Pipeline para Detecção de Intrusão em Redes IoT utilizando Aprendizado de Máquina: Um Estudo de Caso

Nathália C. Martins[1], **Lorena da S. Medeiros** [1], **Lucelia L. Souza**[1], **César L. C. Mattos**[1], **Emanuel B. Rodrigues**[1]

[1]Universidade Federal do Ceará (UFC)
Av. da Universidade, 2853 – CEP 60020-181 – Fortaleza – CE – Brasil

Trabalho de Conclusão de Curso

Abstract. *The growth of IoT (Internet of Things) networks has made them the target of cyber attacks. Therefore, this article aims to propose a pipeline to analyze intrusion in IoT network traffic through Machine Learning techniques. To validate the proposed pipeline, a case study was performed using the IoT-23 database and three classification models: Decision Tree, Random Forest and CatBoost. The results demonstrated the effectiveness of the Random Forest model, which obtained a precision of 98.8% and a recall of 99.5%, and the contribution of attributes in the classification through SHAP. The discussions carried out throughout the article prove the need for well-founded methodologies to obtain reliable results.*

Resumo. *O crescimento das redes IoT (Internet of Things) fez com que elas se tornassem alvo de ataques cibernéticos. Por isso, este artigo tem como objetivo propor um pipeline para analisar intrusão em tráfego de redes IoT por meio de técnicas de Aprendizado de Máquina. Para validar o pipeline proposto, foi realizado um estudo de caso utilizando a base de dados IoT-23 e três modelos de classificação: Árvore de Decisão, Random Forest e CatBoost. Os resultados demonstraram a eficácia do modelo Random Forest, que obteve precisão de 98,8% e recall de 99,5%, e a contribuição de atributos na classificação por meio do SHAP. As discussões realizadas ao longo do artigo comprovam a necessidade de metodologias bem fundamentadas para se obter resultados confiáveis.*

1. Introdução

1.1. Contextualização

O termo Internet das Coisas (IoT - *Internet of Things*) foi proposto pelo pesquisador *Kevin Ashton*, que o cunhou em 1999 enquanto trabalhava na *Proctor & Gamble*. O conceito não se popularizou até 2009, quando um artigo sobre o tema foi publicado no *RFID Journal* conforme [Colombo et al. 2018].

Com o crescente número de dispositivos IoT conectados à rede, a segurança e o gerenciamento desses dispositivos são cada vez mais críticos. O tráfego de rede IoT é complexo e pode incluir diferentes tipos de dispositivos e protocolos de comunicação. A análise desse tráfego pode ser útil para detectar ameaças de segurança, problemas de

desempenho e anomalias no sistema. Dadas as vulnerabilidades presentes nas redes IoT, faz-se necessário o uso de Sistemas de Detecção de Intrusão (IDS - *Intrusion Detection System*). Para atender aos requisitos de um IDS eficaz, os pesquisadores exploraram a possibilidade de usar técnicas de aprendizado de máquina [Ahmad et al. 2021].

1.2. Objetivo Geral

Diante disso, este trabalho tem como objetivo propor um *pipeline* para a detecção de intrusão a partir da análise de tráfego de redes IoT utilizando técnicas de Aprendizado de Máquina. Como forma de experimentar e validar esse *pipeline*, foi realizado um estudo de caso utilizando um conjunto de dados IoT, a saber, a base *IoT-23*[1], que tem sido um recurso usado em múltiplos trabalhos [Chunduri et al. 2021, Singh 2020, Stoian 2020, Aragão et al. 2022]. As técnicas de aprendizado de máquina selecionadas para este estudo foram algoritmos baseados em árvores, uma vez que trabalhos como [Liang and Vankayalapati 2022] e [Stoian 2020] demonstraram que os mesmos obtiveram os melhores resultados. Assim, os algoritmos escolhidos foram Árvore de Decisão[2] e *Random Forest*[3], amplamente utilizados na literatura analisada, e *CatBoost*[4], que não foi mencionado em nenhum dos trabalhos relacionados, mas foi considerado para esse artigo pelo seu suporte nativo a atributos categóricos.

1.3. Justificativa

Na revisão bibliográfica realizada, notou-se muitas vezes uma carência de detalhamento dos aspectos metodológicos dos *pipelines* de aprendizado de máquina. Alguns artigos analisados não faziam qualquer menção ao processamento inicial nos dados, como [Booij et al. 2021] e [Kozik et al. 2021], ou não justificavam o porquê de certas escolhas, como atribuir determinado valor a dados faltantes, a exemplo de [Aragão et al. 2022]. Em outros casos, ações são justificadas sem fundamentação, como foi o caso de [Liang and Vankayalapati 2022], que descartou atributos alegando não ter impacto nos resultados sem se pautar em análises objetivas. Algumas inconsistências também foram encontradas na forma como os algoritmos de aprendizado de máquina foram aplicados, permitindo, por exemplo, vazamento de dados entre os conjuntos de treino e teste, como observado no código disponibilizado em [Aragão et al. 2022].

O estudo de caso apresentado neste trabalho irá comparar os três modelos supracitados considerando as seguintes métricas: acurácia, precisão, *recall* e *f1-score*, atribuídas com a média *'macro avg'*, na qual todas as classes contribuem igualmente para média final. Além disso, também será avaliada a interpretabilidade dos algoritmos por meio do *SHAP (SHapley Additive exPlanations)* [5], que é uma abordagem teórica de jogos para explicar a saída de qualquer modelo de Aprendizado de Máquina.

O restante do trabalho está dividido da seguinte forma: a Seção 2 apresenta a fundamentação teórica; a Seção 3 exibe os principais trabalhos relacionados encontrados na literatura; a Seção 4 descreve a metodologia e solução proposta; a Seção 5 detalha o

[1]https://www.stratosphereips.org/datasets-iot23
[2]https://scikit-learn.org/stable/modules/tree.html
[3]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[4]https://catboost.ai/en/docs/
[5]https://shap.readthedocs.io/en/latest/index.html

estudo de caso realizado, o *pipeline* proposto; a Seção 6 tem-se os resultados e discussões, por último, a Seção 7 apresenta as considerações finais e trabalhos futuros propostos.

