UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Uma abordagem *Ensemble Learning* para modelos de detecção de intrusão para redes industriais

Nicole do Vale Dalarmelina

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Nicole do Vale Dalarmelina

Uma abordagem *Ensemble Learning* para modelos de detecção de intrusão para redes industriais

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestra em Ciências – Ciências de Computação e Matemática Computacional. *EXEMPLAR DE DEFESA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Rodolfo Ipolito Meneguette

USP – São Carlos
Agosto de 2023

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Dalarmelina, Nicole do Vale
D136a Uma abordagem Ensemble Learning para modelos de detecção de intrusão para redes industriais / Nicole do Vale Dalarmelina; orientador Rodolfo Ipolito Meneguette. -- São Carlos, 2023.
62 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2023.

1. Ensemble Learning. 2. IIoT. 3. IDS. 4. Machine Learning. I. Meneguette, Rodolfo Ipolito, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Nicole do Vale Dalarmelina

An Ensemble Learning approach for intrusion detection models for industrial networks

Dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Master in Science. *EXAMINATION BOARD PRESENTATION COPY*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Rodolfo Ipolito Meneguette

USP – São Carlos
August 2023

AGRADECIMENTOS

Os principais agradecimentos são direcionados ao professor doutor Rodolfo Ipolito Meneguette, meu orientador, e ao professor doutor Márcio Andrey Teixeira pelos aconselhamentos assertivos e estímulo permanente, que contribuíram fortemente para a evolução do presente trabalho.

RESUMO

DALARMELINA, N. V. **Uma abordagem *Ensemble Learning* para modelos de detecção de intrusão para redes industriais**. 2023. 62 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

A *Internet* tem se tornado um recurso essencial para a humanidade e para os dispositivos tecnológicos existentes atualmente, tanto dentro de casa – *Internet of Things* (IoT) – quanto dentro de industrias – *Industrial Internet of Things* (IIoT). Todo esse avanço tecnológico pode trazer benefícios, mas também pode oferecer riscos à integridade dos dados se a segurança não for devidamente realizada utilizando Sistemas de Detecção de Intrusão (IDS) eficientes. Neste trabalho é proposto um modelo que poderá ser utilizado por IDSs para redes industriais utilizando *Ensemble Learning*. Para isso são analisadas abordagens para a extração das melhores *features* dos *datasets* utilizados, assim como a aplicação de algoritmos de balanceamento de dados a fim de selecionar as melhores abordagens para o treinamento do modelo proposto viabilizando possíveis retreinamentos do modelo a cada novo ataque encontrado, o modelo desenvolvido no presente trabalho obteve acurácia de 99.93%, concluindo seu treinamento em apenas 1 hora e 34 minutos, enquanto o modelo treinado utilizando os *datasets* sem tratamento obteve acurácia de 99.94% concluindo seu treinamento em 156 horas.

Palavras-chave: Aprendizado em conjunto, IIoT, IDS, Aprendizado de máquina.

ABSTRACT

DALARMELINA, N. V. **An Ensemble Learning approach for intrusion detection models for industrial networks**. 2023. 62 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

The Internet has been turned into an essential resource to humanity and to currently existing technological devices, both indoors – Internet of Things (IoT) – and in industrial environments – Industrial Internet of Things (IIoT). All this progress can bring benefits, yet it can also bring risks to the data integrity if the security has not been properly performed by using effective Intrusion Detection Systems (IDS). In this work it is proposed the training of a model that can be used by IDS industrial networks using Ensemble Learning. For this intent, approaches for extract the best features of the used datasets are analyzed, as well as the use of data balancing algorithms in order to select the best approaches for training the proposed model, enabling possible retraining of the model for each new found attack, the model developed in the present work obtained an accuracy of 99.93%, completing its training in just 1 hour and 34 minutes, while the model trained using the *datasets* without treatment obtained an accuracy of 99.94%, concluding its training in 156 hours.

Keywords: Ensemble Learning, IIoT, IDS, Machine Learning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

A *Internet* tem se tornado um recurso essencial para a humanidade, segundo o trabalho em Petrosyan (2023), cerca de 64% da população mundial faz uso dessa tecnologia. Além disso, não só humanos a utilizam, diversos dispositivos eletrônicos atualmente também se mantêm conectados, como TVs inteligentes, relógios, casas e, até mesmo, automóveis. Assim como para os humanos, a *Internet* passou a ser essencial para esses dispositivos, e essa tecnologia ganhou o nome de *Internet* das Coisas (do inglês, *Internet of Things* - IoT). Segundo estudos aprensentados em Galov (2023), havia aproximadamente 42 bilhões de dispositivos IoT conectados em 2022.

Seguindo a mesma tendência, dispositivos industriais também estão avançando cada vez mais em tecnologia e conectividade, fazendo com que uma nova tecnologia apareça, a *Internet* das Coisas Industrial (do inglês, *Industrial Internet of Things* - IIoT), que permite que máquinas e outros dispositivos industriais possam enviar e receber informações via *Internet*. Essa conexão facilita a implementação de melhorias em questões de eficiência e produtividade no ambiente industrial. Tecnologias IIoT estão sendo utilizadas em infraestruturas críticas, como indústrias petroquímicas, controle de abastecimento de água, entre outras. Os sistemas que controlam esses ambientes são chamados de Sistemas de Controle Industrial (do inglês, *Industry Control Systems* - ICS) e estão em constante evolução para acompanhar as mais avançadas tecnologias.

Todo esse avanço e conectividade, que proporciona diversas facilidades e benefícios, também pode acarretar danos e problemas variados se não for bem utilizado. A partir do momento em que um dispositivo é conectado à *Internet*, se torna um potencial alvo de ataques cibernéticos (ISO/IEC 27000:2018(en), 2018) que podem resultar em grandes prejuízos, como roubo de dados, alteração e exposição de informações confidenciais, entre outros problemas. Existe uma área específica para evitar esse tipo de exposição e perda de dados, chamada de *Cyber* segurança. Essa área se preocupa em garantir a segurança em diferentes dispositivos que se conectam à *Internet*, como *smartphones*, computadores, dispositivos de comunicação em geral e, também, dispositivos do âmbito industrial.

Inúmeras vulnerabilidades estão sendo descobertas em sistemas computacionais a cada ano, e, além isso, o número de invasões e ataques cibernético têm crescido exponencialmente. Para tentar conter essa epidemia cibernética, muitos Sistemas de Detecção de Intrusão (do inglês, *Intrusion Detection Systems* - IDS) vêm sido desenvolvidos, dentre os tipos existentes, vale ressaltar IDSs baseados em rede. IDSs baseados em rede são utilizados para analisar o tráfego e detectar possíveis atividades maliciosas. Para descrever como esses sistemas funcionam, antes, é preciso discorrer um pouco sobre o funcionamento do tráfego de rede. O tráfego de uma rede é constituído por fluxos, que são sequências unidirecionais de pacotes entre *hosts*. Esses fluxos são compostos por informações, como tipos de protocolo, endereço de IP e tamanho do pacote.

Quando uma rede está sendo atacada, suas *features* sofrem alterações de acordo com cada tipo de ataque, como apresentado em Teixeira *et al.* (2018) com ataques reais. Exemplificando, um ataque de reconhecimento apresenta pequenas oscilações no número de pacotes enviados na rede, enquanto um ataque de negação de serviço (do inglês, *Denial of Service* - DoS) apresenta grandes oscilações no tráfego.

Para detectar os padrões de ataques, algoritmos de Aprendizado de Máquina (do inglês, *Machine Learning* - ML) e Aprendizado Profundo (do inglês, *Deep Learning* - DL) podem ser utilizados para gerar um modelo de detecção utilizando *datasets* de tráfegos de rede, além disso, é possível utilizar técnicas de aprendizagem por agrupamento (do inglês, *Ensemble Learning*, que é um tipo de ML, para gerar um modelo ainda mais robusto. Há diversos *datasets* na literatura que podem ser utilizados para treinar modelos de ML e DL, como o Bot-IoT (KORONIOTIS; MOUSTAFA; SITNIKOVA, 2018) e NF-UNSW-NB15-v2 (SARHAN; LAYEGHY; PORTMANN, 2022), porém, é preciso identificar as melhores *features* que compõem os *datasets* para serem utilizadas no treinamento do modelo de identificação de intrusão. Pensando nisso, o presente trabalho tem como objetivo desenvolver um modelo baseado em ML, onde serão exploradas várias abordagens utilizadas no desenvolvimento de um modelo para IDSs, tendo em vista definir uma metodologia de desenvolvimento que possa ser utilizada para criar um modelo de detecção de intrusão efetivo para ser utilizado em redes industriais.

1.1 Definição do problema

Pensando na performance de treinamento de modelos utilizando ML, utilizar bases de dados que contenham *features* "desnecessárias"para determinado ataque podem apresentar críticos desafios na modelagem de dados, como apresentado em Ambusaidi *et al.* (2016), além disso, é preciso se preocupar com o balanceamento do *dataset* utilizado para treinamento, já que um *dataset* desbalanceado pode gerar modelos enviesados, como descrito em Zolanvari, Teixeira e Jain (2018). Sendo assim, a o presente trabalho consiste em selecionar as *features* mais "importantes"para determinados ataques e realizar experimentos de balanceamento dos *datasets* utilizados para responder a seguinte questão de pesquisa: “*Como desenvolver um modelo eficaz e*

robusto que possa ser utilizado por um Sistema de Detecção de Intrusão para Redes Industriais utilizando Ensemble Learning?".

1.2 Objetivos

O presente trabalho tem como principal objetivo desenvolver um modelo para detecção de intrusão em redes industriais utilizando *Ensemble Learning* (ZHANG; MA, 2012) para a obtenção de performances satisfatórias em termos de alta taxa de acertos em predições e tempo de aprendizagem, utilizando *datasets* presentes em Zolanvari, Teixeira e Jain (2018) que contêm um grande número de ataques reais coletados em uma rede industrial de controle de um tanque de armazenamento de água, onde o tratamento e distribuição de água são realizados. Ainda será abordado o impacto que um treinamento realizado com um *dataset* desbalanceado pode gerar nos resultados de treinamento do modelo em relação à falsos negativos, acurácia, entre outras métricas. Como objetivos específicos, o presente trabalho pretende abordar os seguintes tópicos:

- Selecionar as melhores *features* de cada ataque.
- Realizar testes de predição com diferentes bases de dados.
- Balancear os *datasets* utilizados e verificar se há um viés no treinamento do modelo desenvolvido.
- Selecionar a melhor abordagem em termos de: (a) *features*; (b) base de dados. Na qual serão selecionados:
  - (a) *features* específicas de cada ataque ou *features* generalizadas para abranger diversos ataques, conforme melhor performance nas avaliações.
  - (b) base de dados balanceada ou desbalanceada, conforme a melhor performance nas avaliações.
- Utilizar *Ensemble Learning* para desenvolver um modelo para IDSs para detectar anomalias em uma rede industrial.

1.3 Organização do texto

No Capítulo 2 é abordada a fundamentação teórica, onde as tecnologias utilizadas são descritas, assim como os tipos de IDS, tipos de detecção e tipos de ataques que serão explorados no presente trabalho. Os trabalhos que possuem alguma relação com o presente projeto são descritos no Capítulo 3, sendo apontadas as características de cada trabalho, assim como a comparação da presente proposta com o estado da arte.

No Capítulo 4 é apresentada a proposta para o treinamento do modelo utilizando ML, tal como os meios e método que serão utilizados para o desenvolvimento deste trabalho. Os resultados obtidos são descritos no Capítulo 5, bem como as avaliações de seus desempenhos. Por fim, no Capítulo 6, é discorrido sobre os resultados obtidos e desde o início deste trabalho, assim como as contribuições da autora.

CAPÍTULO

2

FUNDAMENTAÇÃO TEÓRICA

Existem inúmeros tipos de ataques computacionais, para o melhor entendimento do que eles são e como se comportam, neste capítulo serão descritos alguns dos ataques mais comuns em redes de computadores, bem como os tipos de IDSs existentes e tipos de detecção, a fim de discorrer sobre os fundamentos teóricos necessários para a implementação da presente proposta e as tecnologias utilizadas.

2.1 Tipos de ataques em redes de computadores

Antes de atacar, *cyber* criminosos precisam estudar seus alvos e entender suas vulnerabilidades, portanto, o ataque de reconhecimento, geralmente, é o primeiro estágio de um ataque. Para isso, ferramentas de varredura de rede são comumente utilizadas a fim de obter informações relevantes sobre o alvo, como a lista de dispositivos conectados à rede, tipologia e possíveis vulnerabilidades. O ataque de reconhecimento, de maneira geral, é utilizado para estudar as melhores formas de atacar um sistema alvo, como descrito em Mazurczyk e Caviglione (2021), coletando detalhes sobre o local físico das vítimas, testando portas abertas e monitorando as atividades que são normalmente realizadas, fornecendo informações valiosas para *cyber* criminosos agirem no momento e lugar de maior vulnerabilidade do sistema.

Conhecendo as vulnerabilidades do alvo e possuindo acesso à rede da vítima, *cyber* criminosos podem optar por ferir um dos pilares da segurança da informação, são eles: confidencialidade, que deve garantir que somente pessoas e máquinas autorizadas tenham acesso a determinado sistema ou serviço; integridade, que deve garantir que informações contidas na rede não tenham sido adulteradas por atividades maliciosas; e disponibilidade, que deve garantir que usuários e máquinas tenham acesso aos recursos necessários a todo momento.

Para ferir um desses pilares, *cyber* criminosos podem realizar o ataque de negação de serviço (do inglês, *Denial of Service* – DoS). Esse tipo de ataque é realizado contra sistemas e

websites há muitos anos e são um grande problema, como discorrido em Tripathi e Hubballi (2021), uma vez que quebra um dos pilares da *cyber* segurança, a disponibilidade, tornando sistemas ou serviços inacessíveis para usuários autorizados. Para que o objetivo seja alcançado, algumas técnicas, como sobrecarga de rede, múltiplas requisições e quebra de conexão entre computadores, são utilizadas, como apresentado em Murini (2014).

Uma variação do ataque DoS é a negação de serviço distribuída (do inglês, *Distributed Denial of Service* – DDoS), esse ataque consiste em infectar diversos computadores, para que estes possam "obedecer" às ordens de um computador mestre. Dessa maneira, todas as máquinas infectadas podem realizar certa requisição a determinado serviço simultaneamente, como descrito em Zargar, Baghaie *et al.* (2012), o que pode esgotar recursos do servidor, fazendo com que este não seja capaz de atender a mais nenhuma solicitação, quebrando o pilar da disponibilidade da segurança da informação. A fim de identificar e bloquear esse tipo de intrusão em sistemas industriais, foram desenvolvidas tecnologias de detecção de intrusão específicas para cada cenário, como a detecção em *host* e a detecção na rede.

