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

DISSERTAÇÃO DE MESTRADO

DETECÇÃO DE INTRUSÃO EM NÓS SENSORES DE REDES DE SENSORES SEM FIO

IAN NÍCOLAS LEMES ALVES

FOZ DO IGUAÇU
2021

Ian Nícolas Lemes Alves

Detecção de Intrusão em Nós Sensores de Redes de Sensores Sem Fio

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

Orientador: Renato Bobsin Machado

Foz do Iguaçu
2021

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

Alves, Ian Nícolas Lemes
Detecção de intrusão em nós sensores de redes de sensores sem fio / Ian Nícolas Lemes Alves; orientador Renato Bobsin Machado. -- Foz do Iguaçu, 2021.
83 p.

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

1. Aprendizado de máquina. 2. Redes de sensores sem fio. 3. Detecção de intrusão. 4. Segurança computacional. I. Machado, Renato Bobsin, orient. II. Título.

Detecção de Intrusão em Nós Sensores de Redes de Sensores Sem Fio

Ian Nícolas Lemes Alves

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

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

Prof. Dr. **Carlos Becker Westphall**
Universidade Federal de Santa Catarina - UFSC

Prof. Dr. **Edgar Manuel Carreno Franco**
Universidade Estadual do Oeste do Paraná - UNIOESTE

Data da defesa: 03 de dezembro de 2021.

Resumo

Considerando a ampla utilização de sistemas computacionais assistidos pela Internet e a sensibilidade dos dados que circulam por esses sistemas, se torna necessária a aplicação de técnicas e sistemas para garantir a segurança desses dados. Os avanços recentes em tecnologias integradas como *Internet of Things*, que apresentam limitações em termos de recursos computacionais, especificamente quando lidam com Redes de Sensores Sem Fio, apresentam desafios para a aplicação de métodos convencionais de segurança, como Sistemas de Detecção de Intrusão. Este trabalho busca aplicar uma solução de segurança baseada em detecção por anomalia, direcionada para detecção de intrusão a nível de nós sensores de RSSF. Isto é feito aplicando e comparando os algoritmos *Naive Bayes, Multilayer Perceptron*, Árvore de Decisão e *Random Forest*. Na avaliação realizada são consideradas as métricas de *recall* de cada tipo de ataque abordado, a média destes e a utilização de recursos em termos de consumo de memória, energia e tempo de execução. Para a realização do treinamento e testes desses algoritmos, foi utilizada a base de dados WSN-DS, que lida especificamente com ataques direcionados a Redes de Sensores Sem Fio. O método proposto baseia-se no processo de *Knowledge Discovery in Databases*, e aplica seleção de atributos na base original, além de análise e ajuste dos parâmetros dos algoritmos, de acordo com os requerimentos dos nós sensores observados. Com a execução dos algoritmos e análise das métricas de performance, foi observada boa performance para os tipos de ataques abordados pelos algoritmos de Árvore de Decisão e *Random Forest*, ambos obtendo performance similar de classificação, no entanto, o *Random Forest* apresentou um consumo de recursos consideravelmente maior do que a Árvore de Decisão.

Palavras-chave: Aprendizado de Máquina, Redes de Sensores Sem Fio, Nós Sensores, Detecção de Intrusão, Segurança Computacional.

Abstract

Considering the wide usage of internet assisted computer systems and the sensitivity of their data, the application of systems and techniques that guarantee the security of that data becomes a necessity. The recent advancements in integrated technologies such as the Internet of Things, that are limited by their available resources, present challenges to the application of conventional security approaches like Intrusion Detection Systems, specially for Wireless Sensor Networks. This work aims to employ a security solution inspired by anomaly detection, applied to intrusion detection at the sensor node level in WSN. This is achieved by employing and comparing the Naïve Bayes, Multilayer Perceptron, Decision Tree and Random Forest algorithms, these comparisons are focused on the individual recall of each attack type, its average and, in terms of resource usage, the memory, energy and run time. The WSN-DS dataset, which deals with attacks to Wireless Sensor Networks, was used for training and testing and the proposed methodology is based on the Knowledge Discovery in Databases process, attribute selection was employed on the original dataset, analysis and tuning were performed to adjust algorithm parameters, according to the requirements of the evaluated sensor nodes. Through the algorithm execution and analysis of the performance metrics, the good performance of both the Decision Tree and the Random Forest algorithms could be observed, with the Random Forest algorithm displaying much higher resource consumption than the Decision Tree Algorithm.

Keywords: Machine Learning, Wireless Sensor Networks, Sensor Nodes, Intrusion Detection, Network Security.

Agradecimentos

Primeiramente, agradeço aos meus pais, Luciene e Crispin, que sempre me incentivaram ao estudo e se dedicaram para que eu tivesse acesso a melhor educação possível, e ao meu irmão, Erick. Aos meus tios, Lila e Wagner, Solange, aos primos Nataly, Analy e Diego que sempre estiveram comigo, me incentivando em todos os momentos de minha vida. Um muito obrigado a todos os meus amigos físicos e virtuais que estavam presentes para me fazer companhia por todos esses anos. Agradeço a todos os professores que fizeram parte da minha educação. Em especial a meu orientador, Renato, e ao Cristiano, que me conduziram nesta jornada, aos meus colegas do grupo de pesquisa do LapSec e aos professores que se dispuseram a fazer parte desta banca.

“We are the sum of our deeds,
not our names.”
Ramza Beoulve

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Abreviaturas

CERT.BR	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
IoT	Internet of Things
SDI	Sistema de Detecção de Intrusão
IDS	Intrusion Detection System
IA	Inteligência Artificial
AI	Artificial Intelligence
AM	Aprendizado de Máquina
ML	Machine Learning
KDD	Knowledge Discovery in Databases
SEMMA	Sample, Explore, Modify, Model, Assess
CRISP-DM	Cross Industry Standard Process for Data Mining
RNA	Redes Neurais Artificiais
MLP	Multilayer Perceptron
OVO	One-Versus-One
OVA	One-Versus-All
RSSF	Redes de Sensores Sem Fio
LEACH	Low-Energy Adaptive Clustering Hierarchy
SPINS	Security Protocols for Sensor Networks
miTESLA	Timed, Efficient, Streaming, Loss-tolerant Authentication Protocol
SNEP	Secure Network Encryption Protocol
DDOS	Distributed Denial of Service
DOS	Denial of Service
BBO	Biogeography-based Optimization
KNN	K-nearest Neighbor
SCA	Sine Cosine Algorithm
PM-CSCA	Polymorphic Mutation Compact Sine Cosine Algorithm
RNN	Random Neural Network
SVM	Support Vector Machine
HADL	Hierarchical Anomaly Detection and Localization
DSR	Dynamic Source Routing

NS-2	Network Simulation-2
EFR	Evolutionary Fuzzy Rules
NB	Naive Bayes
DT	Decision Tree
RF	Random Forest
HL	Hidden Layer
MD	Max Depth
NT	Number of Trees

Capítulo 1

Introdução

A Internet é um componente essencial do modelo de negócio de grande parte das entidades comerciais em operação, além de ser acessada por usuários comuns para troca de informações pessoais e na utilização de diversos serviços. O acesso indevido a essas informações apresenta um grande risco para a integridade dos serviços oferecidos e para a segurança de seus usuários. A área de segurança computacional visa proteger sistemas e recursos contra acessos indevidos ou invasões, e baseiam-se em manter a integridade, confidencialidade e disponibilidade desses dados (Bace, Mell et al., 2001).

Internet of Things (IoT) pode ser definida como um *framework* conceitual que utiliza de dispositivos, serviços e objetos comuns interconectados para possibilitar o desenvolvimento de soluções (Atzori, Iera & Morabito, 2017). Entre os diversos tipos de dispositivos em IoT, existem redes de sensores interconectados por uma rede sem fio, com restrições de processamento e consumo de energia, utilizados para captura e processamentos de dados, estes são chamados Redes de Sensores Sem Fio (RSSF) (Kavitha & Sridharan, 2010).

Considerando a necessidade de transmissão de dados em RSSF, a sensitividade destes e restrições da rede e dispositivos, garantir a segurança e integridade desses dados é fundamental. Diversos tipos de ataques podem ser direcionados a RSSF, e podem ser classificados com base na origem do ataque, capacidades do invasor ou efeito no fluxo de dados (Kavitha & Sridharan, 2010).

Existem sistemas desenvolvidos especificamente para auxiliar na detecção desses tipos de incidente, chamados Sistemas de Detecção de Intrusão (SDI) ou *Intrusion Detection System* (IDS), estes utilizam técnicas automatizadas para identificar e responder a eventos anômalos sem a necessidade de intervenção humana. De acordo com Bace et al. (2001), há duas formas principais de analisar eventos para detecção de intrusão, estas são detecção baseada em mau uso ou assinatura, e detecção baseada em anomalia.

1.1 Proposta da Dissertação

Certos tipos de ataques podem ter um impacto significativo em RSSF, causando indisponibilidade em grandes áreas dessas redes, necessitando apenas do comprometimento de um número pequeno de nós sensores. Muitas técnicas aplicadas para a detecção de intrusão em RSSF realizam o processo de detecção a níveis mais altos da hierarquia da rede, como em *Cluster Heads* ou estações bases, em certos casos isso pode dificultar o processo de detecção e a realização de contramedidas no tempo necessário. Há um pequeno número de trabalhos que lidam com detecção de intrusão ao nível de nós sensores e analisam os possíveis benefícios da aplicação de detecção por anomalia nesse contexto. Desse modo, este trabalho propõe a aplicação de técnicas de detecção de intrusão com aprendizado de máquina para a detecção de múltiplas ameaças (multiclasse) ao nível de nós sensores em RSSF. A aplicação de técnicas de detecção ao nível de nós sensores pode ajudar a garantir a segurança da RSSF, tornando o processo geral de detecção de intrusão mais robusto, além disso, a detecção multiclasse pode acelerar ainda mais a realização de contramedidas, provendo informações mais detalhadas sobre o ataque observado.

Durante o processo de revisão da literatura foram observados diversos tipos de técnicas aplicadas a RSSF para detectar intrusões. Alguns utilizam detecção por assinatura, estratégia que requer um poder de processamento menor, mas não possui a capacidade de identificar ataques desconhecidos, ou seja, que não estão na base de assinaturas. Outros abordam a detecção por anomalia utilizando técnicas de aprendizado de máquina, que geralmente causam um consumo maior de recursos, mas conseguem lidar com ameaças desconhecidas. Entre as técnicas geralmente aplicadas em diversos níveis de RSSF, foram identificados alguns algoritmos de aprendizado de máquina clássicos que apresentam bons resultados em diversos trabalhos, estes foram *Naive Bayes*, Redes Neurais Artificiais e Árvores de Decisão. Além disso, também foi observado o uso de métodos *ensembles* aplicados na detecção de intrusão em RSSF, especificamente *Random Forest*, que demonstra bons resultados além da sua capacidade de lidar com ruídos de classificação e *overfitting*. A partir dessas observações, a aplicação dessas técnicas para detecção de intrusão ao nível de nós sensores foi considerada.

Os algoritmos de *Naive Bayes*, Redes Neurais, Árvores de Decisão e *Random Forests* foram utilizados em experimentos e a performance destes foi comparada levando em consideração as limitações de recursos em nós sensores. Especificamente, foram avaliadas as performances de classificação dos algoritmos e o consumo de recursos. Em termos de performance de classificação, foram considerados principalmente o *Recall* de cada classe e a média dos *Recalls*, que lidam com o conceito de falsos negativos em detecção de intrusão. Para o consumo de recursos foi considerado o tamanho dos modelos treinados, tempo de predição e consumo de energia. Essas métricas foram utilizadas para ajustar os parâmetros dos modelos tentando manter sua performance e consumo de recursos em limites aceitáveis para a aplicação em nós sensores.

Com a execução dos algoritmos e extração das métricas definidas, foi possível avaliar a viabilidade da aplicação destes para detecção de ameaças em nós sensores de RSSF. Essas métricas foram avaliadas realizando comparações com representações gráficas, e com análise de testes estatísticos. Com isso foi observado que os melhores candidatos para aplicações em nós sensores são Árvore de Decisão e *Random Forest*, por obterem os melhores resultados de *recall* individuais e de acurácia balanceada. Porém, dada a natureza do algoritmo *Random Forest*, este demonstrou consumo de recursos consideravelmente maior do que a Árvore de Decisão, o que o torna mais indicado para aplicações com maior poder computacional, tendo em vista a sua capacidade de lidar com ruídos de classificação e de treinamento e *overfitting*.

1.2 Estrutura do Trabalho

O Capítulo 2 apresenta alguns conceitos necessários para compreensão do conteúdo do trabalho. Estes englobam RSSF, detecção de intrusão, SDI, inteligência artificial e aprendizado de máquina.

O Capítulo 3 lida com segurança em RSSF, se aprofundando e conectando alguns conceitos apresentados no Capítulo 2. São apresentados desafios de segurança em RSSF, ameaças a RSSF e o processo de detecção de intrusão em RSSF.

No Capítulo 4 são apresentados alguns trabalhos relacionados ao estado da arte da área de detecção de intrusão em RSSF. Os trabalhos são organizados em categorias que lidam com detecção de intrusão em RSSF e detecção de intrusão ou anomalias em nós sensores.

O Capítulo 5 apresenta o método experimental e materiais. Incluindo discussão do processo de definição e das etapas do método, base de dados e ferramentas utilizadas.

O Capítulo 6 apresenta os resultados obtidos do processo de classificação e da obtenção das métricas de performance. Os resultados também são discutidos neste capítulo utilizando técnicas de visualização e análise estatística.

No Capítulo 7 são apresentadas as conclusões do trabalho, além de possíveis trabalhos futuros.

Capítulo 2

Conceitos Básicos

Este capítulo aborda em termos gerais as áreas de relevância para a execução do trabalho, fornecendo o contexto necessário para compreensão da proposta. Serão introduzidos conceitos básicos de Redes de Sensores Sem Fio (RSSF), ameaças a redes de computadores, Sistemas de Detecção de Intrusão (SDI), Inteligência Artificial (IA) e algoritmos de Aprendizado de Máquina (AM).

2.1 Redes de Sensores Sem Fio

Redes de Sensores Sem Fio são redes de dispositivos capazes de observar características de objetos reais e traduzi-las para um meio digital, estes dispositivos são chamados sensores, sendo que para processar os sinais capturados pelos sensores, geralmente é necessário realizar certas transformações nos mesmos, aplicando técnicas de condicionamento e realizando conversão dos sinais capturados. As técnicas aplicadas geralmente consistem na amplificação do sinal, aplicação de filtros e então a aplicação de conversão analógica-digital. Além disso, é bastante comum a existência de atuadores em RSSF, possibilitando a realização de ações no ambiente observado, de acordo com os sinais capturados pelos sensores. A Figura 2.1 apresenta o processo de aquisição e conversão de dados por sensores e a ação de atuadores (Dargie & Poellabauer, 2010).

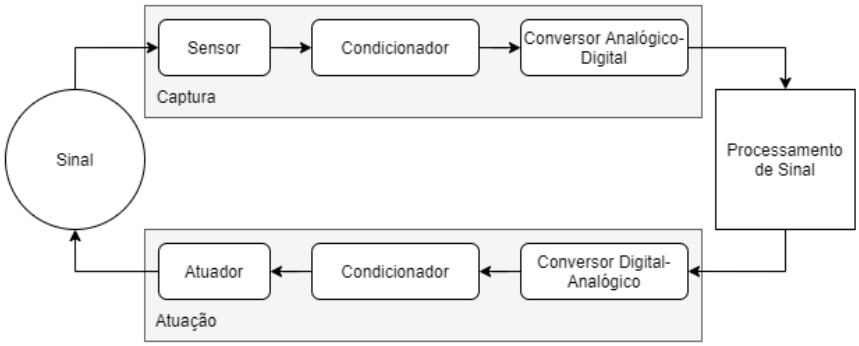

Figura 2.1: Processo de captura e atuação em RSSF.
Fonte: Adaptado de (Dargie & Poellabauer, 2010).

Em RSSF, os nós sensores da rede geralmente são responsáveis por diversas outras tarefas, além de realizarem leituras no ambiente. Em diversos tipos de RSSF, os sensores também são responsáveis por verificar o estado da rede, realizar o processamento dos dados capturados e também de dados de outros sensores, mantendo comunicações com a estação base e outros sensores da rede. A disposição e a distância entre os sensores de uma RSSF definem a topologia da rede, os sensores podem se comunicar diretamente com a estação base ou utilizar outros sensores para propagar suas mensagens. Estes dois modelos de comunicação são chamados *Single-hop* e *Multi-hop* respectivamente, sendo apresentados na Figura 2.2 (Dargie & Poellabauer, 2010).

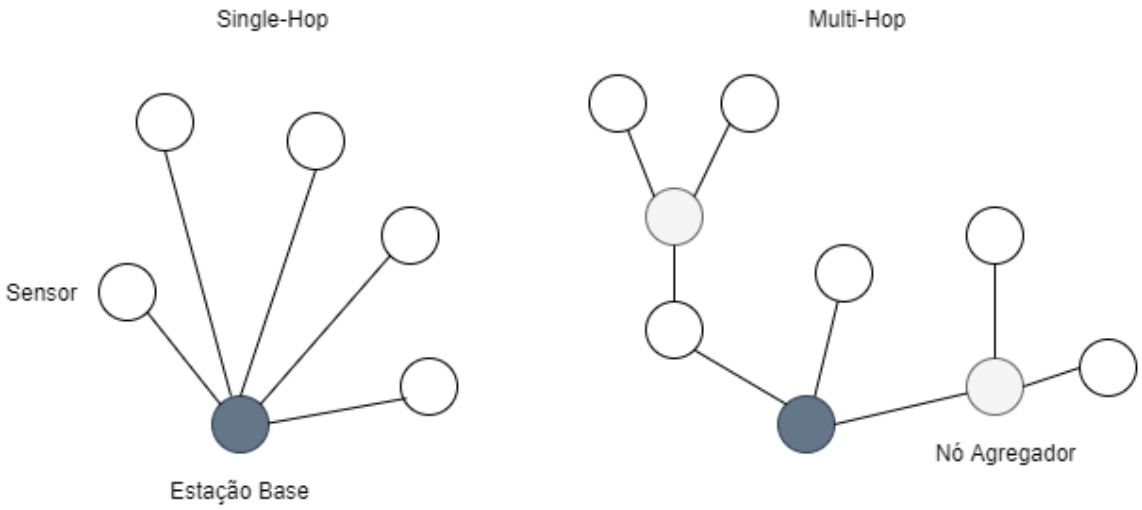

Figura 2.2: Comunicação *Single-hop* e *Multi-hop* em RSSF.
Fonte: Adaptado de (Dargie & Poellabauer, 2010).

As RSSF também apresentam diversas características que criam dificuldades específicas a estas redes, que não são comuns a outras formas de sistemas distribuídos. Estas podem ser descritas como restrições em disponibilidade de recursos e processamento, especificamente podem ser apontados problemas relacionados à (Dargie & Poellabauer, 2010):

- **Consumo de Energia:** Os nós sensores geralmente operam com baterias, o que restringe a quantidade de energia disponível para o processamento de dados e comunicação;
- **Comunicação sem fio:** O meio de comunicação da rede também apresenta desafios, já que existem limitações quanto a atenuação dos sinais utilizados, limitando o alcance da comunicação;
- **Segurança:** Diversas aplicações de RSSF consistem na coleta de dados de alta importância, além disso, a natureza descentralizada e remota da operação dos nós sensores em RSSF facilita a aplicação de ataques a membros da rede.

2.1.1 Arquitetura de um Nó Sensor

O nó sensor é o principal componente de uma RSSF, os quais são responsáveis pela realização das tarefas de comunicação, medição e processamento. Deste modo, as capacidades e performance da rede estão diretamente relacionadas com os recursos disponíveis aos nós sensores. Estes, são compostos principalmente por subsistemas responsáveis pela execução de tarefas de medição, processamento, comunicação e gestão de energia. A Figura 2.3 apresenta uma arquitetura básica de um nó sensor, os subsistemas apresentados na Figura 2.3 são (Dargie & Poellabauer, 2010):

- **Subsistema de Sensor:** Contém os sensores do nó sensor, além de conversores analógico-digital e mecanismos de gestão para compartilhar os mecanismos de conversão entre os sensores;
- **Subsistema de Processamento:** Responsável por integrar os outros subsistemas aos outros mecanismos do nó sensor, como a execução de instruções programáveis. Os principais componentes computacionais estão contidos neste subsistema, como o processador e memórias;
- **Subsistema de Comunicação:** Responsável por realizar a transferência de dados entre o nó sensor e outras entidades da RSSF.

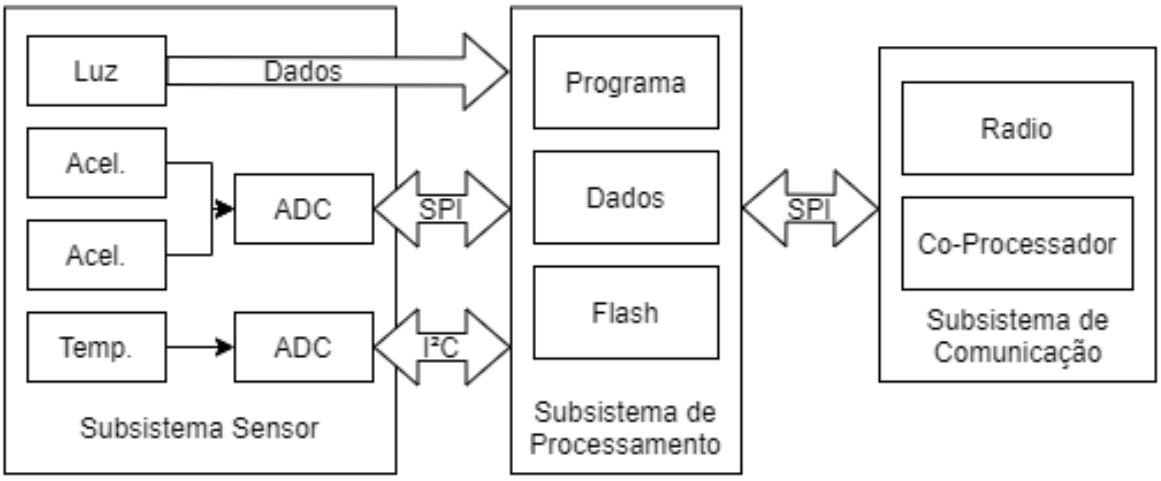

Figura 2.3: Arquitetura de um Nó Sensor de uma RSSF.
Fonte: Adaptado de (Dargie & Poellabauer, 2010).