2. Fundamentação Teórica

2.1. Segurança em IoT (IDS)

O conceito de Internet das Coisas (*IoT*, do inglês *Internet of Things*) está sendo amplamente difundido nos últimos anos, fazendo com que diversos dispositivos sejam conectados à rede, como por exemplo: dispositivos inteligentes ativados por voz, câmeras, TV's, campainhas, geladeiras e automóveis. Porém, alguns deles não foram projetados visando uma garantia de segurança, tornando-os suscetíveis a ataques cibernéticos.

Os dispositivos *IoT* podem receber e/ou enviar dados em períodos predeterminados ou contínuos. Somando isso ao crescente número de dispositivos conectados, infere-se que o fluxo de dados advindos deles é colossal, o que provoca uma demanda de armazenamento com proporcional dimensionamento, podendo, a depender das políticas de segurança adotadas, deixar pontos de vulnerabilidades expostos a pessoas mal intencionadas.

Segundo [Carvalho et al. 2022], erros cometidos por fabricantes de dispositivos que ainda não estão familiarizados com as práticas de segurança são uma das principais causas de riscos e vulnerabilidades. É necessário que haja, por parte desses fabricantes, um compromisso sério com os princípios que regem a segurança da informação, sendo eles: confidencialidade; integridade e disponibilidade.

Para [Boeckl et al. 2019], existem três objetivos de mitigação de alto risco e que estão totalmente relacionados com a pirâmide de segurança da informação (os três princípios supracitados), são eles:

Proteger a segurança do dispositivo: esse objetivo consiste em evitar que um dispositivo seja usado para realizar ataques (um *DDoS*, por exemplo).

Proteger a segurança de dados: esse objetivo consiste em garantir a confidencialidade, integridade e/ou disponibilidade dos dados coletados, armazenados, processados ou transmitidos para ou pelo dispositivo *IoT*.

Proteger a privacidade dos indivíduos: esse objetivo consiste em proteger a privacidade das pessoas afetadas pelo processamento de informações pessoalmente identificáveis.

Na busca de se atingir esses objetivos, várias técnicas têm sido investigadas e desenvolvidas, algumas delas baseadas em detecção de intrusões. Essa identificação pode ser feita por meio de Sistemas Detectores de Intrusões (*IDS*, no inglês, *Intrusion Detection System*), que analisam o tráfego de rede ou de registros de sistemas e de serviços buscando anomalias através de técnicas de Aprendizado de Máquina.

Pensando nisso, [SOUSA et al. 2016] chama atenção para o fato de que a existência de diferentes categorias de ataques pode implicar em diferentes atributos afetados. Ele conclui então que um atributo deve ser selecionado de acordo com o nível de contribuição que ele oferece na detecção de cada perfil de intrusão.

2.2. Ciência de Dados

De acordo com o NIST (*National Institute of Standards and Technology* ou Instituto Nacional de Padrões e Tecnologia), em [Chang and Grady 2019], Ciência de Dados (do inglês *Data Science*) é uma ciência empírica e que realiza o seu processo diretamento nos dados, permitindo o aprendizado a partir dos próprios dados. Consiste na extracão de conhecimento acionável diretamente dos dados, a partir de processos de descobertas, fomulação e testes de hipóteses.

[Chang and Grady 2019] também define o ciclo de vida dos dados como um conjunto de processos em uma aplicação capazes de transformar os dados brutos em conhecimento acionável. O processo analítico, por exemplo, que é uma das etapas desse ciclo, é definida como a síntese do conhecimento a partir da informação.

O campo de Ciência de Dados é multidiscipinar, e portanto, abrange diferentes disciplinas, como estatística, matemática, programação e conhecimento de negócios. Do mesmo modo, o seu ciclo de vida pode envolver técnicas, princípios e métodos de diferentes domínios como a extração e coleta de dados, pré-processamento dos dados como a limpeza e transformação de dados, análise exploratória, modelagem, que pode incluir o uso de Aprendizado de Máquina e visualização de dados.

Dada as suas vastas possibilidades de aplicação, os mais variados setores podem utilizar Ciência de Dados para extrair conhecimento a partir dos dados obtidos. No contexto de estratégias de segurança, por exemplo, em [Sarker et al. 2020] aponta que analistas de segurança podem descobrir informações valiosas, a partir da manipulação e análise de dados de segurança, utilizando técnicas de Ciência de Dados.

Ainda segundo [Sarker et al. 2020], o uso de metodologias de Ciência de Dados no contexto de segurança cibernética, como a compreensão do problema, coleta de dados de segurança de fontes distintas, preparo dos dados, além da construção e atualização do modelo orientado a dados, permite fornecer serviços de segurança inteligentes.

2.3. Aprendizado de Máquina

Aprendizado de Máquina (do inglês *Machine Learning*) é uma subárea da Inteligencia Artificial (IA) que busca desenvolver técnicas e algoritmos computacionais capazes de aprender e identificar padrões em dados e realizar tarefas com base em experiências acumuladas durante o treinamento [Mitchell 1997].

Existem várias técnicas de aprendizado de máquina, incluindo aprendizado supervisionado, onde os algoritmos são treinados usando pares de entrada e saída conhecidos, e aprendizado não supervisionado, onde os algoritmos tentam encontrar padrões em dados sem rótulos [Marquês 2022].

Basicamente dois tipos de problemas que são resolvidos por meio do Aprendizado de Máquina:

Regressão é usada para resolver problemas, nos quais o objetivo é prever um valor numérico contínuo [Harrison 2019]. Esses problemas podem ser resolvidos para prever valores contínuos relacionados ao tráfego, como a quantidade de dados transmitidos, a taxa de transferência de dispositivos, a carga da rede ou o consumo de largura de banda. Essas previsões são úteis para otimizar o dimensionamento de recursos da rede, melho-

rar a eficiência da comunicação entre dispositivos *IoT* e prever demandas futuras para planejar a infraestrutura de rede adequada.