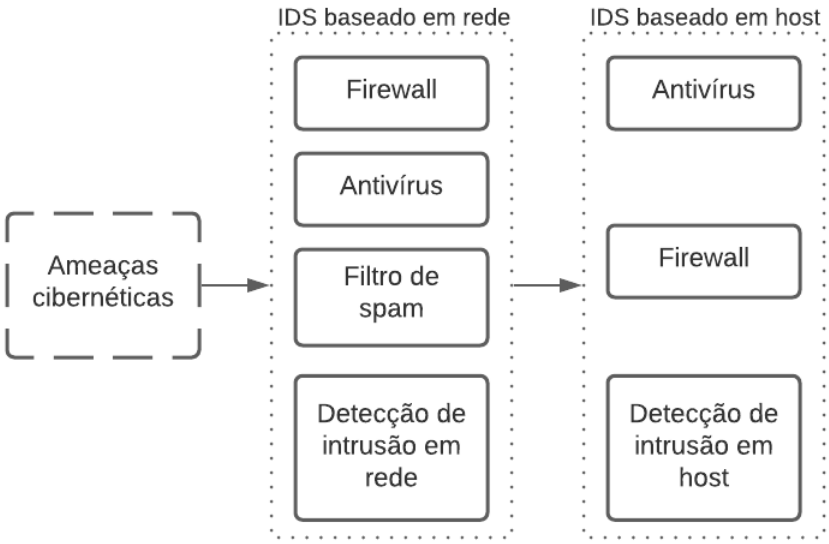
2.2 Sistemas de detecção de intrusão

Sistemas de Detecção de Intrusão (do inglês, *Intrusion Detection System* - IDS), como o nome já sugere, trata-se de sistemas desenvolvidos para detectar atividades maliciosas em determinados cenários. Existem diferentes tipos de IDSs, cada um para um ambiente específico, porém com o mesmo propósito, detectar e evitar intrusões no sistema. Dentre os tipos de IDSs, neste trabalho serão descritas as características de IDSs baseados em *host* (do inglês, *Host Intrusion Detection System* - HIDS) e de IDSs baseados em rede (do inglês, *Network Intrusion Detection System* - NIDS), ilustrados na Figura 1.

HIDSs são sistemas de detecção de intrusão de host, como apresentado em Jose *et al.* (2018), nessa abordagem o principal interesse são os sistemas locais, como computadores e sistemas operacionais, e o objetivo é monitorar estações de trabalho dentro do sistema, garantindo a segurança dentro de cada uma delas. Esse tipo de sistema não é centralizado, pois cada estação de trabalho deve possuir um HIDS que consumirá recursos para controlar todo o comportamento do *host*. Desta forma, o HIDS consegue identificar potenciais atividades maliciosas, ou seja, atividades que fogem do esperado naquele determinado local, como a alteração ou exclusão de um arquivo, por exemplo.

Diferente dos HIDSs, NIDSs são sistemas de detecção de intrusão de rede, como discorrido em Ahmad *et al.* (2020), que, ao invés de monitorar *hosts* e estações de trabalho específicas, conseguem garantir a segurança a nível de rede. Esses sistemas conseguem analisar o fluxo de informação que passa pela rede, monitorando todo o tráfego, portas, requisições e transmissão de pacotes e serviços da rede de computadores. Analisando o tráfego, os NIDSs são capazes de identificar fluxos de rede que fogem do comportamento normal da rede, como o surgimento

Figura 1 – Sistemas tradicionais de segurança cibernética (TEIXEIRA *et al.*, 2021)



de oscilações no tráfego, o que pode significar atividades maliciosas na rede. HIDSs e NIDSs utilizam diferentes métodos para detectar intrusões de acordo com cada objetivo e recursos específicos. Esses métodos podem ser classificados como: baseado em uso indevido ou assinatura; baseado em anomalia; e híbrido.

Sistemas baseados em uso indevido ou assinatura são treinados utilizando uma base de dados que contenha todos os comportamentos anormais da rede ou *host*, apresentando ao modelo os ataques e atividades maliciosas conhecidos. Dessa forma, o modelo consegue identificar as atividades maliciosas a partir da assinatura dos ataques. Essa abordagem é capaz de atingir altos níveis de precisão na identificação das intrusões que foram previamente classificadas, porém, em casos de ataque ainda não conhecidos pela base de dados, o sistema não será capaz de reconhecê-lo como malicioso, como descrito em Ahmad *et al.* (2020), o que pode resultar em muitos falsos negativos, que são situações em que o sistema não identifica um ataque como uma atividade maliciosa. Em ambientes críticos, como os industriais, esse método pode se mostrar ineficaz, já que alguns falsos negativos podem resultar em graves danos e prejuízos.

Seguindo uma abordagem contrária, sistemas baseados em detecção de anomalia utilizam bases de dados que contenham somente, ou majoritariamente, o comportamento normal da rede ou *host*. Essa abordagem se preocupa em identificar eventos e comportamentos anômalos, levando em consideração, principalmente, o comportamento normal do sistema, e não somente o comportamento de ataques e suas assinaturas, diferente dos sistemas baseados em assinatura. Cada evento é analisado pelo sistema, considerando o modelo de treinamento, e é classificado como uma atividade normal ou ataque, com base em uma certa delimitação de comportamentos classificados como “normais”.

Como descrito em Ahmad *et al.* (2020), esse método é capaz de identificar ataques ainda não conhecidos pela base de dados, uma vez que todas as atividades que são consideradas destoantes do usual são tratadas como maliciosas. Porém, isso pode resultar em um alto número de falsos positivos, onde o sistema identifica que uma atividade normal é um ataque, e pode falhar, caso os ataques não apresentem um comportamento tão distinto do comportamento normal dos eventos e atividades da rede ou *host*.

Usufruindo das vantagens de cada uma das abordagens anteriores, os sistemas baseados em detecção híbrida visam combinar as técnicas de abordagens baseadas em assinatura e baseadas em anomalias para intensificar o poder de detecção, identificando ataques conhecidos e ataques ainda não vistos.

2.3 Aprendizado de máquina

Aprendizado de Máquina (do inglês, *Machine Learning* - ML) é uma subárea da Inteligência Artificial, a qual torna possível que sistemas tomem decisões com base em suas próprias experiências. Inicialmente a máquina é submetida a treinamentos, com o objetivo de encontrar padrões para que seu comportamento seja determinado a partir dos dados a ela fornecido, seguindo, resumidamente, a mesma premissa de aprendizado humano, como apresentado em Janiesch, Zschech e Heinrich (2021), progredindo em uma forma de tentativa e erro. Derivado do ML, o Aprendizado Profundo (do inglês, *Deep Learning* – DL) visa resolver problemas ainda mais complexos, como o reconhecimento de pessoas em uma imagem, por exemplo, sendo composto por algoritmos baseados na estrutura de neurônios do cérebro humano, chamados de redes neurais artificiais, que analisam e processam dados e preveem possíveis soluções.

ML pode ser dividido em duas etapas, são elas: treino e execução, ou teste. A etapa de treino corresponde a contextualização da máquina sobre o problema em questão, sendo assim, o sistema é alimentado com um conjunto de dados para a análise de padrões. Tais dados podem ter sido submetidos previamente a uma classificação que indica todas as entradas e saídas conhecidas, determinando a aprendizagem como supervisionada, como apresentado em Janiesch, Zschech e Heinrich (2021), uma vez que a aprendizagem é dependente dessas informações rotuladas para ser capaz de definir os processos que deve executar para alcançar os resultados esperados. Como também podem não ter sido classificados previamente, determinando a aprendizagem como não supervisionada, como também abordado em Janiesch, Zschech e Heinrich (2021), utilizando estratégias de redução de dimensão e agrupamento (do inglês, *clustering*), na qual os dados são analisados e agrupados sem interferência humana, conforme seu grau de similaridade. De posse do modelo treinado, é possível realizar a fase de execução, ou teste, onde o sistema consegue determinar seus próprios meios para predizer os resultados esperados, com base em padrões encontrados na etapa anterior.

É possível encontrar diversos modelos de predição na literatura, os critérios para a

escolha de qual algoritmo utilizar pode variar conforme os tipos de dados que serão utilizados e o tipo de problema que pretende resolver. O algoritmo do k-ésimo vizinho mais próximo (do inglês, *K-Nearest Neighbors* - KNN) é um algoritmo de aprendizagem supervisionada utilizado na classificação de dados baseada em similaridade, é popular por ter simples implementação e uma significante performance no que é proposto, como discorrido em Taunk *et al.* (2019). Este algoritmo realiza a classificação calculando a distância entre as amostras de teste e as amostras de treino para obter o vizinho mais próximo e, assim, rotular os dados a partir da classificação da maioria dos vizinhos mais próximos selecionados, como descrito em Zhang *et al.* (2017).

Outro método poderoso utilizado para predições, mais especificamente para propósitos de agrupamento, é a árvore de decisão (do inglês, *Decision Tree*), que é um modelo que une sucessivas séries e realiza diversos testes, onde uma característica numérica é comparada com um valor em cada teste. Árvores de decisão são compostas por nós que representam características em uma categoria a ser classificada, e cada subseção (os ramos da árvore de decisão) define um valor que pode ser escolhido por um nó. Os algoritmos *Decision Tree*, *Random Forest* e *XGBoost* são tipos de árvore de decisão, fazem parte da família de algoritmos de aprendizagem supervisionada e tem como objetivo principal construir um modelo treinado que possa ser utilizado para predizer a classe ou valor de variáveis alvo por meio de regras de decisão de aprendizagem extraídas dos dados de treinamento. Árvores de decisão podem resolver problemas de classificação e regressão, além de serem simples de entender, como descrito em Priyanka e Kumar (2020), também podem ser utilizadas para classificar tanto dados categóricos como dados numéricos.

2.4 Balanceamento de dados

Sistemas IIoT do mundo real apresentam um número significativamente baixo de ataques e intrusões, conforme citado em Pajouh *et al.* (2019), uma vez que os invasores prezam pelo anonimato e omissão de suas atividades maliciosas, pelo menos até que seu objetivo seja alcançado. Desse modo, as bases de dados disponíveis para o treinamento de IDSs para redes industriais, geralmente, são constituídas por uma pequena parcela de fluxos sob ataque, o que pode ocasionar problemas de detecção.

Uma base de dados que possua mais de 50% de uma só classe, como descrito em Melo (2019), já é considerada desbalanceada. O desbalanceamento de dados pode ser considerado como um crítico problema no processo de treinamento, já que um IDS treinado com uma base de dados desbalanceada tende a apresentar um grande número de alarmes falsos, podendo classificar um ataque como “normal” por não conseguir classificá-lo como uma atividade maliciosa, o que é resultado de um treinamento feito com um baixo número de amostras, ou por ser ataques ainda não conhecidos, como o ataque do dia zero, que se beneficia de vulnerabilidades ainda não conhecidos do sistema.

É possível encontrar algumas tecnologias que foram desenvolvidas para solucionar o

problema de desbalanceamento de dados, as mais populares são as abordagens *Sampling*, cujo objetivo, de acordo com os autores de Melo (2019), é minimizar a disparidade entre as classes presentes na base de dados por meio de uma remostarem. As abordagens *Sampling* podem ser classificadas em *Over Sampling* e *Under Sampling*.

A abordagem *Over Sampling* consiste em replicar as amostras da classe minoritária a partir dos dados originais presentes na base de dados, a fim de aumentar o número das amostras que estão em menor quantidade no *dataset*, fazendo com que a proporção das amostras fique mais igualitária. Essa tarefa pode ser feita através de algoritmos de *clustering* ou aleatoriamente. O ponto positivo dessa abordagem é que nenhum dado presente na base de dados será desperdiçado, em contrapartida, seu custo computacional pode ser elevado (MELO, 2019).

Enquanto o *Over Sampling* aumenta a base de dados criando amostras da classe de menor número, o *Under Sampling* caminha na direção contrária, diminuindo a quantidade de amostras da classe majoritária de acordo com o maior número de ocorrências. Nessa abordagem, as características da classe inferior são preservadas, e, apesar de serem complexas, podem reduzir o tempo computacional e recursos de armazenamento (MELO, 2019). Por esses motivos, para o presente trabalho, foi utilizada essa abordagem para balancear os dados presentes nas bases de dados de treinamento do IDS.

Além do grande volume de dados na composição de um *dataset* de tráfego de uma rede industrial que podem causar o desbalanceamento dele, os dados dos sensores são obtidos durante um longo período, podendo também serem obtidos através de sensores de diferentes frequências, resultando em uma base de dados de larga escala. Utilizar um *dataset* com essa dimensão pode tornar o treinamento e a detecção em tarefas muito demoradas (PAJOUH *et al.*, 2019).

Para diminuir esse tempo e tirar o melhor proveito dos algoritmos de ML para a detecção de ataques do IDS proposto neste trabalho, serão realizados experimentos para selecionar as *features* que possuam maior *score*, ou seja, maior correlação com o campo "*target*" – que determina se o fluxo está normal (0) ou sob ataque (1) – utilizando os seguintes algoritmos de ML: *Random Forest Classifier* e o cálculo do Coeficiente de Correlação de Pearson.

2.5 Treinamento do modelo

Random Forest Classifier (LICENSE, 2020) é um algoritmo de ML presente na biblioteca de código aberto "*scikit-learn*" para a linguagem de programação *Python* (FOUNDATION, 2021). Esse algoritmo utiliza um conceito baseado em treinar simples modelos de predição, chamados de "alunos fracos" (do inglês, *weak learners*), para a mesma tarefa e, a partir da combinação deles, gerar um modelo mais complexo, chamados de "alunos fortes" (do inglês, *strong learners*). Dessa forma, o algoritmo *Random Forest* treina diversas árvores de decisão, *weak learners*, separadamente para que cada uma possa se adaptar a diferentes frações da base de dados, a junção desse treinamento, *strong learner*, consegue classificar todas as variações dos dados

presentes no *dataset*.

O cálculo de correlação de Pearson mensura a relação linear estatística entre duas distintas variáveis com o objetivo de apontar como estão associadas entre si. A fórmula para o cálculo de correlação de Pearson (r) está representada na equação 2.1. Sendo n o número de amostras, x o primeiro elemento e y o segundo. $\bar{X}$ e $\bar{Y}$ representam as médias amostrais de x e y. S representa o desvio padrão, sendo S_x o desvio padrão do primeiro elemento, e S_y o desvio padrão do segundo elemento. Desta forma, a fórmula de correlação de Pearson pode ser calculada resultando em um número Real entre -1 e 1.

Uma associação negativa, ou seja, $r < 0$, indica que, à medida que o valor de um dos elementos aumenta, o valor do outro elemento diminui. Já uma associação positiva, ou seja, $r > 0$, indica que o valor de um dos elementos aumenta conforme o valor do outro elemento aumenta. Uma correlação igual a 0 não quer dizer que os elementos não possuem correlação, somente que não possuem relação linear entre si. Uma correlação -1 ou 1 é chamada de correlação perfeita, pois aponta que a variância de um elemento pode ser exatamente estabelecida ao saber a variância do outro, como apresentado em Filho e Júnior (2009).

$$r = \frac{1}{n-1} \sum \left(\frac{x_i - \bar{X}}{S_x} \right) \left(\frac{y_i - \bar{Y}}{S_y} \right) \tag{2.1}$$

De posse da base de dados balanceada e possuindo somente *features* relevantes para a detecção dos ataques de reconhecimento e DDoS, esses dados podem ser submetidos ao treinamento do modelo de detecção utilizando a aprendizagem em conjunto (do inglês, *Ensemble Learning*). A aprendizagem em conjunto, como o nome já sugere, se fundamenta da premissa de combinar modelos de predição para que, treinando-os para uma mesma tarefa, como discorrido em Cardeal e Coutinho (2019), um modelo mais complexo possa ser gerado a partir desse conjunto de modelos. Métodos *ensemble* utilizam múltiplos algoritmos de aprendizagem para alcançar uma performance melhor quando comparado à performance desses mesmos algoritmos separadamente. A base de aprendizagem do *ensemble learning* pode ser composta por algoritmos de aprendizagem simples, *weak learners*, como também por algoritmos mais complexos, *strong learners*, o que resulta em performances superiores, como observado em Zhou (2009).

