até que o critério de parada seja satisfeito.

A estratégia de busca *forward selection* se refere a uma busca que tem o conjunto inicial vazio. A exploração é então iniciada a partir do estado pai onde são criadas variações do estado atual e gerando estados filhos, adicionando *features* a este subconjunto que antes eram ausentes no estado pai. 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 *features* nos subconjuntos se torna mais rápido a comparação e classificação.

O *dataset* NF-UQ-NIDS-V2 também passou por um processo de pré-processamento e posteriormente foi utilizado nos experimentos Exp10, Exp11 e Exp12, para avaliar a seleção de *features* e os classificadores.

Figure 6. Experimentos realizados com o *dataset* NF-UQ-NIDS-V2 com seleção de *features* através do método Seleção Sequencial de *Features*.



4.4.5. Materiais utilizados

Neste trabalho foi utilizado a plataforma Microsoft Azure para realização dos experimentos. É uma plataforma em nuvem que disponibiliza um ambiente de execução em Python. A configuração da máquina utilizada contava com processador Intel® Xeon® Platinum 8370C de terceira geração (Ice Lake), 28 GB de memória de 56 GB de armazenamento. Para a criação e execução da abordagem proposta, foram baseadas em bibliotecas Python. A biblioteca *Pandas* foi utilizada no carregamento do *dataset* NF-UQ-NIDS-v2, armazenando em uma variável para manipulação. A biblioteca *Numpy* foi utilizada para manipulação de *arrays* e matrizes, onde serviu para agrupar os ataques em suas respectivas classes. Também foi utilizada a biblioteca *Scikit-Learn* para instanciar e manipular os classificadores baseados em árvores de decisão, onde também possibilitou o uso de métodos de avaliação de *features* como o Ganho de Informação.

5. Resultados

Em relação aos tempos, a DT mostrou-se a abordagem menos custosa em todos seus experimentos. A DT se mostrou quase 10x mais rápida entre os outros classificadores. Sua estrutura de dados e operação justificam seus baixos tempos de treino e teste.

As abordagens com RF se mostraram as mais custosas em relação ao tempo de treino, em todos os experimentos precisou de mais de 3.000 segundos para completar a etapa. No processo de detecção de intrusão as atualizações de modelos possuem grande relevância. A dinamicidade e elasticidade presente em redes IoT contempla a inserção e remoção de diversos nós ou dispositivos. Essas mudanças podem alterar o comportamento da rede constantemente e tornar necessário atualizar os modelos de detecção com maior frequência. Portanto, quanto menor o tempo necessário para realizar o treinamento do modelo, menor será o tempo esperando um novo modelo ou utilizando um modelo desatualizado. Portanto, a DT se mostra a mais adequada nesse quesito.

O tempo de teste dos experimentos com RF, no geral foram maior que as DT, porém melhor do que os experimentos com a ET, que possui um tempo mais elevado. O tempo de teste indica o tempo necessário para a análise e classificação do tráfego, portanto é um atributo muito importante.

De modo geral, destaca-se que as abordagens com seleção de *features* foram capazes de alcançar reduções nos tempos de treino e teste em relação aos experimentos sem seleção de *features*. Desse modo, é possível concluir que a quantidade de *features* pode influenciar diretamente no tempo necessário para treinar e testar os classificadores baseados em árvores de decisão.

Para discutir a respeito da melhor abordagem, deve-se considerar na escolha o cenário e aplicação onde ela será implantada. Caso haja necessidade de resposta imediata de um incidente, a DT tende a ser a melhor abordagem, principalmente a configuração avaliada no experimento Exp04, que utilizou Ganho de Informação para redução de *features*, ela apresentou a melhor acurácia balanceada dentre as DTs e os menores tempos de treino e de predição.

Consideramos no trabalho três métricas de desempenho de classificação: Acurácia, Acurácia Balanceada e Precisão. A Acurácia e a Precisão neste caso podem nos trazer uma falsa percepção da realidade pois podem ser influenciadas por classes com uma grande quantidade de dados. Nesse caso, tanto a classe Benigna e DDoS possuem uma enorme quantidade de dados e obtiveram uma alta taxa de detecção, influenciando positivamente na Acurácia e Precisão que então ficaram todas acima de 98%. No entanto algumas classes tiveram taxas de detecção menores e por possuírem poucos dados não influenciaram essas métricas gerais. A Acurácia Balanceada corrige este problema, pois faz a média das taxas de detecção de todas as classes.