Classificação outro tipo de problema comum resolvido pelo aprendizado de máquina é a classificação no qual um algoritmo é treinado para classificar os dados de entrada em variáveis discretas [Grus 2016], podendo ser dividida em (classificação binária) ou em várias classes (classificação multiclasse). Em relação à classificação, problemas de tráfego de redes *IoT* podem envolver a categorização de dispositivos, pacotes de dados ou fluxos de tráfego em diferentes classes ou tipos. Por exemplo, a classificação pode ser usada para identificar o tipo de dispositivo *IoT* (sensores, atuadores, câmeras, etc.) ou para detectar padrões de tráfego anômalos, como ataques cibernéticos ou comportamento malicioso. Essa classificação é valiosa para a segurança da rede *IoT* e para a identificação e mitigação de possíveis ameaças ou falhas na comunicação.

O aprendizado de máquina tem aplicações em diversas áreas, como medicina, finanças, automação industrial e muito mais. À medida que a quantidade de dados disponíveis cresce exponencialmente, o aprendizado de máquina se torna cada vez mais relevante, permitindo que sistemas computacionais tomem decisões mais precisas e eficientes com base em informações relevantes extraídas dos dados.

3. Trabalhos Relacionados

Em seguida, serão discutidos de forma breve os artigos que apresentaram maior relação com o tema abordado neste trabalho, considerando detecção de intrusão em redes IoT e aprendizado de máquina.

No trabalho de [Chunduri et al. 2021], foi realizado um estudo para detecção de *Botnet IoT* por meio de classificação multiclasse. Para isso, o trabalho utilizou dois conjuntos de dados recentes que tratam desse assunto: *IoT-23* e *MedBIoT*[6]. Foram consideradas seis variantes de ataques de *Botnet IoT* de ambos os conjuntos de dados, que foram categorizadas em 3 classes, a saber, *Mirai, Bashlite e Torii*. Os experimentos foram realizados com diferentes algoritmos de classificação, sendo eles: *Random Forest*, *Gradient Boost - GBM*, *K-nearest neighbors - KNN* e *Support Vector Machine - SVM*. O *Random Forest* superou todos os outros com uma acurácia de 99,88%.

O estudo de [Hegde et al. 2020] apresentou uma abordagem baseada em aprendizado de máquina para a detecção de atividades de *botnet* em tráfego de rede IoT. Ele propõe um *pipeline* de detecção que inclui extração de características, seleção de características, pré-processamento e treinamento de modelos. Foram utilizados cinco algoritmos de aprendizado de máquina diferentes: *Naive Bayes*, KNN, Árvore de Decisão, *Random Forest* e *SVM*. Os resultados mostraram que a combinação de características estatísticas e de fluxo permitiu uma acurácia de detecção acima de 99%, independentemente do algoritmo de classificação utilizado. Além disso, o estudo comparou o desempenho dos modelos em diferentes condições de tráfego de rede e mostrou que eles são capazes de detectar com eficácia atividades de *botnet* em condições realistas.

Em [Stoian 2020], o autor abordou a usabilidade de algoritmos de aprendizado de máquina na detecção de anomalias de forma supervisionada em redes IoT utilizando a versão menor do conjunto de dados *IoT-23*. Os algoritmos implementados foram *Ran-*

[6]https://cs.taltech.ee/research/data/medbiot/

dom Forest, Naive Bayes, AdaBoost, Perceptron Multicamadas e *SVM*. Foi considerada uma abordagem de classificação multiclasse, na qual o algoritmo *Random Forest* obteve as melhores métricas, dentre elas, uma acurácia de 99,5%. No pré-processamento, os dados passaram pela etapa de seleção, visualização, padronização e codificação dos rótulos, além da aplicação de correlação estatística visando a eliminação de atributos não relacionados com o rótulo. Em função de limitações técnicas, a base foi dividida aleatoriamente em 4 partes e os algoritmos foram aplicados em cada uma delas. E para analisar o desempenho dos modelos, utilizou-se a média das métricas obtidas em cada partição. Por esse motivo, como trabalho futuro, o estudo sugere que o experimento seja refeito utilizando a base completa e os rótulos originais.

[Singh 2020] abordou o desenvolvimento de um modelo de aprendizado de máquina para classificação binária do tráfego de rede IoT, indicando se é maligno ou benigno. Para realizar esse objetivo, foi utilizada uma versão menor do conjunto de dados *IoT-23*, o algoritmo *Random Forest* e um meta-estimador que combina três classificadores, a saber, *Random Forest, Naive Bayes Gaussiano* e *KNN*, utilizando votação forçada para a decisão final. A qualidade do modelo foi avaliada por meio da acurácia e da matriz de confusão.

[Liang and Vankayalapati 2022] propõem em seu trabalho o uso de alguns métodos de aprendizado de máquina e aprendizado profundo para analisar os dados da base *IoT-23*. Os modelos avaliados nesse estudo foram: *Naive Bayes*, SVM, Árvore de Decisão e Redes Neurais Convolucionais. A principal métrica utilizada para avaliar os modelos citados foi a acurácia, mas o desempenho de cada um deles também foi analisado por meio de seus tempos de execução. Dessa forma, os resultados apontaram que o melhor algoritmo foi a Árvore de Decisão, com uma acurácia de 73% e um tempo de execução, considerando treino e teste, de apenas 3 segundos. O algoritmo que obteve o pior desempenho foi o *Naive Bayes* com apenas 30% de acurácia, enquanto o que apresentou o maior tempo de execução foi o *SVM*, com 5849 segundos.