Em um *ensemble* de média, como o *Random Forest*, o modelo combina múltiplas predições de modelos treinados independentemente se a performance do modelo foi satisfatória ou não. Já no *ensemble* médio ponderado pesa a contribuição de cada *learner* pelas melhores predições utilizando regressão linear ou regressão logística, fornecendo um modelo mais confiável, esse método recebe o nome de *Stacking*, e por esse motivo, esse foi o método de *Ensemble Learning* escolhido para treinar o modelo de detecção de intrusão para redes industriais apresentado no presente trabalho utilizando regressão logística (do inglês, *Logistic Regression*). A regressão logística é um método popularmente utilizado para modelar respostas binárias de dados, como discorrido em Das (2020), geralmente a resposta representada pelo número 1 indica sucesso,

enquanto a 0 indica o contrário.

2.6 Conclusão

Neste capítulo foram apresentados os ataques em redes de computadores que serão estudados no presente trabalho, assim como os tipos de IDSs existentes e as formas como podemos construir um para redes industriais, discorrendo sobre os fundamentos teóricos necessários para a implementação do IDS utilizando *Ensemble Learning*.

CAPÍTULO

3

TRABALHOS CORRELATOS

Criminosos cibernéticos estão evoluindo constantemente e alterando seus ataques para escapar de mecanismos de segurança. Por esse motivo, a utilização de avançados mecanismos de segurança se tornou essencial na identificação e prevenção de futuros ataques. Neste capítulo serão descritos trabalhos que possuam relação com o presente projeto, sendo apontadas características importantes de cada um, assim como a comparação da presente proposta com o estado da arte.

3.1 Estado da arte

Foi proposto em Ambusaidi *et al.* (2016) um algoritmo baseado em informações mútuas que seleciona analiticamente as *features* ideais para a classificação, sendo capaz de lidar com *features* dependentes de modo linear e não linear. Ainda foi desenvolvido um IDS chamado Máquina de Vetor de Suporte de Mínimos Quadrados (do inglês, *Least Square Support Vector Machine* - LSSVM-IDS) utilizando as *features* selecionadas anteriormente, classificando os ataques em DoS, Probe, R2L e U2L.

Em Apruzzese *et al.* (2018) foram analisadas tecnologias para problemas encontrados no âmbito de segurança cibernética, como análise de *malware* e detecção de intrusão e de *spam* com o intuito de verificar se tais técnicas supriam as necessidades de identificação e prevenção de tais problemas. De acordo com os autores, os resultados mostraram que as tecnologias analisadas ainda eram afetadas por certas deficiências que acabam por reduzir a efetividade da segurança no espaço cibernético. As abordagens se mostraram vulneráveis aos ataques e necessitavam de constantes retreinamentos e ajustes de parâmetros não automatizados. Os autores ainda destacam que quando o mesmo classificador é aplicado para a identificação de diferentes ameaças, o desempenho da detecção de intrusões e detecção de *spam* cai.

Uma proposta de detecção de intrusão baseada em árvore foi desenvolvida em Sarker

et al. (2020), onde as características do fluxo de rede são, inicialmente, ranqueadas de acordo com sua importância, o que, segundo os autores, minimiza a complexidade computacional do modelo de detecção de intrusão generalizado desenvolvido. A abordagem se mostrou eficaz para o objetivo do trabalho.

Métodos de ML presentes na literatura para *cyber* análise no suporte de detecção de intrusão são descritos em Buczak e Guven (2016) e mostram, segundo os autores, que ainda há uma lacuna na disponibilidade de dados rotulados, que poderiam ser utilizados com foco em segurança cibernética para treinar algoritmos de ML da melhor forma. Em Teixeira *et al.* (2018) são apresentados *datasets* de ataques reais em um tanque de armazenamento de água, onde o tratamento e distribuição de água são realizados. Os autores desenvolveram um Sistema de Supervisão e Aquisição de Dados (do inglês, *Supervisory Control And Data Acquisition* - SCADA) que fornece recursos que podem ser utilizados para treino, validação e comparação de resultados em pesquisas em segurança cibernética, e ainda destacam que o cenário de teste fornece um bom entendimento dos efeitos e consequências dos ataques em ambientes reais SCADA.

Em Keshk *et al.* (2017) é apresentada uma abordagem para a seleção de relevantes *features* na detecção de intrusão em sistemas SCADA, propondo uma técnica para a preservação da privacidade baseada no coeficiente de correlação e no mecanismo de agrupamento Maximização de Expectativa (do inglês, *Expectation Maximisation* - EM). Já em Zargar, Baghaie *et al.* (2012) é proposta uma abordagem utilizando Análise de Componentes Principais (do inglês, *Principal Components Analysis* - PCA) para reduzir a quantidade de *features* com o intuito de melhorar a taxa de detecção de intrusão em uma abordagem utilizando o algoritmo KNN. Os resultados dos experimentos se mostraram eficientes para os objetivos do trabalho, contudo os autores destacam que, para trabalhos futuros, diferentes métodos de detecção de intrusão podem ser utilizados.

Também utilizando o classificador KNN, foi identificado em Pajouh *et al.* (2019) comportamentos suspeitos em redes *backbone* IoT, para a seleção das *features* foram feitas análises de componentes e análise discriminante linear do módulo de dimensão. De acordo com os autores, a abordagem superou os modelos anteriores na detecção de ataques Usuário para Raiz (do inglês, *User to Root* - U2R) e Remoto para Local (do inglês, *Remote to Local* - R2L). No contexto de balanceamento do *dataset* para o treinamento de IDSs, foi apresentado em Zolanvari, Teixeira e Jain (2018) um estudo sobre os efeitos que *datasets* desbalanceados podem gerar no treinamento de modelos utilizando ML no contexto de segurança na área de IIoT.

Uma abordagem que utilizou *Ensemble Learning* foi apresentada em Arya e Gupta (2023), onde os autores utilizaram o coeficiente de correlação de Pearson (do inglês, *Pearson Correlation Coefficient* - PCC) e a "Floresta de Isolamento" (do inglês, *Isolation Forest*) para escolher as features mais apropriadas, testando a performance do IDS construído utilizando o classificador Random Forest, fazendo uso das bases de dados Bot-IoT e NF-UNSW-NB15-v2. Já em Mohy-Eddine *et al.* (2023) foi proposta a utilização do *Ensemble Learning* para uma

seleção de *features* utilizando quatro técnicas renomadas para a redução de features irrelevantes. Foram gerados dois conjuntos de *features* reduzida e combinadas utilizando técnicas de união e intersecção.

Os trabalhos encontrados na literatura, os quais foram descritos nesse capítulo, em sua maioria, tem como objetivo o desenvolvimento de um modelo para IDSs que possa suprir as necessidades de cada cenário específico, diferentemente desses trabalhos, o trabalho proposto visa analisar a melhor abordagem em termos de seleção de *features*, balanceamento de dados e algoritmo de predição para desenvolver um IDS para redes industriais utilizando *Ensemble Learning*. Espera-se como principais contribuições:

- Uma análise da escolha das melhores *features* para os ataques de Reconhecimento e DDoS.
- Redução nos campos e balanceamento dos *datasets* para verificar se haverá vieses no treinamento de modelos para IDSs.
- Modelo de detecção de anomalias em redes industriais utilizando *Ensemble Learning*.

3.2 Comparando a proposta e o estado da arte

A Tabela 1 ilustra as principais diferenças entre a abordagem proposta e os trabalhos encontrados na literatura, é possível observar que o presente trabalho aborda questões como a seleção de *features*, que não está presente em Apruzzese *et al.* (2018), Teixeira *et al.* (2018) e em Arya e Gupta (2023) e balanceamento de dados, que, dentre os trabalhos citados, só é tratado em Zolanvari, Teixeira e Jain (2018), onde são apresetados os efeitos que bases de dados desbalanceadas podem gerar no treinamento de modelos ML com foco em segurança.

A presente proposta ainda visa o desenvolvimento de um modelo que poderá ser utilizado por IDSs - representado na coluna IDS - , o que também foi proposto em Ambusaidi *et al.* (2016), Sarker *et al.* (2020), Teixeira *et al.* (2018), Keshk *et al.* (2017), Zargar, Baghaie *et al.* (2012) e Pajouh *et al.* (2019), também é focado em cenários industriais onde é utilizado IIoT, o que só é abordado em Teixeira *et al.* (2018), que apresentou um sistema SCADA em um ambiente industrial onde ocorria o tratamento e distribuição de água, em Keshk *et al.* (2017) e em Zolanvari, Teixeira e Jain (2018), que, apesar de tratar sobre o cenário industrial, não tem como proposta principal o desenvolvimento do modelo.

Também é proposta a utilização de ML, utilizado em Apruzzese *et al.* (2018), Teixeira *et al.* (2018), Zargar, Baghaie *et al.* (2012) e Pajouh *et al.* (2019), mas, mais especificamente, a utilização de *Ensemble Learning* no intuito de treinar um modelo mais robusto, que somente é encontrado em Mohy-Eddine *et al.* (2023). Ainda é apresentada uma redução no tamanho das *features* das bases de dados, a fim de reduzir o tempo de aprendizagem do modelo, fazendo uma

Tabela 1 – Comparação da proposta com os trabalhos citados neste capítulo considerando os objetivos específicos presentes na Seção 1.2

Trabalho	Seleção das *features*	IDS	IIoT	ML	Balanceamento do *dataset*	*Ensemble Learning*
Ambusaidi *et al.* (2016)	✓	✓	×	×	×	×
Apruzzese *et al.* (2018)	×	×	×	✓	×	×
Sarker *et al.* (2020)	✓	✓	×	×	×	×
Teixeira *et al.* (2018)	×	✓	✓	✓	×	×
Keshk *et al.* (2017)	✓	✓	✓	×	×	×
Zargar, Baghaie *et al.* (2012)	✓	✓	×	✓	×	×
Pajouh *et al.* (2019)	✓	✓	×	✓	×	×
Zolanvari, Teixeira e Jain (2018)	✓	×	✓	✓	✓	×
Arya e Gupta (2023)	✓	×	✓	✓	✓	✓
Mohy-Eddine *et al.* (2023)	✓	✓	✓	✓	×	✓
Presente trabalho	✓	✓	✓	✓	✓	✓

análise sobre a otimização do tempo de treinamento do modelo e possíveis retreinamentos do modelos sem perder performance na predição de ataques nos fluxos de rede.

3.3 Conclusão

Nesse capítulo foi discorrido brevemente sobre trabalhos presentes na literatura que tratam de assuntos pertinentes ao presente trabalho, apontando suas características e comparando suas implementações com a proposta deste trabalho.

CAPÍTULO

4

UMA ABORDAGEM *ENSEMBLE LEARNING* PARA MODELOS DE DETECÇÃO DE INTRUSÃO PARA REDES INDUSTRIAIS

Na literatura é possível encontrar diversos trabalhos com o objetivo de desenvolver um modelo para IDSs, como descrito no Capítulo 3, utilizando diferentes tecnologias, porém ainda há uma lacuna presente na detecção de intrusões em redes IIoT, onde é preciso que a segurança seja muito bem trabalhada, uma vez que um desvio de informações ou manipulação de dados pode ter resultados catastróficos. Portanto, neste trabalho é proposto o treinamento de um modelo para IDS com ênfase em redes IIoT utilizando aprendizagem em conjunto (do inglês, *Ensemble Learning*). Para isso, serão analisadas as *features* mais importantes para o treinamento adequado do modelo. Além disso, também será proposta uma análise da performance do modelo treinado com a base de dados balanceada, discutindo sobre possíveis vieses que podem comprometer a detecção de intrusões utilizando *datasets* desbalanceados no treinamento, assim como o tempo para treinamento com a base de dados tratada e reduzida e sua performance comparada ao modelo treinado com a base de dados sem a redução.

4.1 *IIDEnsemble*

Neste trabalho foi proposto um modelo de detecção de intrusão para redes industriais, intitulado *IIDEnsemble*, sistema de detecção de intrusão industrial utilizando *Ensemble Learning* (do inglês, *Industrial Intrusion Detection system using Ensemble Learning*), com o objetivo de otimizar a performance de IDSs utilizados em redes industriais, para isso o modelo proposto foi desenhado e está ilustrado na Figura 2, a descrição de cada ítem da imagem está apresentada a seguir.

- (a) Base de dados

  - Para montar o modelo proposto nesse trabalho, foi preciso utilizar bases de dados contendo fluxos de redes industriais, as quais foram submetidas a ataques de DDoS e de Reconhecimento, encontradas em Teixeira *et al.* (2018).

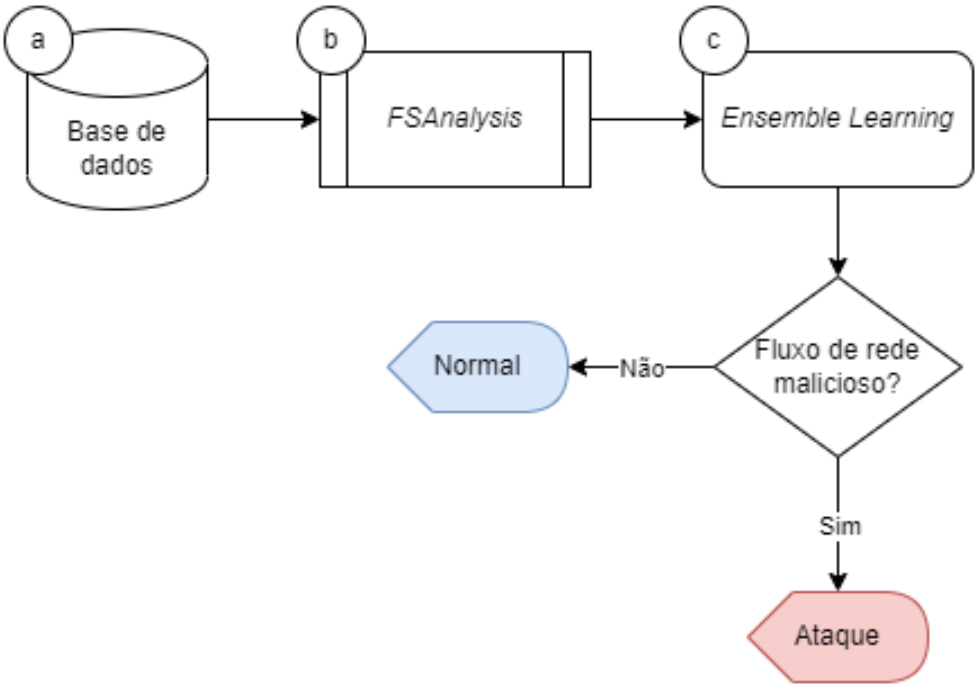
- (b) *FSAnalysis*

  - De posse das bases de dados contendo fluxos - normais e sob ataque - de redes industriais, foi preciso implementar o módulo de pré-processo *FSAnalysis* (seleção de *features* e análise, do inglês *Feature Selection Analysis*) (DALARMELINA *et al.*, 2022) para a seleção das melhores *features* para a predição do modelo, além da realização do balanceamento dos dados, para que não houvesse viéses no treinamento do modelo.

- (c) *Ensemble Learning*

  - Foram treinados dois modelos distintos utilizando a aprendizagem em conjunto, chamada de *Ensemble Learning*, com a finalidade de comparar a performance de modelos treinados com uma redução no tamanho de cada *feature* e a performance de modelos treinados sem essa redução. Ao fim destes testes, o modelo de melhor performance foi selecionado, sendo capaz de decidir se o fluxo de entrada era malicioso ou não e classifica-lo como "normal"ou "ataque".