Portanto, vamos considerar na análise principalmente a acurácia balanceada.

Apesar de ser mais leve, a DT apresentou taxas de Acurácia Balanceada inferiores as abordagens Ensemble. Além disso, as Árvores de Decisão estão sujeitas a problemas de ajuste excessivo *overfitting* por problemas de ruído durante o treinamento. A RF e a ET podem ser consideradas como alternativas robustas a ruídos de classificação e treinamento, os quais são comuns para aplicações de detecção de intrusão, além de

apresentarem maior robustez a *overfitting*.

6. Discussões

Nesta seção, são apresentados e discutidos os resultados obtidos pelas diferentes configurações executadas em diversos experimentos. Para isso foi construída uma tabela comparativa entre as métricas gerais mais relevantes a serem discutidas e foi criado um gráfico sobre as taxas de Acurácia Balanceada obtidas pelas abordagens nos experimentos. Por fim, são discutidos outros aspectos que valem ser pontuados.

A Tabela 1 apresenta os resultados obtidos pelas diferentes configurações da abordagem nos 12 experimentos em relação a Acurácia, Acurácia Balanceada, Precisão, tempo de treino e tempo de teste. Os valores de tempo de treino nesta tabela foram arredondados para uma melhor visualização.

Table 1. Tabela comparativa entre os experimentos realizados

Experimentos	Treino (seg)	Teste (seg)	Acurácia	Acurácia B.	Precisão
Exp01 (DT)	422	0,65	99,04%	80,72%	99,18%
Exp02 (RF)	3.954	62,32	99,14%	84,91%	99,32%
Exp03 (ET)	2.205	100,58	99,12%	85,09%	99,26%
Exp04 (IG+DT)	240	0,44	98,90%	83,70%	99,20%
Exp05 (IG+RF)	3.796	73,64	98,95%	85,35%	99,28%
Exp06 (IG+ET)	1.754	116,83	98,86%	85,44%	99,23%
Exp07 (RFE+DT)	389	0,45	99,04%	81,18%	99,17%
Exp08 (RFE+RF)	3.529	62,42	99,12%	84,55%	99,31%
Exp09 (RFE+ET)	1.820	90,12	99,08%	83,49%	99,25%
Exp10 (SFS+DT)	291	0,58	99,02%	81,36%	99,16%
Exp11 (SFS+RF)	3.146	63,15	99,17%	85,54%	99,32%
Exp12 (SFS+ET)	2.030	86,50	99,15%	83,49%	99,27%

Em todos os experimentos a Acurácia e a Precisão apresentaram altas taxas, próximas ou acima de 99%. Esses valores foram alcançados principalmente porque estas métricas são fortemente influenciadas pelo resultados das classes que possuem muitas instâncias e pouco influenciadas pelas classes que possuem poucas instâncias. O *dataset* NF-UQ-NIDS-v2 possui muitas instâncias de trafego benigno e de ataques de negação de serviço, estas classes tiveram elevadas taxas de detecção, desse modo, a acurácia e precisão foram extremamente altas, mesmo tendo classes que o modelo não conseguiu ter bom desempenho.