Baseando-se nos trabalhos de [Liang and Vankayalapati 2022] e [Stoian 2020], [Aragão et al. 2022] propôs uma abordagem baseada em aprendizado de máquina onde 20 algoritmos foram testados com o conjunto de dados *IoT-23*. Sua principal métrica de avaliação foi a acurácia, mas, a exemplo de [Liang and Vankayalapati 2022], considerou o tempo de treino em sua análise. Uma diferença de seu trabalho foi o enfoque no pré-processamento dos dados e na otimização dos modelos. Os resultados obtidos apontaram que os modelos Árvore de Decisão, *Extra Trees* e *Random Forest* apresentaram os melhores desempenhos, tendo todos alcançado 99,86% de acurácia. Quanto ao tempo de treinamento, houve um empate de nove algoritmos, todos com 100 treinos em uma hora. Dentre eles estavam a Árvore de Decisão e algoritmos derivados do *Naive Bayes*.

Nosso trabalho destaca-se em relação aos demais por abordar o detalhamento de informações sobre etapas importantes, como o pré-processamento dos dados, a implementação do modelo *CatBoost* na detecção de intrusão em redes IoT e a interpretação da contribuição dos atributos por meio do SHAP. Enquanto outros trabalhos não mencionam o processamento inicial dos dados ou não justificam suas escolhas, este estudo ressalta a importância de preencher essas lacunas. Além disso, foram identificadas inconsistências na aplicação dos algoritmos nos trabalhos relacionados, como vazamento de dados entre conjuntos de treino e teste. Essa identificação ressalta a ne-

cessidade de uma abordagem de detecção de intrusão em redes IoT que usa aprendizado de máquina com uma metodologia mais rigorosa e fundamentada, tornando este trabalho uma contribuição relevante.

4. Metodologia e Solução Proposta

Visando fornecer um passo a passo consistente para detectar intrusão em tráfego de redes IoT por meio de aprendizado de máquina, propõe-se um *pipeline* com quatro etapas: i) extração dos dados; ii) pré-processamento; iii) seleção e otimização dos modelos; e iv) avaliação e interpretação dos modelos. O fluxograma dessas etapas está sintetizado na Figura 1. Nas subseções a seguir detalhamos cada uma das etapas.

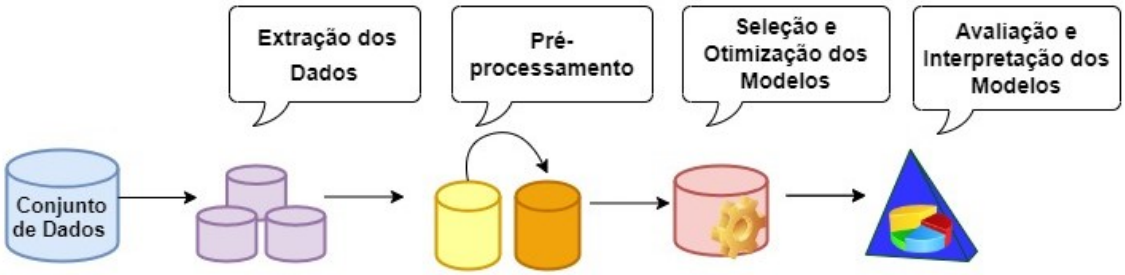

Figura 1. Fluxograma da metodologia empregada no pipeline proposto.

4.1. Conjunto de dados

Nos últimos anos, muitos conjuntos de dados no domínio de segurança para IoT foram propostos, cada um com suas vantagens e desvantagens. As bases de dados IoT são conjuntos de dados coletados a partir de dispositivos conectados à internet, que abrangem uma ampla gama de aplicações e setores. Essas bases de dados são essenciais para a compreensão e a análise do mundo conectado, fornecendo informações valiosas para diversas áreas de pesquisa, em particular para o treinamento e avaliação de sistemas de detecção de intrusão baseados em aprendizado de máquina. São exemplos de bases voltadas para esse contexto: *IoT-23*, utilizada para análise de tráfego de rede IoT, e *MedBIoT*, focada em aplicações de IoT na área da saúde.

4.2. Extração dos dados

A etapa de extração de dados consiste no processo de obtenção de dados relevantes para o contexto proposto. Para dados provenientes de múltiplas fontes, faz-se necessária a criação de regras para gerar uma base única. Nesses casos, é necessário o uso de artifícios para se extrair dados brutos e transformá-los em um conjunto de dados que poderá ser utilizado posteriormente para o problema de interesse.

4.3. Pré-processamento

Com os dados coletados, a próxima etapa é o seu pré-processamento. Essa etapa consiste na estruturação e organização do conjunto de dados brutos por meio dos seguintes processos: i) inspeção dos atributos; ii) limpeza dos dados; e iii) transformação dos dados.

A tarefa de inspeção inicial dos atributos consiste na avaliação semântica de cada um deles. Isso permite a deleção de algum atributo devido à sua baixa contribuição ou

redundância para o contexto do problema que está inserido. Já a tarefa de limpeza de dados refere-se ao processo de padronização de dados, bem como o tratamento de valores ausentes e a eliminação de rótulos. Por sua vez, na tarefa de transformação de dados é feita a conversão do tipo de variável, assim como a codificação de dados, como por exemplo transformar *strings* para valores numéricos. Nessa etapa, técnicas como *one-hot encoding* e *feature hashing* podem ser usadas.

Tanto o processo de limpeza quanto o de transformação de dados são importantes para adequação da base de dados ao modelo de aprendizado de máquina que será implementado. Por esse motivo, elas podem variar e ter demandas específicas de acordo com o tipo de algoritmo que será executado. Logo, alguns processos imprescindíveis para um determinado modelo podem não ser essenciais para outros.

4.4. Seleção e Otimização dos Modelos

Nesta etapa, determina-se o tipo de classificação a ser abordada (binária ou multiclasse). Depois, selecionam-se modelos de aprendizado de máquina a serem utilizados que desempenhem bem na abordagem escolhida. Como forma de evitar o vazamento de dados, sugere-se o uso da validação cruzada aninhada, que divide os dados múltiplas vezes em treinamento, validação e teste.