Figura 2 – *IIDEnsemble*

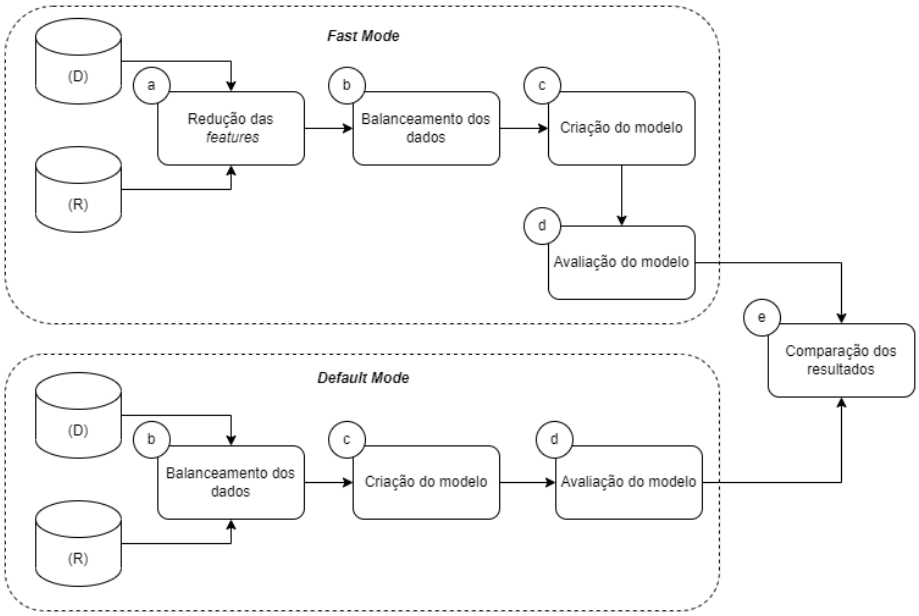


Para a construção do *IIDEnsemble*, inicialmente o *dataset* foi submetido ao módulo de pré-processamento *FSAnalysis*, onde foi analisada e escolhida a melhor abordagem para a seleção de *features* e a decisão de balancear ou não os dados para a criação do modelo.

Com os resultados obtidos nessa etapa explicados no Capítulo 5, observou-se que as *features* que obtiveram performances superiores no treinamento do modelo, dentre as 25 presentes nos *datasets*, foram as selecionadas pelo PCC, são elas: *DstPort*; *SrcLoss*; *DstLoss* e *PLoss*, e o treinamento do modelo realizado com a base de dados balanceada também obteve melhores resultados.

Foi observado que a base de dados contendo ataques de reconhecimento fazia uso de 3.4GB de memória, contendo 5 *features* do tipo *object*, 8 do tipo *float64* e 12 do tipo *int64*. De posse dessas informações, foi analisada uma maneira de reduzir o tamanho dessa base sem perder nenhum dado. Portanto, foi utilizada a função do *pandas*, "*to_numeric*" para realizar a conversão dos tipos dessas *features*, de modo que essas pudessem ocupar menos memória, então, *features* do tipo *float64* foram convertidas para *float32* e *features* do tipo *int64* foram convertidas para *int32* e *int8*, de acordo com o tamanho do seu valor. Dessa forma, o uso de memória dessa base de dados caiu para 2.8GB. Para comparar se essa redução no *dataset* faria uma diferença significante no treinamento do modelo, a etapa de tratamento de dados foi segmentada em uma sub-etapa dividida em dois modos, o modo rápido (do inglês, *Fast Mode* - FM) e o modo padrão (do inglês, *Default Mode* - DM), como ilustrado na Figura 3, as quais as bases de dados, sendo (D) a base de dados que contém os ataques de DDoS e (R) a base de dados que contém os ataques de Reconhecimento, foram submetidas separadamente.

Figura 3 – Etapa de tratamento das bases de dados



O FM consiste em (a) reduzir as *features* utilizando o método descrito anteriormente para reduzir o tamanho ocupado por elas. Após a redução das *features*, os dados são balanceados (b) utilizando o método *Under Sampling*, para que haja a redução do custo computacional na geração do modelo. Em seguida, é utilizado *Ensemble Learning*, mais especificamente

o *Stacking Classifier* - fazendo uso dos *learners Decision Tree Classifier*, *Random Forest Classifier*, *KNeighbours Classifier*, *XGB Classifier* - , para criar o modelo (c) a partir da base de dados tratada. Por fim, o modelo criado tem sua performance avaliada (d) utilizando o algoritmo *LogisticRegression*, do módulo "*linear_model*" da biblioteca de código aberto "*sklearn*" (PEDREGOSA *et al.*, 2011), extraindo a acurácia e a matriz de confusão para que outros cálculos possam ser realizados. De posse do modelo treinado, é possível partir para a etapa de testes, onde o modelo já é capaz de identificar fluxos de rede maliciosos, classificando-os como ataque ou normal. Já o DM consiste em realizar os mesmos processos do FM, porém, sem o processo (a), de forma que as *features* continuem ocupando a mesma quantidade de memória. Ao final dos experimentos DM e FM, foi selecionada a abordagem FM por obter melhores resultados quando comparados aos resultados de DM utilizando cálculos como acurácia, sensibilidade, FAR e UR extraídos da matriz de confusão, bem como o tempo de aprendizagem e de predição.

4.2 Base de dados

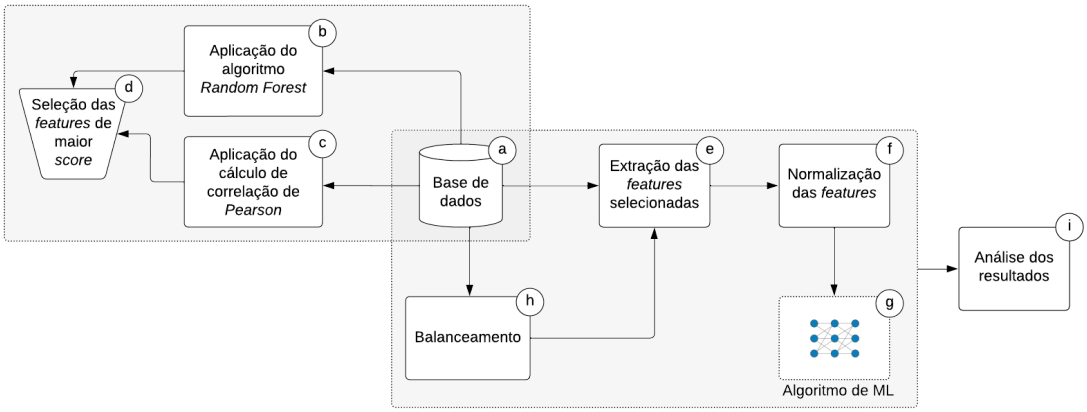
Para executar a proposta do atual projeto foram utilizados dois *datasets* encontrados em Teixeira *et al.* (2018), onde foram montadas bases de dados com fluxos de rede industrial reais, que foram submetidas a ataques de DDoS e de Reconhecimento. Uma base de dados contendo ataques de DDoS - sendo 51,73% dos ataques do tipo DoS; 11,19% dos ataques do tipo DDoS; e 37,07% dos ataques do tipo DDoS *Spoofing* - e outra base de dados contendo 13 tipos diferentes de ataque de Reconhecimento.

Cada base de dados utilizada contém 25 *features* que representam uma característica do fluxo, como os *bits* de destino por segundo, pacotes de destino, porcentagem de pacotes descartados, número da porta de destino, entre outras características importantes para o fluxo de rede, assim como a classificação desse fluxo, sendo o valor "0"a classificação de fluxo "normal"e "1"o fluxo sob ataque.

4.3 *FSAnalysis*

Essa etapa do projeto consistiu em realizar experimentos para selecionar a melhor abordagem no contexto de seleção das melhores *features* e seleção da abordagem do *dataset* (balanceado ou desbalanceado). Para a seleção das melhores *features* foram utilizados o algoritmo *Random Forest* e o PCC. O fluxograma da arquitetura do *FSAnalysis*, ilustrado na Figura 4, tem como elemento centralizador a base de dados (a). Serão utilizadas duas diferentes bases de dados que contêm amostras de fluxos normais e ataques em uma rede industrial, cada base de dados contém um tipo de ataque diferente, sendo eles, ataque de Reconhecimento e de DDoS. A partir da base de dados serão realizados experimentos utilizando o algoritmo *Random Forest* (b) e, paralelamente, o PCC (c), para encontrar as *features* de maior relevância para cada abordagem.

Figura 4 – Fluxograma do *FSAnalysis* (DALARMELINA *et al.*, 2022)



Para utilizar o algoritmo *Random Forest*, o método *RandomForestClassifier*, da biblioteca "*scikit-learn*", foi empregado. Mesmo com os mesmos dados de treinamento, a melhor combinação de predição encontrada a partir desse método pode variar, no presente trabalho isso não poderia acontecer pois um dos objetivos era encontrar as melhores *features* para serem utilizadas no treinamento de um IDS para redes industriais e os experimentos devem poder ser replicados posteriormente. Portanto, para controlar a aleatoriedade da inicialização das amostras usadas ao construir as árvores de decisão, foi definido o valor "0" para o parâmetro "*random_state*" do método, assim como o valor "10" no parâmetro "*n_estimators*", definindo que o modelo iria gerar somente 10 árvores de decisão para treinamento.

Após a realização dos experimentos utilizando *Random Forest*, observou-se que as *features DstBytes*, *DstLoad* e *DstRate* são as que trazem características mais importantes ao avaliar a base de dados que contém ataques de DDoS (D), uma vez que estas apresentaram *scores* acima de 0,15. Na base de dados que contém ataques de Reconhecimento (R) foi observado que as *features pLoss*, *Dport*, *SrcPkts* e *TotPkts* foram as mais relevantes, obtendo *scores* acima de 0,15. Na literatura é possível encontrar diversas métricas para a avaliação dos melhores *scores* resultantes do cálculo PCC, em Cohen (1992) é considerado que valores entre 0,50 e 1 podem ser interpretados como de grande magnitude, enquanto outros autores acreditam que essa faixa seja um pouco menor, de 0,70 ou, até mesmo, 0,80 a 1. A faixa valor para a escolha do melhor *score* pode variar de acordo com cada contexto e propósito, portanto, neste trabalho, para o critério de seleção das melhores *features*, foi determinado que somente as *features* que obtiveram *score* maior que 0,55 seriam selecionadas. Com isso, notou-se que as melhores *features* utilizando a base de dados (D) foram *DstLoss*, *pLoss*, *Mean* e *SrcLoss*, e utilizando a base de dados (R), *DstLoss*, *Dport*, *SrcLoss* e *pLoss*, alcançando *scores* superiores a 0,55. Os detalhes dos resultados estão descritos no Capítulo 5.

De posse das *features* mais relevantes para cada ataque, os dados presentes nas bases utilizadas foram normalizados utilizando a classe *StandardScaler*, do módulo "*preprocessing*"

da biblioteca "*sklearn*" (PEDREGOSA *et al.*, 2011), e divididos em duas partes, sendo 80% para treinamento e 20% para teste. Após esse pré-processamento, a classe *LogisticRegression*, do módulo "*linear_model*" da biblioteca de código aberto "*sklearn*" (PEDREGOSA *et al.*, 2011), foi utilizada para criar o modelo de predição dos fluxos de rede fazendo uso das *features* de maior importância, selecionadas conforme descrito anteriormente.

Para a realização dos testes de predição para a escolha da melhor abordagem, sendo ela utilizando as bases na íntegra e utilizando as bases balanceadas, três cenários foram utilizados, são eles: (1) aplicação das *features* selecionadas utilizando o algoritmo *Random Forest*; (2) aplicação das *features* selecionadas utilizando o PCC; (3) aplicação de todas as *features* presentes em ambos as bases de dados. A mesma rotina de treinamento foi realizada utilizando os *datasets* (D) e (R). Utilizando o algoritmo de predição *Logistic Regression* foi analisado tempo de aprendizagem, acurácia, sensibilidade, taxa de alarme falso (do inglês, *False Alarm Rate* - FAR), taxa de não detecção (do inglês, *Undetected Rate* - UR) e coeficiente de correlação de Matthews (do inglês, *Matthews Correlation Coefficient* - MCC), que são alguns cálculos extraídos da matriz de confusão, ilustrada na Tabela 2, que pode ser interpretada da seguinte maneira:

- Fluxo de entrada: Valor a ser analisado pelo algoritmo. Esse valor pode ser 0 ou 1, sendo 0 a representação numérica do fluxo de rede normal, e 1 o fluxo sob ataque.
- Verdadeiro Negativo (do inglês, *True Negative*): Representa o número de fluxos normais classificados corretamente como fluxo normal.
- Falso Positivo (do inglês, *False Positive*): Representa o número de fluxos normais classificados erroneamente como fluxo sob ataque.
- Falso Negativo (do inglês, *False Negative*):,Representa o número de fluxos sob ataque classificados erroneamente como fluxo normal.
- Verdadeiro Positivo (do inglês, *True Positive*): Representa o número de fluxos anormais classificados corretamente como ataques.

A acurácia (do inglês, *Acuracy*), é uma das métricas mais utilizadas para a avaliação de performance de modelos ML pois representa a porcentagem de predições feitas corretamente considerando o número total de predições. A fórmula para o cálculo da acurácia está representada na Equação 4.1.

$$Acuracy = \frac{TP+TN}{TP+TN+FP+FN} x100 \tag{4.1}$$

A FAR apresenta a porcentagem de tráfego normal classificado erroneamente como ataque. Esse cálculo está representado na Equação 4.2.

Tabela 2 – Matriz de confusão

		Predição	
		Classificado como normal	**Classificado como ataque**
Entrada	**Fluxo normal**	Verdadeiro Negativo (TN)	Falso Positivo (FP)
	Fluxo sob ataque	Falso Negativo (FN)	Verdadeiro Positivo (TP)

$$FAR = \frac{FP}{FP+TN} x100 \tag{4.2}$$

No contexto de redes industriais, a UR pode ser considerada mais importante que a FAR, pois essa métrica representa a fração de tráfego sob ataque que foi classificado erroneamente como fluxo normal, e pode ser calculada utilizando a Equação 4.3.

$$UR = \frac{FN}{FN+TP} x100 \tag{4.3}$$

Uma métrica muito importante para avaliar IDSs treinados com bases desbalanceadas é o MCC, pois mensura a qualidade da classificação, apresentando a correlação entre as predições e os valores reais. A fórmula que será utilizada para calcular o MCC está representada na Equação 4.4.

$$MCC = \frac{TPxTN - FPxFN}{\sqrt{(TP+FP)x(TP+FN)x(TN+FP)x(TN+FN)}} x100 \tag{4.4}$$

Por fim, será avaliada a sensibilidade dos modelos, calculando a taxa de verdadeiro positivo (do inglês, *True Positive Rate* ou *Sensitivy*), para mensurar a eficácia na predição de atividades anormais classificadas corretamente como ataques. Para isso, será utilizada a Equação 4.5.

$$Sensitivity = \frac{TP}{TP+FN} x100 \tag{4.5}$$

Os resultados mostraram que, apesar da diferença não ser consideravelmente grande, a performance do modelo treinado com as bases balanceadas foi melhor, alcançando resultados superiores especialmente em termos de tempo de aprendizagem, MCC e sensibilidade, como detalhado no Capítulo 5.