2.1.2 Aplicações de RSSF

RSSF podem ser aplicadas em diversas áreas, desde monitoramento de ambientes até sistemas de assistência médica. Algumas das aplicações mais importantes destas redes se concentram em áreas de alto risco ou que necessitem de monitoramento constante. Alguns exemplos são:

- **Monitoria de estruturas:** Monitoria e manutenção de estruturas geralmente ocorrem em algumas etapas sendo realizada por equipes de especialistas através de inspeções, estas requerem uma grande quantidade de tempo e recursos, mas são importantes para evitar a ocorrência de eventos catastróficos como o colapso de prédios ou pontes. RSSF tem sido aplicadas no processo de inspeção global, responsável pela detecção de danos capazes de afetar estruturas inteiras. Estas redes geralmente podem ser implantadas sem interrupções na área, requerem pouca manutenção e possibilitam realizar correlações entre diferentes dados de sensores, auxiliando na detecção de falhas estruturais menos visíveis (Dargie & Poellabauer, 2010);
- **Assistência médica:** Há diversas propostas de aplicações de RSSF na área de assistência médica, estas visam prover informações detalhadas sobre doenças e prevenção, integração com a infraestrutura de saúde em operações de resgate e emergência, sistemas de monitoramento de saúde não intrusivos e auxiliar no processo de atendimento a pacientes em situações de internação (Dargie & Poellabauer, 2010);
- **Monitoria de vulcões ativos:** O monitoramento de vulcões ativos geralmente se da pela utilização de diversos dispositivos de alto custo, com difícil implantação e requerem alimentação de energia externa. Uma alternativa é a utilização de RSSF, que tem custo menor, e apresenta menos dificuldades na implantação. Um protótipo de RSSF foi criado e testado por Werner-Allen, Lorincz, Ruiz, Marcillo, Johnson, Lees & Welsh (2006);

2.2 Segurança em Redes de Computadores

O Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT .br) descreve incidentes de segurança como, qualquer evento adverso, relacionado com a segurança de sistemas de computação ou redes de computadores. Exemplos citados são, tentativas de adquirir acesso não autorizado a sistemas ou dados, comprometer a disponibilidade de sistemas, ou realizar modificações não autorizadas em um sistema. A Figura 2.4 apresenta o total de incidentes reportados ao CERT.br por ano, entre 1999 e 2020.

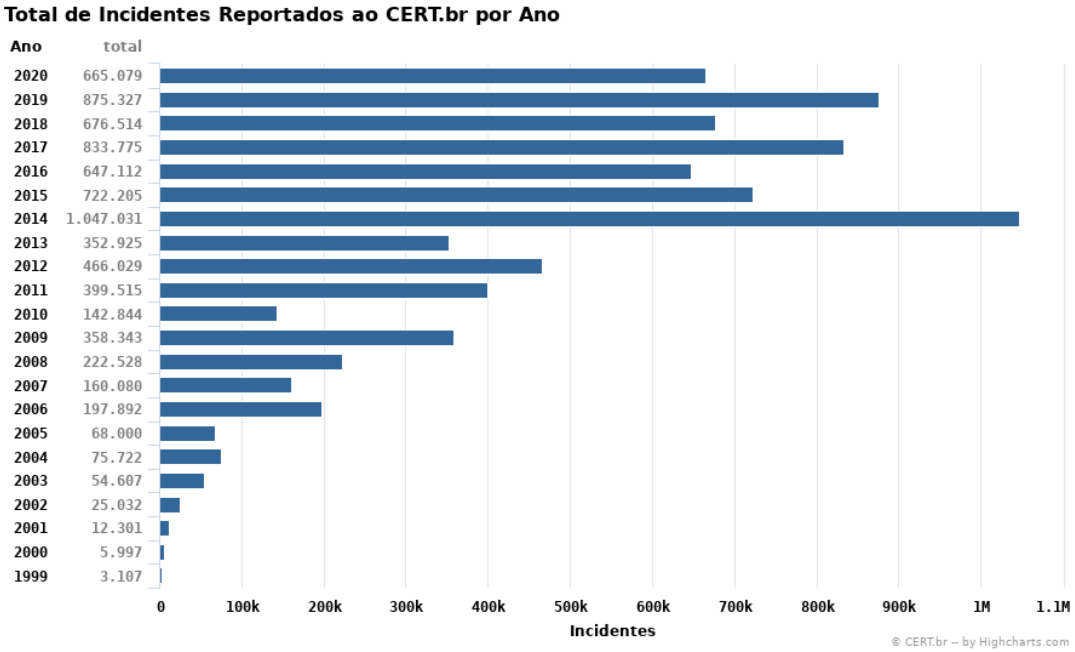

Figura 2.4: Total de incidentes reportados ao CERT.br por ano.
Fonte: (*Total de Incidentes Reportados ao CERT.br por Ano*, 2020).

2.2.1 Ameaças a Redes de Computadores

Ataques a sistemas geralmente afetam componentes específicos dos mesmos. Alguns tipos principais de ataques são descritos em Bace et al. (2001), estes são classificados por sua capacidade de comprometer:

- **Confidencialidade:** Permite que o atacante tenha acesso a dados do sistema sem devida autorização;
- **Integridade:** Permite que o atacante realize mudanças ao estado do sistema ou de dados contidos neste;
- **Disponibilidade:** Impede que usuários comuns do sistema tenham acesso a determinados recursos do mesmo;
- **Controle:** Permite que o atacante conceda um nível de privilégio não condizente com as políticas de controle de acesso do sistema, o que permite subsequente violações de confidencialidade, integridade ou disponibilidade;

2.2.2 Sistemas de Detecção de Intrusão

Sistemas desenvolvidos com o propósito de detectar incidentes de segurança são chamados Sistemas de Detecção de Intrusão. Estes utilizam técnicas automatizadas para identificar e responder a eventos de intrusão sem a necessidade de intervenção humana. Eventos de intru-

são são descritos por Bace et al. (2001) como tentativas de comprometer a confidencialidade, integridade ou disponibilidade de um sistema.

Classificação de SDI

Existem vários tipos de SDI que são caracterizados principalmente por suas capacidades de monitoramento e análise, Campello & Weber (2001) descreve subdivisões para classificar SDI em termos de método de detecção, arquitetura, tipo de resposta e frequência de uso. A Figura 2.5 apresenta esta classificação.

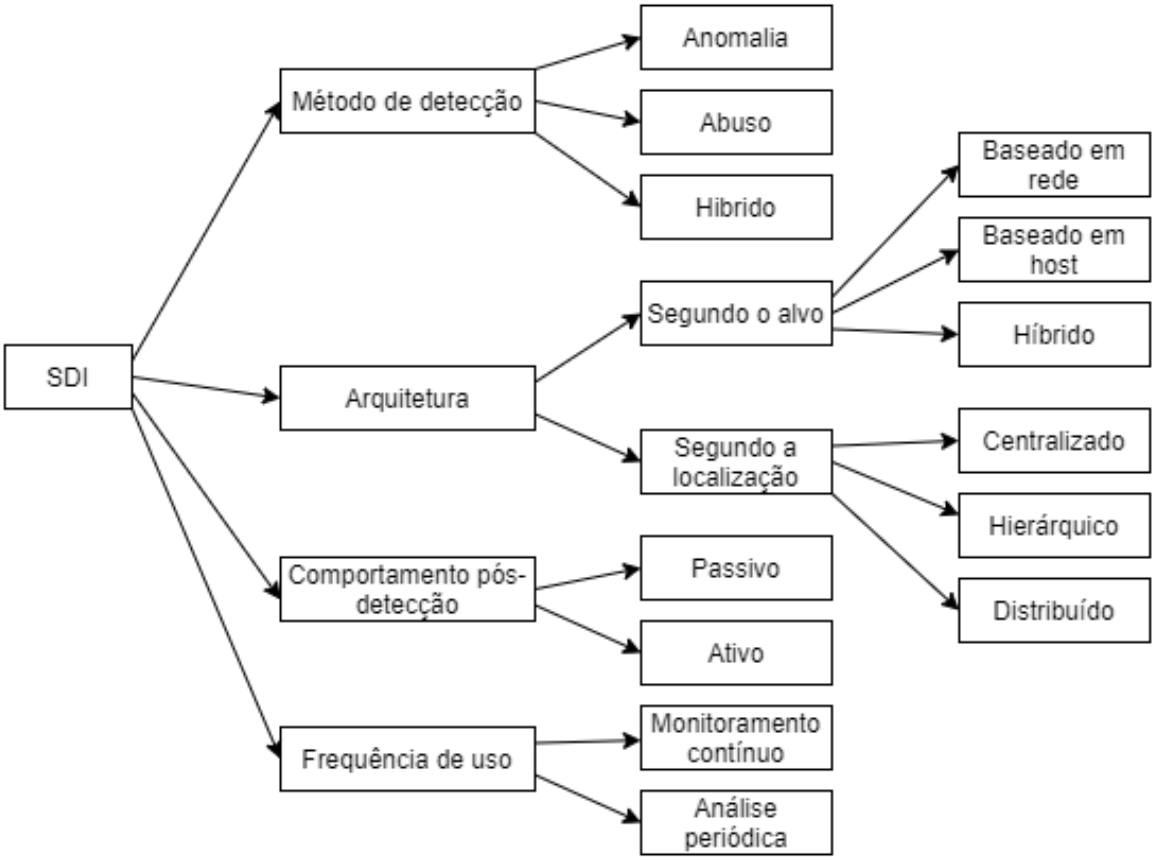

Figura 2.5: Classificação de SDI.
Fonte: Adaptado de (Campello & Weber, 2001).

De maneira similar, Bace et al. (2001) aponta três principais componentes para classificação de SDI, fontes de informação, forma de análise e tipo de resposta:

- **Fontes de Informação:** Informações sobre os eventos. São utilizadas pelo sistema para decidir se o evento analisado é uma intrusão;
- **Análise:** Responsável por verificar as informações coletadas e identificar se um evento é uma intrusão;

- **Resposta:** Ações tomadas pelo sistema quando uma intrusão é identificada, podem ser ações ativas, que envolvem alguma intervenção automatizada pelo sistema, ou passivas, que reportam o evento identificado para especialistas humanos.

Em termos de arquitetura, Bace et al. (2001) descreve dois componentes principais, o *Host*, que é o sistema onde o SDI é executado, e o *Target* (Alvo), que é o sistema que o SDI monitora. Grande parte dos primeiros SDI eram executados no mesmo sistema que monitoravam, o que é chamado de Co-localização de *Host-Target*. Isso ocorria principalmente pela prevalência de mainframes e do custo de *hardware*, e apresentava problemas para segurança do sistema. SDI modernos geralmente adotam o modelo de Separação de *Host-Target*, no qual o SDI não é executado no mesmo sistema que monitora. Os tipos de estratégias de controle descritas são:

- **Centralizada:** Todas as tarefas realizadas pelo SDI (monitoramento, detecção e notificação) são executadas de forma centralizada por um único agente;
- **Parcialmente Distribuída:** As tarefas de monitoramento e detecção são controladas por um agente, e as tarefas de notificação são divididas hierarquicamente e direcionadas para um agente central;
- **Distribuída:** As tarefas de monitoramento, detecção e realização de respostas são realizadas por vários agentes individuais;

A frequência de uso descreve o período de tempo entre a observação de um evento. Na tarefa de análise, SDI podem ser classificados como (Bace et al., 2001):

- ***Batch*:** Envia informações observadas para análise em intervalos de tempo pré-definidos, não possibilita a execução ativa de respostas;
- **Tempo Real:** Envia as informações observadas para análise continuamente, possibilitando respostas rápidas aos eventos observados;

De acordo com Bace et al. (2001), há duas formas de analisar eventos para detecção de intrusão, chamadas de detecção baseada em mau uso ou assinatura, e detecção baseada em anomalia.

- **Assinatura:** Técnica comumente utilizada em grande parte dos SDI comerciais, a qual utiliza registros de eventos conhecidos e os compara com eventos ocorrendo na rede ou sistema para detectar intrusões;
- **Anomalia:** SDI que realizam detecção por anomalia são capazes de identificar eventos anômalos na rede ou sistema, sem conhecimento prévio destes eventos, sendo que geralmente utilizam técnicas de aprendizado de máquina para modelar o comportamento normal;

Outras formas de detecção de intrusão são detecção baseada em especificação (Uppuluri & Sekar, 2001), e detecção baseada em confiança (Wang, Huang, Zhao & Rong, 2008).

- **Especificação:** Consiste da união de conceitos de detecção por assinatura e anomalia. Neste esquema, especialistas criam especificações de comportamento normal do sistema ou rede, e o sistema então, deve ser capaz de identificar eventos que desviem do comportamento esperado como eventos anômalos;
- **Confiança:** Aplica conceitos de *Evidence Chain* e *Trust Fluctuation* para avaliar o valor de confiança de cada nó na rede. A *Evidence Chain* para cada nó representa uma série incremental de comportamentos que o mesmo apresenta, com semelhanças a comportamentos não aceitos pelo sistema ou rede, quanto maior esta corrente de evidências, menor o valor de confiança do nó. O *Trust Fluctuation* é responsável por identificar mudanças no valor de confiança de um nó, geralmente mudanças drásticas no valor de confiança de um nó indicam eventos anômalos;

Em termos de respostas à eventos analisados, Bace et al. (2001) descreve:

- **Ativa:** Realizam ações automatizadas quando eventos de intrusão são identificados, estas podem ser, coleta de mais informações, realizar mudanças no ambiente ou tomar ações contra o atacante;
- **Passiva:** Consiste na emissão de alarmes ou notificações aos usuários do sistema, para que os mesmos respondam adequadamente ao evento de intrusão;

Além destas formas em comum de classificação, Bace et al. (2001) também descreve a classificação de um SDI em termos de suas fontes de informação, estes podem ser:

- **SDI em Rede (*Network Based* IDS):** Detectam intrusões através da captura e análise de pacotes da rede, desta maneira estes SDI são capazes de operar em diversos hosts conectados na rede;
- **SDI baseado em *Host* (*Host Based* IDS):** Operam em um computador específico, possibilitando acesso a grande parte das informações do host e não apenas sobre trafego da rede;
- **SDI baseado em Aplicação (*Application Based* IDS):** São especializações de SDI baseados em host, estes operam sobre uma aplicação específica no *host*;

2.3 Inteligência Artificial

Inteligência artificial pode ser descrita como a área de pesquisa com foco no estudo dos processos do pensamento humano, e a replicação destes através de sistemas computacionais.

Em Russell & Norvig (2010), IA é definida em termos de seu processo de pensamento, raciocínio e comportamento, especificamente, nas suas capacidades de:

- **Pensar de forma humana:** Compara-se o processo de resolução de um problema pela IA com o processo adotado no pensamento humano;
- **Pensar racionalmente:** O processo de pensamento racional é descrito em termos de criar estruturas de informação que possam ser analisadas de forma lógica para obter um resultado correto;
- **Agir de forma humana:** Demonstrar comportamento similar ao de um humano, sendo capaz de passar no teste de Turing (Turing, 1950);
- **Agir racionalmente:** Um agente racional deve ser capaz de realizar ações de acordo com o pensamento racional afim de obter o resultado correto, ou na ausência deste, o melhor resultado possível;

Através da replicação desses processos por sistemas computacionais, estes conseguem solucionar problemas com ou sem auxílio de interação humana. Uma forma simples de solução de problemas através de IA, é realizada pela utilização de Agentes Inteligentes. Neste contexto, um agente pode ser definido como qualquer coisa capaz de perceber o seu ambiente através de algum tipo de sensor, e capaz de agir sobre o mesmo através de atores (Russell & Norvig, 2010). Estes agentes são guiados por alguma métrica de performance que define seu sucesso com base em suas ações.

Desta forma, um agente racional opera sobre o conhecimento que têm de um ambiente, buscando otimizar uma métrica de performance definida. Esta forma de resolução de problemas é bastante dependente do tamanho do domínio do mesmo. Quanto maior o número de estados possíveis para uma determinada solução, maior o custo de mapear estes, para que a resolução do problema seja efetiva.

Uma forma de evitar este problema é através da especialização de agentes racionais, como agentes de resolução de problemas, que especificam um objetivo para o problema e realizam uma sequência de passos para atingir este. Isto é feito através da formulação apropriada do problema, para definir as possíveis ações do agente e os possíveis estados do ambiente. Um problema bem formulado deve conter uma série de possíveis estados, um estado inicial, ações que o agente pode realizar, uma função de transição, um teste para definir se um estado é um possível objetivo, e uma função de custo para cada ação (Russell & Norvig, 2010). Para encontrar a solução para o problema proposto são aplicados algoritmos de busca, que avaliam cada possível mudança de estado e definem a melhor sequência entre o estado inicial e um estado objetivo, considerando uma função de custo ou métrica de performance definida no problema.

2.3.1 Aprendizado de Máquina e KDD

Alan Turing (Turing, 1950) contempla a possibilidade de criar um programa capaz de reproduzir o funcionamento e os tipos de informações contidas na mente humana, de modo manual, estimando cerca de 60 programadores trabalhando por 50 anos a uma taxa de 1000 caracteres de programa por dia. Isto aponta a necessidade da criação de formas mais eficientes e automáticas de desenvolvimento de máquinas capazes de 'pensar'.

Aprendizado de máquina é descrito em Witten, Frank & Hall (2017) como técnicas utilizadas para encontrar padrões em dados, que facilitem o processo de tomada de decisão e melhorem o entendimento de um certo domínio. Técnicas de aprendizado de máquina geralmente se preocupam em criar descrições estruturais do que é aprendido, através da observação de exemplos conhecidos de um domínio, estas descrições são então utilizadas para avaliar novos exemplos não conhecidos. Este processo não é apenas útil por suas capacidades preditivas, mas também para expandir o conhecimento de especialistas sobre um determinado domínio.

Uma das principais aplicações de aprendizado de máquina é no processo de mineração de dados, este consiste na aplicação de técnicas automatizadas para extração de conhecimento de bases com um grande volume de dados. A mineração de dados é descrita em Azevedo & Santos (2008) como uma das fases do processo de *Knowledge Discovery in Databases* (KDD), modelo que descreve o processo de obtenção de conhecimento em bases de dados, descrito anteriormente.

O processo de KDD é definido em Fayyad, Piatetsky-Shapiro & Smyth (1996), como o processo de mapeamento de dados de baixo nível e transformação destes para outros formatos capazes de expressar mais informações sobre os mesmos, como um modelo descritivo do processo. A Figura 2.6 apresenta os passos do processo de KDD como descrito por (Fayyad et al., 1996).

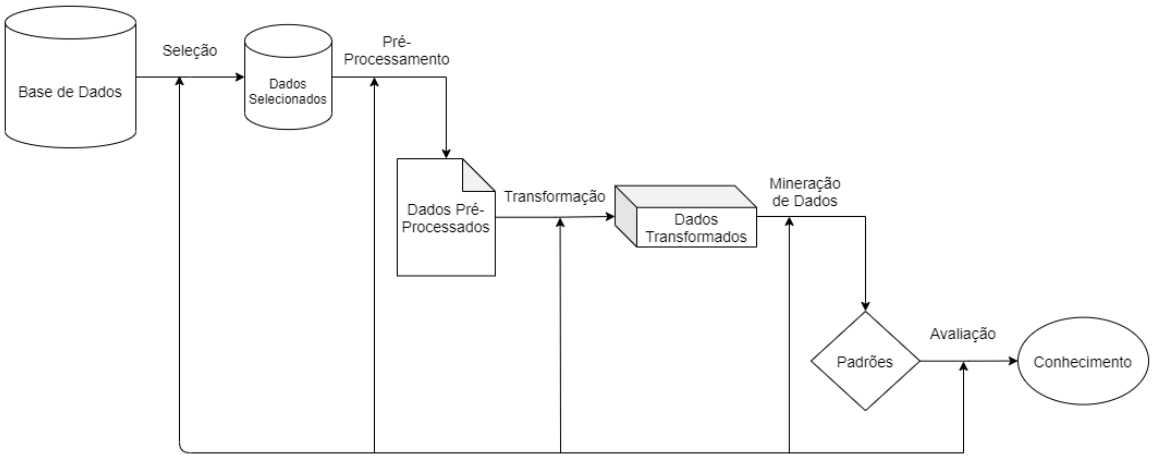

Figura 2.6: Processo de KDD.
Fonte: Adaptado de (Fayyad et al., 1996).

As fases do processo de KDD são descritas em (Fayyad et al., 1996), sendo divididas em 5 etapas:

- **Seleção:** Após a definição do domínio do problema e a identificação das metas para o processo de KDD, o conjunto de dado a ser explorado é definido;
- **Pré-processamento:** O conjunto de dados deve passar por um pré-processamento, onde são realizadas certas operações para reduzir ruídos ou redundâncias no conjunto de dados;
- **Transformação:** A base pré-processada é analisada para verificar a relevância dos dados representados, os atributos da base são verificados para garantir que os dados contidos nas instâncias sejam relevantes para o processo da descoberta de conhecimento;
- **Mineração de Dados:** Nesta etapa o método de mineração de dados é selecionado, este deve estar de acordo com as metas definidas para o processo de KDD, o método selecionado é aplicado para criação de um modelo dos dados e extrair padrões da base;
- **Avaliação:** O processo de avaliação consiste na análise e interpretação dos padrões extraídos, esta pode ser auxiliada por técnicas estatísticas e de visualização. Este processo é iterativo, após a análise, o conhecimento obtido pode ser utilizado para melhorar o modelo, realizando mudanças em outros passos do processo de KDD;

Também existem outros modelos que buscam criar modelos para o processo de descoberta de conhecimento (Azevedo & Santos, 2008), conforme cita SEMMA e CRISP-DM. SEMMA é uma sigla que descreve as 5 (cinco) etapas do processo, *Sample, Explore, Modify, Model, Asses*:

- ***Sample*:** Consiste na seleção de dados para o processo, geralmente uma amostra representativa de uma base de dados com um grande conjunto de dados é extraída;
- ***Explore*:** Os dados selecionados são explorados em busca de padrões ou anomalias, visando ganhar melhor entendimento do domínio;
- ***Modify*:** Consiste na realização de transformações nos dados, visando uma melhor performance no processo de criação de modelos;
- ***Model*:** Através da realização de buscas por um software ou algoritmo específico, um modelo que represente os dados é criado;
- ***Assess*:** É a fase de avaliação das informações extraídas dos dados através da extração e comparação de métricas de performance.

O processo CRISP-DM (*Cross-Industry Standard Process for Data Mining*) é cíclico, consistindo de 6 etapas (Azevedo & Santos, 2008):

- ***Business understanding*:** Consiste em adquirir um bom entendimento das metas e requerimentos do processo, e a partir deste, definir um problema de mineração de dados, e realizar um planejamento do processo;

- ***Data understanding*:** Nesta etapa os dados são coletados e uma familiarização com os mesmos ocorre, visando identificar problemas, verificar relações entre certos conjuntos de dados ou criar hipóteses sobre os mesmos;
- ***Data preparation*:** Consiste na criação de uma base de dados utilizando o conhecimento e dados adquiridos nas etapas anteriores;
- ***Modeling*:** Diversas técnicas de modelagem dos dados são selecionadas e avaliadas para aplicação no processo, seus parâmetros também são ajustados para valores considerados ótimos;
- ***Evaluation*:** Os modelos obtidos são avaliados e o processo definido para construção dos mesmos são revisados e ajustados caso necessário;
- ***Deployment*:** Os modelos são então aplicados a fim de atingir as metas definidas ao início do processo;

A Figura 2.7 apresenta o processo de mineração de dados de acordo com o modelo de referência CRISP-DM (Witten et al., 2017).

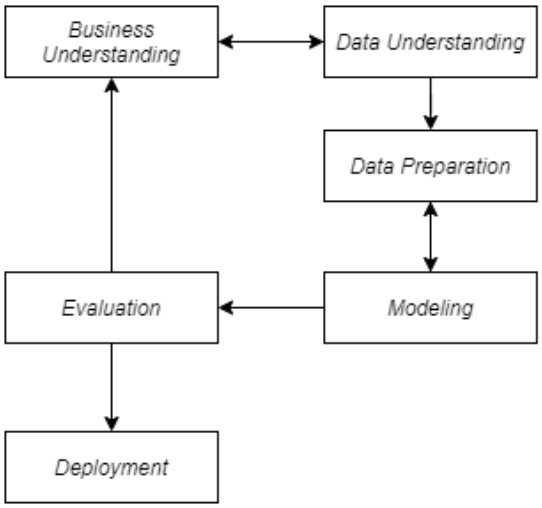

Figura 2.7: Processo de mineração de dados de acordo com o modelo CRISP-DM.
Fonte: Adaptado de (Witten et al., 2017).