Nessa técnica, há dois *loops*, um interno e outro externo. No *loop* interno, ajusta-se cada modelo a partir do subconjunto de treino, selecionando os hiperparâmetros com um outro subconjunto de validação. O *loop* externo, a partir da média das métricas obtidas em cada partição de teste, determina a capacidade de generalização do modelo com os melhores hiperparâmetros. Na composição do *loop* externo, pode-se usar, por exemplo, métodos como o *StratifiedKFold*[7], e no *loop* interno métodos como *GridSearchCV*[8] ou *RandomSearchCV*[9], todos da biblioteca *Scikit-Learn*[10].

4.5. Avaliação e Interpretação dos Modelos

Para avaliar o desempenho dos modelos, utilizam-se métricas de avaliação comumente usadas em problemas de classificação, como precisão, *recall*, *f1-score* e acurácia. Essas métricas mostram a relação entre a taxa de acertos e erros proporcionadas pela classificação do modelo, a saber: Verdadeiro Positivo (VP), Verdadeiro Negativo (VN), Falso Positivo (FP) e Falso Negativo (FN).

Importante mencionar que, ao calcular essas métricas em problemas de classificação multiclasse, podemos usar valores médios ponderados ou não ponderados para fornecer uma medida geral do desempenho do modelo em todas as classes. Além disso, é possível usar outros recursos específicos para problemas de classificação multiclasse, como a matriz de confusão, que fornece uma visão mais detalhada dos resultados obtidos para cada classe.

Após a aplicação dos modelos de classificação, pode-se utilizar a técnica *SHAP* para interpretar e compreender a relevância dos atributos do conjunto de dados no momento da predição de classes realizada por cada algoritmo. O *SHAP* é um método de

[7]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html
[8]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[9]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html
[10]https://scikit-learn.org/stable/index.html

explicabilidade que fornece uma visão detalhada de como cada atributo contribui para a tomada de decisão dos modelos. Ele atribui um valor de importância a cada atributo, permitindo ranqueá-los de acordo com seu grau de relevância.

5. Estudo de caso

O estudo de caso deste trabalho consiste em aplicar o *pipeline* proposto na seção anterior na base de dados de tráfego de rede *IoT-23* para detectar múltiplas classes de ataque utilizando modelos de Aprendizado de Máquina baseados em árvore. Os seguintes algoritmos foram escolhidos: *CatBoost*, Árvore de Decisão e *Random Forest*.

Os experimentos foram realizados em computadores com processador de 11ª geração do Intel Core i7-11800H, CPU 4.60GHz, 16 GB de RAM DDR3 e sistema operacional *Windows 11 Pro 64 bits*. Além disso, foram utilizados os ambientes de desenvolvimento *Jupyter Notebook*, *Google Colab* e *Visual Studio Code*. *Python* foi usado como linguagem de programacão, juntamente com as bibliotecas *Numpy, Pandas, SHAP, Scikit-Learn, Seaborn e Matplotlib* onde estão disponíveis em repositório GitHub[11].

5.1. Conjunto de dados

A base *IoT-23*, também chamada de *Aposemat IoT-23*, foi criada especificamente para fins de pesquisa e desenvolvimento de técnicas de detecção e classificação de intrusão em tráfego de rede IoT. Essa base de dados apresenta uma diversidade de protocolos de comunicação e tipos de tráfego de rede associados aos dispositivos IoT. Esses registros de tráfego incluem informações sobre o fluxo de dados, como endereços IP de origem e destino, portas de comunicação, protocolos utilizados, duração das transações, tamanho dos pacotes, entre outros atributos relevantes.

Neste experimento, foi utilizada a base de dados que é composta por 23 capturas (ou cenários) de tráfego de rede de dispositivos IoT, sendo 20 capturas de *malware* e 3 capturas de tráfego benigno. Essas capturas foram realizadas no Laboratório Stratosphere, grupo AIC, FEL, Universidade CTU, República Tcheca, entre 2018 e 2019, tendo sido a base publicada pela primeira vez em 2020.

Os cenários maliciosos foram obtidos executando-se uma amostra de *malware* em um *Raspberry Pi*, enquanto o tráfego de rede benigno foi obtido a partir de três dispositivos IoT, a saber: uma lâmpada LED inteligente *Philips HUE*, um assistente pessoal inteligente doméstico *Amazon Echo* e uma fechadura inteligente *Somfy*.

Há duas versões da base *IoT-23* disponíveis para *download*: uma completa, de 21 GB, que contém os arquivos *.pcap* e *conn.log.labeled* originais; e outra mais leve, de 8,8 GB, contendo apenas o arquivo *conn.log*. Neste trabalho, utilizou-se a versão menor com instâncias rotuladas.

5.2. Aplicação do *Pipeline* proposto

De acordo com a metodologia explicitada anteriormente, esta subseção detalhará o estudo de caso, abordando a aplicação do *pipeline* proposto e informando as tarefas que foram executadas em cada etapa apresentada na Figura 1.

[11] https://github.com/Projeto-UFC-SiDi/projeto-ufc-sidi

5.2.1. Etapa 1: Extração de dados

Primeiramente, os arquivos do tipo *conn.log.labeled*, relativos às capturas que compõem a base *IoT-23*, foram colocados em pastas distintas. Após isso, foi desenvolvido e executado um *script* em *Python* visando a obtenção de dados de cada uma das pastas e sua posterior concatenação em uma base única. Desta forma, foi possível a obtenção de um arquivo *CSV* com um conjunto de dados de 1.444.674 linhas e 21 colunas.

5.2.2. Etapa 2: Pré-processamento dos dados

A primeira tarefa da etapa de pré-processamento foi a inspeção de atributos. Nesta etapa, 7 atributos foram excluídos após análise semântica: *ts, uid, id.orig_h, id.resp_h, local_orig, local_resp, resp_pkts*.