4.4 *Ensemble Learning*

A aprendizagem em conjunto pode ser composta por algoritmos de aprendizagem simples, como também por algoritmos mais complexos, o que pode resultar em performances superiores, em um *ensemble* médio ponderado é mensurada a contribuição de cada *learner* através das melhores predições utilizando regressão linear ou logística. No presente trabalho

esse método de aprendizagem, denominado "empilhamento"(do inglês, *Stacking*), foi utilizado a partir da biblioteca "*sklearn.ensemble*"e os algoritmos empregados para o empilhamento foram os algoritmos de árvore de decisão *Decision Tree*, *Random Forest* e *XGBoost*, e o algoritmo de aprendizagem supervisionada KNN.

Ao final do *Stacking* foi utilizado o algoritmo *Logistic Regression*, que é um dos métodos mais utilizados para modelar respostas binárias de dados, para treinar o modelo gerado. De posse do modelo, foi possível avalia-lo comparando as predições com os valores reais, e gerando a matriz de confusão para que sua performance pudesse ser avaliada.

4.5 Conclusão

Neste capítulo foi descrita a proposta do presente trabalho, uma abordagem de *Ensemble Learning* para modelos de detecção de intrusão para redes industriais, detalhando as etapas percorridas para a execução da proposta, como o desenvolvimento do módulo *FSAnalysis* e tratamentos de dados focados em performance e tempo de aprendizagem dos modelos, apresentando o fluxograma proposto e destacando a metodologia utilizada. Os resultados obtidos com os experimentos e testes dos modelo treinados estão detalhados no Capítulo 5, e o código implementado para a execução dos experimentos apresentados neste capítulo estão disponíveis em Dalarmelina (2023).

CAPÍTULO

5

RESULTADOS

Para o desenvolvimento do presente trabalho foram escolhidas bases de dados presentes em (TEIXEIRA *et al.*, 2018). Os dados presentes nesses *datasets* foram coletados em uma rede industrial de tratamento e distribuição de água, onde um tanque de armazenamento de água sofreu ataques de DDoS e Reconhecimento – para mais detalhes, consultar (TEIXEIRA *et al.*, 2018). As bases de dados foram divididas por ataque, portanto, para o presente trabalho foram utilizados dois *datasets*, um contendo os ataques de DDoS (D) e o outro, contendo os ataques de Reconhecimento (R).

5.1 *FSAnalysis*

Nessa seção serão detalhados os resultados obtidos pelo módulo *FSAnalysis*, o qual é resposável por realizar a seleção das *features* das bases de dados considerando bases balanceadas e desbalanceadas, analisando cada um dos cenários para a seleção do que obtiver a melhor performance. Para o desenvolvimento e execução do *FSAnalysis* foram utilizados diferentes bases de dados contendo ataques de DDoS (D) e de Reconhecimento (R), esse módulo engloba a seleção das *features* de maior *score* em cada uma das bases de dados, assim como a extração das *features* encontradas das bases de dados desbalanceadas e balanceadas, separadamente, para que experimentos com o algoritmo de predição *Logistic Regression* pudessem ser realizados. Para a avaliação dos resultados, foram utilizados cálculos extraídos da matriz de confusão, sendo eles acurácia, sensibilidade, FAR, UR e MCC.

5.1.1 Seleção das features

Na primeira etapa de desenvolvimento desse módulo foi realizada a seleção das *features* de maior relevância para cada ataque utilizando o algoritmo *Random Forest* e o cálculo PCC. Para utilizar o algoritmo *Random Forest*, o método *RandomForestClassifier*, da biblioteca "*scikit-learn*", foi empregado. Para controlar a aleatoriedade da inicialização das amostras usadas ao

Tabela 3 – *Features* selecionadas da base (D) utilizando o algoritmo *Random Forest*

Features	*Scores*	Descrições
DstBytes	0.272996	Contagem de destino/origem.
DstLoad	0.182034	Bits de destino por segundo.
DstRate	0.181972	Pacotes de destino por segundo.

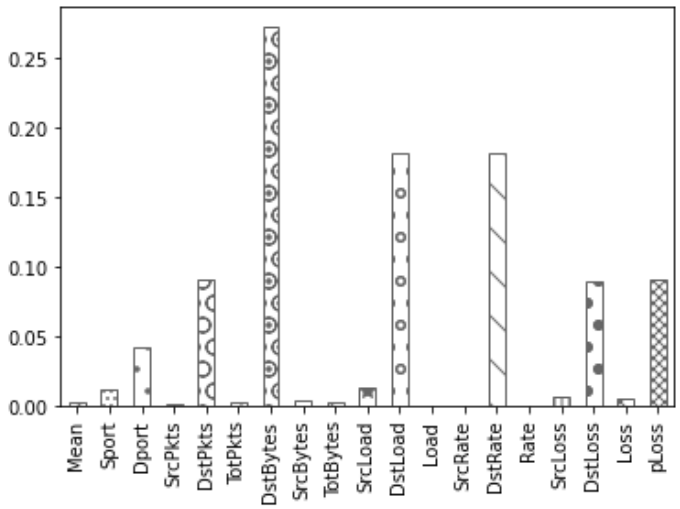
Tabela 4 – *Features* selecionadas na base (R) utilizando o algoritmo *Random Forest*

Features	*Scores*	Descrições
pLoss	0.309460	Porcentagem de pacotes descartados.
Dport	0.199118	Número da porta de destino.
SrcPkts	0.183496	Contagem de pacotes de origem/destino.
TotPkts	0.180653	Contagem total de pacotes de transação.

construir as árvores de decisão, foi definido o valor "0" para o parâmetro "*random_state*" do método, assim como o valor "10" no parâmetro "*n_estimators*", definindo que o modelo iria gerar somente 10 árvores de decisão para treinamento.

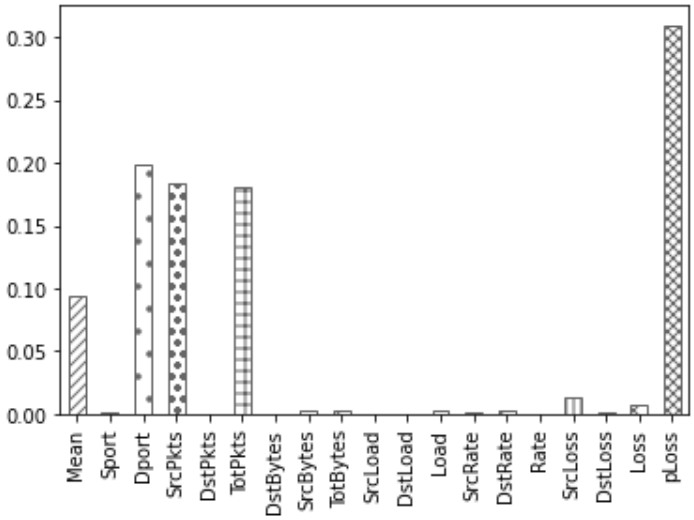
Analisando o gráfico ilustrado na Figura 5 é possível observar que as *features* *Destination Bytes* (DstBytes), *Destination Load* (DstLoad) e *Destination Rate* (DstRate) foram as que trouxeram características mais importantes ao avaliar o *dataset* (D), apresentando *scores* acima de 0,15. Na Tabela 3 são descritas as funções de cada *feature*, bem como o *score* obtido por cada uma delas.

Figura 5 – *Features* mais relevantes na base (D) encontrados pelo modelo *Random Forest*



Ao realizar os mesmos procedimentos com a base (R), foi observado que *features* diferentes obtiveram *score* maior. É possível observar na Figura 6 que as *features* mais relevantes foram: pLoss, Dport, *Source Packets* (SrcPkts) e *Total Packets* (TotPkts), obtendo *scores* acima de 0,15. As *features* de maior *score* são descritas na Tabela 4, bem como o *score* obtido por cada uma delas.

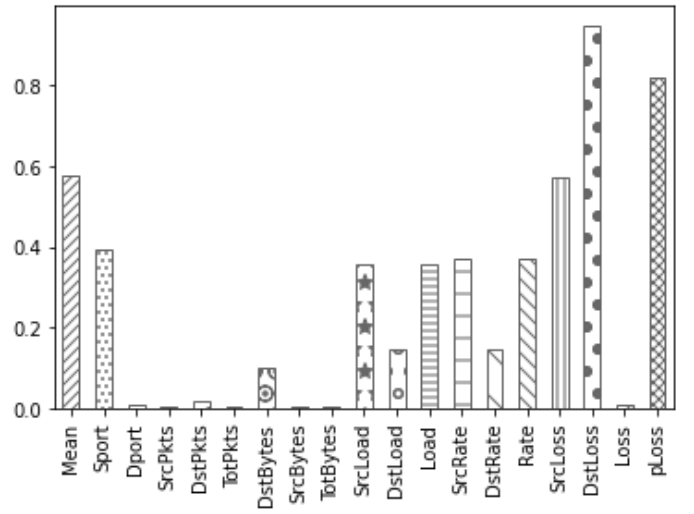
Figura 6 – *Features* mais relevantes na base (R) encontradas pelo modelo *Random Forest*



Desta forma, a seleção das melhores *features* para cada base de dados utilizando o algoritmo *Random Forest* foi realizada. Feito isso, o cálculo PCC foi utilizado para que as *features* que obtiverem maior *score* de correlação com o campo "*target*" – campo da base de dados que indica se o fluxo está normal (0) ou sob ataque (1) – pudessem ser identificadas.

Aplicando o PCC no *dataset* (D), foi observado que as *features* que obtiveram maior *score* de correlação foram DstLoss, pLoss, *Mean flow* (Mean) e SrcLoss, como ilustrado na Figura 7, obtendo *scores* superiores a 0,55.

Figura 7 – Correlação das *features* com o campo *target* na base (D) determinada pelo cálculo PCC



Ao realizar os mesmos procedimentos com a base (R), foi observado que as *features* DstLoss, Dport, SrcLoss e pLoss obtiveram *score* maior que 0,55, como ilustrado na Figura 8, apresentando maior correlação com o campo *target*.

Neste trabalho, para o critério de seleção das melhores *features*, foi determinado que

Figura 8 – Correlação das *features* com o campo *target* na base (R) determinada pelo cálculo PCC

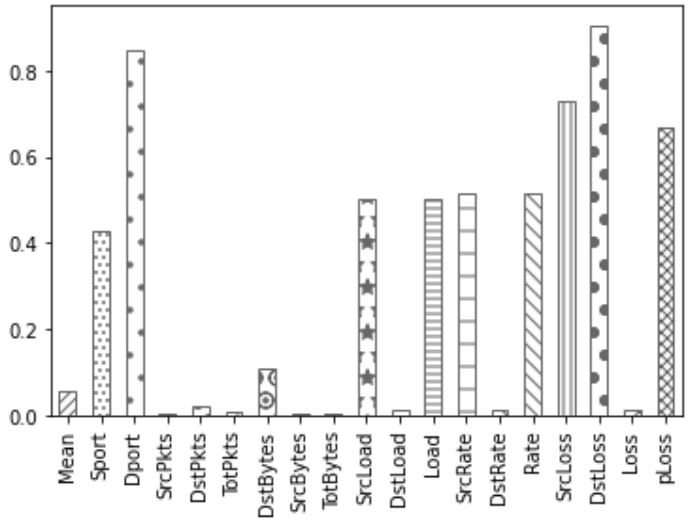


Tabela 5 – *Features* selecionadas na base (D) utilizando o cálculo PCC

Features	*Scores*	Descrições
DstLoss	0.949714	Pacotes de destino descartados.
pLoss	0.818986	Porcentagem de pacotes descartados.
Mean	0.578331	Média de duração dos fluxos ativos.
SrcLoss	0.571413	Pacotes fonte descartados.

Tabela 6 – *Features* selecionadas na base (R) utilizando o cálculo PCC

Features	*Scores*	Descrições
DstLoss	0.905914	Pacotes de destino descartados.
Dport	0.847983	Número da porta de destino.
SrcLoss	0.729565	Pacotes origem descartados.
pLoss	0.671127	Porcentagem de pacotes descartados.

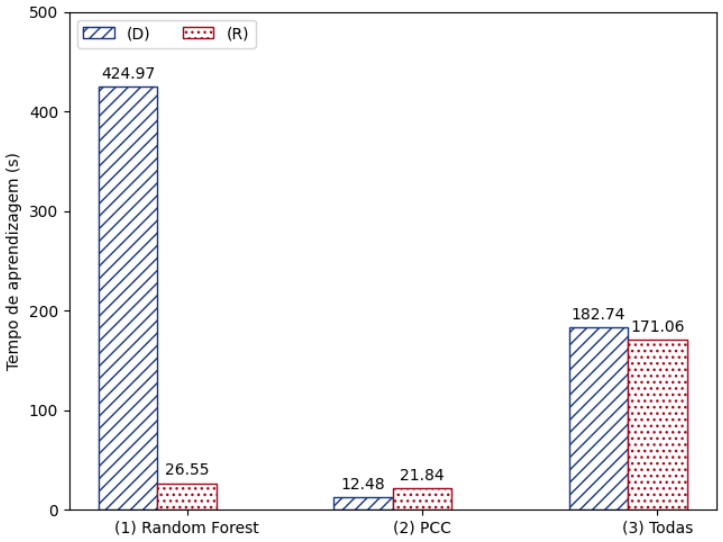
somente as *features* que obtiveram *score* maior que 0,55 seriam selecionadas. A Tabela 5 e a Tabela 6 apresentam as *features* selecionadas, assim como o *score* e descrição de cada uma delas.

Para a realização dos testes de predição, três cenários foram utilizados, são eles: (1) aplicação das *features* selecionadas utilizando o algoritmo *Random Forest*; (2) aplicação das *features* selecionadas utilizando o cálculo PCC; (3) aplicação de todas as *features* presentes em ambos os *datasets*. A mesma rotina de treinamento foi realizada utilizando as bases (D) e (R).

Analisando a Figura 9, onde foi discriminado o tempo consumido por cada aprendizagem, é possível observar que o tempo de aprendizagem utilizando a base (D) no cenário (1) foi o maior dentre os cenários e *datasets* utilizados, contabilizando 424,97 segundos, enquanto, no cenário (2), utilizando a mesma base de dados, o tempo de aprendizagem foi o menor, contabilizando 12,48 segundos. Quando observado apenas o tempo de aprendizagem utilizando a base (D), nota-se que o cenário (3) obteve o segundo mais longo tempo de aprendizagem, enquanto que, se

observado somente o tempo utilizando a base (R), o cenário (3) obteve o tempo de aprendizagem mais longo dentre os três cenários.

Figura 9 – Tempo de aprendizagem consumido utilizando o algoritmo *Logistic Regression*



5.1.2 Balanceamento das bases de dados

Para verificar se as bases de dados utilizadas no presente trabalho estavam balanceados, foi necessário discriminar a proporção de cada classe presente em cada uma delas. Analisando a Figura 10 é possível observar que o número de amostras de fluxos normais é bem maior que a quantidade de amostras do fluxo sob ataque.

Na base (D) haviam 7.206.421 amostras de fluxo normal, enquanto as amostras de fluxo sob ataque somavam 675.832, tendo uma proporção de 1:10,66 – 1 fluxo sob ataque a cada 10,66 fluxos normais. Representando uma grande desproporção de dados, o que classifica a base de dados como desbalanceada. Já na base (R), a proporção de amostras normais para cada amostra sob ataque foi ainda maior, 1:16,47, sendo 6.646.523 amostras normais de fluxo de rede e 403.466 de fluxo de rede sob ataque.