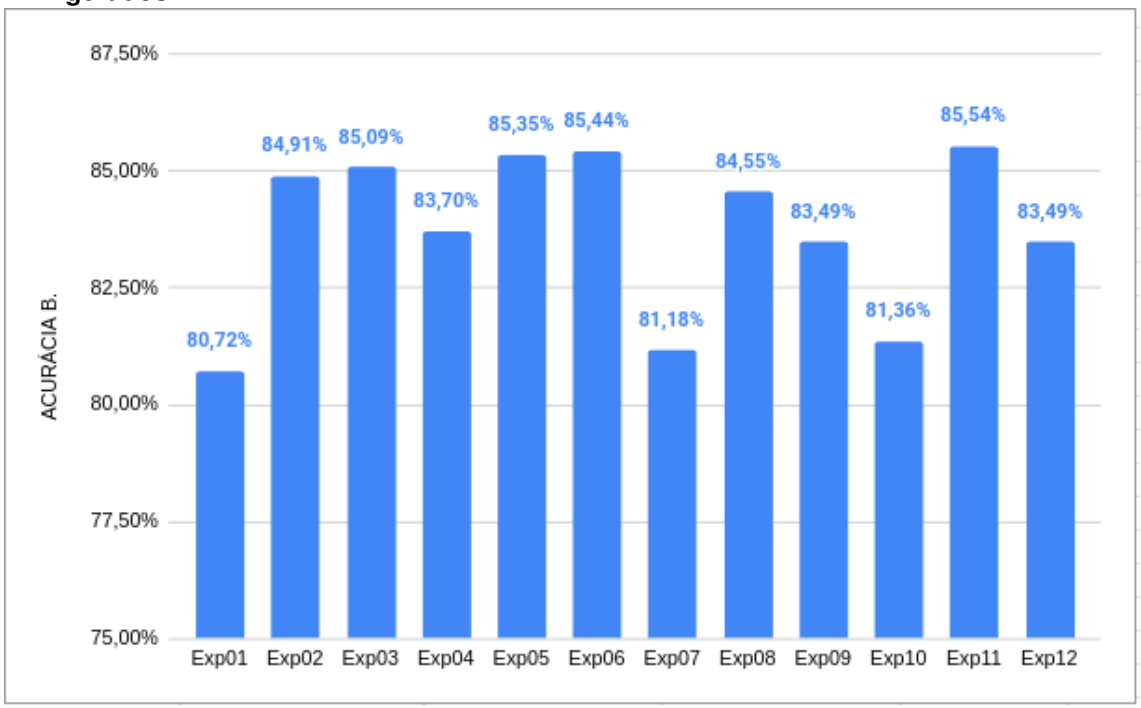
Já a métrica de Acurácia Balanceada contorna o problema anterior, pois realiza a média das taxas de detecção das classes. Os experimentos com DT apresentaram Acurácia Balanceada inferior aos demais métodos em todos os cenários, com ou sem seleção de *features*. Os experimentos utilizando DT (Exp01, Exp04, Exp07 e Exp10) não apresentaram boas taxas de detecção em comparação com os demais experimentos.

Conforme pode ser observado no gráfico apresentado na Figura 7, a DT apresentou Acurácia Balanceada de 80,72% (Exp01), 83,71% (Exp04), 81,18% (Exp07) e 81,36% (Exp10). Portanto, o melhor desempenho da DT foi no Exp04, onde foi utilizado o Ganho de Informação para selecionar as *features*.

Os experimentos utilizando a RF como classificador apresentaram taxas de Acurácia Balanceada muito próximas em todos as abordagens, 83,91% (Exp02), 85,36% (Exp05), 84,55% (Exp08) e 85,54% (Exp11), conforme pode ser visualizado na Figura 7. A maior taxa de Acurácia Balanceada foi alcançada no Exp11, onde o *dataset* foi reduzido pela Seleção Sequential de *Features*.

O melhor resultado dentre todas as abordagens avaliadas sem seleção de *features* foi a ET, ela obteve 85,10% de Acurácia Balanceada, No experimento com seleção de *features* por Ganho de Informação (Exp06) essa métrica aumentou para 85,44%. No entanto, o desempenho da ET foi menor nos experimentos Exp09 e Exp12, onde *dataset* foi reduzido através da Seleção Sequencial de *Features* e Eliminação Recursiva de *Features*, respectivamente.

Figure 7. Gráfico comparativo da Acurácia Balanceada dos diferentes modelos gerados.



Com relação aos tempos, a DT mostrou-se a abordagem menos custosa. Sua estrutura e operação justificam seus baixos tempos de treino e teste. A iteração do algoritmo é mínima comparado com as demais abordagens. Realizando uma média do tempo de treino, considerando todos os experimentos entre a DT e RF, a DT é quase dez vezes mais rápida. Em questão de atualizações de modelos, conforme o tráfego de entrada é registrado e processado, a DT apresenta melhor desempenho. A mesma interpretação pode ser considerada para o tempo de teste que é muito inferior em relação as demais abordagens, logo a DT, caso tenha boas taxas de detecção, seria a mais recomendada para detectar ataques em ambientes com restrições de recursos e/ou com a necessidade de resposta à incidentes de forma imediata.