Os dois primeiros atributos são relativos ao *timestamp* e ao *id* da captura. Foram dispensados por serem considerados de baixa relevância para a análise do tráfego de rede, pois são dados relativos ao processo de captura e não a características do tráfego em si.

Do mesmo modo, os endereços IP envolvidos no processo de captura, *id.orig_h* e *id.resp_h*, também foram desconsiderados, por se entender que tratam-se de parâmetros específicos do cenário original considerado na criação da base de dados *IoT-23*. Se esses atributos fossem usados para treinar o modelo de aprendizado de máquina, poderia ocorrer um *overfitting* dos dados, pois os modelos poderiam ficar condicionados apenas a esse cenário e o IDS não poderia ser generalizado para ser usado em outros ambientes. Além disso, em um cenário de ataque, os IPs de origem podem ser forjados.

Seguindo essa mesma lógica de evitar parâmetros que sejam específicos do cenário, os atributos que informam se a conexão e a resposta foram originadas localmente também foram dispensados: *local_orig e local_resp*. Por fim, o atributo referente ao número de pacotes enviado do dispositivo (*resp_pkts*) foi dispensado por ser proporcional ao atributo correspondente ao número de pacotes enviados para o dispositivo (*orig_pkts*), podendo ser um ponto de redundância para a classificação.

A segunda tarefa da etapa de pré-processamento foi a limpeza de dados, onde foi realizada uma padronização dos rótulos que caracterizavam o tipo de ataque ou a ausência dele (tráfego benigno/normal). Por exemplo, rótulos com os nomes *'- Benign -'* e *'(empty) Benign -'* foram mudados para *'-Benign-'*. Após a padronização dos rótulos, a base passou a ter 12 classes: *PartOfAHorizontalPortScan, Okiru, Benign, DDoS, C&C, Attack, C&C-HeartBeat, C&C-FileDownload, C&C-Torii, FileDownload, C&CHeartBeatFile-Download e C&C-Mirai*. O número total de instâncias desses rótulos variavam de 1 unidade a um valor acima de 825.000, evidenciando que o conjunto de dados era bastante desbalanceado.

Com o objetivo de criar um modelo mais robusto para a classificação de ataques, optou-se pela eliminação dos rótulos que possuíam menos de 50 instâncias, resultando na remoção de 5 deles. A base de dados final que foi utilizada para o treinamento e teste dos modelos continha os 7 seguintes rótulos: *Attack*, *Benign*, *C&C*, *C&C-HeartBeat*, *DDoS*, *Okiru* e *PartOfAHorizontalPortScan*, cujas descrições são apresentadas na Tabela 1.

A terceira tarefa da etapa de pré-processamento foi a transformação/codificação de

Tabela 1. Descrição das classes da base de dados utilizada neste trabalho

Classes	Descrição
Attack	Rótulo genérico atribuído a anomalias que não podem ser identificadas.
Benign	Rótulo genérico atribuído a capturas que não são suspeitas.
C&C	Ataque que assume o controle do dispositivo para ordená-lo a realizar vários ataques no futuro.
C&C-HeartBeat	O servidor que controla o dispositivo infectado envia mensagens periódicas para verificar o status do dispositivo infectado.
DDoS	O dispositivo infectado está executando um ataque de negação de serviço distribuída.
Okiru	O ataque é realizado pela *botnet* Okiru.
PartOfAHorizontalPortScan	As informações são coletadas de um dispositivo para um ataque futuro.

dados. O *CatBoost*, um dos modelos de aprendizado de máquina avaliados neste estudo de caso, consegue lidar nativamente com atributos categóricos ou ausentes por meio do parâmetro *cat_features*. Com este parâmetro exclusivo do *CatBoost*, foi possível especificar uma lista de atributos do tipo categórico como argumento, facilitando a execução do modelo e evitando novas etapas para transformar estes atributos. Isso porque o *CatBoost* utiliza o *one-hot encoding* por padrão para estes recursos, tornando desnecessária outra etapa para codificação destes dados categóricos. Nessa etapa, 6 atributos foram passados para o parâmetro *cat_features* e, sendo assim, transformados automaticamente: *history, service, proto, conn_state, id.orig_p e id.resp_p*.

Para os outros modelos testados neste estudo de caso (*Random Forest* e Árvore de Decisão), no intuito de replicar os mesmos efeitos que o *CatBoost* proporciona com o *cat_features*, foram utilizados alguns métodos para transformação e codificação de dados. Para os atributos *duration, orig_bytes e resp_bytes*, o tipo de variável foi transformado para que posteriormente os valores ausentes, os quais eram representados pelo caractere '-' na base de dados original, pudessem ser substituídos pela mediana dos dados de treino. Além disso, para os atributos categóricos *history, service, proto e conn_state*, a técnica *one-hot encoding* foi escolhida para transformar e codificar os dados.

Por fim, foi necessário tratar também os atributos *id.orig_p e id.resp_p*, que são relativos às portas de origem e destino utilizadas pelos protocolos de aplicação nas conexões. A forma mais correta de considerar esses dois atributos nos modelos de aprendizado de máquina é tratando-os como variáveis do tipo categórico, e não numérico. Além disso, foi observado na base de dados *IoT-23* que havia muitas instâncias com valores diferentes de portas. Optou-se por utilizar a técnica *feature hashing* no *pipeline* para transformar esses dois atributos das portas, pois a mesma cumpre a função de lidar com o dado categórico e é também considerada a mais adequada pela sua eficiência para atributos com muitas instâncias distintas. Por conta de recursos computacionais, o parâmetro *n_features* foi configurado como 2^5, significando que cada um dos atributos de porta foi

codificado em 32 posições.