2.3.2 Algoritmos Clássicos de Aprendizado de Máquina

Os algoritmos utilizados para extração de conhecimento no processo de KDD geram modelos para representar o conhecimento da base de dados. Estes modelos podem então ser utilizados para realizar predições sobre novos dados. A seguir serão apresentados alguns algoritmos de aprendizado de máquina que serão avaliados no método experimental deste trabalho para detecção de intrusão a nível de sensores.

Árvore de Decisão

As árvores de decisão são modelos hierárquicos de aprendizado de máquina para aprendizado supervisionado, compostas por nós internos de decisão e nós finais, chamados nós folha, estes representam as classes do problema. A estrutura de uma árvore de decisão é construída através da divisão recursiva de um conjunto de dados em regiões, cada nó folha representa uma região específica deste conjunto (Alpaydin, 2020). A Figura 2.8 apresenta uma árvore de decisão simples e a divisão das regiões, no conjunto de dados relacionado.

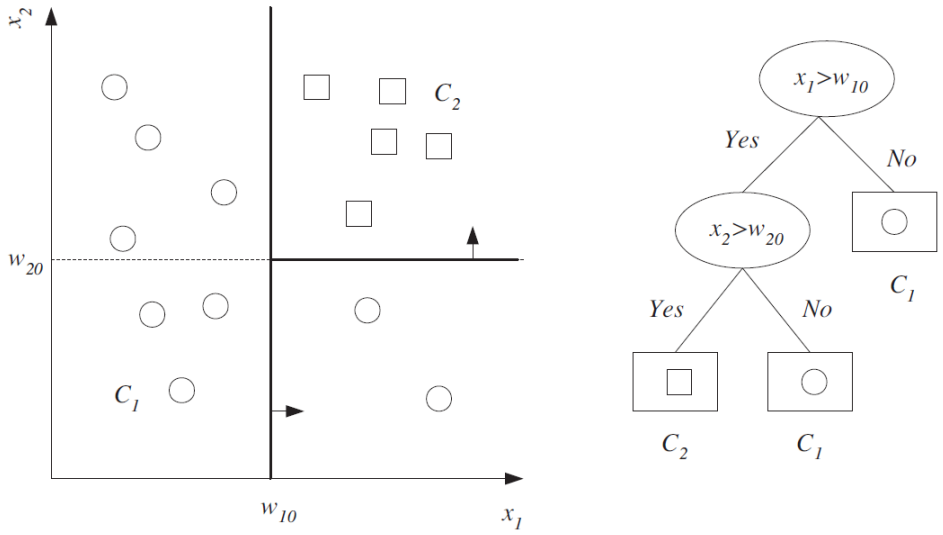

Figura 2.8: Árvore de decisão construída a partir de um conjunto de dados.
Fonte: (Alpaydin, 2020).

A indução de uma árvore de decisão a partir de um conjunto de dados ocorre através da avaliação de atributos específicos da base, estes são selecionados, e testes são criados para definir o *split* em cada nó de decisão. Um nó é considerado nó objetivo ou folha, quando uma porcentagem alta dos exemplos naquele nó pertence a mesma classe. Para árvores de decisão utilizadas para classificação, a qualidade dos splits é avaliada geralmente com a aplicação de uma medida de ganho de informação (Alpaydin, 2020).

A aplicação dessa métrica se dá pela sua capacidade de mensurar o quão adequado é um atributo para garantir que a árvore criada dê uma classificação correta, minimizando a profundidade da mesma. Para chegar ao ganho de informação, é necessária a definição da entropia, que em teoria da informação, é a medida de quantidade de informação. A entropia define o nível de desordem ou incerteza de uma variável, reduzida com a aquisição de informação (Russell & Norvig, 2010). Dado um conjunto de exemplos S contendo exemplos positivos e negativos, onde $p+$ é a proporção de exemplos positivos e $p-$ de exemplos negativos, a entropia para a classificação binária de S é Equação 2.1 e para classificação onde existem c valores objetivos (multiclasse) é Equação 2.2 (Mitchell, 1997):

$$Entropia(S) = -p_+ log_2 p_+ - p_- log_2 p_- \tag{2.1}$$

$$Entropia(S) = \sum_{i=1}^{c} -p_i log_2 p_i \tag{2.2}$$

Com a definição de entropia, é possível representar a medida de ganho de informação. Sendo o $Ganho(S, A)$, onde S é o conjunto de exemplos e A é um atributo, temos Equação 2.3, onde os Valores(A) são todos os possíveis valores para o atributo e Sv é o subconjunto de S para qual o atributo A tem valor v (Mitchell, 1997):

$$Ganho(S, A) = Entropia(S) - \sum_{v \epsilon Valores(A)} \frac{|S_v|}{|S|} Entropia(S_v) \tag{2.3}$$

Outra medida de impureza descrita por Breiman (1996) é chamada de Gini, apresentada na Equação 2.4. Esta descreve a probabilidade de classificação incorreta de uma nova instância classificando-a de forma aleatória. A impureza de Gini é descrita em termos da probabilidade de cada classe pj, sendo representada por $Gini(p)$. Esta medida geralmente prefere *splits* que favoreçam a classe majoritária pertencer a um nó puro:

$$Gini(p) = \sum_j p_j (1 - p_j) \tag{2.4}$$

Redes Neurais Artificiais

Redes Neurais Artificiais (RNA) são construídas a partir de várias unidades de processamento simples, altamente interconectadas. Estas redes são inspiradas em sistemas biológicos de aprendizado, como o cérebro humano. O funcionamento básico destas redes se da pela propagação de informações pelas camadas da rede, as unidades de processamento (neurônios) recebem as entradas e produzem uma saída de acordo, esta saída pode então ser utilizada por um ou mais neurônios nas próximas camadas como suas entradas (Mitchell, 1997). A Figura 2.9 apresenta um *perceptron*, um tipo de RNA (Alpaydin, 2020).

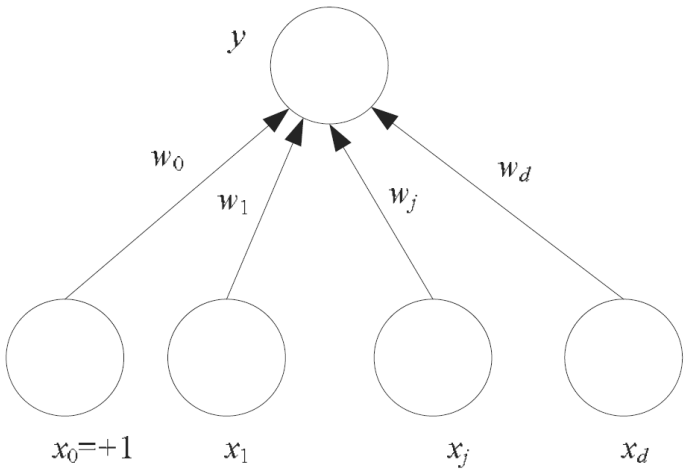

Figura 2.9: Uma RNA do tipo *perceptron*, x são as unidades de entrada e y é a unidade de saída.
Fonte: (Alpaydin, 2020).

Perceptrons são tipos comuns de RNA, que utilizam como entrada valores reais, usando uma combinação linear destas entradas para produzir uma saída apropriada. Caso o resultado da combinação seja maior ou menor que um limiar pré-estabelecido, nesta combinação, cada entrada é associada com um certo peso que determina sua contribuição para a saída do *perceptron*. Um *perceptron* simples consegue representar algumas funções simples como AND, OR, NAND e NOR, funções linearmente separáveis em uma superfície de decisão, para representar funções não linearmente separáveis é necessária a utilização de um *perceptron* com múltiplas camadas. A Figura 2.10 apresenta a superfície de decisão de duas funções diferentes (Mitchell, 1997).

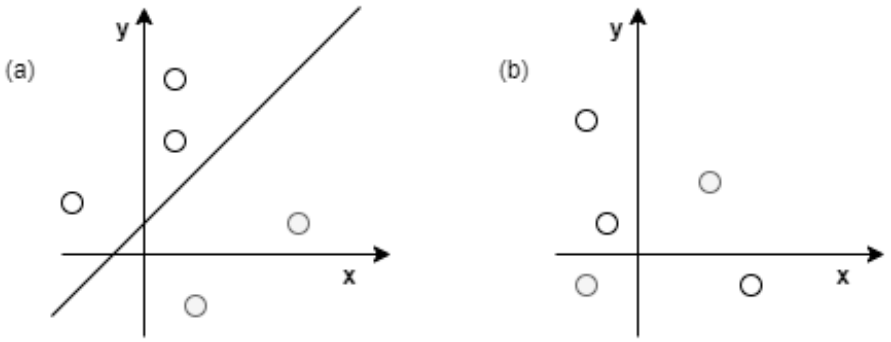

Figura 2.10: (a) representa uma função linearmente separável, (b) apresenta uma função não linearmente separável.
Fonte: Adaptado de (Mitchell, 1997).

Além da função de ativação de cada neurônio, também é necessário definir o tipo de conexão entre os neurônios, as duas principais maneiras descrevem redes do tipo *feed-forward* e recorrentes. Redes *feed-forward* fazem a conexão entre os neurônios em uma direção, indo

da camada de entrada até a camada de saída, sem loops. Já nas redes recorrentes, os neurônios tem a capacidade de utilizar suas saídas como suas entradas, isto permite que a rede tenha características únicas não presente em redes *feed-forward* como memória de curto prazo. Um *Perceptron*, por exemplo, é um tipo de rede neural *feed-forward* de uma única camada (Russell & Norvig, 2010).

Um *Multilayer Perceptron* (MLP) é uma classe de perceptrons capaz de representar funções complexas através da utilização de múltiplas camadas de unidades de processamento interconectadas. O algoritmo de retro propagação é utilizado para treinamento de redes com várias camadas, e é responsável por ajustar os pesos das unidades de cada camada da rede. Quando aplicado em uma rede *feed-forward*, os pesos da rede são inicializados com números pequenos gerados aleatoriamente e cada exemplo de treinamento é propagado pela rede, os erros são então calculados e retro propagados. Por fim, os pesos das unidades da rede são então ajustados de acordo com o erro (Mitchell, 1997).

Naive Bayes

O teorema de Bayes descreve um método capaz de calcular a melhor hipótese dado um conjunto de dados. Esta pode ser descrita como a hipótese mais provável, com base nos dados e nas probabilidades das hipóteses neste conjunto. Neste teorema, a notação $P(h)$ representa a probabilidade inicial da hipótese h, $P(D)$ representa a probabilidade que o conjunto de dados de treinamento D será observado. Com isso, $P(D|h)$ descreve a probabilidade de observar D quando a hipótese h é verdadeira. $P(h|D)$ descreve a probabilidade que h é verdadeira quando D é observado, está é chamada de probabilidade a posteriori e é o objetivo em problemas de aprendizado de máquina quando aplicando este tipo de técnica probabilística (Mitchell, 1997). Com isso o teorema de Bayes é dado por:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{2.5}$$

O classificador *Naive Bayes* aplica os conceitos desse teorema para a resolução de problemas de classificação em aprendizado de máquina. Para esta tarefa, cada instancia do conjunto de dados é descrita em termos de uma conjunção de seus atributos. O classificador deve apresentar a hipótese (classe) mais provável de acordo com os valores dos atributos apresentados, de acordo com o teorema de Bayes:

$$v_{MAP} = argmax_{v_j} P(a_1, a_2 ... a_n | v_j) P(v_j) \tag{2.6}$$

Nesta (v_{MAP}) é o valor objetivo da instância (classe), a_n é o valor de um atributo, v_j é uma classe e $P(a_1 ... a_n|v_j)$ é a probabilidade da combinação dos valores dos atributos em uma

classe. O classificador Naive Bayes assume que os valores dos atributos são condicionalmente independentes dado uma determinada classe. Isto não pode ser garantido na maioria dos conjuntos de dados comuns, no entanto, o classificador *Naive Bayes* apresenta boa performance mesmo quando essa característica não é verdadeira (Mitchell, 1997). Com isso este classificador pode ser representado por:

$$v_{NB} = argmax_{v_j} P(v_j) \prod_i P(a_i|v_j) \tag{2.7}$$

Esta equação descreve o classificador como o produto de cada atributo com relação a uma classe com a mesma classe. A hipótese criada por este classificador então corresponde as estimativas observadas, e o processo de classificação de uma nova instância se da pela aplicação da mesma equação (Mitchell, 1997).

2.3.3 Classificadores em Ensemble

Ensembles são conjuntos de classificadores que tem suas decisões combinadas por algum método de agregação, geralmente aplicados com a finalidade de melhorar a acurácia de classificação. Para que um classificador *ensemble* tenha uma acurácia melhor que um classificador convencional, deve haver a possibilidade de diversidade no algoritmo utilizado, no conjunto de dados ou através da utilização de diferentes algoritmos (Dietterich, 2000).

Os métodos de agregação são aplicados para buscar melhorar o desempenho dos algoritmos do *ensemble*, mas esses também podem ser utilizados para possibilitar a aplicação de algoritmos em diversos problemas, através da aplicação de algoritmos para classificação binária na resolução de problemas multiclasse, por exemplo. Isto é feito através da aplicação de estratégias de decomposição binária, como *One-Versus-One* (OVO) e *One-Versus-All* (OVA) e combinando os resultados dos classificadores através de um método de agregação (Galar, Fernández, Barrenechea, Bustince & Herrera, 2011).

Existem diferentes estratégias para agregar os resultados obtidos dos algoritmos de classificação binária em problemas multiclasse. Galar et al. (2011) aplica estratégias como *voting*, *weighted voting*, *binary tree of classifiers* e *decision directed acyclic graph* com a estratégia de decomposição binária OVO, e *maximum confidence strategy* ou *dynamically ordered one-vs-all* com a estratégia OVA. A Figura 2.11 apresenta o modelo genérico de um ensemble.

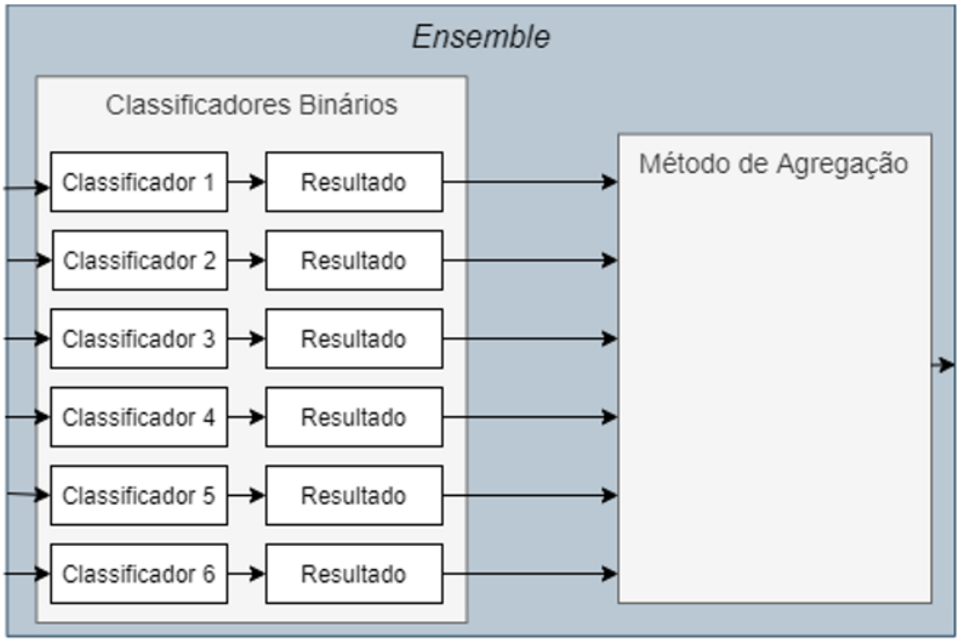

Figura 2.11: Arquitetura de um *Ensemble*.

Entre os tipos de *ensemble*, Polikar (2006) descreve *Bagging*, *Boosting*, *Stacking* e suas variações. *Boosting* baseia-se na observação que *weak learners* (descritos como modelos que atingem acurácia de classificação um pouco melhor que 50%) podem ser transformados em *strong learners* (descritos como modelos que apresentem boa acurácia de acordo com o problema). Esta técnica realiza amostragem dos dados e os resultados dos classificadores são combinados por voto majoritário. Esta amostragem visa criar o conjunto de treinamento com o maior nível de informação possível em cada classificador. Um algoritmo de *boosting* cria 3 *weak classifiers*. C1 treina com um subconjunto aleatório dos dados, C2 treina com o subconjunto que apresenta o maior ganho de informação dos dados a partir de C1. C2 é treinado em um conjunto de dados contendo metade dos dados classificados corretamente por C1 e a outra metade de dados classificados incorretamente. O classificador C3 é treinado utilizando as instâncias com classificações diferentes em C1 e C2, estes 3 classificadores são então combinados utilizando voto majoritário.

AdaBoost é uma versão mais geral de Boosting. As versões mais utilizadas são *AdaBoost.M*1 e *AdaBoost.R*, que possibilitam classificação multiclasse e regressão. Em *AdaBoost* são geradas hipóteses combinadas através de voto majoritário ponderado. As hipóteses são geradas treinando *weak classifiers* com instâncias extraídas através de um processo incremental. O processo de atualizar a distribuição dos dados de treinamento incrementalmente garante que as instâncias classificadas erroneamente pelo classificador anterior tem uma probabilidade maior de inclusão no próximo classificador.

Em *Stacking*, um ensemble de classificadores é criado e suas saídas são utilizadas como entradas de outro nível de meta-classificadores, que tem o propósito de mapear as saídas do *ensemble* de classificadores do nível anterior e as classes corretas.

Bagging (*bootstrap aggregating*) tenta garantir a diversidade do *ensemble* utilizando versões dos dados de treinamento, obtidos através de amostragem. Os resultados dos classificadores são agregados utilizando voto majoritário. O autor descreve *Bagging* como uma boa alternativa quando há uma limitação na quantidade dos dados disponíveis. Os subconjuntos de dados obtidos através de amostragem são chamados *Bootstraped* replicas e cada subconjunto é utilizado no treinamento de um classificador (Polikar, 2006).

Uma variação de *Bagging* é o algoritmo de *Random Forest*, construído utilizando arvores de decisão, através da variação de certos parâmetros de treinamento. Estes parâmetros podem ser conjuntos obtidos por amostragem da base de treinamento, mas também podem ser subconjuntos diferentes de atributos (Polikar, 2006). A Figura 2.12 apresenta o funcionamento do algoritmo de *Random Forest*.

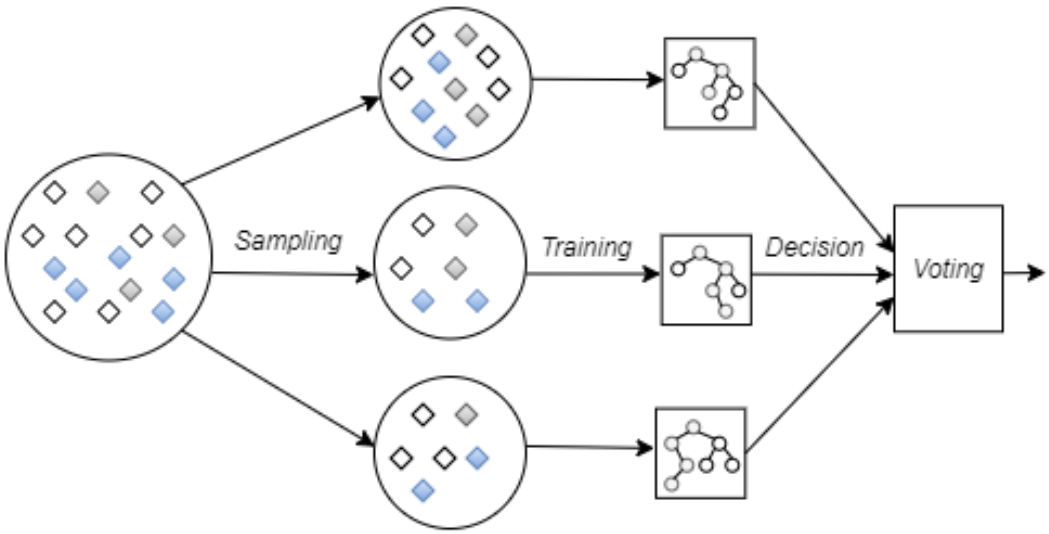

Figura 2.12: Algoritmo *Random Forest*.

Random Forest é descrito em (Breiman, 2001), o autor descreve e prova que o algoritmo é resistente a *overfitting* mesmo com o aumento do número de árvores na floresta, além de ser resistente a ruído nos dados. O autor também demonstra que os modelos de *Random Forest* com menor erro de generalização, são aqueles que possuem uma menor correlação entre os seus classificadores.

Capítulo 3

Segurança em Redes de Sensores Sem Fio

Neste capítulo são abordados conceitos relevantes a segurança computacional em Redes de Sensores Sem Fio (RSSF). Considerando especificamente as limitações dessas redes e dispositivos, além de questões de segurança relacionadas.

3.1 Considerações de Segurança em RSSF

Devido as características e aplicações de RSSF, estas sofrem com diversas limitações em termos de recursos de hardware dos nós sensores, aos meios de comunicação e dificuldades provenientes do ambiente em que essas redes são dispostas. Com isso RSSF são especialmente suscetíveis a ameaças de segurança como ataques ao *software* dos nós sensores, aos meios de comunicação entre eles e a estação base, e até ao *hardware* dos nós sensores. Neste capítulo são discutidos os incidentes relacionados ao software de sensores e as comunicações realizadas entre os sensores e as estações base. Especificamente são abordados os ataques de *Blackhole*, *Greyhole*, *Flooding* e *Scheduling*.

Vhatkar & Atique (2013) descrevem as dificuldades enfrentadas por RSSF são descritas em termos de problemas provenientes dos objetivos de design de RSSF, incluindo dificuldades de roteamento nestas redes. Os autores também apontam que RSSF são comumente construídas para propósitos específicos, portanto os objetivos de design das mesmas podem variar, mas as considerações mais comuns são citadas:

- **Nós Sensores:** As principais características dos nós sensores se dão pelo seu tamanho reduzido, baixo custo e baixo consumo de energia. Essas características são importantes para possibilitar a utilização de RSSF em diversos ambientes e facilitar a implementação da rede, isto causa uma redução no poder de processamento destes nós;
- **Características da Rede:** Considerando o grande número de nós na rede, esta deve ser escalável e se adaptar a distribuição de sensores cobrindo grandes distâncias. Além disso, os nós na rede devem ser capazes de realizar auto configuração após a sua instalação, para fácil criação da rede, com isto também é considerada a capacidade da rede operar na

presença de falhas pela desativação de nós, por exemplo;

- **Segurança e Comunicação:** Com as limitações de banda das conexões utilizadas pelos sensores, protocolos especiais de comunicação são criados, para manter reduzido o consumo de recursos dos nós, além de considerar as limitações do meio de comunicação sem fio. As informações que circulam na RSSF também devem ser adequadamente transferidas entre os membros, e considerações devem ser feitas para evitar acessos não autorizados (Qualidade de Serviço e Segurança);

Considerando estes objetivos propostos, Vhatkar & Atique (2013) descreve os problemas gerados devido à capacidade limitada de energia, ao posicionamento dos nós sensores, aos recursos de *hardware* limitados, ao processo de distribuição dos nós durante a formação da rede, às características da rede e dificuldades do ambiente, à agregação de dados, aos requerimentos do processo de captação de dados pelos sensores e à escalabilidade da rede. Todos estes contribuem para a maior suscetibilidade a ameaças de segurança em RSSF, e parte importante para a detecção destas ameaças e a realização de contramedidas é a consideração destas características e suas ramificações na criação da rede.

Hac (2003) descreve alguns dos possíveis efeitos que podem ocorrer quando a segurança do meio de comunicação dos sensores é comprometida, como a inserção de código malicioso e interceptação de mensagens dos nós, além da injeção de mensagens falsas na rede.