As abordagens com RF se mostraram as mais custosas em relação ao tempo de treino, em todos os experimentos precisou de mais de 3.000 segundos para completar a etapa. No processo de detecção de intrusão as atualizações de modelos possuem grande

relevância. A dinamicidade e elasticidade presente em redes IoT contempla a inserção e remoção de diversos nós ou dispositivos. Essas mudanças podem alterar o comportamento da rede constantemente e tornar necessário atualizar os modelos de detecção com maior frequência. Portanto, quanto menor o tempo necessário para realizar o treinamento do modelo, menor será o tempo esperando um novo modelo ou utilizando um modelo desatualizado. No experimento Exp11, RF com SFS como abordagem de seleção de *Features*, o tempo de treino foi de 3.146 segundos, apresentando uma boa redução em relação ao tempo de treino com a base original Exp02 que levou 3.954 segundos.

O tempo de teste dos experimentos com RF, no geral foram maior que as DT, porém melhor do que os experimentos com a ET, que possui um tempo mais elevado. O tempo de teste indica o tempo necessário para a análise e classificação do tráfego, portanto é um atributo muito importante.

A ET foi menos custosa que a RF em tempo de treino, para alguns experimentos o tempo da ET foi a metade da RF. Nos experimentos utilizando a seleção de *features* por Ganho de Informação, a RF precisou de 3795 segundos para ser treinada no experimento Exp05 e a ET precisou de 1754 segundos no experimento Exp06.

De modo geral, destaca-se que as abordagens com seleção de *features* foram capazes de melhorar a Acurácia Balanceada para todos os classificadores em relação aos experimentos utilizando o *dataset* original sem utilização da técnica de seleção de *features*. Além disso, a DT e a ET tiveram reduções em seus tempos de treino e teste. A RF alcançou a redução apenas no tempo de treino. Desse modo, é possível concluir que a quantidade de *features* pode influenciar diretamente no tempo necessário para treinar e testar os classificadores baseados em árvores de decisão.

Para discutir a respeito da melhor abordagem, deve-se considerar na escolha o cenário e aplicação onde ela será implantada. Caso haja necessidade de resposta imediata de um incidente, a DT tende a ser a melhor abordagem, principalmente a configuração avaliada no experimento Exp04, que utilizou Ganho de Informação para redução de *features*, ela apresentou a melhor acurácia balanceada dentre as DTs e os menores tempos de treino e de predição. No entanto, a DT apresentou taxas de Acurácia Balanceada inferiores as abordagens *Ensemble*. Além disso, as Árvores de Decisão estão sujeitas a problemas de ajuste excessivo *(overfitting)* por problemas de ruído durante o treinamento [T.K. et al. 2021]. A RF e a ET podem ser consideradas como alternativas robustas a ruídos de classificação e treinamento, os quais são comuns para aplicações de detecção de intrusão, além de apresentarem maior robustez a *overfitting*.

Em um cenário onde há necessidade de maior confiabilidade na classificação, a RF do experimento Exp11, utilizando a Seleção Sequencial de *Features*, classificou melhor por ter apresentado uma boa taxa de Acurácia Balanceada, com um tempo de teste relativamente baixo. Como a abordagem está inserida na fog, o treinamento que apresentou um alto tempo poderia ser designado para a nuvem, onde tem o recurso de processamento maior.

De modo geral, as abordagens avaliadas neste trabalho apresentaram excelentes resultados quanto a falsos positivos, as taxas de *recall* para a classe benigna ficaram acima de 99%, indicando que de todos os eventos benignos existentes no *dataset*, mais de 99% foram identificados como benignos, ou seja a taxa de falsos positivos foi baixa.

Do mesmo modo, as abordagens avaliadas neste trabalho apresentaram excelentes resultados quanto a falsos negativos, ou seja, todos os experimentos apresentaram taxas de precisão para o trafego benigno acima dos 99,16%, evidenciando que a quantidade de instancias classificadas erroneamente como benigna foi baixa. Isso evidencia que houve erros de classificação entre as classes de ataque.

Os erros entre os ataques podem ser justificados por diversas variáveis. Primeiro, devido a base de criação do *dataset* NF-UQ-NIDS-V2 ter sido concebida através de uma mistura de quatro *datasets* distintos, contendo registros de diferentes tipos de ataques, desde a nível de rede até de aplicações. Outro ponto importante foi o desbalanceamento de classes, por mais que utilizamos a técnica de SMOTE no processo de treinamento, quando passamos para a etapa de teste, existem poucos registros para serem testados de algumas classes minoritárias.