Por conta dessa desproporção de dados presente nas bases de dados utilizadas, foi preciso balancear ambas as bases, para isso foi utilizado a *RandomUnderSampler*, uma classe da biblioteca "*imblearn*" presente na linguagem *Python* (FOUNDATION, 2021).

Figura 10 – Amostras presentes na base (D) e na base (R)

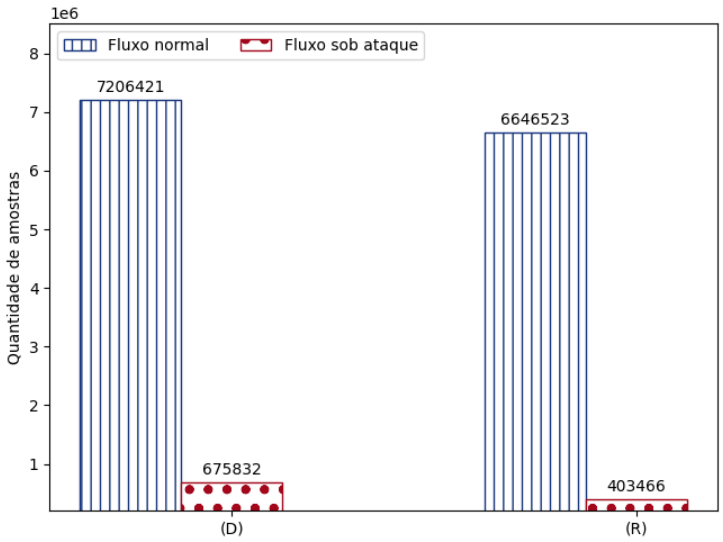
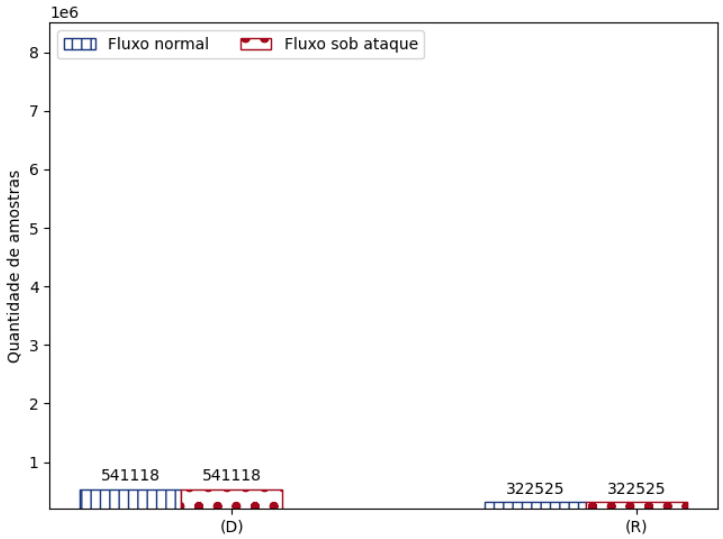


Figura 11 – Amostras presentes nas bases de dados após o balanceamento

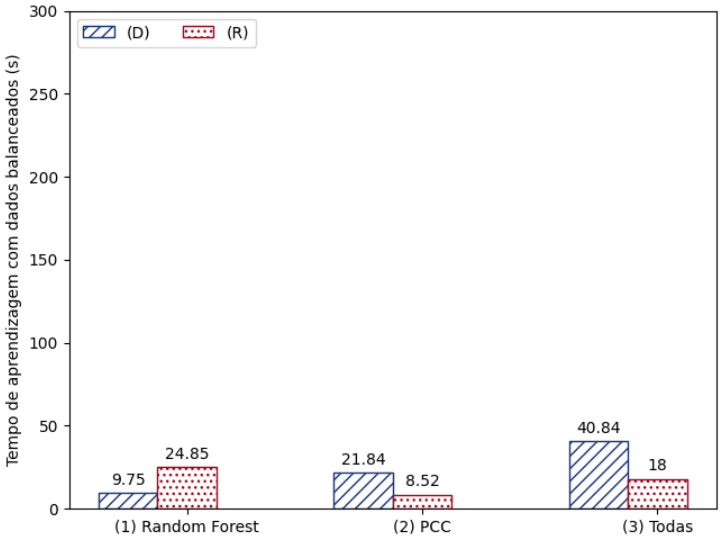


Essa classe executa a técnica de *Under Sampling* aleatoriamente, ou seja, escolhe de maneira aleatória as amostras da classe majoritária que serão deletadas da base de dados. Dessa forma, a base (D), após balanceado, apresenta proporção 1:1, com 541.118 amostras de cada classe, assim como a base (R), que, após aplicados os algoritmos de balanceamento, apresentou uma proporção de 1:1, sendo 322.525 amostras de cada classe, como ilustrado na Figura 11. Após o balanceamento dos dados, os mesmos testes descritos na Subseção 5.1.1 foram realizados, o tempo de aprendizagem foi contabilizado e está ilustrado na Figura 12.

5.1.3 Avaliação do FSAnalysis

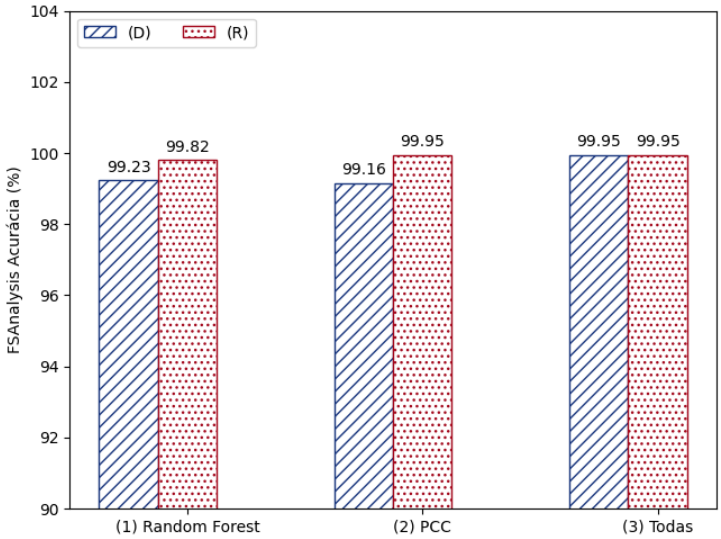
Para a avaliação da performance dos experimentos descritos nesta seção, foram utilizadas métricas derivadas da matriz de confusão, as métricas utilizadas foram acurácia; FAR; UR; MCC

Figura 12 – Tempo de aprendizagem com as bases balanceadas



e sensibilidade. Pra calcular a acurácia foi utilizada a Equação 4.1. Analisando a Figura 13 é possível observar que o cenário (3) obteve maior acurácia, 99.95% com *dataset* (R) e 99,50% com o *dataset* (D), isso acontece porque todas as *features* do tráfego da rede foram consideradas.

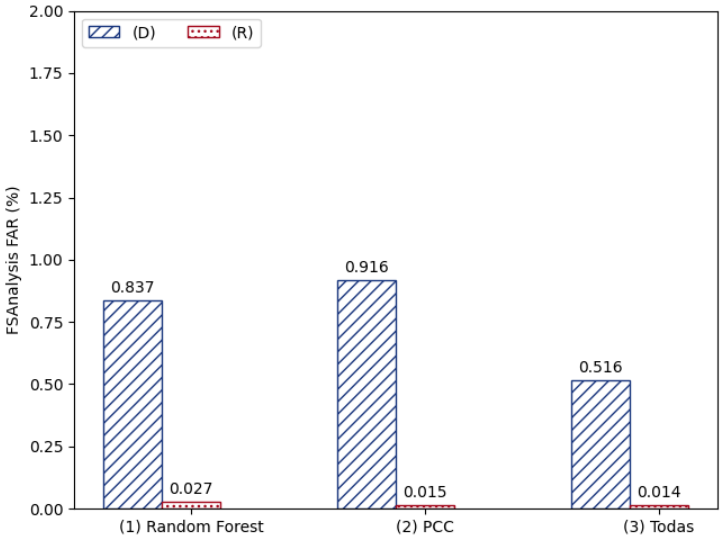
Figura 13 – Acurácia obtida utilizando o algoritmo *Logistic Regression*



Apesar da acurácia ter se mostrado satisfatória, apresentando mais de 99% em todos os testes, é necessário levar em consideração outras métricas para avaliar a performance de modelos treinados com *datasets* desbalanceados. Portanto também foi calculada a FAR, também conhecida como *False Alarm Rate*, em inglês, que apresenta a porcentagem de tráfego normal classificado erroneamente como ataque. Ao aplicar a Equação 4.2 para o cálculo de FAR, os resultados obtidos nos cenários estudados foram satisfatórios, apresentando porcentagem menor que 1%.

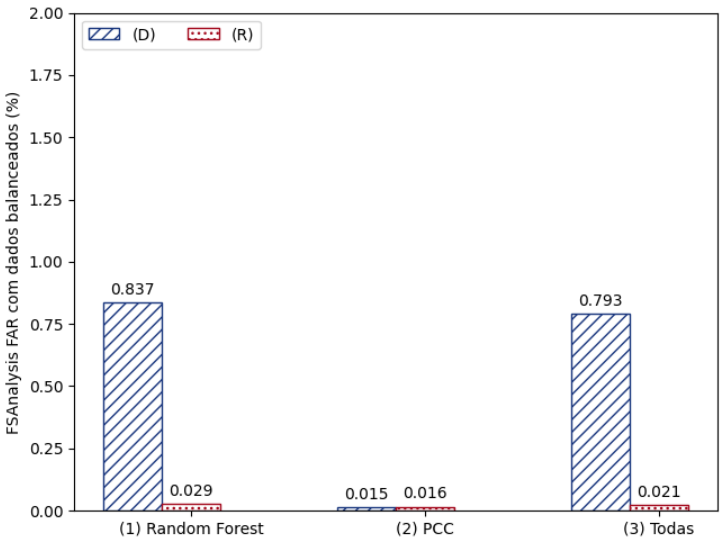
Ao analisar a Figura 14 é possível observar que, utilizando o *dataset* que contém os ataques de DDoS, o IDS classificou mais fluxos normais erroneamente se comparado ao treina-

Figura 14 – FAR obtida utilizando o algoritmo *Logistic Regression*



mento com o *dataset* contendo ataques de Reconhecimento. O mesmo pôde ser observado com os experimentos realizados utilizando *datasets* balanceados, como ilustrado na Figura 15. Nesse teste, as *features* selecionadas pelo PCC se mostrou mais eficaz por apresentar a menor taxa de alarme falso dos três testes realizados – 0,015% com a base de dados (D) e 0,016% com a base (R).

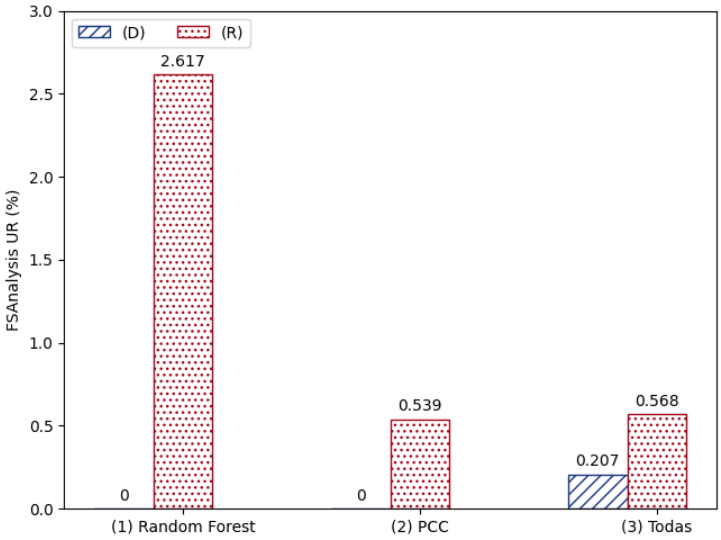
Figura 15 – FAR obtida utilizando o algoritmo *Logistic Regression* com bases balanceadas



Para representar a porcentagem de tráfego sob ataque que foi classificado erroneamente como fluxo normal durante os experimentos, foi utilizado o cálculo da UR, ou Undetected Rate em inglês, que pode ser representado pela Equação 4.3.

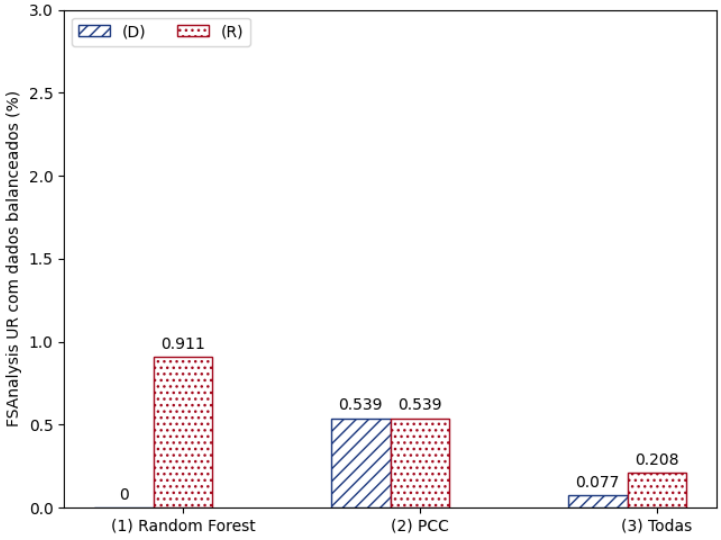
Considerando a Figura 16 apesar de apresentar baixos resultados no cálculo de UR, o IDS treinado com a base (R) obteve a maior taxa de falsos negativos, o que pode ser um grande problema no contexto industrial. Após o balanceamento das bases de dados, o mesmo cálculo de UR foi aplicado ao IDS. Analisando a Figura 17 é possível observar que, apesar de melhores

Figura 16 – UR obtida utilizando o algoritmo *Logistic Regression*



resultados, apresentando 0,91% como a maior taxa, o desempenho do IDS treinado no cenário (2) caiu, apresentando uma taxa de 0,53% de falsos negativos para a base (D), taxa que era de 0% com o treinamento feito com a base de dados desbalanceada.

Figura 17 – UR obtida utilizando o algoritmo *Logistic Regression* com bases balanceadas



Uma métrica importante para avaliar IDSs treinados com bases de dados desbalanceadas é o MCC, pois mensura a qualidade da classificação, apresentando a correlação entre as predições e os valores reais. A fórmula utilizada para calcular o MCC dos IDSs avaliados está representada na Equação 4.4. Analisando a Figura 18 é possível observar que as avaliações dos três cenários se mostraram satisfatórias para os objetivos esperados no presente trabalho, alcançando uma média de 96,87% nos IDSs treinados no cenário (1), 97,27% no (2) e 98,27% no (3). Apesar de obter resultados promissores, o MCC dos testes realizados com os IDSs treinados com os *datasets* balanceados se mostraram melhores, salvo os testes no cenário (3) que apresentaram uma média de MCC 0,63% mais alta, como ilustrado na Figura 19.