Uma das considerações mais importantes na configuração de uma RSSF é a escolha do seu protocolo de roteamento. Como não há garantia de infraestrutura nestas redes os protocolos utilizados devem ser capazes de lidar com falhas nos nós sensores e as restrições impostas pela limitação na disponibilidade de energia pelos membros da rede. Um dos protocolos de roteamento mais populares é o protocolo *Low-Energy Adaptive Clustering Hierarchy* (LEACH) (Heinzelman, Chandrakasan & Balakrishnan, 2000) e (Heinzelman, Chandrakasan & Balakrishnan, 2002), que visa principalmente facilidade de formação, maior tempo de vida do sistema, latência mínima e qualidade da rede. Esta qualidade é definida em termos dos dados agregados da rede, garantindo o fluxo correto e consistente das informações do domínio da RSSF.

A arquitetura do protocolo LEACH é descrita por Heinzelman et al. (2000) e Heinzelman et al. (2002) como um protocolo com arquitetura do tipo especifica de aplicação. Considerando que os domínios onde são aplicadas RSSF geralmente lidam com monitoramento remoto de ambientes, há muitas vezes dados redundantes que não são uteis para o usuário final do sistema, sendo preferível que o usuário final tenha uma visão de alto nível das informações da rede. Com isto o protocolo LEACH propõe a utilização de técnicas de *clustering*, estes então agregam parte dos dados da rede que são processados e enviados ao usuário final. Este processo ocorre através da organização dos nós em *clusters* locais, com a definição de um nó como um *Cluster Head* para cada um destes conjuntos, os nós comuns destes transmitem seus dados para o *Cluster Head* que faz a agregação dos dados e transmite-os para a estação base. Para lidar com os problemas de consumo de energia provenientes da utilização de *cluster heads,* estes não são

fixos para todo o tempo de vida da rede, escolhidos de acordo com a disponibilidade de energia de cada nó nos *clusters* e mudando em cada etapa da operação do LEACH (estas etapas são chamadas de *rounds*). A Figura 3.1 apresenta algumas configurações de *clusters* e *cluster heads* na mesma RSSF.

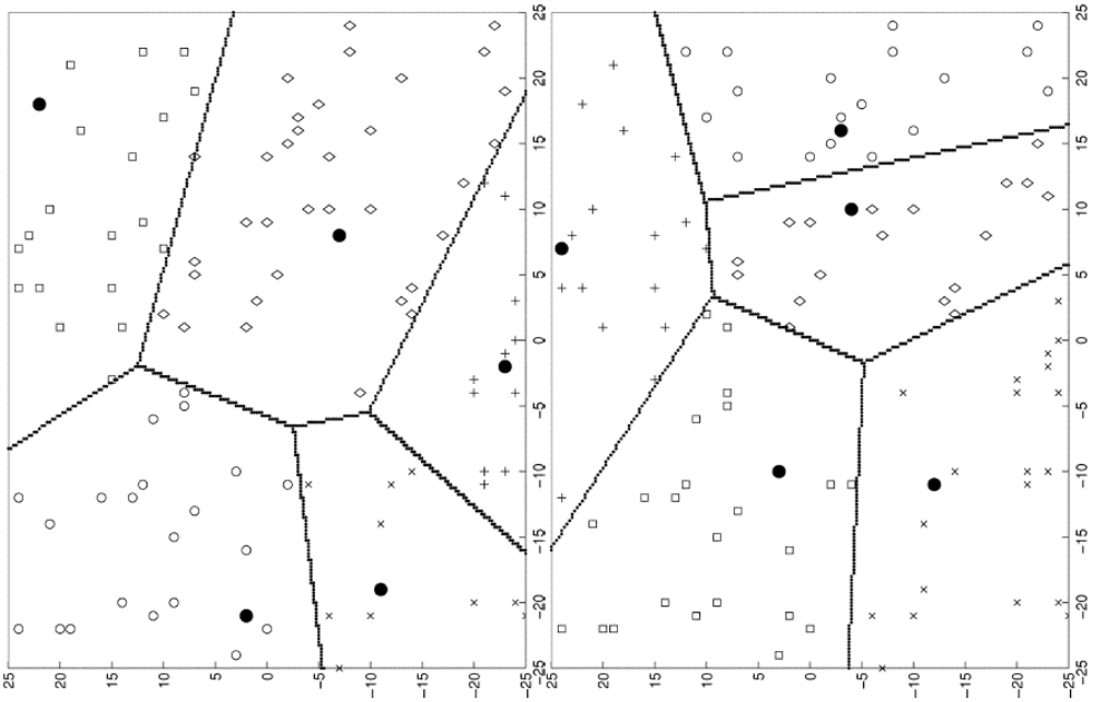

Figura 3.1: Diferentes configurações de *Cluster Heads* e *Clusters* em uma RSSF.
Fonte: (Heinzelman et al., 2002).

Heinzelman et al. (2000) descreve o protocolo LEACH como capaz de reduzir os custos energéticos de comunicação em até oito vezes em comparação com técnicas de transmissão direta e *minimum-transmission-energy routing*, com a desativação do primeiro nó no protocolo LEACH ocorrendo até 8 vezes mais tarde que nestes outros métodos, e até 3 vezes mais tarde em comparação com protocolos de *clustering* estático.

3.2 Ameaças à RSSF

Em termos gerais, ameaças a RSSF funcionam de maneira similar a ameaças de segurança em redes de computadores convencionais, mas devido às restrições e decisões de design, nestas redes estes ataques se tornam mais eficazes. Existem diversos esquemas de segurança propostos para redes sem fio comuns com difícil implementação no contexto de RSSF, principalmente pela natureza destas redes, sendo compostas por um grande número de membros que se auto-organizam e criam topologias dinâmicas. A existência de alguma forma de centralização, como, por exemplo, as estações bases que se comunicam com os membros da rede, podem facilitar a implementação de esquemas de segurança (Pathan, Lee & Hong, 2006). Ainda existem con-

siderações a segurança individual de membros da rede, nestes casos, detecção de ameaças a realização de contramedidas na estação base ou em pontos intermediários na rede pode não ser suficiente para lidar com diversos tipos de ameaça de maneira eficaz.

Pathan et al. (2006) descreve ataques contra RSSF como sendo direcionados aos mecanismos de segurança da rede ou contra mecanismos básicos da mesma. Alguns tipos de ataques mais comuns citados são de negação de Serviço, ataques ao fluxo de informação, ataque *Sybil*, *Blackhole* ou *Sinkhole*, *Hello Flood* e *Wormhole*. A seguir são descritos alguns ataques importantes para a execução deste trabalho.

3.2.1 *Blackhole* e *Grayhole Attacks*

Blackhole e *Grayhole* são ataques de negação de serviço que operam na camada de rede da RSSF e comprometem o processo de roteamento, já que todos os nós na rede agem como roteadores. Os ataques do tipo *Blackhole* comprometem um ou mais nós da rede, estes nós alteram as rotas de seus vizinhos se passando por nós com prioridade alta, fazendo com que grande parte ou todos os pacotes das proximidades sejam enviados apenas para o nó comprometido (Roosta, Shieh & Sastry, 2006).

O funcionamento deste tipo de ataque em redes LEACH pode ocorrer com a comunicação de níveis de energia altos do nó comprometido, fazendo com que este seja selecionado como *Cluster Head*. Com o recebimento dos pacotes do *cluster* pelo *Cluster Head* comprometido, este então, derruba todos os pacotes recebidos, interferindo no funcionamento correto da rede (Tripathi, Gaur & Laxmi, 2013). Ataques de *Grayhole* funcionam de maneira similar aos ataques de *Blackhole*, a diferença está na derrubada seletiva de pacotes, este processo pode fazer com que o nó derrube apenas pacotes vindo de outros nós específicos, ou derrubando todos os pacotes recebidos por um certo período, e depois retornando a operação normal (Tripathi et al., 2013).

Os autores Almomani, Al-Kasasbeh & Al-Akhras (2016) realizaram simulações para criar uma base de dados de ataques contra RSSF, e nestas simulações do protocolo LEACH, os ataques de *Blackhole* e *Grayhole* são implementados através da aplicação da técnica de *advertisement* de *Cluster Head*.

3.2.2 *Flooding Attacks*

Magotra & Kumar (2014) descrevem ataques de *Flooding* são descritos em RSSF que utilizam o protocolo LEACH. O tipo de ataque descrito é conhecido como ataque *HELLO Flood*, este tipo de ataque pode ocorrer em redes que utilizam pacotes do tipo *HELLO* para estabele-

cer suas rotas. Estas redes geralmente assumem que quando recebem um pacote deste tipo de algum outro nó, o outro nó está em uma distância transmissível, e então a rota é estabelecida. Com isso, é possível que transmissores maliciosos de alto alcance enviem esse pacote para uma certa área da rede, estabelecendo rotas inviáveis.

Ataques do tipo *HELLO Flood* são apenas um tipo de ataques de *Flooding*. Almomani et al. (2016) simularam ataques de *Flooding* através do envio de um grande número de pacotes de seleção de *Cluster Head* pelo nó comprometido, consumindo mais energia dos sensores e tempo na tentativa de determinar a qual *Cluster Head* adequado um nó qualquer deve se associar.

3.2.3 *Scheduling Attacks*

Ataques de *Scheduling* são descritos por Almomani & Al-Kasasbeh (2015) como ataques de negação de serviço que ocorrem durante a fase de *setup* no protocolo LEACH. Quando um *Cluster Head* define o seu TDMA (*Time-Division Multiple Acess*) *schedule* para os intervalos de transmissão de dados, um nó agindo como o *Cluster Head* define *schedules* conflitantes para os nós daquele *cluster*, causando colisões e perda de pacotes.

Almomani et al. (2016) simularam ataques de *scheduling* através da atribuição dos mesmos intervalos de transmissão para todos os nós do *cluster* ou a atribuição de intervalos iguais para subconjuntos de nós dentro do *cluster*.

3.3 Detecção de Intrusão em RSSF

3.3.1 Esquemas de Segurança em RSSF

Os autores Hac (2003) descrevem certos protocolos de segurança para RSSF, estes são chamados SPINS (*Security Protocols for Sensor Networks*) e lidam com as dificuldades de segurança impostas pelo *design* de RSSF. São citados protocolos como miTESLA (*Timed, Efficient, Streaming, Loss-tolerant Authentication protocol*) e SNEP (*Secure Network Encryption Protocol*) que lidam com os processos de autenticação de *streaming* e confidencialidade nos dados respectivamente.

O processo de broadcast com autenticação descrito por Hac (2003), consiste na encriptação dos dados com uma chave, estes dados então só podem ser decriptados pelos receptores apropriados. Este tipo de *broadcast* funciona de forma assimétrica, caso contrário, se um dispositivo na rede for comprometido, este poderia ser utilizado para forjar mensagens na rede.

Este tipo de mecanismo assimétrico geralmente tem altos custos computacionais, que não estão disponíveis em RSSF, neste contexto são aplicados protocolos como o miTESLA, que garante a assimetria através da utilização de chaves simétricas com atrasos, reduzindo assim os custos computacionais relacionados. Outro mecanismo utilizado para garantir confidencialidade e *data freshness* é o SNEP, através da utilização de um contador em cada nó, incrementado com o envio de mensagens. Este contador é utilizado no processo de encriptação das mensagens garantindo a segurança e expondo a ordem de envio das mensagens para os membros da rede.

Outra consideração na segurança de RSSF é a segurança dos meios de comunicação. Os tipos de dados trocados pelos sensores nesses meios podem ser classificados de acordo com a sensibilidade da informação. Uma taxonomia é descrita por Hac (2003), que define três níveis de segurança para os dados em uma RSSF, estes são:

- **Nível 1:** Reservado para código móvel, considerado o tipo de informação mais importante que circula pela rede;
- **Nível 2:** Reservado para informações relacionadas com características de localização;
- **Nível 3:** Reservado para informação específica do domínio da aplicação;

Quanto menor o nível do conjunto de dados em questão, maior o nível de medidas de segurança requerido pelo mesmo. Estes níveis também descrevem outras características dos dados além de suas necessidades de segurança, por exemplo, código móvel descrito pelo Nível 1 é menos frequente do que mensagens contendo dados do Nível 3, o que possibilita a tomada de decisões de projeto específicas no desenvolvimento de mecanismos de segurança para a RSSF.

Os autores Pathan et al. (2006), apresentaram um *survey* de diferentes esquemas de segurança para RSSF, este inclui diversos esquemas que lidam com ataques de DDOS (especificamente *jamming*), *spoofing* de informação, *flooding*, ataques de *blackhole* e *wormhole*. Dentre as estratégias citadas estão a aplicação de técnicas de criptografia simétrica, utilização de conceitos de ataques de *wormhole* para responder a ataques de *jamming*, roteamento geográfico e técnicas para garantir maior resiliência da rede com considerações na distribuição de energia e roteamento. Além disso, também é citado um esquema holístico, que lida com a performance da RSSF em segurança, longevidade e resiliência, este esquema lida com todas as camadas de rede para tentar garantir essas características.

3.3.2 Processo de Detecção de Intrusão em RSSF

O processo de detecção de intrusão em RSSF pode ocorrer através da utilização de Sistemas de Detecção de Intrusão convencionais em estações bases, este tipo de detecção de intrusão aplicado em conjunto com outros protocolos e práticas descritas nas seções anteriores contribuem para um nível maior de segurança da RSSF. No entanto, a visão da rede da perspectiva

da estação base pode criar atrasos na detecção de ameaças e na realização das contramedidas adequadas.

Os autores El Mourabit, Bouirden, Toumanari, Moussaid et al. (2015) realizaram um estudo comparativo de diferentes técnicas de detecção de intrusão em RSSF, com foco em técnicas que utilizam aprendizado de máquina para realizar detecção por anomalia. Estes algoritmos são *K-mens*, *Naive bayes*, *Support-Vector Machine*(SVM) e *Random Forest*, aplicados com a base de dados KDDCup'99. Os algoritmos também são avaliados em custos computacionais e comparados com 2 tipos de sensores utilizados em RSSF, Mica2 e Telosb. Os autores observam que a utilização de detecção por anomalia em RSSF apresenta bons resultados, além do consumo de recursos apresentados pelos algoritmos estar contido no limite aceitável para o poder de processamento de alguns tipos de nós sensores. Os autores também apontam o potencial da aplicação de técnicas que utilizam processamento distribuído ou hierárquico para a detecção de intrusão para lidar com as dificuldades provenientes dos recursos limitados da rede.

Capítulo 4

Trabalhos Relacionados

4.1 Estado da Arte

Este capítulo apresenta trabalhos científicos observados que demonstram o estado da arte de detecção de intrusão em Redes de Sensores Sem Fio (RSSF). As seções foram organizadas para apresentar de aspectos abrangentes até mais específicos, relevantes para o trabalho proposto.

4.2 Detecção de Intrusão em RSSF

Garofalo, Di Sarno & Formicola (2013) propuseram uma arquitetura de Sistema de Detecção de Intrusão (SDI) para RSSF que visa uma boa taxa de detecção através da aplicação de árvores de decisão e economia no consumo de energia com a aplicação de técnicas de detecção leve. A arquitetura proposta consiste em um agente central e diversos agentes locais, estes são distribuídos nos nós da RSSF enquanto o agente central é implementado na estação base. Os agentes locais realizam um processo de detecção "fraca", caso um evento de intrusão seja identificado o nó é adicionado a uma *blacklist* e as rotas contendo aquele nó são modificadas. Além disso, um alerta é enviado para o agente central que define se o evento é realmente um evento de intrusão. O processo de detecção de intrusão no agente central ocorre pela aplicação de uma árvore de decisão treinada com um *dataset* sintético gerado por simulação, este contém eventos de tráfego normal e ataques de *sinkhole*. Diferentes tipos de árvore de decisão são comparadas no processo experimental, dentre elas, CART, CHAID e C5,0. Todas as técnicas apresentaram performance de classificação satisfatória.

Os autores Pan, Fan, Chu, Zhao & Liu (2021) apresentaram um modelo leve de detecção de intrusão para RSSF, que aplica *K-Nearest Neighbors* (KNN) e *Sine Cosine Algorithm* (SCA) para aumentar a acurácia e reduzir a taxa de alarmes falsos do processo de detecção. Os autores implementam o SDI proposto em uma arquitetura de rede que combina computação na nuvem e na névoa. O processo de detecção é implementado na nuvem, recebendo suporte da camada

de névoa para processamento e armazenamento de dados. Um SCA compacto é aplicado no processo de otimização do KNN, neste SCA é aplicada uma estratégia de *Polymorphic Mutation*, para reduzir a perda de precisão proveniente da aplicação do método compacto de SCA. Nos experimentos com as bases de dados NSL-KDD e UNSW-NB15 a abordagem PM-CSCA + KNN apresentou boa performance de classificação.

Saeed, Ahmadinia, Javed & Larijani (2016) propuseram a utilização de uma *Random Neural Network* (RNN) aplicada a detecção de intrusão em RSSF, especificamente na detecção de ataques de degradação de performance. O autor argumenta que pelo fato da RNN utilizar uma quantidade menor de memória, consumir menos energia e detectar comportamentos anômalos no sistema, esta pode ser uma boa alternativa na utilização de técnicas de aprendizado de máquina em RSSF. A RSSF utilizada é composta por microcontroladores reais para a manutenção de temperatura em um ambiente compartilhado, um computador central é responsável por gerenciar os dados recebidos dos sensores. O mecanismo de detecção de intrusão é implementado no computador, sendo medido o *overhead* de tempo de execução e consumo de energia. O autor apresenta uma acurácia na detecção de 97,23%, com pouco *overhead* de performance e 10,45% de aumento no consumo de energia.

A estratégia de detecção de intrusão apresentada pelos autores Batiha, Prauzek & Krömer (2019) se baseia em *ensemble* de Redes Neurais Artificiais (RNA) rasas. Cada RNA modela um tipo diferente de intrusão, e por serem rasas podem ser utilizadas em dispositivos com restrições de processamento e energia. Com a criação de um modelo de RNA para cada tipo de ataque, é possível a implementação dos modelos necessários em nós que apresentem uma predisposição a serem atingidos por estes tipos específicos de ataques. O *dataset* utilizado para testes e treinamento é o WSN-DS, que contém 4 classes de ataques, *Blackhole, Greyhole, Flooding* e *Scheduling*, além de tráfego normal. A capacidade de diferentes RNA modelarem diferentes tipos de ameaças específicas no *ensemble* é beneficial em RSSF, considerando os recursos limitados destas redes. Os *ensembles* avaliados apresentam boa performance de classificação.

Os autores Borkar, Patil, Dalgade & Hutke (2019) propuseram a utilização de *Adaptive Chicken Swarm Optimization* para a seleção de *Cluster Heads* de cada *round* na organização topológica da RSSF. Além da implementação de um SDI para reportar nós maliciosos. Um ensemble do tipo *Rotated Random Forest* é aplicado para o processo de seleção de atributos da base KDD'99, utilizada para o processo de treinamento e testes do *adaptive Support-Vector Machine* (SVM). O processo de identificação de uma ameaça proveniente de um nó é dividido em duas etapas. Inicialmente o nó é identificado como malicioso ou normal através de um método de troca de mensagens, e então os dados do nó são processados pelo algoritmo para verificar o tipo de ataque. As ameaças consideradas da base KDD'99 são DOS, *probe*, U2R e R2L. Os testes com a metodologia proposta apresentam boa performance de classificação.

O sistema apresentado por Singh, Virmani & Gao (2020) aplica regras *fuzzy* para detecção de intrusão em RSSF, operando em três (3) fases: extração de atributos, computação de um valor de *membership* e aplicação de regras *fuzzy*. O sistema tem foco na classificação de nós

em três (3) categorias “vermelha”, “laranja” e “verde”, que simbolizam o nível de ameaça que o nó oferece a rede. A base WSN-DS foi selecionada para treinamento e testes. O processo de seleção de atributos é realizado utilizando *Correlation-Based Feature Selection* e os atributos selecionados são is_CH, $Data\_sent\_to\_base$, $data\_s$, $data\_r$ e $consumedenergy$, além de um novo parâmetro criado, o $packetdeliveryratio$. As métricas de classificação observadas apresentam boa performance, e o sistema proposto é eficaz em prevenir a entrada de nós maliciosos na rede.

Os autores Jiang, Zhao & Xu (2020) propuseram um método de detecção de intrusão para RSSF, que consiste na aplicação do algoritmo *sequence backward selection* para redução de dimensionalidade dos dados, seguida da aplicação do algoritmo *LightGBM* para a classificação de ameaças. As ameaças detectadas são identificadas pelo dataset WSN-DS e são, *Blackhole, Grayhole, Scheduling* e *Flooding*. O algoritmo de *sequence backward selection*, aplicado para seleção de atributos, realiza uma *greedy search* removendo atributos da base até encontrar um subconjunto com um certo número de atributos que tenha uma avaliação aceitável. O algoritmo *LightGBM* é um algoritmo do tipo *gradient boosting* baseado em Árvores de Decisão. Os experimentos foram realizados por meio de comparações entre subconjuntos de atributos selecionados e diversos algoritmos de classificação. A performance do algoritmo proposto é boa em termos das métricas de classificação, além do algoritmo apresentar baixo consumo de memória.

Alqahtani, Gumaei, Mathkour & Maher Ben Ismail (2019) desenvolveram um algoritmo para detecção de intrusão em RSSF baseado na aplicação de algoritmos genéticos com um classificador *extreme gradient boosting*, este apresentando boa performance em *datasets* altamente desbalanceados. O algoritmo proposto é dividido em quatro (4) etapas: geração da população de parâmetros, seleção da população de parâmetros, treinamento da função de decisão e avaliação da função de *fitness*. A base de dados WSN-DS foi utilizada para o processo experimental, e comparações com outros algoritmos de *boosting* foram realizadas, como *AdaBoost*, *gradient boosting* e *extreme gradient boosting*. O algoritmo proposto tem melhor performance de classificação que os outros algoritmos apresentados, além de apresentar um tempo menor de classificação.

Os autores Dong, Yan & Zhang (2020) propuseram um modelo de detecção de intrusão para RSSF baseado em *information gain ratio* e *ensembles* do tipo *Bagging*. O *information gain ratio* é utilizado na seleção de atributos do tráfego dos nós sensores e o algoritmo de *Bagging* é utilizado para construção do *ensemble* que realiza o processo de classificação de ataques. Este *ensemble* é composto de árvores de decisão C4,5. Através de um processo iterativo de treinamento e poda, estas árvores são melhoradas e um método de agregação de voto majoritário é utilizado para a classificação final do *ensemble*. O método experimental utiliza os *datasets* WSN-DS e NSL-KDD para treinamento e testes, e comparações são realizadas entre o algoritmo proposto e os algoritmos PCA-SVM, *Naive Bayes*, *Bayesian Network*, IG-C4,5, *Boosting*-C5,0 e *Artificial Neural Network*. Os tipos de ataques classificados das bases

são *Blackhole, Grayhole, Flooding, Scheduling, Probe*, DoS, U2R e R2L. O método proposto apresenta uma melhora em termos de acurácia em comparação com os outros algoritmos.

Batiha & Krömer (2020) apresentaram um algoritmo de classificação e regressão baseado em uma técnica híbrida de classificação *evolutionary-fuzzy*. Esta é aplicada no treinamento de diversos classificadores na tarefa de detecção de intrusão em RSSF. *Evolutionary Fuzzy Rules* (EFR) é descrita como uma técnica com alta acurácia, baixos requerimentos de performance computacional e consumo de energia e flexibilidade, requerimentos para os dispositivos que formam RSSF. Os autores propõem a utilização destas EFR na detecção de tipos específicos de ataques, estes são os ataques identificados pelo dataset WSN-DS, *Blackhole, Grayhole, Scheduling* e *Flooding*. O *dataset* foi separado em *datasets* menores para classificar os tipos específicos de ataque, estes foram utilizados para evoluir os EFR de cada tipo de ataque. Os modelos apresentaram boa performance de classificação em termos gerais, no entanto, no caso do modelo de detecção de ataques *Grayhole*, este apresentou uma taxa elevada de falsos positivos.