5.2.3. Etapa 3: Seleção e Otimização dos Modelos

A técnica de validação cruzada aninhada foi implementada utilizando a função *StratifiedKFold* do *Scikit-Learn* para construção do *loop* externo e o *GridSearchCV* para o *loop* interno. No total foram executados cinco *folds* para cada *loop*. Cada subconjunto de treino foi usado para ajustar os hiperparâmetros dos algoritmos em cada iteração. Para isso, foram repassadas configurações de hiperparâmetros para o argumento *param_grid* do *GridSearchCV* considerando cada algoritmo.

Para o *DecisionTreeClassifier*, utilizou-se *max_depth* = ['None', 5, 10] e *min_samples_split* = [2, 5, 10]. Os valores testados para o *CatBoostClassifier* foram *iterations* = [100], *learning_rate* = [1, 0.1, 0.01], *depth* = ['None', 4, 8, 10] e *loss_function* = ['MultiClass']. Para o *RandomForestClassifier*, utilizou-se *n_estimators* = [100, 200], *max_depth* = [3, 5, 30] e *min_samples_split* = [2, 5]. Os melhores estimadores obtidos foram utilizados para predizer os rótulos das instâncias do subconjunto de teste, e as métricas de avaliação foram armazenadas para um posterior cálculo de média.

5.2.4. Etapa 4: Avaliação e Interpretação dos Modelos

No presente artigo, a interpretação de modelos de aprendizado de máquina é feita usando a biblioteca *SHAP*, que permite a visualização dos atributos mais importantes na tomada de decisão do modelo. O *SHAP* foi utilizado em todos os modelos analisados, sendo necessário criar um objeto *TreeExplainer* passando o melhor modelo treinado e o conjunto de dados de treinamento. O parâmetro *feature perturbation* foi definido como *tree path dependent* para garantir que a técnica *SHAP* seja aplicada corretamente para modelos com divisões categóricas.

Em seguida, os valores *SHAP* podem ser calculados para um conjunto de dados de teste usando o método *shap values* do objeto *TreeExplainer*. Isso gera uma matriz de valores *SHAP* que indicam a contribuição de cada atributo para a saída do modelo em cada instância de teste. Finalmente, é possível gerar uma visualização dos valores *SHAP* através da plotagem de um gráfico de resumo usando a função *summary plot*.

As métricas de avaliação dos modelos foram calculadas por meio da função *classification_report* da biblioteca *Scikit-Learn*. Essas métricas fornecem uma medida da qualidade e desempenho dos modelos de classificação. A função *classification_report* é chamada passando como argumentos os rótulos verdadeiros *(y_test)* e os rótulos preditos pelo modelo *(y_pred)*. O parâmetro *output_dict = 'True'* é utilizado para retornar um dicionário contendo as métricas calculadas.

Em seguida, o nosso código extrai desse dicionário as médias das métricas acurácia, precisão, *recall* e *F1-score* calculadas para cada classe, as quais são armazenadas em listas separadas para cada *fold*. Por fim, são calculadas as médias finais das 4 métricas considerando todos os *folds*.

Nessa etapa de avaliação dos modelos, ao importar as bibliotecas *Seaborn* e *Mat-*

plotlib, foi gerada uma matriz de confusão para cada *fold* durante a validação cruzada aninhada, criando uma lista para armazená-las. Vale ressaltar que as matrizes de confusão foram preenchidas usando *np.pad*, para garantir que todas as matrizes tivessem a mesma dimensão.

6. Resultados e Discussões

Nesta seção são apresentados e discutidos os resultados obtidos pelos classificadores no estudo de caso de detecção de intrusão em redes *IoT* descrito anteriormente, levando em conta aspectos de desempenho e interpretabilidade dos modelos de Aprendizado de Máquina testados.

6.1. Métricas

Quando se trata de IDS, é importante analisar métricas que mostrem o desempenho dos classificadores quanto à quantidade de falsos positivos (FP) e falsos negativos (FN). Quando o tráfego benigno é classificado como malicioso, caracterizando um FP, o IDS pode barrar esse tráfego por acreditar ser um risco para o sistema. Esses falsos alarmes podem sobrecarregar o time de segurança que deverá analisar todos os alarmes. Mas o oposto também é prejudicial, ou seja, o IDS pode considerar o tráfego malicioso como benigno, caracterizando um FN. Nesse caso, o ataque cibernético vai ser bem sucedido e o sistema pode ser comprometido, incorrendo em perdas severas. Enquanto a acurácia mede o desempenho geral dos classificadores, as métricas que representam de forma mais fidedigna os efeitos de FP e FN são a precisão e o *recall*, respectivamente. O *F1-Score* é uma média harmônica entre a precisão e o *recall*.

A Tabela 2 apresenta as médias e desvios-padrões das métricas de todos os classificadores testados. De um modo geral, todos os algoritmos se saíram muito bem na tarefa de detecção de ameaças. Todos apresentaram altos índices de acurácia, com o *Random Forest* apresentando um valor um pouco inferior. No entanto, no nosso entender, as métricas de precisão e *recall* são mais importantes para o problema de detecção de intrusão, pois as mesmas levam em consideração os falsos positivos e falsos negativos de maneira mais precisa. Considerando essas duas métricas em conjunto, o algoritmo *Random Forest* se sobressai frente aos demais, atingindo uma precisão de 98,8% e um *recall* de 99,5%. A Árvore de Decisão também se mostrou uma ótima opção de classificador no presente estudo de caso. É interessante observar um custo-benefício apresentado pelo algoritmo *CatBoost*. Ele se mostrou muito fácil de usar devido à sua habilidade de lidar com dados categóricos e dados faltantes, mas no entanto foi o pior algoritmo em termos de *recall*, não conseguindo tratar tão bem os casos de falsos negativos.