7. Conclusão

Este trabalho teve como objetivo propor uma abordagem para detectar e identificar intrusões em ambientes de computação fog e IoT. A abordagem proposta é baseada na detecção de anomalias, onde visa detectar e identificar desvios no comportamento padrão dos fluxos de rede. A partir disso, investigou-se a capacidade de três classificadores baseados em Árvores de Decisão para detecção multiclasse. Além disso, utilizou-se técnicas de seleção de *features* para minimizar tanto os tempos de treinamento e testagem do classificador, como maximizar as taxas de detecção e classificação dos fluxos.

De modo a encontrar os seletores de *features* e classificadores que alcançam melhor desempenho no cenário de *Fog* e IoT, foram realizados doze experimentos com diferentes configurações de classificadores e seletores de *features*. Os resultados obtidos através destes experimentos demonstram que o Exp11, que utilizou como seletor de *features* o SFS e o classificador RF, apresentou a melhor taxa de Acurácia Balanceada entre todos os experimentos deste trabalho, ou seja, em um cenário onde há a uma maior necessidade em classificar os ataques com confiabilidade no modelo. Por contrapartida, o seu tempo de treino, que representa o tempo necessário para atualizar e disponibilizar um modelo atualizado foi superior aos classificadores DT e ET. Em um contexto onde há necessidade de utilizar modelos atualizados com criticidade, as abordagens com DT parecem ser mais recomendáveis, devido ao seu custo inferior em relação aos demais classificadores considerados.

Trabalhos futuros podem ser realizados em cima de redes acadêmicas federadas, onde há uma gama de serviços de rede IoT em diversos setores, como *datacenters*, câmeras, sensores industriais e etc. Com essa rede agregada disponível, sugerimos a implantação de um serviço de captura de fluxos com NetFlow ou IPFIX em um ponto de agregação da rede, para que a partir desse processo, seja gerado um novo *dataset* disponibilizado para o campo de pesquisa da Universidade Federal de Santa Catarina (UFSC), como a âmbito nacional, em cooperação com as demais universidades interconectadas com a Rede Nacional de Ensino e Pesquisa (RNP).

References

Atnafu, S. and Acharya, A. (2021). Comparative analysis of intrusion detection attack based on machine learning classifiers. *Indian Journal of Artificial Intelligence and*

Neural Networking, 1:22–28.

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16.

Karegowda, A. G., Manjunath, A., and 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.

Liu, H. and Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, 9(20).

Marin Tordera, Eva, M.-B. X., Jordi, Jukan, A., Ren, G.-J., and Zhu, J. (2017). Do we all really know what a fog node is? current trends towards an open definition. *Computer Communications*, 109:117–130.

Miranda, C., Kaddoum, G., Bou-Harb, E., Garg, S., and Kaur, K. (2020). A collaborative security framework for software-defined wireless sensor networks. *IEEE Transactions on Information Forensics and Security*, 15:2602–2615.

Moualla, S., Khorzom, K., and Jafar, A. (2021). Improving the performance of machine learning-based network intrusion detection systems on the unsw-nb15 dataset. *Computational Intelligence and Neuroscience*, 2021.

Priyadarshini, R. and Barik, R. K. (2019). A deep learning based intelligent framework to mitigate ddos attack in fog environment. *Journal of King Saud University-Computer and Information Sciences*.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Rokach, L. (2016). Decision forest: Twenty years of research. *Information Fusion*, 27:111–125.

Sarhan, M., Layeghy, S., Moustafa, N., and Portmann, M. (2020). Netflow datasets for machine learning-based network intrusion detection systems. *arXiv preprint arXiv:2011.09144*.

Sarhan, M., Layeghy, S., Moustafa, N., and Portmann, M. (2021). Towards a standard feature set of nids datasets. *arXiv preprint arXiv:2101.11315*.

Satyanarayanan, M. (2015). A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets. *GetMobile: Mobile Computing and Communications*, 18(4):19–23.

Shafi, Q., Basit, A., Qaisar, S., Koay, A., and Welch, I. (2018). Fog-assisted sdn controlled framework for enduring anomaly detection in an iot network. *IEEE Access*, 6:73713–73723.

T.K., B., Annavarapu, C. S. R., and Bablani, A. (2021). Machine learning algorithms for social media analysis: A survey. *Computer Science Review*, 40:100395.