Otoum, Kantarci & Mouftah (2020) propuseram um SDI que aplica um ensemble com *Random Forest, Density Based Spatial Clustering of Applications with Noise* e *Restricted Boltzmann Machine* como classificadores base e *Bayesian Combination Classification* (Dependente e Independente) como combinador. Os dados relevantes dos sensores são reportados pelos *Cluster Heads*, os quais são separados em dois (2) segmentos de dados utilizados para criação de dois *ensembles* com as configurações propostas. Para realizar a avaliação foi criada uma simulação utilizando *Network Simulator-3* com 40 sensores que se comunicam utilizando o protocolo DSR, com cinco (5) *clusters*, cada um com 8 (oito) nós sensores. Além disso, o dataset NSL-KDD também é utilizado. Os resultados observados na comparação da performance de classificação apontam para melhor o desempenho foi alcançado pelo *ensemble* com *Dependent Bayesian Combination Classification*. A comparações realizadas são dos modelos base individualmente, e dos modelos base em *ensemble* utilizando *Bayesian Combination Classification* (Dependente e Independente).

Os autores Park, Cho, Kim et al. (2018) propuseram a utilização de um *ensemble* do tipo *Random Forest* para a detecção de ataques do tipo DoS. Os experimentos são realizados com a base WSN-DS e os ataques descritos são *Blackhole, Greyhole, Scheduling* e *Flooding*. A avaliação é realizada com o auxílio de uma matriz de confusão e métricas de performance de classificação. Os resultados dos experimentos com *Random Forest* são comparados com os resultados obtidos de uma rede neural artificial do trabalho original da WSN-DS, sendo que as métricas de performance de classificação com o *Random Forest* apresentaram resultados melhores que a rede neural.

4.3 Detecção de Intrusão ou Anomalias no Nível de Sensores em RSSF

Ding, Fei, Du & Xu (2014) descrevem um método de detecção de anomalias em RSSF baseado em aprendizado *online* utilizando *ensembles* com a aplicação de *Biogeography-based Optimization* (BBO). Os autores utilizam os dados de sensores para construir o *dataset*, cada nó sensor da rede tem um *ensemble* criado através dos dados de treinamento, estes nós se comunicam com os *Cluster Heads* que também criam *ensembles*. Considerando a natureza espaço-temporal dos dados de treinamento, o processo de criação linear dos *ensembles* e a natureza *online* do método proposto, apenas versões mais novas dos *ensembles* de cada nó são mantidas. BBO é aplicada no processo de poda dos *ensembles*, e um processo de aprendizado *online* é utilizado para continuamente atualizar os *ensembles* da rede. São aplicados testes para verificar a performance de classificação de um *ensemble* do tipo *bagging* composto por SVM de uma classe. Os resultados apresentados são do *ensemble* global criado pela agregação dos *ensembles* dos nós sensores pelos *Cluster Heads* e então, pela agregação dos *ensembles* criados pelos *Cluster Heads*. São apresentadas versões com e sem poda com aplicação de BBO, apresentando boa performance em ambos os casos.

O SDI proposto por Alruhaily & Ibrahim (2019) possui duas camadas de detecção, uma com a aplicação de *Naive Bayes* nos sensores da RSSF para classificação dos pacotes inspecionados, e outra com a aplicação de *Random Forest* na nuvem para detecção multiclasse. A detecção é dividida e classificada em *Edge-based* em nós sensores e *Cloud-based* na nuvem. Os nós sensores aplicam *Naive Bayes* pare detecção binária diferenciando tráfego normal de malicioso, as instâncias identificadas como maliciosas são então enviadas para a nuvem onde detecção multiclasse é realizada para identificar o tipo de ameaça. O dataset WSN-DS é aplicado para treinamento e testes. As ameaças identificadas são *Blackhole, Grayhole, Scheduling* e *Flooding*, e os experimentos aplicados apresentam performance de classificação satisfatória.

Os autores Yahyaoui, Abdellatif & Attia (2019) propuseram um protocolo hierárquico de detecção por anomalia, chamado *Hierarchical Anomaly Detection and Localization* (HADL), que utiliza um SVM de uma classe em nós da rede, e um algoritmo de *deep learning* na estação base. As contribuições propostas deste protocolo são, um mecanismo de detecção por anomalia executado apenas quando necessário, com um sistema híbrido de aprendizado de máquina adaptado ao nível de recursos do sensor, e um sistema baseado em confiança, que utiliza dados estatísticos para localizar nós maliciosos. Para os experimentos foram utilizados o simulador Castalia 3,2 (simula um ambiente de WSN), e o *dataset* KDD99 para o processo de treinamento e testes. São observados bons resultados com uma alta taxa de detecção de 95% quando observada uma taxa alta de pacotes de um nó malicioso, além de conseguir detectar nós suspeitos mesmo com uma taxa baixa de pacotes.

O SDI proposto pelos autores Wang, Li, Cheng, Bhatti & Dai (2018) é composto por 3

módulos, aquisição e pré-processamento de dados, análise e detecção, e o módulo de resposta. A abordagem de detecção de intrusão utiliza SVM. Para a detecção é proposto um modelo de detecção de 3 níveis, sendo estes, *Cluster Member, Cluster Head* e *Sink Nodes*. O treinamento e detecção são feitos hierarquicamente, começando dos *Cluster Members*, passando pelos *Cluster Heads* e então pelo *sink node*. A QualNet foi utilizada como plataforma de testes (simulação). O algoritmo apresenta uma boa taxa de detecção e baixa taxa de alarmes falsos, demonstrando a viabilidade de SVM para detecção de intrusão em RSSF.

Capítulo 5

Detecção de Intrusão em Nós Sensores de RSSF

5.1 Aprendizado de Máquina para Detecção de Intrusão em Nós Sensores

A proposta do trabalho consiste na aplicação de algoritmos de aprendizado de máquina para detecção de intrusão por anomalia, realizando classificação multiclasse, em nós sensores de Redes de Sensores Sem Fio (RSSF). Especificamente, o foco é verificar a viabilidade da aplicação dos algoritmos *Naive Bayes* (NB), *Multilayer Perceptron* (MLP), *Decision Trees* (DT) e *Random Forest* (RF) em nós sensores, para realizar a detecção e identificação de intrusões. Para a realização do protocolo experimental, estes algoritmos foram comparados em termos de performance de classificação e consumo de recursos (performance de execução). As métricas selecionadas, para a comparação, foram escolhidas com base na análise da literatura e conforme a importância para o processo de detecção de intrusão. Além disso, foram consideradas algumas características representativas de nós sensores em consumo de recursos.

O trabalho tem foco na execução da detecção por anomalia em nós sensores, afim de implementar um sistema de detecção de intrusão baseado em rede em trabalhos futuros. É considerada detecção ativa operando de forma parcialmente distribuída através da aplicação de modelos treinados para detecção em nós da RSSF.

Os algoritmos foram selecionados considerando a utilização de algoritmos vinculados a diferentes paradigmas, vinculados ao aprendizado de máquina. Estes também são comumente utilizados em outros trabalhos que lidam com detecção de intrusão e especificamente detecção de intrusão em RSSF.

Para o ajuste dos parâmetros dos algoritmos selecionados, foram realizados testes para verificar o impacto na performance de classificação e consumo de recursos computacionais. Na fase de classificação foi aplicado o *10Fold Cross Validation*, e para as métricas de execução foram utilizados *train* e *test split*. Na avaliação de desempenho foi realizada apenas uma predição apenas para cada algoritmo, ao passo que no caso do consumo de energia aplicou-se

10 (dez) predições, tendo em vista que com poucas predições não é possível mensurar o consumo de energia no intervalo. Para as métricas de classificação e de custo computacional foram considerados os seguintes aspectos:

- Idealmente, a acurácia balanceada de um algoritmo deve ser maior que 0,6;
- O tamanho do modelo treinado do algoritmo deve se manter próximo a 1/10 do tamanho máximo observado nos sensores convencionais observados, 12800B (bytes);

O método experimental se baseia nas etapas do processo de *Knowledge Discovery in Databases* (KDD). A base de eventos de segurança utilizada para validar o modelo foi a WSN-DS (Almomani et al., 2016), que lida com ameaças a RSSF. Não foi necessária a aplicação de técnicas de pré-processamento, tendo em vista que a base foi criada sinteticamente. A WSN-DS foi submetida a técnicas de seleção de atributos na etapa de transformação dos dados. O processo de mineração de dados consistiu na utilização dos algoritmos de classificação selecionados, fornecendo subsídios para o cálculo das métricas que foram comparadas na etapa de avaliação. A Figura 5.1 apresenta a delineação do método proposto.

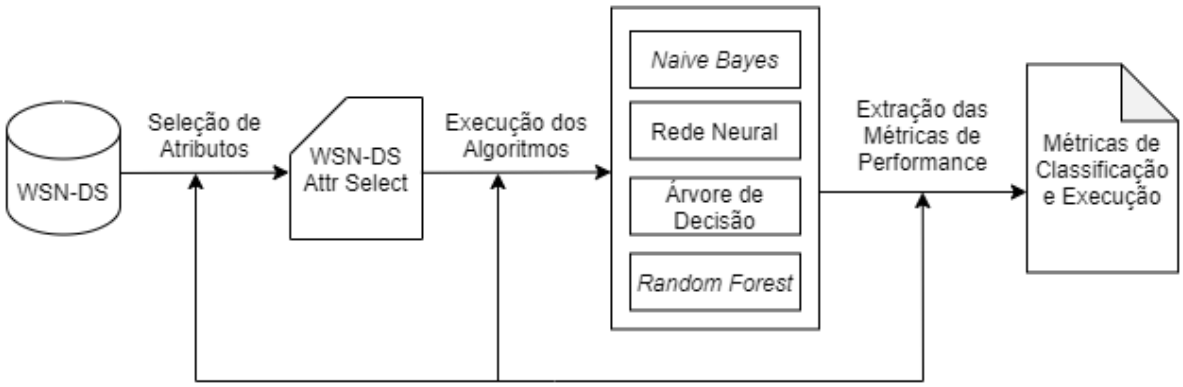

Figura 5.1: Delineação do método proposto.

5.1.1 Seleção dos Algoritmos

Os algoritmos utilizados foram selecionados por representarem diferentes paradigmas entre algoritmos de aprendizado de máquina, além de serem comumente aplicados em outros trabalhos que lidam com detecção de intrusão por anomalia em redes comuns e RSSF. Através de pesquisas na literatura, observa-se que os algoritmos selecionados geralmente apresentam boa performance de classificação.

Algoritmos Clássicos

Os autores Elmrabit, Zhou, Li & Zhou (2020) realizaram comparações entre diversos algoritmos, incluindo *Naive Bayes*, Árvore de Decisão e Rede Neural, em 3 (três) *datasets* de detecção de intrusão. Nestes modelos, os autores aplicaram classificação multiclasse, sendo que os algoritmos de Rede Neural e Árvore de Decisão apresentaram boa performance nas métricas observadas (Acurácia, Precisão, *Recall* e *F-Score*), para as 3 (três) bases. Já o método *Naive Bayes* não apresentou boa performance consistentemente.

Em outro trabalho, os autores Amor, Benferhat & Elouedi (2004) compararam os algoritmos *Naive Bayes* e Árvore de decisão, utilizando o *dataset* KDD'99, e obtiveram performance de classificação similares.

Já Folleco, Khoshgoftaar, Van Hulse & Bullard (2008) observaram o impacto de ruídos nos dados, comparando diversos algoritmos, entre eles, *Naive Bayes* e Árvore de Decisão C4.5. Os autores observaram grande impacto dos ruídos no desempenho da Árvore de Decisão, ao passo que o *Naive Bayes* apresentou-se robusto aos ruídos introduzidos.

Algoritmos de *Ensemble*

Algoritmos do tipo *ensemble* foram explorados por sua escalabilidade e bons resultados na redução de viés e pela resistência a ruído. Dentre os algoritmos de ensemble explorados na literatura, estão *boosting*, *bagging* e *stacking*. Os algoritmos do tipo *bagging*, especificamente o *Random Forest* apresentaram boa performance de classificação e consumo de recursos, além de apresentarem bons resultados em trabalhos com foco em RSSF e Detecção de Intrusão. Os autores Elmrabit et al. (2020), por exemplo, também consideraram o *Random Forest* em suas comparações, e verificaram que ele apresentou a melhor performance entre todos os algoritmos explorados, para classificação binária e multiclasse. O ruído nos dados de treinamento e de classificação não são considerados neste trabalho, mas são comuns em RSSF. No trabalho de Folleco et al. (2008), o impacto de ruídos para o *Random Forest* foi avaliado, e os resultados demonstram a robustez de modelos de *Random Forest*.

5.1.2 Base de Eventos de Segurança WSN-DS

A base de eventos de segurança computacional, selecionada para os experimentos, foi a WSN-DS (Almomani et al., 2016), construída utilizando o *software* de simulação *Network Simulation-2* e considerando o protocolo de roteamento *Low-Energy Adaptive Clustering Hierarchy* (LEACH). A base foi criada com o intuito de representar os principais tipos de ataque de *Denial of Service* (DoS), para auxiliar no processo de detecção de intrusão por anomalia em

RSSF. A WSN-DS foi utilizada por simular o tráfego de uma RSSF, além de modelar apenas alguns tipos de ataques comuns nestas redes com foco específico em redes LEACH, limitando assim o escopo dos experimentos. A base escolhida contém os seguintes atributos:

- **Node ID:** Identificador de um nó sensor, em um *round* e estágio;
- **Time:** Tempo de simulação do nó;
- **Is CH:** Define se o nó é *Cluster Head* (CH) (1) ou *Normal Node* (0);
- **Who CH:** O ID do CH no round atual;
- **RSSI:** *Received Signal Strength Indication* entre o nó e seu CH no *round* atual;
- **Distance to CH:** A distância entre o nó e o CH no *round* atual;
- **Max distance to CH:** A distância máxima entre um CH e os nós daquele *cluster*;
- **Average distance to CH:** A distância média entre nós no *cluster* a seu CH;
- **Current energy:** Energia atual do nó no *round*;
- **Energy consumption:** A quantidade de energia consumida no *round* anterior;
- **ADV_CH send:** O número de mensagens de *advertise* CH's enviadas para os nós;
- **ADV_CH receive:** O número de mensagens de *advertise* CH recebidas pelos nós;
- **Join_REQ send:** O número de *join requests* enviadas pelos nós para os CH;
- **Join_REQ receive:** O número de *join requests* recebidas pelos CH dos nós;
- **ADV_SCH send:** O número de mensagens de *advertise* TDMA *schedule* enviadas aos nós;
- **ADV_SCH receive:** O número de mensagens de TDMA *schedule* recebidas pelos nós;
- **Rank:** A ordem do nó no TDMA *schedule*;
- **Data sent:** O número de pacotes de dados enviados do sensor para o seu CH;
- **Data received:** O número de pacotes de dados recebidos dos CH;
- **Data sent to BS:** O número de pacotes de dados enviados a *Base Station* (BS);
- **Distance CH to BS:** A distância entre um CH e a BS;
- **Send Code:** O *sending code* do *cluster*;
- **Attack Type:** Classe objetivo, representa o tipo de ataque identificado.

A base WSN-DS é composta por 4 (quatro) classes de ataques *Blackhole Attack, Grayhole Attack, Flooding Attack, Scheduling Attack* e pela classe de eventos Normal. A base contém 374661 registros representando os 4 tipos de ataques de negação de serviço apresentados. Desses registros, 340066 pertencem a classe Normal, 3312 representam ataques de *Flooding*, 6638 TDMA, 14596 *Grayhole* e 10049 *Blackhole*, sendo assim uma base desbalanceada.

OS autores da base (Almomani et al., 2016) também descrevem para cada tipo de ataque modelado, quais seus parâmetros simulados. Estes são:

- ***Blackhole Attack:*** as instâncias de ataques são realizadas através da atribuição da tarefa de CH ao nó, este então derruba todos os pacotes recebidos dos nós do *Cluster* que seriam enviados a estação base;
- ***Grayhole Attack:*** simulado de forma similar ao ataque *Blackhole*, com a atribuição da tarefa de CH ao nó e a derrubada de pacotes a caminho da estação base. A principal diferença é que os pacotes são derrubados de forma seletiva, ou aleatória, este sendo o caso para as simulações realizadas;
- ***Flooding Attack:*** estes ataques foram simulados através do envio de um grande número de mensagens do tipo ADV_CH aos outros nós da rede, para a simulação foram utilizados intervalos de entre 10 e 50 para o número de mensagens enviadas;
- ***Scheduling Attack:*** implementados através de atribuição de intervalos de transmissão (TDMA) iguais para nós da rede. São simulados através da atribuição do mesmo intervalo para todos os nós de um CH, para conjuntos de 2 (dois) nós ou para conjuntos de 5 (cinco) nós.

5.1.3 Seleção das Métricas de Performance

A proposta considera a realização da classificação multiclasse nos sensores, já que o treinamento dos modelos não será realizado nos sensores. Portanto, a métrica de predição contempla as restrições da RSSF. O conjunto de métricas de performance selecionadas contém algumas relacionadas ao consumo de recursos (métricas de execução) e algumas relacionadas a predição (métricas de classificação). A captura das métricas de execução visa observar a performance dos algoritmos em comparação com os recursos comumente disponíveis para nós sensores, já as métricas de classificação lidam com a qualidade dos resultados da predição dos algoritmos.

Métricas de Execução

As métricas de execução para avaliação dos algoritmos foram selecionadas com base na observação de *hardware* de sensores encontrados na literatura. As métricas definidas são o tamanho do modelo, o consumo de energia e o tempo de execução para a predição de um registro. O tamanho dos modelos é considerado tendo em vista a quantidade de memória limitada de nós sensores. O consumo de energia é usado, tendo em vista que uma das maiores restrições existentes em nós sensores lidam com a disponibilidade de energia. Por fim, o tempo de execução tem associação com a possível indisponibilidade de recursos dos nós sensores durante o

processo de predição, além do consumo de energia associado com este. A Tabela 5.1 apresenta um apanhado dos dispositivos comumente utilizados em RSSF, observados nos trabalhos dos autores Jaladi, Khithani, Pawar, Malvi & Sahoo (2017), Vujović & Maksimović (2015) e Vieira, Coelho, da Silva & da Mata (2003):

Tabela 5.1: *Hardware* de nós sensores. Fonte: O Autor.

Modelo	CPU	RAM	Armazenamento
Arduino Mega 2560	?	8KB	256KB
MicaZ	ATMEGA128	4KB	128KB
TelosB	TI MSP430	10KB	48KB
Iris	ATMEGA1281	8KB	128KB
Cricket	ATMEL128L	4KB	512KB
Lotus	ARM NXP LPC1758	64KB	512KB
Raspberry Pi	ARM BCM2835	256-512MB	2-64GB
AT90LS8535	?	512B	8b
ATMega103L	?	4KB	128b
PIC16F8X	?	1B	68b
MSP430F149	?	2048B	60b
Atmel AT91M42800A	?	?	8KB
MC68HC05PV8A	?	192B	?
80C51RD+	?	1024B	64KB
DragonBall MC9328MX1	?	128KB	?

Considerando os recursos disponíveis nos dispositivos citados, foi definido um valor de referência para o tamanho dos modelos treinados, consistindo em 1/10 (um décimo) da capacidade de memória disponível ao maior nó sensor convencional, ou seja, cerca de 12.8KB (12800B). O *Raspberry Pi* não foi considerado na definição deste valor de referência, já que os recursos destes dispositivo não correspondem aos recursos dos outros dispositivos observados.

Métricas de Classificação

As métricas de classificação comumente utilizadas para avaliação de algoritmos de aprendizado de máquina foram consideradas no planejamento deste trabalho. Desse modo, as medidas selecionadas para performance de classificação dos modelos foram (Hossin & Sulaiman, 2015):

- **Acurácia:** Taxa total de instâncias classificadas corretamente;
- **Precisão:** Taxa de todas as instâncias classificadas como positivas (intrusões), sendo realmente positivas, importante para mensurar falsos positivos;
- ***Recall*:** Taxa de instâncias positivas na base classificadas corretamente, importante pare mensurar falsos negativos;

- ***F1 Score*:** Média ponderada da precisão e *recall* (1 melhor valor, 0 pior valor), busca demonstrar a performance do modelo de acordo com a precisão e o *recall*.

A métrica mais importante para o presente trabalho é o *Recall*, o qual pode ajudar a avaliar os falsos negativos, que consistem no pior caso para o processo de detecção de intrusão. Este caso ocorre quando uma instância de intrusão é classificada como tráfego normal, e com isso, o ataque não é identificado e bloqueado. Como métrica de referência foi definida a Acurácia Balanceada, tendo o *Recall* de cada classe como suporte. Idealmente, a Acurácia Balanceada não deve ser inferior a 0,6. Este valor foi definido observando uma média dos menores valores observados com a realização de testes de classificação dos algoritmos, sem considerar limitações de *hardware*. Estes testes foram realizados utilizando a base final aplicada no trabalho.

5.2 Materiais

Nesta seção são descritos os materiais utilizados para a realização deste trabalho, separando-se em *hardware* e *software*.

5.2.1 Hardware

Para a realização dos experimentos foram utilizadas máquinas virtuais disponíveis através da plataforma *Google Colab*. Tal ambiente foi utilizado para a realização dos experimentos de classificação e para a extração de métricas de execução. Para mensurar os níveis do consumo de energia, um sistema sem virtualização com Linux foi utilizado. As especificações de *hardware* destes são descritas a seguir.

- **Métricas de consumo de energia:** Intel® Core™ i7-4790 CPU @3,60GHz, 16GB 1866MHz DDR3 Ram;
- **Colab de Métricas de Classificação:** Intel(R) Xeon(R) CPU @ 2,30GHz, 12GB Ram;
- **Colab de Métricas de Execução:** Intel(R) Xeon(R) CPU @ 2,20GHz, 12GB Ram.

5.2.2 Software

Para a consturção do modelo e execução dos experimentos foi utilizado a linguagem de programação Python, com a biblioteca Sklearn para a maior parte das tarefas de implementação de algoritmos, além de outras bibliotecas de suporte para extração de algumas métricas, como Pickle para verificar o tamanho dos modelos em memória, e PyJoules para obter o consumo de

energia. O Scipy e Matplotlib também foram utilizados para testes estatísticos e representações gráficas. O *software* de aprendizado de máquina Weka foi utilizado para realizar testes iniciais e a seleção de atributos no *dataset* WSN-DS. Todos os softwares utilizados são listados a seguir:

- Python 3.7;
- Sklearn 0.22.2;
- Pandas 1.1.5;
- PyJoules 0.5.1;
- Pickle 4.0;
- Weka 3.8;
- Scipy 1.4.1;
- Matplotlib 3.2.2.

5.3 Método Experimental

Nesta seção é descrito o processo de confecção do presente trabalho, utilizando como base para o método experimental o processo de KDD (Pg. 26).

5.3.1 Seleção dos Dados e Transformação

Seleção dos Dados

Diversas bases de dados, no contexto de detecção de intrusão, foram consideradas para serem utilizadas neste trabalho, incluindo KDD99 (Tavallaee, Bagheri, Lu & Ghorbani, 2009) e NLS-KDD (Dhanabal & Shantharajah, 2015). A base selecionada foi a WSN-DS (Almomani et al., 2016), por tratar especificamente eventos de detecção de intrusão em RSSF. A escolha desta base também ajudou na limitação do escopo das ameaças tratadas, considerando apenas as classes contidas na base.