Figura 18 – MCC obtida utilizando o algoritmo *Logistic Regression*

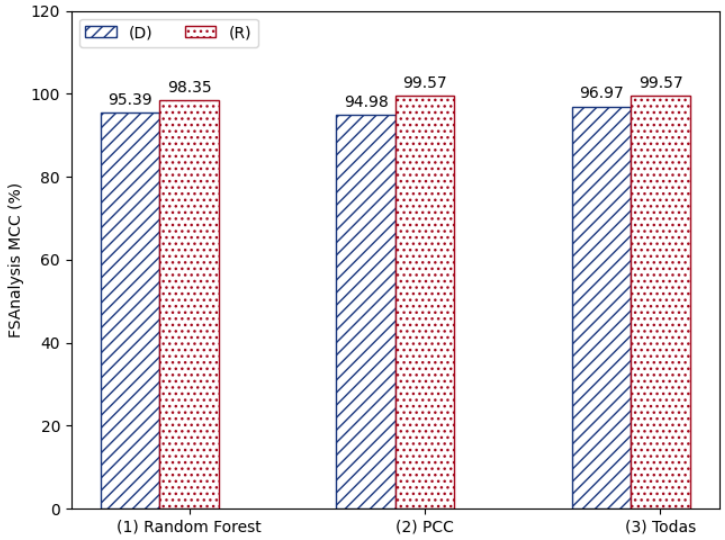
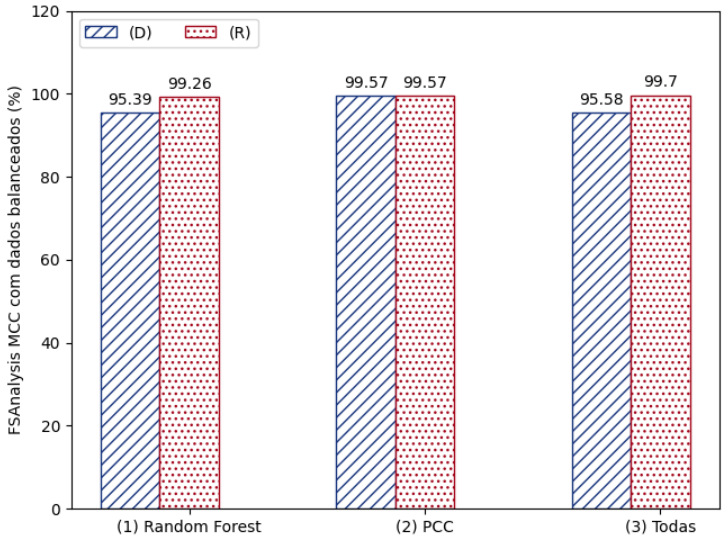


Figura 19 – MCC obtida utilizando o algoritmo *Logistic Regression* com bases balanceadas



Por fim, foi avaliada a sensibilidade dos modelos, calculando a Taxa de Verdadeiro Positivo, em inglês *True Positive Rate* ou *Sensitivy*, para mensurar a eficácia na predição de atividades anormais classificadas corretamente como ataques. Para isso, foi utilizada a equação 4.5. Observando a Figura 20 é possível constatar que os modelos apresentaram alta sensibilidade na detecção de intrusões no tráfego de rede mesmo sendo treinados com *datasets* desbalanceados. Ao avaliar os modelos treinados com os *datasets* balanceados, como ilustrado na Figura 21, a sensibilidade na detecção foi ainda maior nos IDSs treinados com a base (R) quando comparados aos modelos treinados com a base desbalanceado.

Considerando o resultado das avaliações dos experimentos realizados com o algoritmo de predição *Logistic Regression* foi observado que, apesar da diferença não ser tão grande, a performance dos IDSs treinados com os *datasets* balanceados foi mais satisfatória, alcançando resultados superiores principalmente em termos de tempo de aprendizagem, MCC e sensibilidade.

Figura 20 – Sensibilidade obtida utilizando o algoritmo *Logistic Regression*

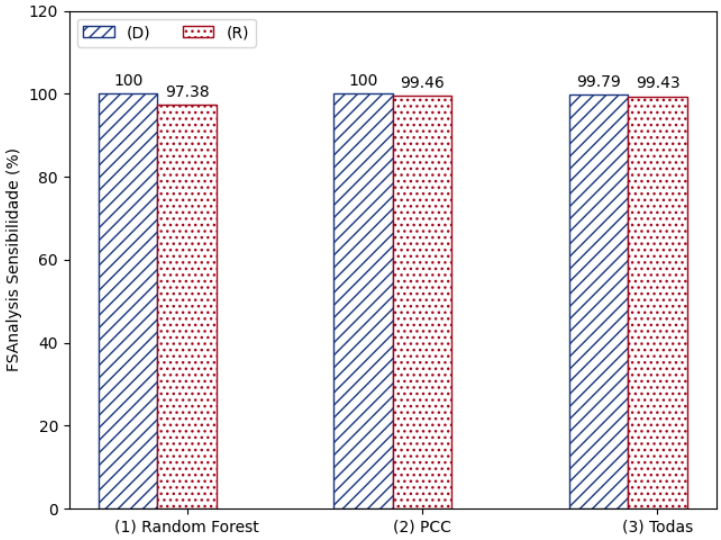
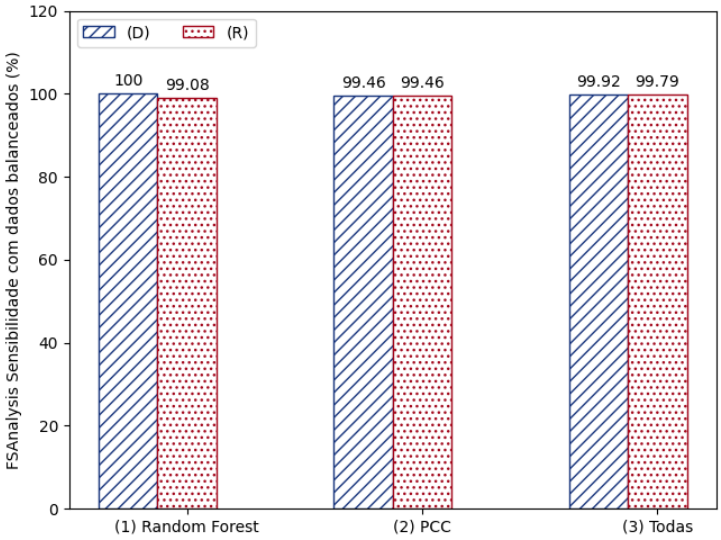


Figura 21 – Sensibilidade obtida utilizando o algoritmo *Logistic Regression* com bases balanceadas



5.2 Avaliação do *IIDEnsemble*

Para treinar o modelo foi feita uma redução nas bases de dados, detalhada na Seção 4.1, e testes foram realizados para mensurar a performance do modelo treinado sem essa redução (DM) e do modelo treinado com a redução (FM) utilizando a melhor abordagem encontrada no módulo *FSAnalysis*, ou seja, utilizando as *features* selecionadas com o cálculo PCC e utilizando as bases de dados balanceadas. A partir da matriz de confusão, algumas métricas foram utilizadas no presente trabalho para que a avaliação da performance do modelo treinado pudesse ser realizada, são elas acurácia; FAR; UR e sensibilidade, ou taxa de verdadeiro positivo.

A fórmula para o cálculo da acurácia está representada na Equação 4.1, e os resultados obtidos estão ilustrados na Figura 22, na qual é possível observar que os resultados tiveram uma diferença de 0.03% e 0.01%, respectivamente, para os *datasets* (D) e (R).O cálculo da FAR utilizado está representado na Equação 4.2 e os resultados obtidos foram apresentados na

Figura 23, que também obtiveram diferença de 0.03% para a base de dados (D) e 0.02% para a base (R).

Figura 22 – Acurácia do modelo proposto

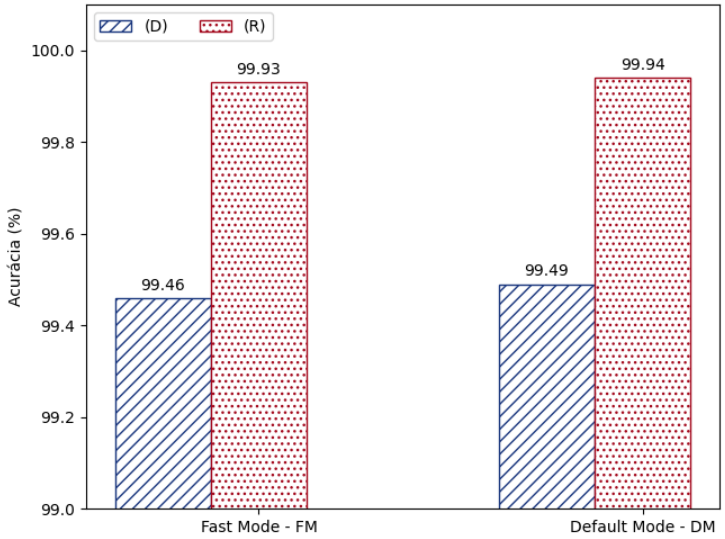
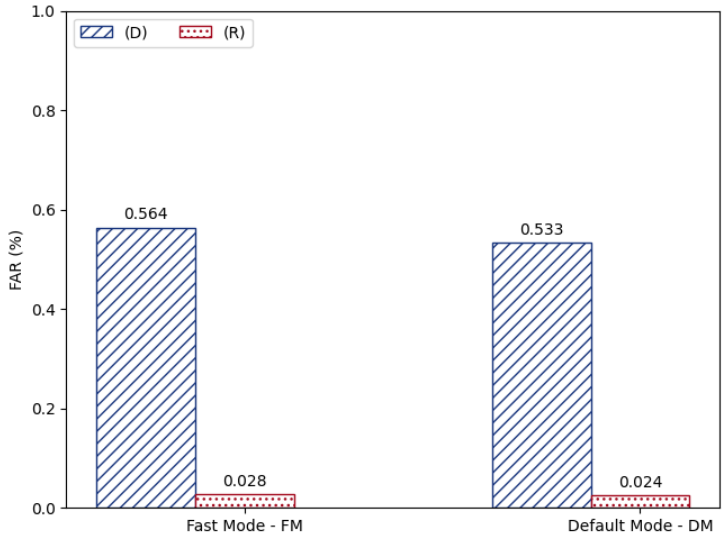


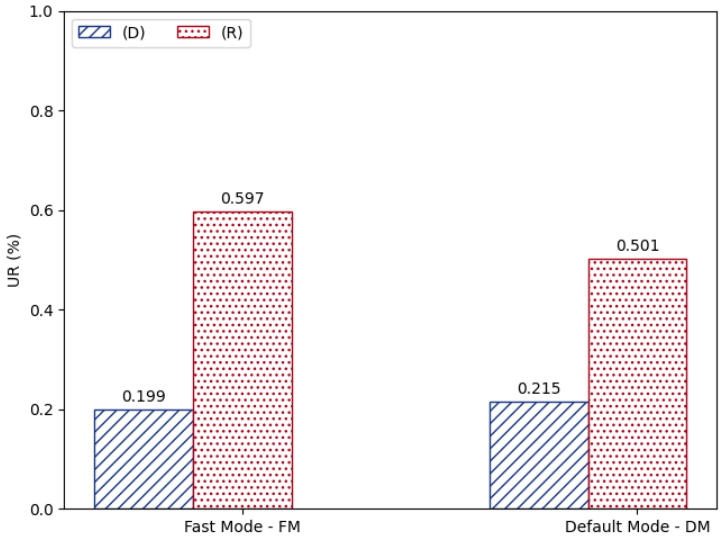
Figura 23 – FAR do modelo proposto



No contexto de redes industriais, a UR pode ser considerada mais importante que a FAR, pois essa métrica representa a fração de tráfego sob ataque que foi classificado erroneamente como fluxo normal, e pode ser calculada utilizando a Equação 4.3. Analisando o gráfico ilustrado na Figura 23 é possível observar que os resultados obtidos tiveram uma diferença maior nesse cálculo, apesar de ainda ser uma pequena diferença, o FM obteve 0.1% e 0.09% a mais que o DM na taxa de não detecção.

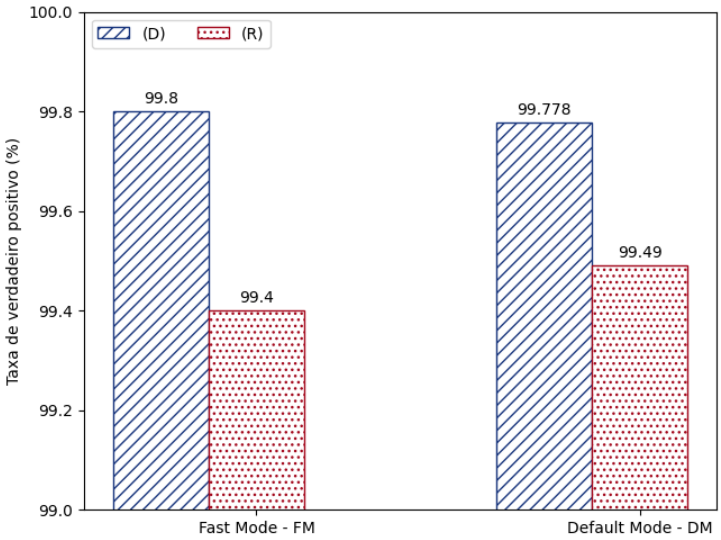
Foi avaliada a sensibilidade dos modelos, calculando a Taxa de Verdadeiro Positivo para mensurar a eficácia na predição de atividades anormais classificadas corretamente como ataques. Para isso, foi utilizada a Equação 4.5. Os resultados obtidos com esse cálculo foram ilustrados

Figura 24 – UR do modelo proposto



na Figura 25, na qual é possível observar que o FM obteve uma performance 0.03% melhor que o DM com o *dataset* (D).

Figura 25 – Taxa de verdadeiro positivo do modelo proposto



Como os dois modos obtiveram resultados similares em todos os cálculos realizados, apresentando uma diferença abaixo de 0.1% em 3 das 4 equações calculadas, tanto o tempo de aprendizagem quanto o tempo de predição também foram comparados, a fim de avaliar a performance dos modos. A Figura 26 ilustra o tempo, em horas, percorrido pelos modelos nos dois modos apresentados. Nesse cenário, o FM obteve vantagem, pois concluiu o treinamento do modelo em 1.34 hora, equivalente a aproximadamente 80 minutos, e em 0.92 hora, equivalente a aproximadamente 55 minutos, nas bases de dados (D) e (R) respectivamente, enquanto o DM levou 156 horas, o equivalente a aproximadamente 6 dias, para (D) e 128 horas, equivalente a aproximadamente 5 dias, para (R). Pensando em uma proposta na qual um modelo possa ser retreinado a cada novo ataque apresentado a ele, o FM pode ser uma boa escolha, por conta da

sua capacidade de treinamento em tempo reduzido. Além disso, analisando a comparação do tempo de predição dos dois modos, apresentada na Figura 27, o FM também se sobressai, uma vez que esse concluiu a predição em 0.0067% e 0.0065% a menos que o DM utilizando (D) e (R) respectivamente, esses valores podem parecer mínimos, porém podem ser cruciais em uma rede crítica, como a industrial.