Transformação dos Dados

A base obtida difere da versão apresentada no trabalho original (Almomani et al., 2016), mas é a versão descrita no trabalho dos autores Batiha et al. (2019) sobre *Ensembles* com Re-

des Neurais em RSSF. A aplicação de *10-Fold Cross-Validation* com *Multilayer Perceptron* no Weka apresenta resultados muito próximos dos resultados apresentados no trabalho original.

Uma importante atividade foi a seleção de atributos através da análise individual dos atributos representados pela base e a sua relação com as classes. Além disso, como métrica de suporte foram utilizados algoritmos de seleção de atributos disponíveis no WEKA, medindo especificamente a correlação de atributos com as classes utilizando Correlação de Pearson (Hall, 2000) e a medida de ganho de informação de cada atributo (Quinlan, 1986). A Figura 5.2 apresenta o processo de seleção de atributos e escolha da base.

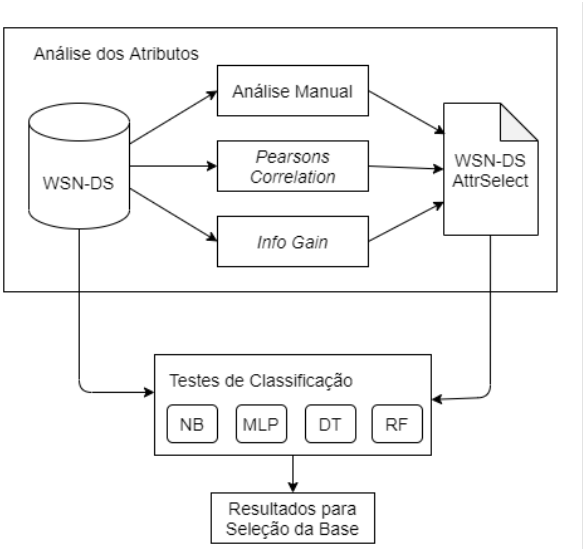

Figura 5.2: Seleção de atributos e escolha da base.

Os atributos da base original e a sua relevância para a tarefa de classificação, são:

- **id:** É o identificador do nó sensor, e possui pouca relevância para treinamento e classificação na tarefa de detecção de intrusão;
- **Time:** Indicar o tempo de simulação, ou seja, a quantidade de tempo o nó sensor está ativo na rede de sensores. Esta métrica pode ser relevante, pois a idade do nó na rede pode indicar a confiabilidade do mesmo, no entanto, alguns ataques utilizam nós sensores que já faziam parte da rede;
- **Is_CH:** Define se o nó é *Cluster Head* (1) ou nó normal (0). Considerando a existência de ataques que se passam por CH, este atributo é relevante;
- **who_CH:** É o identificador do CH do nó no round atual. Com a proposta de remoção do atributo id, este atributo deixa de ser relevante, no entanto, é possível que a união dos atributos id e who_CH apresente informações relevantes para a tarefa de classificação;
- **Dist_To_CH:** Indica a distância entre o nó e seu CH. Considerando as classes de ataques da base, atributos de distância entre um nó e seu *Cluster Head* não parecem ser relevantes;

- **ADV_S:** Indica o *ADV_CH send* na base original, ou seja, o número de *broadcasts* de *advertise* como CH enviados a nós. Este atributo tem relação com o comportamento de ataques representados na base;
- **ADV_R:** Indica o *ADV_CH receive* na base original, ou seja, o número de *broadcasts* de *advertise* como CH recebidos pelos nós. Este atributo tem relação ao comportamento de ataques representados na base;
- **JOIN_S:** Indica o *Join_REQ send* na base original, ou seja, o número de *join requests* enviadas pelos nós para um CH, este atributo pode descrever características de ataques de *flooding*, no entanto, na simulação que gera a WSN-DS, *flooding* é simulado utilizando mensagens de ADV_CH. Para a WSN-DS especificamente este atributo pode não ser relevante, mas, a união deste atributo e JOIN_R podem apresentar características de perda de pacotes que caracterizam outros ataques;
- **JOIN_R:** Indica o *Join_REQ receive* na base original, ou seja, o número de *join requests* recebidas pelo CH dos nós. Este atributo pode descrever características de ataques de *flooding*, no entanto, na simulação que gera a WSN-DS, *flooding* é simulado utilizando mensagens de ADV_CH. Para a WSN-DS especificamente este atributo pode não ser relevante, mas, a união deste atributo e JOIN_S podem apresentar características de perda de pacotes que caracterizam outros ataques;
- **SCH_S:** Corresponde ao *ADV_SCH send* na base original, ou seja, ao número de *broadcasts* de *scheduling* enviado para os nós. É representativo de ataques de *scheduling* na base;
- **SCH_R:** Indica o *ADV_SCH receive* na base original, ou seja, o número de *broadcasts* de *scheduling* recebidos pelos nós. É representativo de ataques de *scheduling* na base;
- **Rank:** Indica a ordem do nó no *schedule*. Pode ser representativo de ataques de *scheduling* (de acordo com a descrição da base, *scheduling* é simulado, em parte, através da atribuição do mesmo *time slot* para vários nós);
- **Data_S:** Indica o *Data sent* na base original, ou seja, o número de pacotes enviados de um nó para o seu CH, similar ao JOIN_S e JOIN_R;
- **Data_R:** Indica o *Data received* na base original, ou seja, o número de pacotes recebidos de um nó pelo seu CH, similar ao JOIN_S e JOIN_R;
- **Data_Sent_To_BS:** Corresponde ao número de pacotes de dados enviados a *base station*. Este atributo parece não ser relevante considerando os ataques representados na base, mas pode ser relevante na classificação de outros tipos de ataques;
- **dist_CH_To_BS:** Indica a distância entre o CH e a *base station*. Considerando as classes de ataque classificadas, este atributo não parece relevante;
- **send_code:** É Descrito como *sending code* do *cluster*, o significado dos códigos não é especificado;

- **Consumed_Energy:** Quantidade de energia consumida no *round* anterior, pode ser indicativo da situação da rede (diversos ataques aumentam o consumo de energia consideravelmente), sem contexto, este atributo pode não contribuir muito para a tarefa de classificação;
- **Attack_type:** Indica a classe do registro;

Para suporte a análise dos atributos, foi realizada a seleção de atributos no WEKA, com o modo de seleção de atributos *10-Fold Cross-Validation Stratified* para validação. Os resultados do processo de seleção de atributos são apresentados na Tabela 5.2. As técnicas aplicadas para a seleção foram:

- ***Pearsons correlation*:** Realiza a avaliação dos atributos medindo a correlação entre o atributo e o valor da classe (utilizando Correlação de Pearson) (Hall, 2000);
- ***Info Gain*:** Realiza a avaliação dos atributos medindo o ganho de informação destes em relação à classe (Quinlan, 1986).

Tabela 5.2: Resultados da seleção de atributos. Fonte: O Autor.

Pearsons Correlation	***Info Gain***
Is_CH: 0,831 +- 0	ADV_S: 0,391+- 0
JOIN_S: 0,566 +- 0	Is_CH: 0,348+- 0
SCH_R: 0,499 +- 0	Consumed Energy: 0,339+- 0,001
ADV_R: 0,407 +- 0	DATA_S: 0,276+- 0
ADV_S: 0,33 +- 0,002	Rank: 0,236+- 0
JOIN_R: 0,327 +- 0,001	send_code: 0,22+- 0
DATA_S: 0,317 +- 0	JOIN_S: 0,22+- 0
send_code 0,312 +- 0	Dist_To_CH: 0,22+- 0
Dist_To_CH: 0,309 +- 0	ADV_R: 0,185+- 0
Time: 0,302 +- 0	SCH_R: 0,18+- 0
SCH_S: 0,295 +- 0,001	SCH_S: 0,154+- 0
Rank: 0,197 +- 0	who_CH: 0,13 +- 0
who_CH: 0,193 +- 0,001	Data_Sent_To_BS: 0,129+- 0
id: 0,193 +- 0,001	id: 0,119+- 0
dist_CH_To_BS: 0,162	JOIN_R: 0,117+- 0
DATA_R: 0,154 +- 0,001	dist_CH_To_BS: 0,084+- 0,001
Data_Sent_To_BS: 0,136 +- 0,001	Time: 0,083+- 0
Consumed Energy: 0,075 +- 0,001	DATA_R: 0,058+- 0,001

Os resultados dessa seleção são apresentados de forma ranqueada, números mais próximos de 1 (um) apresentam atributos mais relevantes para a classificação. Considerando os

resultados da execução dos algoritmos de seleção de atributos, e a avaliação individual de relevância de cada atributo, a Tabela 5.3 apresenta a lista dos atributos mantidos e removidos da base.

Tabela 5.3: Atributos mantidos e removidos. Fonte: O Autor.

Atributos Mantidos	**Atributos Removidos**
Time	id
Is_CH	who_CH
Dist_To_CH	Data_Sent_To_BS
ADV_S	dist_CH_To_BS
ADV_R	
JOIN_S	
JOIN_R	
SCH_S	
SCH_R	
Rank	
Data_S	
Data_R	
send_code	
Consumed_Energy	
Attack_type	

Após a seleção de atributos com os algoritmos e a análise dos dos mesmos, foram realizados testes com a base original e a base modificada para verificar a performance da nova base. Os algoritmos aplicados foram *Naive Bayes* (ComplementNB), *Multilayer Perceptron* (1 camada intermediária com 3 neurônios), Árvore de Decisão (profundidade 5) e *Random Forest* (5 árvores com profundidade 5). As Tabelas 5.4 e 5.5 apresentam os resultados desses testes, utilizando *10-Fold Cross Validation Stratified*.

Tabela 5.4: Comparação das bases 1. Fonte: O Autor.

	Original, NB	**Attr Select, NB**	**Original, MLP**	**Attr Select, MLP**
Acurácia Balanceada	0,658 (0,083)	0,684 (0,046)	0,218 (0,054)	0,779 (0,051)
Blackhole Recall	0,365 (0,201)	0,331 (0,175)	0,100 (0,300)	0,419 (0,283)
Flooding Recall	0,491 (0,207)	0,703 (0,154)	0,000 (0,000)	0,960 (0,064)
Grayhole Recall	0,683 (0,217)	0,799 (0,128)	0,003 (0,009)	0,656 (0,319)
Normal *Recall*	0,856 (0,093)	0,855 (0,092)	0,987 (0,040)	0,993 (0,006)
TDMA *Recall*	0,893 (0,153)	0,733 (0,155)	0,001 (0,002)	0,870 (0,146)

Tabela 5.5: Comparação das bases 2. Fonte: O Autor.

	Original, DT	**Attr Select, DT**	**Original, RF**	**Attr Select, RF**
Acurácia Balanceada	0,863 (0,065)	0,878 (0,056)	0,883 (0,050)	0,859 (0,059)
Blackhole Recall	0,858 (0,177)	0,820 (0,095)	0,895 (0,064)	0,776 (0,143)
Flooding Recall	0,992 (0,006)	0,992 (0,006)	0,928 (0,066)	0,877 (0,148)
Grayhole Recall	0,723 (0,145)	0,714 (0,144)	0,793 (0,153)	0,789 (0,120)
Normal *Recall*	0,999 (0,003)	0,999 (0,003)	0,996 (0,005)	0,991 (0,007)
TDMA *Recall*	0,744 (0,224)	0,867 (0,138)	0,804 (0,175)	0,863 (0,139)

Na maioria das execuções foram observadas diferenças pequenas em termos da performance de classificação da Acurácia Balanceada e do *Recall* de cada classe. Exceto no caso de *Multilayer Perceptron*, na qual a performance observada na base com seleção de atributos foi muito maior do que a performance da base original. Considerando as diferenças de performance e a redução de dimensionalidade da base, a versão reduzida com seleção de atributos foi utilizada para os experimentos.

5.3.2 Implementação e Execução dos Algoritmos

A linguagem Python com a biblioteca *Sklearn* foi utilizada para implementação dos algoritmos. Para a extração das métricas de performance de classificação foram realizados testes utilizando *10-fold cross validation stratified*. A extração das métricas de performance de execução foi realizada através da técnica de *train* e *test splits*, para verificar o desempenho de uma predição, ou conjunto de predições no caso do consumo de energia.

Métricas de Classificação Selecionadas do Sklearn

Dentre as métricas de classificação disponíveis, as citadas nas seções anteriores foram selecionadas, com a adição de uma nova métrica, a *Balanced Accuracy* ou Acurácia Balanceada que é definida como a média do *Recall* de cada classe, sendo útil para sumarizar o impacto de cada classe na performance de algoritmos para toda a base.

Algoritmos Selecionados do Sklearn

Os algoritmos selecionados foram *Naive Bayes*, Rede Neural, Árvore de Decisão e *Random Forest*. As implementações foram executadas em ambientes do *Google Colab*, os quais são chamados *Notebooks*. Um *Notebook* foi utilizado para as implementações que lidam com

a extração das métricas de classificação, e outro foi utilizado para a extração das métricas de execução. Como não é possível obter os dados de consumo de energia utilizando virtualização, esta parte foi implementada em um computador local. Para verificar a viabilidade dos modelos, considerando as limitações de sensores, foram realizados testes exaustivos verificando o ganho de performance de classificação (com foco em *Recall* individual) e consumo de recursos (com foco no tamanho do modelo treinado). O propósito foi definir os parâmetros aceitáveis para comparação das implementações de cada algoritmo.

Naive Bayes

Dentre as implementações de *Naive Bayes* disponíveis no *sklearn*, a mais apropriada para a base é a *ComplementNB*, recomendada para *datasets* desbalanceados. Os modelos criados pelo *Naive Bayes* são compactos, portanto, não foi necessária a verificação do tamanho dos modelos durante os testes com parâmetros. Apenas um parâmetro do algoritmo *ComplementNB* tinha relevância para a criação do modelo utilizando a base, (*norm=true* ou *norm=false*, *false* sendo o valor padrão). *Naive Bayes* tem suporte nativo a classificação multiclasse. O *ComplementNB* com parâmetro *default* foi selecionado. A Tabela 5.6 apresenta a comparação entre as execuções do algoritmo.

Tabela 5.6: Comparação dos algoritmos. Fonte: O Autor.

	ComplementNB (norm=true)	**ComplementNB (norm=false)**
Acurácia Balanceada	0,601 (0,035)	0,684 (0,046)
Blackhole Recall	0,325 (0,173)	0,331 (0,175)
Flooding Recall	0,491 (0,180)	0,703 (0,154)
Grayhole Recall	0,802 (0,125)	0,799 (0,128)
Normal *Recall*	0,844 (0,099)	0,855 (0,092)
TDMA *Recall*	0,543 (0,123)	0,733 (0,155)

Rede Neural

A implementação de *Multilayer Perceptron* (MLP) do *sklearn* é bastante custosa. O maior tamanho de *perceptron* possível observado (através de testes exaustivos), seguindo as restrições impostas tem 2 (duas) *Hidden Layers* (HL), com 5 (cinco) neurônios em cada, uma variação com 1 (um) *Hidden Layer*, e 8 (oito) neurônios também é possível dentro nos limites. Os outros parâmetros disponíveis na implementação do *sklearn* são mantidos nas configurações recomendadas. Para realizar classificação multiclasse, é aplicada a função *Softmax* como a função de saída. O *MLPerceptron* com 2 (duas) *Hidden Layers* com 5 (cinco) neurônios cada foi selecionada, por apresentar melhor performance de classificação, e tamanho do modelo levemente menor. A Tabela 5.7 apresenta a comparação entre as execuções do algoritmo.

Tabela 5.7: Comparação dos algoritmos. Fonte: O Autor.

	MLPerceptron (HL = 2, 5)	**MLPerceptron (HL = 1, 8)**
Acurácia Balanceada	0,809 (0,043)	0,771 (0,062)
Blackhole Recall	0,311 (0,186)	0,334 (0,221)
Flooding Recall	0,988 (0,013)	0,954 (0,081)
Grayhole Recall	0,850 (0,087)	0,675 (0,344)
Normal *Recall*	0,991 (0,006)	0,993 (0,006)
TDMA *Recall*	0,904 (0,146)	0,899 (0,144)

Árvore de Decisão

O principal parâmetro para o ajuste do tamanho do modelo gerado pela árvore de decisão é a profundidade máxima da árvore, através da aplicação de teste para verificação dos modelos gerados, onde 2 (dois) candidatos são propostos. Na profundidade 7, alguns modelos gerados ultrapassam o limite de 12800B (bytes), mas como esse excedente é pequeno a profundidade 7 é considerada. Na profundidade 6 todos os modelos estão abaixo do limite máximo estipulado, porém a performance de classificação é mais baixa. A seleção da profundidade máxima (MD) considera se a diferença das performances de classificação é significativa suficiente para justificar alguns modelos excedendo o tamanho. Este classificador tem suporte nativo a classificação multiclasse, e a implementação de árvore de decisão utilizada é uma versão otimizada do algoritmo CART, aplicando *Gini* como medida de impureza. O modelo com profundidade máxima de 6 (seis) foi selecionado, já que a diferença na performance entre este e o modelo de profundidade 7 (sete) é pequena, e todos os *Recalls* já estão no limite aceitável. A Tabela 5.8 apresenta a comparação entre as execuções do algoritmo.

Tabela 5.8: Comparação dos algoritmos. Fonte: O Autor.

	***Decision Tree* (MD = 7)**	***Decision Tree* (MD = 6)**
Acurácia Balanceada	0,898 (0,046)	0,872 (0,066)
Blackhole Recall	0,757 (0,201)	0,773 (0,204)
Flooding Recall	0,954 (0,085)	0,996 (0,006)
Grayhole Recall	0,916 (0,040)	0,725 (0,146)
Normal *Recall*	0,993 (0,004)	0,999 (0,003)
TDMA *Recall*	0,868 (0,135)	0,867 (0,138)

Random Forest

Os parâmetros de ajuste do modelo gerado pelo algoritmo *Random Forest* são o número de árvores e a profundidade máxima de cada árvore. Durante os testes foi observado um aumento considerável no tamanho dos modelos a medida que se aumenta a profundidade máxima das árvores, enquanto o aumento do número de árvores não apresentava um aumento significativo no tamanho dos modelos. A menor profundidade para manter uma performance aceitável

é 2 (dois), mesmo com um número muito grande de árvores, se as profundidades são muito pequenas a performance de classificação é baixa. Dentre os modelos avaliados, alguns ultrapassam o limite estipulado de 12800B, mas de forma similar as árvores de decisão, estes serão considerados se o ganho de performance de classificação for considerável.

Este classificador tem suporte nativo a classificação multiclasse e utiliza uma técnica de *perturb-and-combine*, garantindo a diversidade dos classificadores através da introdução de aleatoriedade na construção destes. A predição do *ensemble* é uma média das predições dos classificadores. Cada árvore no *RandomForestClassifier* é construída através de *bootstrap sampling*, e os *splits* em cada nó na construção das árvores são encontrados observando todos os atributos ou um *subset* aleatório de tamanho *max_features*. Para impureza é aplicada a medida Gini. O modelo selecionado utiliza número de árvores (NT) igual a 4 (quatro) e profundidade máxima (MD) igual a 4 (quatro) . As Tabelas 5.9 e 5.10 apresentam a comparação entre as execuções do algoritmo.

Tabela 5.9: Comparação dos algoritmos 1. Fonte: O Autor.

	RF (NT = 8, MD = 3)	**RF (NT = 6, MD = 3)**
Acurácia Balanceada	0,760 (0,118)	0,657 (0,116)
Blackhole Recall	0,665 (0,253)	0,388 (0,357)
Flooding Recall	0,514 (0,421)	0,395 (0,443)
Grayhole Recall	0,854 (0,071)	0,809 (0,147)
Normal *Recall*	0,988 (0,009)	0,990 (0,008)
TDMA *Recall*	0,780 (0,234)	0,704 (0,272)

Tabela 5.10: Comparação dos algoritmos 2. Fonte: O Autor.

	RF (NT = 5, MD = 4)	**RF (NT = 4, MD = 4)**
Acurácia Balanceada	0,842 (0,081)	0,792 (0,106)
Blackhole Recall	0,827 (0,081)	0,726 (0,262)
Flooding Recall	0,803 (0,326)	0,624 (0,423)
Grayhole Recall	0,780 (0,124)	0,808 (0,095)
Normal *Recall*	0,991 (0,007)	0,988 (0,009)
TDMA *Recall*	0,810 (0,177)	0,815 (0,152)

5.3.3 Avaliação dos Resultados

O processo de avaliação dos resultados foi realizado com a aplicação de métodos estatísticos, visando verificar se existem diferenças estatisticamente significativas entre os as abordagens aplicadas. Para isto foi definida a aplicação de testes de normalidade dos dados, seguida de testes de significância estatística, e por fim, um pós-teste.

Testes de Normalidade

Estes testes tentam verificar se uma amostra de dados, pertence ou não a uma distribuição normal. Existem algumas técnicas diferentes para fazer essa verificação, geralmente lidando com análise gráfica ou estatística. Se um conjunto de dados demonstrar pertencer a uma distribuição normal, de acordo com estes testes, ele pode ser submetido a análises estatísticas com métodos paramétricos, caso contrário, métodos não paramétricos devem ser aplicados.

Testes de normalidade estatísticos geralmente provêm um valor estatístico e um p-valor. O valor estatístico pode ser utilizado para realizar uma interpretação mais profunda dos resultados, enquanto, o p-valor é utilizado para interpretar os testes de forma direta. No caso de testes de normalidade, o p-valor indica se a amostra faz parte de uma distribuição normal (gaussiana) ou não. Estes testes presumem inicialmente que a amostra faz parte de uma distribuição normal (hipótese nula ou H0), com isso é definido um nível de certeza (a), geralmente 5% (0,05), que é utilizado para interpretar o p-valor, caso $p <= a$, H0 é rejeitada, caso $p > a$, H0 não pode ser rejeitada.

Foram identificados 3 (três) principais testes de normalidade que poderiam ser aplicados aos dados, os quais foram aplicados no processo de avaliação:

- ***Shapiro-Wilk*:** Geralmente mais indicado para amostras pequenas (perto dos milhares de observações ou menos), retorna a W-statistic e o p-valor (Shapiro & Wilk, 1965);
- ***D'Agostino K2*:** Calcula *kurtosis* e *skewness* dos dados, e utiliza estes para determinar se os dados fazem parte de uma distribuição normal (D'agostino, Belanger & D'Agostino Jr, 1990);
- ***Anderson-Darling*:** Retorna uma lista de valores críticos, ao invés de um p-valor (Anderson & Darling, 1954);

Testes de Significância

Testes de significância podem ser utilizados para verificar se existem diferenças estatisticamente significativas entre dois ou mais grupos. Estes podem ser aplicados utilizando conhecimento prévio do conjunto de dados. Se pertencem a uma distribuição normal, por exemplo, aplicam-se testes paramétricos. Caso não haja conhecimento prévio do conjunto de dados, devem ser utilizados testes não-paramétricos. Dentre os testes observados, considerando a natureza não pareada dos dados, e a existência de mais de dois conjuntos de amostras para comparação, os testes utilizados foram:

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

- **Kruskal-Wallis:** Teste não-paramétrico, aplicado para comparar 2 ou mais amostras independentes (Kruskal & Wallis, 1952);

Pós-Teste

A aplicação de um teste de significância como o ANOVA ou Kruskal-Wallis identifica se existem diferenças significativas entre 2 ou mais amostras, mas para verificar qual das amostras difere, é necessário aplicar uma técnica de pós-teste. Para realizar esta comparação foi selecionado o pós-teste de Dunn (Dunn, 1961), que realiza comparações em pares para cada grupo, e identifica quais pares apresentam diferenças significativas.

Capítulo 6

Resultados e Discussões

Neste capítulo são apresentados os resultados experimentais, acompanhados de observações e discussão dos mesmos.

6.1 Performance de Execução

A definição de limites ou guias para o consumo de recursos dos algoritmos considera as especificações de dispositivos de nós sensores. Existem certas dificuldades em transpor completamente as especificações de um nó sensor a um computador convencional, portanto, alguns pontos específicos de performance foram selecionados, como tempo de execução, tamanho de memória operacional e consumo de energia. O principal ponto de comparação selecionado foi o tamanho da memória operacional, já que o consumo de energia e tempo de uso de CPU pode variar entre redes e aplicações. No entanto, mesmo com as dificuldades de comparações, a observação do tempo de predição e consumo de energia também fornece informações importantes para compreender a viabilidade da aplicação dos algoritmos em diferentes situações.

Na avaliação do tamanho dos modelos, as restrições de tamanho consideradas inicialmente eram de 4KB (kilobytes), mas uma quantidade razoável dos algoritmos ultrapassava esse tamanho, mesmo utilizando os parâmetros mínimos. Levando em conta possíveis otimizações que podem ser aplicadas em implementações em sensores, tratamento da base de dados para reduzir a dimensionalidade e as diferenças entre as linguagens de programação utilizadas em sensores e em computadores convencionais, foi considerado um valor maior, ainda condizente com o que existe em nós sensores. O maior valor observado em sensores convencionais foi 128KB. Considerando este cenário e as possíveis diferenças em implementação citadas, foi considerado adequado a utilização de $1/10$ desta capacidade de memória de 12,8kb (não considerando *Raspberry Pi*). O processo de ajuste dos parâmetros do algoritmo foi descrito na Seção 5.3.2 (Pg. 61), considerando estas definições.

As Tabelas 6.1 e 6.2 apresentam os resultados dos algoritmos para extração das métricas de execução. Nestes experimentos foram considerados o tamanho do modelo treinado, o tempo de uma predição, e o consumo de energia pela CPU, memória e total do sistema. Para o consumo

de energia, as métricas são representativas do consumo de energia em 10 predições, já que para alguns dos algoritmos não é possível medir o consumo de energia de apenas uma predição. Estes são a média e desvio padrão de 10 execuções de cada algoritmo selecionado, utilizando *train* e *test split* de 25%.

Tabela 6.1: Métricas de execução 1. Fonte: O Autor.

	NB (ComplementNB)	**MLP (*HL* = 5, 5)**
Tamanho do Modelo (B)	2067,00 (0,00)	12077,40 (1034,18)
Tempo de Predição (s)	0,00069 (0,00005)	0,00051 (0,00004)
Consumo de Energia: CPU (J)	43687,44 (472,64)	15157 (5727,07)
Consumo de Energia: RAM (J)	2909,33 (856,15)	2054,88 (1166,08)
Consumo de Energia: Total (J)	64595,44 (23006,59)	24671,66 (9301,83)

Tabela 6.2: Métricas de execução 2. Fonte: O Autor.

	DT (MD = 6)	**RF (NT = 4, MD = 4)**
Tamanho do Modelo (B)	8681,40 (701,37)	13696,40 (384,53)
Tempo de Predição (s)	0,00040 (0,00017)	0,00119 (0,00015)
Consumo de Energia: CPU (J)	10871 (8188,33)	100307,66 (12312,81)
Consumo de Energia: RAM (J)	1763,22 (1324,40)	14065,11 (1401,79)
Consumo de Energia: Total (J)	17842,55 (13427,11)	154079,44 (16339,75)

O consumo de energia dos algoritmos foi medido utilizando a biblioteca *PyJoules*, que obtém uma estimativa do consumo de energia por componentes do computador em um intervalo de tempo, obtido através de performance *counters* da CPU. Alguns dos algoritmos não apresentavam nenhum consumo de energia para uma única predição, possivelmente pelo curto tempo de uma única predição. Com isso, para obter medidas de consumo de energia de todos os algoritmos, foram realizadas 10 predições para cada algoritmo. As Figuras 6.1, 6.2 e 6.3 apresentam o consumo médio de energia registrado por cada algoritmo para a CPU, memória e consumo Total.

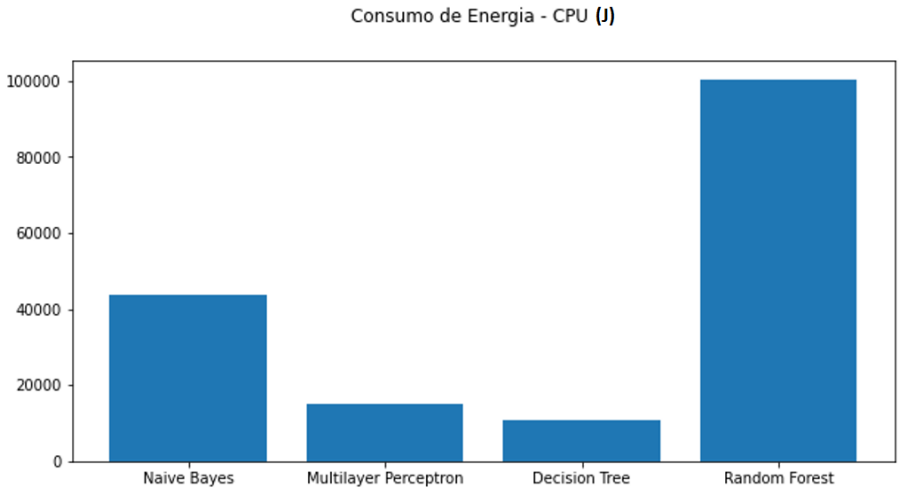

Figura 6.1: Consumo de energia pela CPU.

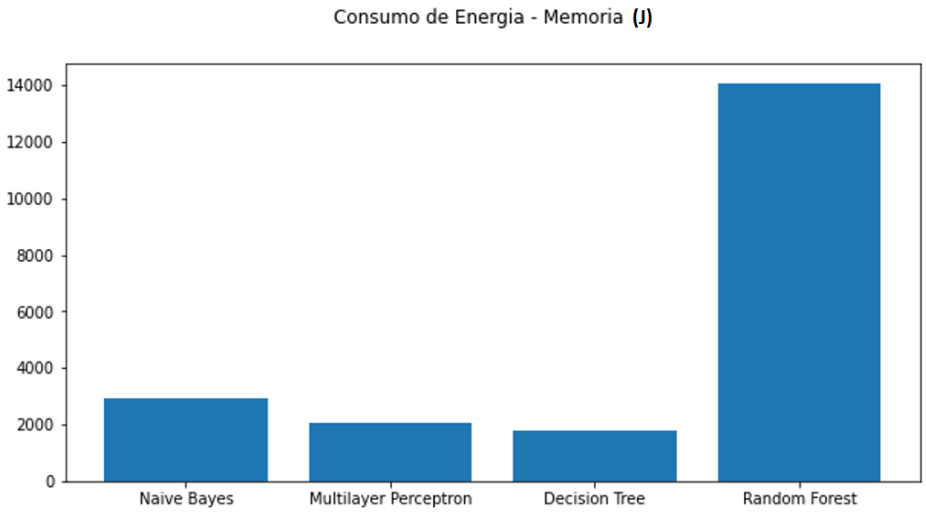

Figura 6.2: Consumo de energia pela memória.

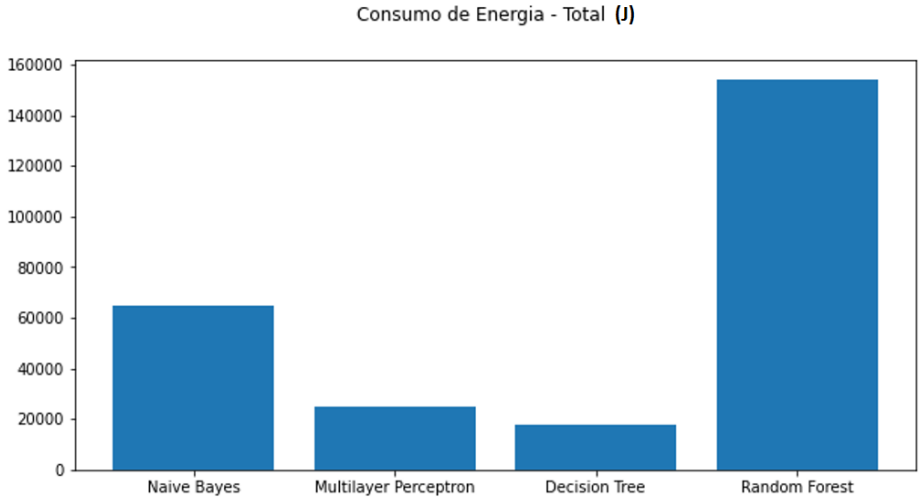

Figura 6.3: Consumo de energia total.

O consumo de energia apresenta um comportamento similar ao consumo de CPU, memória e consumo total para os quatro modelos observados. Nas 3 (três) medições o algoritmo Árvore de Decisão apresentou o menor consumo, seguidos da Rede Neural, enquanto o algoritmo de *Random Forest* apresenta o maior consumo, quase dez (10) vezes maior que o modelo com menor consumo. Observando o tempo de predição do algoritmo, é possível observar uma correlação entre tempos de predição mais altos e um consumo maior de energia, considerando o funcionamento do PyJoules, já que as medições também são representativas de outros processos do computador. Mas considerando a diferença de complexidade dos algoritmos, esta correlação com o tempo de predição não deve ser o único fator de importância, já que na execução em um nó sensor, maiores tempos de predição além acarretar um consumo maior de energia, também ocupam tempo de execução da CPU.

Os algoritmos também foram avaliados observando o tamanho dos modelos treinados,

tempo de predição e o consumo de energia. Estes foram comparados com a Acurácia Balanceada observada para cada algoritmo. As Figuras 6.4, 6.5 e 6.6 demonstram graficamente a relação entre essas métricas.

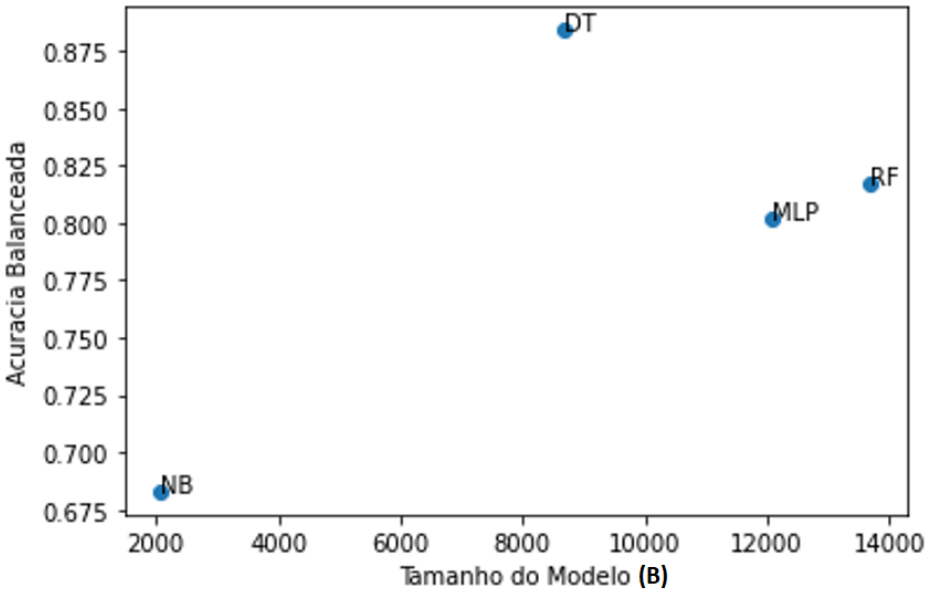

Figura 6.4: Acurácia Balanceada X Tamanho do Modelo.

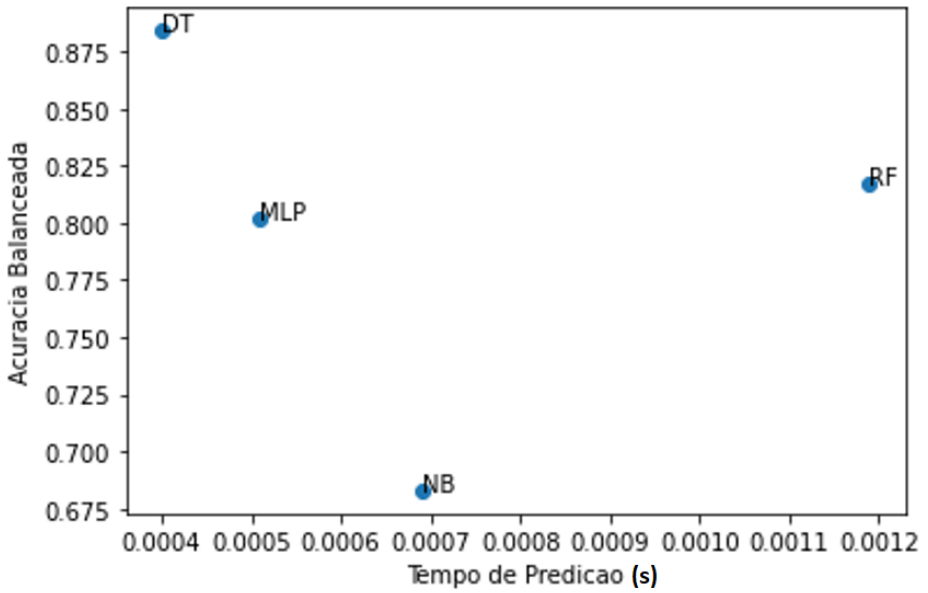

Figura 6.5: Acurácia Balanceada X Tempo de Predição.

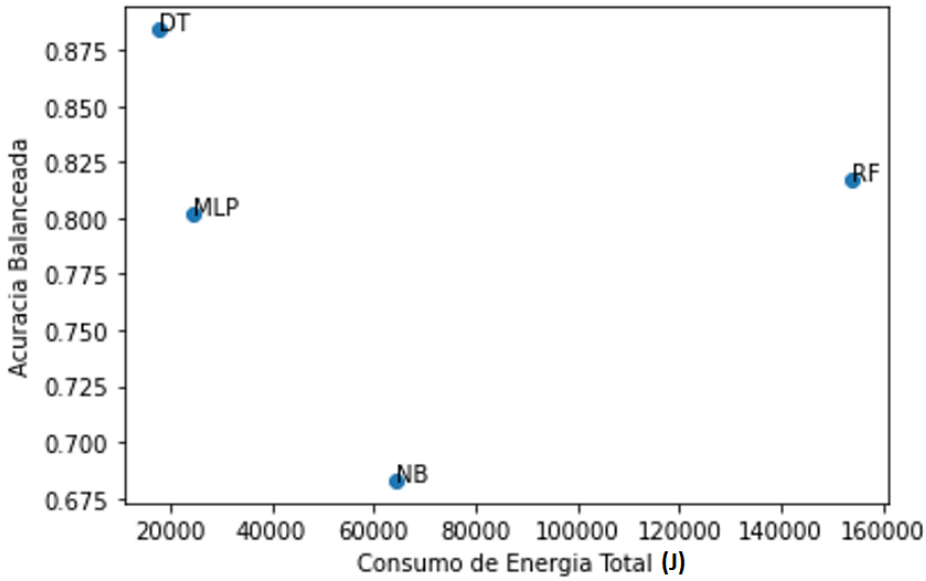

Figura 6.6: Acurácia Balanceada X Consumo de Energia Total.

A maior complexidade do algoritmo *Random Forest* pode ser observada com o maior consumo de energia, tempo de execução e tamanho do modelo. Isto se dá principalmente pelo funcionamento de algoritmos *ensemble*, que sao compostos por um conjunto de modelos, neste caso, um conjunto de arvores de decisão, criando um modelo maior. Além disso, como é realizada uma classificação por modelo, e então a agregação, o processo leva mais tempo e consequentemente, consome mais energia.

O algoritmo *Naive Bayes*, que obteve a menor performance de acurácia balanceada também apresenta o menor tamanho de modelo, mas um tempo de predição e consumo de energia consideráveis para a performance observada, o que é esperado para este algoritmo, que mantém uma grande quantidade de informações em seu modelo, além de realizar um número considerável de operações no processo de classificação.

O algoritmo *Multilayer Perceptron* apresenta boa performance de classificação pela acurácia balanceada, e apresenta um consumo de energia e tempo de predição pequenos, mas apresenta o segundo maior tamanho de modelo, próximo ao *Random Forest*. O processo de classificação é rápido em algoritmos *Multilayer Perceptron*, mais tempo é gasto geralmente no processo de treinamento. Já o tamanho do modelo tende a ser considerável pelo grande número de informações relacionadas aos neurônios da rede e suas conexões.

A melhor performance, em termos gerais, foi da Árvore de Decisão, que apresenta a maior acurácia balanceada, com o menor tempo de predição e consumo de energia, além do segundo menor tamanho de modelo. A árvore de decisão é um algoritmo com um modelo bastante simples, principalmente considerando as limitações impostas durante o processo de treinamento, além disso também tem um processo de classificação rápido, através da realização de uma série de comparações, que diminui com a profundidade da árvore.

A acurácia balanceada foi selecionada para comparações em geral por sumarizar bem a performance dos algoritmos com foco em evitar falsos negativos. Em termos gerais, a maior acurácia balanceada foi observada no algoritmo de Árvore de Decisão, seguido pelo *Random Forest*, com o *Naive Bayes* demonstrando a pior métrica, mas ainda no limite estabelecido de 0,6. Considerando apenas a acurácia balanceada, os algoritmos de Árvore de Decisão e *Random Forest* apresentaram a melhor performance, e considerando o tempo de predição, consumo de energia e tamanho do modelo, a Árvore de Decisão é o melhor algoritmo observado para implementação em um nó sensor. No entanto, o *Random Forest* pode ser considerado como uma alternativa robusta a ruídos de classificação e treinamento, os quais são comuns para aplicações de detecção de intrusão, principalmente em RSSF, além apresentarem maior robustez a *overfitting* (Folleco et al., 2008).

6.2 Performance de Classificação

Considerando os desafios provenientes de nós sensores (Vhatkar & Atique, 2013), a utilização dos algoritmos para detecção de intrusão no nível destes pode trazer diversos benefícios para o processo global de detecção de intrusão. Com a aplicação destas técnicas é possível identificar intrusões próximo ao ponto de ataque, tendo assim um tempo menor de detecção e aplicação de contramedidas. A utilização de técnicas de aprendizado de máquina também apresenta vantagens em relação a técnicas baseadas em assinatura pela sua capacidade de identificar ataques desconhecidos (Bace et al., 2001).

No entanto, as limitações existentes em nós sensores representam dificuldades no que tange a eficácia dos algoritmos, portanto, é importante a definição de limites de consumo de recursos, o que limita também a performance de classificação dos modelos. Com a aplicação de técnicas de classificação multiclasse, pode ocorrer ainda maior degradação da performance de classificação, especialmente das métricas principais selecionadas, como Acurácia Balanceada, que representa uma média dos *Recalls* de cada classe, e o *Recall* individual de cada classe. O objetivo da tarefa de detecção de intrusão multiclasse nesse contexto é a identificação de ameaças, tentando reduzir o número de falsos-negativos, e por tal motivo as principais métricas de performance têm relação com o *Recall*. Com a natureza desta aplicação, a performance aceitável de Acurácia Balanceada para este trabalho foi definida próxima a 0,6,

As Tabelas 6.3 e 6.4 apresentam os resultados obtidos da execução dos algoritmos para a extração das métricas de classificação. Estes são a média e desvio padrão de cada algoritmo selecionado, de 10 execuções de 10 Fold Cross Validation Stratified.

Tabela 6.3: Métricas de classificação 1. Fonte: O Autor.

	NB (ComplementNB)	**MLP (HL = 5, 5)**
Acurácia Balanceada	0,683 (0,007)	0,802 (0,066)
Acurácia	0,835 (0,002)	0,968 (0,007)
Precisão	0,946 (0,000)	0,971 (0,015)
Recall	0,835 (0,002)	0,968 (0,007)
F1 Score	0,879 (0,001)	0,965 (0,011)
Blackhole Recall	0,332 (0,012)	0,326 (0,076)
Flooding Recall	0,697 (0,024)	0,966 (0,139)
Grayhole Recall	0,798 (0,010)	0,834 (0,116)
Normal *Recall*	0,855 (0,002)	0,994 (0,003)
TDMA *Recall*	0,732 (0,018)	0,892 (0,093)

Tabela 6.4: Métricas de classificação 2. Fonte: O Autor.

	DT (MD = 6)	**RF (NT = 4, MD = 4)**
Acurácia Balanceada	0,884 (0,004)	0,817 (0,073)
Acurácia	0,981 (0,001)	0,970 (0,007)
Precisão	0,981 (0,001)	0,972 (0,008)
Recall	0,981 (0,001)	0,970 (0,007)
F1 Score	0,980 (0,001)	0,969 (0,009)
Blackhole Recall	0,826 (0,011)	0,678 (0,240)
Flooding Recall	0,998 (0,002)	0,785 (0,278)
Grayhole Recall	0,732 (0,012)	0,810 (0,052)
Normal *Recall*	0,999 (0,000)	0,990 (0,005)
TDMA *Recall*	0,867 (0,012)	0,823 (0,106)

Além da Acurácia Balanceada, também é importante considerar o *Recall* individual para cada classe, já que, para os ataques com menor representação na base, a Acurácia Balanceada ou outras medidas que lidam com médias de métricas do modelo podem não ser muito bem representativas. A Figura 6.7 apresenta o *Recall* de cada classe observada para cada algoritmo.

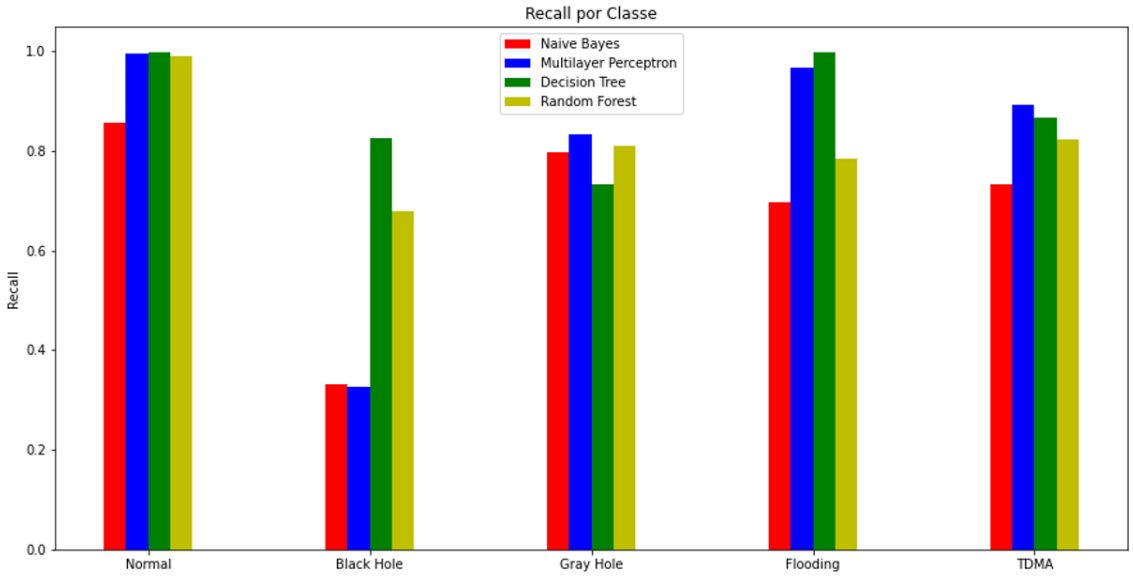

Figura 6.7: *Recall* por classe.