Figura 26 – Tempo de aprendizagem do modelo proposto

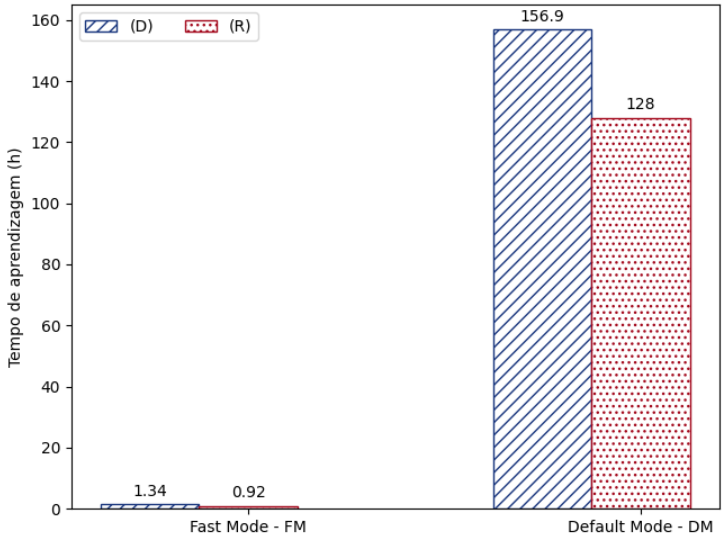
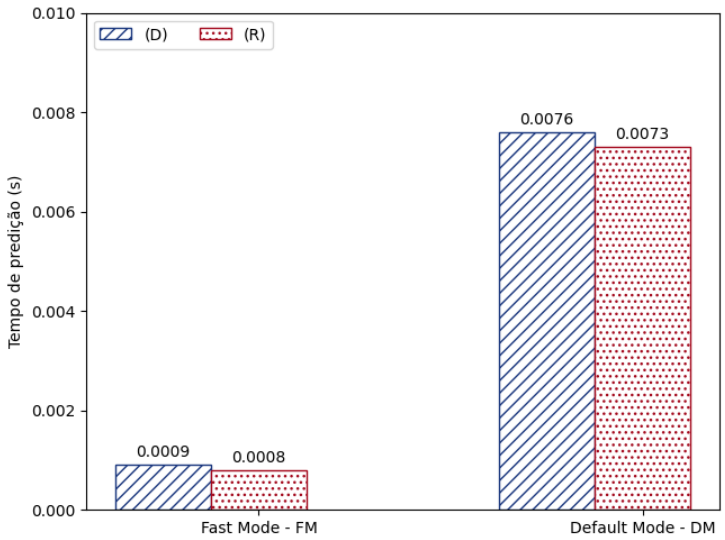


Figura 27 – Tempo de predição do modelo proposto



5.3 Conclusão

Neste capítulo foi discorrido sobre os resultados obtidos na execução do presente trabalho, os resultados apresentados foram calculados utilizando equações derivadas da matriz de confusão para analisar a performance do módulo *FSAnalysis* e treinamento do modelo criado, denominado *IIDEnsemble*, utilizando *Ensemble Learning*. Ao final da análise foi descrito que, considerando

um cenário no qual o modelo possa ser retreinado a cada novo ataque apresentado a ele, o *IIDEnsemble*, treinado com a redução das bases, será eficiente, uma vez que seu tempo de treinamento é consideravelmente menor que o tempo levado por modelos treinados sem a redução nas bases.

CAPÍTULO

6

DISCUSSÃO

No presente trabalho foram apresentados os métodos utilizados para analisar a melhor abordagem para o desenvolvimento de um modelo eficaz e robusto para ser utilizados em redes industriais utilizando ML. Utilizando *datasets* presentes na literatura foram abordadas questões como a escolha das melhores *features* para o treinamento do modelo e a comparação do treinamento realizado com *datasets* desbalanceados e balanceados.

Com os resultados desses experimentos, foi realizado o treinamento de um modelo de detecção de intrusão para redes industriais utilizando *Ensemble Learning*, considerando possíveis cenários onde o modelo deverá passar por um retreinamento a cada novo ataque encontrado, o que faz com que o tempo levado para o treinamento do modelo seja considerado. O modelo treinado que obteve os melhores resultados no presente trabalho, apresentou acurácia de 99.93% levando somente 1 hora e 34 minutos para seu completo treinamento, enquanto o modelo treinado sem os tratamentos realizados, obteve acurácia de 99.94% levando 156 horas para completar seu treinamento. Em termos de tempo de predição, o modelo que obteve melhores resultados conseguiu realizar as predições em 0.0009 segundos, enquanto o outro modelo levou 0.0076 segundos.

6.1 Contribuições

Os resultados obtidos deste trabalho são:

- Capítulo de livro
  - DALARMELINA, N. d. V.; ARORA, P.; KAUR, B.; MENEGUETTE, R. I.; TEIXEIRA, M. A. Using ml and dl algorithms for intrusion detection in the industrial internet of things. In: AI, Machine Learning and Deep Learning. CRC Press, 2023. p. 243–256. (DALARMELINA *et al.*, 2023)

- Conferência Internacional
  - DALARMELINA, N. d. V.; TEIXEIRA, M. A.; ANDRADE, F. R. H.; JÚNIOR, L. A. P.; FILHO, G. P. R.; MENEGUETTE, R. I. Fsanalysis: um mecanismo de seleção de features e análise considerando bases balanceadas e desbalanceadas. Iberian Conference on Information Systems and Technologies - CISTI, IEEE, 2022. (DALARMELINA *et al.*, 2022).

A autora do presente trabalho também teve colaboração no artigo publicado em um workshop nacional:

- SANTOS, H.; BARRETO, J.; DALARMELINA, N.; TEIXEIRA, M.; MENEGUETTE, R. Similitude de ocorrências de csam na internet e o registro perante às autoridades no estado de são paulo. In: Anais do V Workshop de Computação Urbana. Porto Alegre, RS, Brasil (SANTOS *et al.*, 2021).

REFERÊNCIAS

AHMAD, Z.; KHAN, A. S.; SHIANG, C. W.; ABDULLAH, J.; AHMAD, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. 2020. Disponível em: <https://doi.org/10.1002/ett.4150>. Citado nas páginas 22, 23 e 24.

AMBUSAIDI, M. A.; HE, X.; NANDA, P.; TAN, Z. Building an intrusion detection system using a filter-based feature selection algorithm. **IEEE Transactions on Computers**, v. 65, n. 10, p. 2986–2998, 2016. Citado nas páginas 18, 29, 31 e 32.

APRUZZESE, G.; COLAJANNI, M.; FERRETTI, L.; GUIDO, A.; MARCHETTI, M. On the effectiveness of machine and deep learning for cyber security. In: **2018 10th International Conference on Cyber Conflict (CyCon)**. [S.l.: s.n.], 2018. p. 371–390. Citado nas páginas 29, 31 e 32.

ARYA, L.; GUPTA, G. P. Ensemble filter-based feature selection model for cyber attack detection in industrial internet of things. In: IEEE. **2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)**. [S.l.], 2023. v. 1, p. 834–840. Citado nas páginas 30, 31 e 32.

BUCZAK, A. L.; GUVEN, E. A survey of data mining and machine learning methods for cyber security intrusion detection. **IEEE Communications Surveys Tutorials**, v. 18, n. 2, p. 1153–1176, 2016. Citado na página 30.

CARDEAL, E.; COUTINHO, B. **Modelos de Predição | Ensemble Learning**. 2019. Disponível em: <https://medium.com/turing-talks/turing-talks-24-modelos-de-prediÃğÃčo-ensemble-learning-aa02ce01afda>. Citado na página 27.

COHEN, J. Statistical power analysis. **Current directions in psychological science**, Sage Publications Sage CA: Los Angeles, CA, v. 1, n. 3, p. 98–101, 1992. Citado na página 37.

DALARMELINA, N. d. V. **IIDEnsemble**. 2023. Disponível em: <https://github.com/nicoledalarmelina/iidensemble>. Citado na página 40.

DALARMELINA, N. d. V.; ARORA, P.; KAUR, B.; MENEGUETTE, R. I.; TEIXEIRA, M. A. Using ml and dl algorithms for intrusion detection in the industrial internet of things. In: **AI, Machine Learning and Deep Learning**. [S.l.]: CRC Press, 2023. p. 243–256. Citado na página 57.

DALARMELINA, N. d. V.; TEIXEIRA, M. A.; ANDRADE, F. R. H.; JúNIOR, L. A. P.; FILHO, G. P. R.; MENEGUETTE, R. I. Fsanalysis: um mecanismo de seleção de features e análise considerando bases balanceadas e desbalanceadas. **Iberian Conference on Information Systems and Technologies - CISTI**, IEEE, 2022. Citado nas páginas 11, 34, 37 e 58.

DAS, A. Logistic regression. In: ____. **Encyclopedia of Quality of Life and Well-Being Research**. Cham: Springer International Publishing, 2020. p. 1–2. ISBN 978-3-319-69909-7. Disponível em: <https://doi.org/10.1007/978-3-319-69909-7_1689-2>. Citado na página 27.

FILHO, D. B. F.; JÚNIOR, J. A. S. Desvendando os mistérios do coeficiente de correlação de pearson (r). **Revista Política Hoje**, v. 18, n. 1, p. 115–146, 2009. Citado na página 27.

FOUNDATION, P. S. **Python**. 2021. Disponível em: <https://www.python.org/>. Citado nas páginas 26 e 45.

GALOV, N. **How Many IoT Devices Are There in 2021? [All You Need To Know]**. 2023. Disponível em: <https://techjury.net/blog/how-many-iot-devices-are-there/#gref>. Citado na página 17.

ISO/IEC 27000:2018(en). **Information technology — Security techniques — Information security management systems — Overview and vocabulary**. [S.l.], 2018. Disponível em: <https://www.iso.org/obp/ui/#iso:std:iso-iec:27000:ed-5:v1:en>. Acesso em: 27/05/2021. Citado na página 17.

JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. 2021. Disponível em: <https://doi.org/10.1007/s12525-021-00475-2>. Citado na página 24.

JOSE, S.; MALATHI, D.; REDDY, B.; JAYASEELI, D. A survey on anomaly based host intrusion detection system. **Journal of Physics: Conference Series**, IOP Publishing, v. 1000, n. 1, p. 012049, apr 2018. Disponível em: <https://dx.doi.org/10.1088/1742-6596/1000/1/012049>. Citado na página 22.

KESHK, M.; MOUSTAFA, N.; SITNIKOVA, E.; CREECH, G. Privacy preservation intrusion detection technique for scada systems. In: **2017 Military Communications and Information Systems Conference (MilCIS)**. [S.l.: s.n.], 2017. p. 1–6. Citado nas páginas 30, 31 e 32.

KORONIOTIS, N.; MOUSTAFA, N.; SITNIKOVA, B. T. E. **Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset**. [S.l.], 2018. Disponível em: <https://doi.org/10.48550/arXiv.1811.00701>. Citado na página 18.

LICENSE, B. (Ed.). **RandomForestClassifier**. BSD License, 2020. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>. Citado na página 26.

MAZURCZYK, W.; CAVIGLIONE, L. Cyber reconnaissance techniques. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 64, n. 3, p. 86–95, feb 2021. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/3418293>. Citado na página 21.

MELO, C. **Como lidar com dados desbalanceados?** 2019. Disponível em: <https://sigmoidal.ai/como-lidar-com-dados-desbalanceados/>. Citado nas páginas 25 e 26.

MOHY-EDDINE, M.; GUEZZAZ, A.; BENKIRANE, S.; AZROUR, M.; FARHAOUI, Y. An ensemble learning based intrusion detection model for industrial iot security. **Big Data Mining and Analytics**, TUP, v. 6, n. 3, p. 273–287, 2023. Citado nas páginas 30, 31 e 32.

MURINI, C. T. Análise dos sistemas de detecção de intrusão em redes: Snort e suricata comparando com dados da darpa. **UFSM, TCC, Janeiro**, 2014. Citado na página 22.

PAJOUH, H. H.; JAVIDAN, R.; KHAYAMI, R.; DEHGHANTANHA, A.; CHOO, K.-K. R. A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in iot backbone networks. **IEEE Transactions on Emerging Topics in Computing**, v. 7, n. 2, p. 314–323, 2019. Citado nas páginas 25, 26, 30, 31 e 32.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011. Citado nas páginas 36 e 38.

PETROSYAN, A. **How Many IoT Devices Are There in 2023? [All You Need To Know]**. 2023. Disponível em: <https://www.statista.com/statistics/617136/digital-population-worldwide/>. Citado na página 17.

PRIYANKA; KUMAR, D. Decision tree classifier: a detailed survey. 2020. Disponível em: <https://doi.org/10.1504/IJIDS.2020.108141>. Citado na página 25.

SANTOS, H.; BARRETO, J.; DALARMELINA, N.; TEIXEIRA, M.; MENEGUETTE, R. Similitude de ocorrências de csam na internet e o registro perante às autoridades no estado de são paulo. In: **Anais do V Workshop de Computação Urbana**. Porto Alegre, RS, Brasil: SBC, 2021. p. 209–222. ISSN 2595-2706. Disponível em: <https://sol.sbc.org.br/index.php/courb/article/view/17115>. Citado na página 58.

SARHAN, M.; LAYEGHY, S.; PORTMANN, M. **Towards a Standard Feature Set for Network Intrusion Detection System Datasets**. [S.l.], 2022. Citado na página 18.

SARKER, I. H.; ABUSHARK, Y. B.; ALSOLAMI, F.; KHAN, A. I. Intrudtree: a machine learning based cyber security intrusion detection model. **Symmetry**, MDPI, v. 12, n. 5, p. 754, 2020. Citado nas páginas 30, 31 e 32.

TAUNK, K.; DE, S.; VERMA, S.; SWETAPADMA, A. A brief review of nearest neighbor algorithm for learning and classification. In: **2019 International Conference on Intelligent Computing and Control Systems (ICCS)**. [S.l.: s.n.], 2019. p. 1255–1260. Citado na página 25.

TEIXEIRA, M.; ZOLANVARI, M.; KHAN, K.; JAIN, R.; MESKIN, N. Flow-based intrusion detection algorithm for supervisory control and data acquisition systems: A real-time approach. **IET Cyber-Physical Systems: Theory Applications**, 05 2021. Citado nas páginas 11 e 23.

TEIXEIRA, M. A.; SALMAN, T.; ZOLANVARI, M.; JAIN, R.; MESKIN, N.; SAMAKA, M. Scada system testbed for cybersecurity research using machine learning approach. **Future Internet**, v. 10, n. 8, 2018. ISSN 1999-5903. Disponível em: <https://www.mdpi.com/1999-5903/10/8/76>. Citado nas páginas 18, 30, 31, 32, 34, 36 e 41.

TRIPATHI, N.; HUBBALLI, N. Application layer denial-of-service attacks and defense mechanisms: A survey. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 54, n. 4, may 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3448291>. Citado na página 22.

ZARGAR, G. R.; BAGHAIE, T. *et al.* Category-based intrusion detection using pca. **Journal of Information Security**, Scientific Research Publishing, v. 3, n. 04, p. 259, 2012. Citado nas páginas 22, 30, 31 e 32.

ZHANG, C.; MA, Y. **Ensemble Machine Learning**. [S.l.]: Springer New York, NY, 2012. ISBN 978-1-4419-9326-7. Citado na página 19.

ZHANG, S.; LI, X.; ZONG, M.; ZHU, X.; WANG, R. Efficient knn classification with different numbers of nearest neighbors. **IEEE transactions on neural networks and learning systems**, IEEE, v. 29, n. 5, p. 1774–1785, 2017. Citado na página 25.

ZHOU, Z.-H. Ensemble learning. **Encyclopedia of biometrics**, v. 1, p. 270–273, 2009. Citado na página 27.

ZOLANVARI, M.; TEIXEIRA, M. A.; JAIN, R. Effect of imbalanced datasets on security of industrial iot using machine learning. In: IEEE. **2018 IEEE international conference on intelligence and security informatics (ISI)**. [S.l.], 2018. p. 112–117. Citado nas páginas 18, 19, 30, 31 e 32.


ICMC
USP
SÃO CARLOS