Considerando o *Recall* individual de cada tipo de ataque, observa-se que os algoritmos *Naive Bayes* e *Multilayer Perceptron*, que apresentam acurácia balanceada maior que 0,6, apresentam um *Recall* individual para ataques do tipo *Black Hole* menor que 0,4. Durante os testes, para definição dos parâmetros dos algoritmos, não foi possível obter um *Recall* aceitável para ataques *Black Hole*, e em alguns casos, TDMA e *Gray Hole*. Este fato ocorreu mesmo sem limitar os algoritmos em tamanho do modelo, isso se dá pela natureza altamente desbalanceada da base WSN-DS, que também é comum ao domínio de detecção de intrusão em redes de computadores.

Especificamente considerando a baixa performance de classificação de *Naive Bayes* e *Multilayer Perceptron* para ataques do tipo *Black Hole*, este pode ser atribuído as similaridades entre ataques *Black Hole* e *Gray Hole*, principalmente na sua implementação na base WSN-DS, na qual a única diferença na simulação destes dois ataques é a diferença nos números de pacotes enviados a estação base, descritos na Seção 5.1.2 (Pg. 53). Além disso, considerando que os nós responsáveis por realizarem transmissões a estação base são os CH, também é possível que ataques *Black Hole* sejam erroneamente classificados como tráfego normal, caso o modelo desenvolva algum viés quanto ao número de pacotes enviados a estação base.

A capacidade de lidar com conjuntos de dados desbalanceados pelos algoritmos de Árvore de Decisão e *Random Forest* é aparente observando os *Recalls* de cada classe. Estes apresentam boa performance em *Recall* para todas as classes, com o algoritmo de Árvore de Decisão obtendo melhor *recall* em 4 (quatro) das 5 (cinco) classes de ataques. A menor performance de classificação do *Random Forest*, em comparação com a Árvore de Decisão, se dá pelas limitações impostas pelos meio proposto (nós sensores), em situações sem limitação de recursos, o *Random Forest* apresenta performance superior à Árvore de Decisão para o mesmo conjunto de dados.

6.3 Avaliação das Métricas

Para avaliar as métricas de classificação extraídas foram aplicados testes estatísticos, visando verificar a existência de diferenças significativas entre as medidas. Foi considerado especificamente a Acurácia Balanceada, sendo esta a média dos *Recall*s das classes. O conjunto de dados para análise foi obtido através da aplicação de *10-Repeated 10-Fold Cross Validation Stratified*. Portanto, para cada algoritmo havia 10 médias do processo de *Cross Validation* para avaliação. Inicialmente foram aplicados testes de normalidade, para verificar qual dos testes de variância pré-selecionados deveria ser aplicado. Os testes de normalidade aplicados foram Shapiro-Wilk, D'Agostino K2 e Anderson-Darling. Para a aplicação de Shapiro-Wilk e D'Agostino K2 foi considerado um valor de significância de 95% ($a = 0,05$), portanto, caso o $p - valor > 0,05$, a amostra provavelmente faz parte de uma distribuição normal, enquanto, para o caso de um $p - valor <= 0,05$, a amostra provavelmente não provém de uma distribuição normal. No caso de Anderson-Darling, este provê um conjunto de valores críticos, ao invés de um $p - valor$. Estes valores representam limites de significância pré-definidos, nos quais o teste é aplicado podendo aceitar ou rejeitar a hipótese sendo testada. Através da aplicação destes testes, o conjunto de resultados, com as 40 medidas de Acurácia Balanceada, provavelmente não faz parte de uma distribuição normal, a avaliação para cada teste de normalidade foi:

- **Shapiro-Wilk:** Normalidade não confirmada ($Stat = 0,835$, $p = 0,0000388327$);
- **D'Agostino K2:** Normalidade não confirmada ($Stat = 6,767$, $p = 0,0339331021$);
- **Anderson-Darling:** Normalidade não confirmada ($Stat : 2,417$);

Considerando a não normalidade dos dados, o teste de variância aplicado foi de Kruskal-Wallis, um teste não paramétrico utilizado para a comparação de duas (2) ou mais amostras não pareadas. As quatro (4) amostras foram compostas por 10 (dez) valores de Acurácia Balanceada de cada algoritmo testado, com um valor de significância de 95% ($a = 0,05$), portanto, caso seja observado um $p - valor > 0,05$, provavelmente as a mostras não apresentam diferenças estatisticamente significativas, enquanto, para o caso de um $p - valor <= 0,05$, as amostras provavelmente apresentam diferenças estatisticamente significativas. Após a aplicação do teste de Kruskal-Wallis, foi observado que havia diferenças significativas entre as amostras ($Stat = 34,071$, $p = 0,0000001914$).

Para verificar quais amostras específicas apresentavam diferenças significativas, foi aplicado o Pós-Teste de Dunn, o teste recomendado para comparações par a par após a aplicação de Kruskal-Wallis. Este foi aplicado com um valor de significância de 95% ($a = 0,05$), portanto, caso seja observado um $p - valor > 0,05$, provavelmente as a mostras não apresentam diferenças significativas, enquanto, para o caso de um $p - valor <= 0,05$, as amostras provavelmente apresentam diferenças significativas. A correção de Bonferroni foi utilizada para controlar a taxa de erro familiar. Com a aplicação do pós-teste foram observadas diferenças significativas

entre o algoritmo de Árvore de Decisão e os algoritmos *Naive Bayes* e *Multilayer Perceptron*, além de diferenças entre o algoritmo de *Random Forest* e o algoritmo *Naive Bayes*. A Tabela 6.5 apresenta os $p - valor$ observados na aplicação do pós-teste, e os valores médios de Acurácia Balanceada dos algoritmos comparados, as comparações com diferenças significativas são denotadas por com um asterisco (*) na coluna do p-valor.

Tabela 6.5: Aplicação do pós-teste de Dunn. Fonte: O Autor.

	Acurácia Balanceada	**p-Valor**
***Naive Bayes* – MLP**	0,683 (0,007) – 0,802 (0,066)	0,053751
Naive Bayes – Decision Tree	0,683 (0,007) – 0,884 (0,004)	0,00000003791622*
Naive Bayes – Random Forest	0,683 (0,007) – 0,817 (0,073)	0,008406*
MLP – *Decision Tree*	0,802 (0,066) – 0,884 (0,004)	0,008406*
MLP – *Random Forest*	0,802 (0,066) – 0,817 (0,073)	1,000000
Decision Tree – Random Forest	0,884 (0,004) – 0,817 (0,073)	0,053751

De modo geral, considerando o consumo de recursos e performance de classificação dos algoritmos, observa-se que a Árvore de Decisão é o melhor candidato para aplicação em nós sensores. O *Random Forest* também pode ser considerado em dispositivos com maior disponibilidade de recursos.

Durante os testes iniciais com os algoritmos para definição da base, descritos na Seção 5.3.1 (Pg. 60), observou-se a performance destes com o aumento da disponibilidade dos recursos. Neste caso, comparando Árvore de Decisão e *Random Forest*, o segundo apresentou ganhos mais significativos para todas as métricas de classificação utilizadas com o aumento do número e profundidade das árvores.

Capítulo 7

Conclusão

Neste trabalho foram utilizados algoritmos de aprendizado de máquina com o propósito de realizar detecção de intrusão para múltiplas ameaças ao nível de nós sensores de Redes de Sensores Sem Fio (RSSF). Foram utilizados especificamente os algoritmos *Naive Bayes, Multilayer Perceptron*, Árvore de decisão e *Random Forest*, os quais foram ajustados considerando métricas de execução para que sua aplicação em nós sensores fosse viável, visando um equilíbrio entre consumo de recursos e performance de classificação.

Para cada algoritmo foi definido um conjunto de parâmetros que apresentava performance de classificação aceitável, em termos de *recall* de cada classe, e que se mantivesse em um intervalo aceitável em relação ao tamanho do modelo. Estes testes foram realizados utilizando a base de dados WSN-DS após a realização de um processo de seleção de atributos que removeu certos atributos que apresentavam impactos negativos a performance de classificação.

Em relação a performance de execução dos modelos observou-se que um baixo consumo de energia dos algoritmos *Multilayer Perceptron* e Árvore de Decisão, e consumo elevado do algoritmo *Random Forest*. Essas métricas apresentam relação com o tempo de predição, que também é mais alto no *Random Forest*. Em relação ao tamanho do modelo, o *Random Forest* também apresentou o maior tamanho, seguido do *Multilayer Perceptron* e Árvore de Decisão, com *Naive Bayes* apresentando um modelo significantemente menor do que os demais.

Neste contexto, também foi considerado o desempenho da classificação dos modelos, sendo que os resultados foram comparados através de testes estatísticos em termos da Acurácia Balanceada, além de observações em termos do *Recall* individual de cada classe. Nos testes estatísticos, foram observadas diferenças significativas entre a performance da Árvore de Decisão (0,884) x *Naive Bayes* (0,683) e *Multilayer Perceptron* (0,802), e entre *Random Forest* (0,817) x *Naive Bayes* (0,683). Nota-se então a melhor performance de Acurácia Balanceada dos modelos de Árvore de Decisão e *Random Forest*. Através da observação do *Recall* de cada classe para os modelos, também fica aparente a capacidade que Árvore de Decisão e *Random Forest* possuem de lidar com bases altamente desbalanceadas, mantendo um *Recall* aceitável, ao contrário de *Naive Bayes* e *Multilayer Perceptron*, que apresentam um *Recall* baixo para ataques do tipo *Black Hole*.

Considerando a performance e consumo de recursos dos modelos, observa-se que o mo-

delo de Árvore de Decisão é o mais indicado para a aplicação, já que devido à estrutura do *Random Forest*, este apresenta um nível de consumo de recursos mais elevado, sem apresentar ganho de performance de classificação significativo. Com maior disponibilidade de recursos, o *Random Forest* deve apresentar performance de classificação consideravelmente melhor do que a Árvore de Decisão. O *Random Forest* também é robusto a ruído de classificação e *overfitting*, o que se demonstra importante em aplicações em RSSF, então este pode ser considerado se houver disponibilidade de recursos adequada.

A partir dos resultados obtidos e observações realizadas durante a execução deste trabalho, foram considerados possíveis futuros trabalhos:

- Realizar comparações entre os melhores algoritmos observados neste trabalho e outros algoritmos não testados;
- Adaptar e aplicar os modelos observados em um ambiente simulado de RSSF ou em hardware real de nós sensores;
- Realizar experimentos com técnicas específicas de pré-processamento da base WSN-DS, para lidar com desbalanceamento de classes;
- Implementar o método proposto neste trabalho como parte de um IDS de RSSF de múltiplos níveis.

Referências Bibliográficas

Almomani, I. & Al-Kasasbeh, B. (2015). Performance analysis of leach protocol under denial of service attacks, *2015 6th International Conference on Information and Communication Systems (ICICS)*, IEEE, pp. 292–297. Citado na página 40.

Almomani, I., Al-Kasasbeh, B. & Al-Akhras, M. (2016). Wsn-ds: A dataset for intrusion detection systems in wireless sensor networks, *Journal of Sensors* **2016**. Citado 6 vezes nas páginas 39, 40, 50, 51, 53 e 56.

Alpaydin, E. (2020). *Introduction to machine learning*, MIT press. Citado 3 vezes nas páginas 29, 30 e 31.

Alqahtani, M., Gumaei, A., Mathkour, H. & Maher Ben Ismail, M. (2019). A genetic-based extreme gradient boosting model for detecting intrusions in wireless sensor networks, *Sensors* **19**(20): 4383. Citado na página 45.

Alruhaily, N. M. & Ibrahim, D. M. (2019). A multi-layer machine learning-based intrusion detection system for wireless sensor networks. Citado na página 47.

Amor, N. B., Benferhat, S. & Elouedi, Z. (2004). Naive bayes vs decision trees in intrusion detection systems, *Proceedings of the 2004 ACM symposium on Applied computing*, pp. 420–424. Citado na página 51.

Anderson, T. W. & Darling, D. A. (1954). A test of goodness of fit, *Journal of the American statistical association* **49**(268): 765–769. Citado na página 65.

Atzori, L., Iera, A. & Morabito, G. (2017). Understanding the internet of things: definition, potentials, and societal role of a fast evolving paradigm, *Ad Hoc Networks* **56**: 122–140. Citado na página 14.

Azevedo, A. I. R. L. & Santos, M. F. (2008). Kdd, semma and crisp-dm: a parallel overview, *IADS-DM* . Citado 2 vezes nas páginas 26 e 27.

Bace, R. G., Mell, P. et al. (2001). Intrusion detection systems. Citado 6 vezes nas páginas 14, 21, 22, 23, 24 e 72.

Batiha, T. & Krömer, P. (2020). Evolutionary fuzzy rules for intrusion detection in wireless sensor networks, *International Conference on Intelligent Networking and Collaborative Systems*, Springer, pp. 149–160. Citado na página 46.

Batiha, T., Prauzek, M. & Krömer, P. (2019). Intrusion detection in wireless sensor networks by an ensemble of artificial, *Intelligent Decision Technologies 2019: Proceedings of the 11th KES International Conference on Intelligent Decision Technologies (KES-IDT 2019), Volume 1*, Vol. 142, Springer, p. 323. Citado 2 vezes nas páginas 44 e 56.

Borkar, G. M., Patil, L. H., Dalgade, D. & Hutke, A. (2019). A novel clustering approach and adaptive svm classifier for intrusion detection in wsn: A data mining concept, *Sustainable Computing: Informatics and Systems* **23**: 120–135. Citado na página 44.

Breiman, L. (1996). Some properties of splitting criteria, *Machine Learning* **24**(1): 41–47.

Citado na página 30.

Breiman, L. (2001). Random forests, *Machine learning* **45**(1): 5–32. Citado na página 35.

Campello, R. S. & Weber, R. F. (2001). Sistemas de detecçao de intrusao, *Minicurso procedente do 19º Simpósio Brasileiro de Redes de Computadores* . Citado na página 22.

D'agostino, R. B., Belanger, A. & D'Agostino Jr, R. B. (1990). A suggestion for using powerful and informative tests of normality, *The American Statistician* **44**(4): 316–321. Citado na página 65.

Dargie, W. & Poellabauer, C. (2010). *Fundamentals of wireless sensor networks: theory and practice*, John Wiley & Sons. Citado 4 vezes nas páginas 17, 18, 19 e 20.

Dhanabal, L. & Shantharajah, S. (2015). A study on nsl-kdd dataset for intrusion detection system based on classification algorithms, *International journal of advanced research in computer and communication engineering* **4**(6): 446–452. Citado na página 56.

Dietterich, T. G. (2000). Ensemble methods in machine learning, *International workshop on multiple classifier systems*, Springer, pp. 1–15. Citado na página 33.

Ding, Z., Fei, M., Du, D. & Xu, S. (2014). Online anomaly detection method based on bbo ensemble pruning in wireless sensor networks, *Life System Modeling and Simulation*, Springer, pp. 160–169. Citado na página 47.

Dong, R.-H., Yan, H.-H. & Zhang, Q.-Y. (2020). An intrusion detection model for wireless sensor network based on information gain ratio and bagging algorithm., *Int. J. Netw. Secur.* **22**(2): 218–230. Citado na página 45.

Dunn, O. J. (1961). Multiple comparisons among means, *Journal of the American statistical association* **56**(293): 52–64. Citado na página 66.

El Mourabit, Y., Bouirden, A., Toumanari, A., Moussaid, N. et al. (2015). Intrusion detection techniques in wireless sensor network using data mining algorithms: comparative evaluation based on attacks detection, *International Journal of Advanced Computer Science and Applications* **6**(9): 164–172. Citado na página 42.

Elmrabit, N., Zhou, F., Li, F. & Zhou, H. (2020). Evaluation of machine learning algorithms for anomaly detection, *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, IEEE, pp. 1–8. Citado na página 51.

Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P. (1996). From data mining to knowledge discovery in databases, *AI magazine* **17**(3): 37–37. Citado 2 vezes nas páginas 26 e 27.

Fisher, R. (1954). The analysis of variance with various binomial transformations, *Biometrics* **10**(1): 130–139. Citado na página 65.

Folleco, A., Khoshgoftaar, T. M., Van Hulse, J. & Bullard, L. (2008). Software quality modeling: The impact of class noise on the random forest classifier, *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, IEEE, pp. 3853–3859. Citado 2 vezes nas páginas 51 e 72.

Galar, M., Fernández, A., Barrenechea, E., Bustince, H. & Herrera, F. (2011). An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes, *Pattern Recognition* **44**(8): 1761–1776. Citado na página 33.

Garofalo, A., Di Sarno, C. & Formicola, V. (2013). Enhancing intrusion detection in wireless

sensor networks through decision trees, *European Workshop on Dependable Computing*, Springer, pp. 1–15. Citado na página 43.

Hac, A. (2003). *Wireless sensor network designs*, Citeseer. Citado 3 vezes nas páginas 37, 40 e 41.

Hall, M. A. (2000). Correlation-based feature selection of discrete and numeric class machine learning. Citado 2 vezes nas páginas 57 e 59.

Heinzelman, W. B., Chandrakasan, A. P. & Balakrishnan, H. (2002). An application-specific protocol architecture for wireless microsensor networks, *IEEE Transactions on wireless communications* **1**(4): 660–670. Citado 2 vezes nas páginas 37 e 38.

Heinzelman, W. R., Chandrakasan, A. & Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks, *Proceedings of the 33rd annual Hawaii international conference on system sciences*, IEEE, pp. 10–pp. Citado 2 vezes nas páginas 37 e 38.

Hossin, M. & Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations, *International journal of data mining & knowledge management process* **5**(2): 1. Citado na página 54.

Jaladi, A. R., Khithani, K., Pawar, P., Malvi, K. & Sahoo, G. (2017). Environmental monitoring using wireless sensor networks (wsn) based on iot, *Int. Res. J. Eng. Technol* **4**(1): 1371–1378. Citado na página 54.

Jiang, S., Zhao, J. & Xu, X. (2020). Slgbm: An intrusion detection mechanism for wireless sensor networks in smart environments, *IEEE Access* **8**: 169548–169558. Citado na página 45.

Kavitha, T. & Sridharan, D. (2010). Security vulnerabilities in wireless sensor networks: A survey, *Journal of information Assurance and Security* **5**(1): 31–44. Citado na página 14.

Kruskal, W. H. & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis, *Journal of the American statistical Association* **47**(260): 583–621. Citado na página 66.

Magotra, S. & Kumar, K. (2014). Detection of hello flood attack on leach protocol, *2014 IEEE International Advance Computing Conference (IACC)*, IEEE, pp. 193–198. Citado na página 39.

Mitchell, T. M. (1997). *Machine Learning*, McGraw-Hill. Citado 5 vezes nas páginas 29, 30, 31, 32 e 33.

Otoum, S., Kantarci, B. & Mouftah, H. T. (2020). A novel ensemble method for advanced intrusion detection in wireless sensor networks, *Icc 2020-2020 ieee international conference on communications (icc)*, IEEE, pp. 1–6. Citado na página 46.

Pan, J.-S., Fan, F., Chu, S.-C., Zhao, H.-Q. & Liu, G.-Y. (2021). A lightweight intelligent intrusion detection model for wireless sensor networks, *Security and Communication Networks* **2021**. Citado na página 43.

Park, T., Cho, D., Kim, H. et al. (2018). An effective classification for dos attacks in wireless sensor networks, *2018 Tenth international conference on ubiquitous and future networks (ICUFN)*, IEEE, pp. 689–692. Citado na página 46.

Pathan, A.-S. K., Lee, H.-W. & Hong, C. S. (2006). Security in wireless sensor networks: issues and challenges, *2006 8th International Conference Advanced Communication Technology*,

Vol. 2, IEEE, pp. 6–pp. Citado 3 vezes nas páginas 38, 39 e 41.

Polikar, R. (2006). Ensemble based systems in decision making, *IEEE Circuits and systems magazine* **6**(3): 21–45. Citado 2 vezes nas páginas 34 e 35.

Quinlan, J. R. (1986). Induction of decision trees, *Machine learning* **1**(1): 81–106. Citado 2 vezes nas páginas 57 e 59.

Roosta, T., Shieh, S. & Sastry, S. (2006). Taxonomy of security attacks in sensor networks and countermeasures, *The first IEEE international conference on system integration and reliability improvements*, Vol. 25, Citeseer, p. 94. Citado na página 39.

Russell, S. & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*, 3 edn, Prentice Hall. Citado 3 vezes nas páginas 25, 29 e 32.

Saeed, A., Ahmadinia, A., Javed, A. & Larijani, H. (2016). Random neural network based intelligent intrusion detection for wireless sensor networks, *Procedia Computer Science* **80**: 2372–2376. Citado na página 44.

Shapiro, S. S. & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples), *Biometrika* **52**(3/4): 591–611. Citado na página 65.

Singh, N., Virmani, D. & Gao, X.-Z. (2020). A fuzzy logic-based method to avert intrusions in wireless sensor networks using wsn-ds dataset, *International Journal of Computational Intelligence and Applications* **19**(03): 2050018. Citado na página 44.

Tavallaee, M., Bagheri, E., Lu, W. & Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set, *2009 IEEE symposium on computational intelligence for security and defense applications*, IEEE, pp. 1–6. Citado na página 56.

Total de Incidentes Reportados ao CERT.br por Ano (2020).
URL: *https://www.cert.br/stats/incidentes/* Citado na página 21.

Tripathi, M., Gaur, M. S. & Laxmi, V. (2013). Comparing the impact of black hole and gray hole attack on leach in wsn, *Procedia Computer Science* **19**: 1101–1107. Citado na página 39.

Turing, A. (1950). Mind, *Mind* **59**(236): 433–460. Citado 2 vezes nas páginas 25 e 26.

Uppuluri, P. & Sekar, R. (2001). Experiences with specification-based intrusion detection, *International Workshop on Recent Advances in Intrusion Detection*, Springer, pp. 172–189. Citado na página 24.

Vhatkar, S. & Atique, M. (2013). Design issues, characteristics and challenges in routing protocols for wireless sensor networks, *International Journal of Computer Applications* **975**: 8887. Citado 3 vezes nas páginas 36, 37 e 72.

Vieira, M. A. M., Coelho, C. N., da Silva, D. j. & da Mata, J. M. (2003). Survey on wireless sensor network devices, *EFTA 2003. 2003 IEEE Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No. 03TH8696)*, Vol. 1, IEEE, pp. 537–544. Citado na página 54.

Vujović, V. & Maksimović, M. (2015). Raspberry pi as a sensor web node for home automation, *Computers & Electrical Engineering* **44**: 153–171. Citado na página 54.

Wang, F., Huang, C., Zhao, J. & Rong, C. (2008). Idmtm: A novel intrusion detection mechanism based on trust model for ad hoc networks, *22nd International Conference on Advanced Information Networking and Applications (aina 2008)*, IEEE, pp. 978–984. Citado na

página 24.

Wang, L., Li, J., Cheng, J., Bhatti, U. A. & Dai, Q. (2018). Dos attacks intrusion detection algorithm based on support vector machine, *International Conference on Cloud Computing and Security*, Springer, pp. 286–297. Citado na página 47.

Werner-Allen, G., Lorincz, K., Ruiz, M., Marcillo, O., Johnson, J., Lees, J. & Welsh, M. (2006). Deploying a wireless sensor network on an active volcano, *IEEE internet computing* **10**(2): 18–25. Citado na página 20.

Witten, I. H., Frank, E. & Hall, M. A. (2017). *Data Mining: Practical Machine Learning Tools and Techniques*, 4 edn, Morgan Kaufmann. Citado 2 vezes nas páginas 26 e 28.

Yahyaoui, A., Abdellatif, T. & Attia, R. (2019). Hierarchical anomaly based intrusion detection and localization in iot, *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, IEEE, pp. 108–113. Citado na página 47.