Tabela 2. Média e desvio-padrão das métricas obtidas para cada modelo

Algoritmos	Acurácia	Precisão	*Recall*	*F1-Score*
Random Forest	0,988 +- 0,0	0,988 +- 0,001	0,995 +- 0,001	0,991 +- 0,001
Árvore de Decisão	0,991 +- 0,0	0,988 +- 0,002	0,994 +- 0,001	0,991 +- 0,001
CatBoost	0,991 +- 0,0	0,988 +- 0,003	0,925 +- 0,009	0,946 +- 0,008

6.2. Matriz de Confusão

A matriz de confusão é uma tabela usada para avaliar o desempenho de um algoritmo de classificação. Na diagonal principal da matriz, encontram-se os valores que representam as classificações corretas, enquanto fora dela, os valores indicam os erros de classificação.

A fim de analisar o desempenho do IDS referente às classes de ataques individuais, a matriz de confusão média do modelo *Random Forest* é apresentada na Figura 2. Mais uma vez comprova-se a baixa ocorrência de erros de detecção, evidenciada pelos altos valores na diagonal principal da matriz. No entanto, é importante ressaltar que o algoritmo *Random Forest* se confundiu algumas vezes ao considerar um ataque do tipo *PartOfAHorizontalPortScan* como tráfego benigno. De fato, observa-se no mundo real que ataques do tipo varredura de portas têm se tornado cada vez mais difíceis de detectar devido à habilidade de camuflagem dos atacantes.

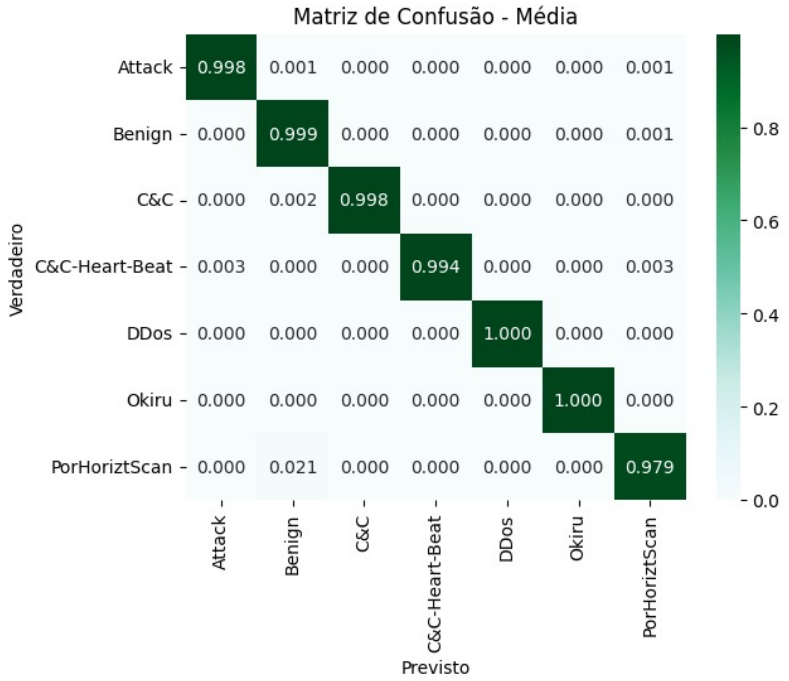

Figura 2. Matriz de confusão média do modelo *Random Forest*

O resultado da matriz de confusão média do modelo Árvore de Decisão é apresentada na Figura 3 e demonstra baixa ocorrência de erros de detecção, refletidos pelos altos valores na diagonal principal. Porém, houve confusão em alguns casos, classificando ataques *PartOfAHorizontalPortScan* como tráfego benigno. Isso destaca o bom desempenho geral do modelo Árvore de Decisão, porém reconhece suas limitações quando ao desafio de detectar varreduras de portas, que se tornaram mais difíceis devido à camuflagem dos atacantes no mundo real.

A análise da matriz de confusão média de um modelo chamado *Catboost* como mostra na Figura 4, chama a atenção o fato de que o algoritmo *Catboost* teve algumas confusões ao classificar ataques do tipo *C&C-HeartBeat* como tráfego benigno. O termo *C&C-HeartBeat* refere-se a uma técnica usada em ciberataques, especificamente em *botnets* (redes de computadores infectados e controlados remotamente por um servidor

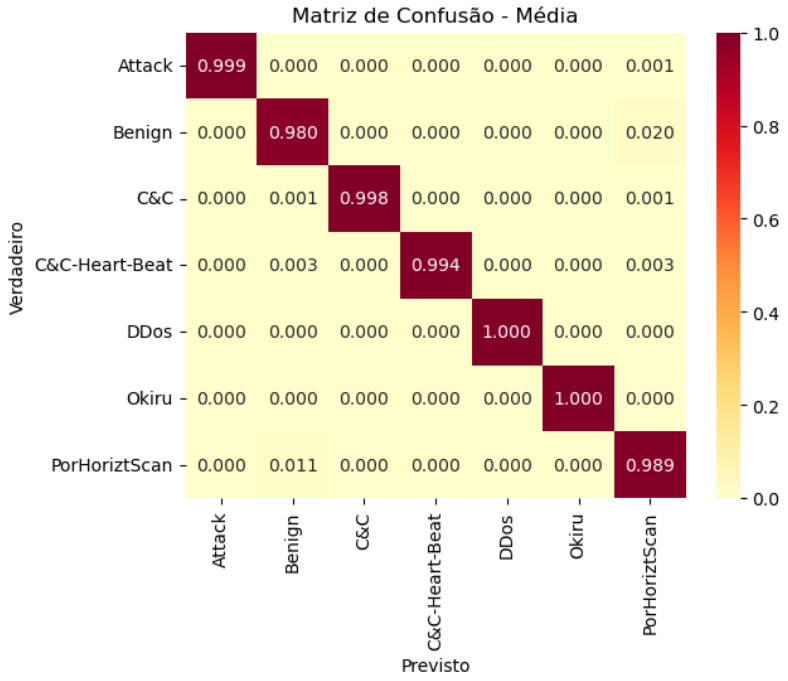

Figura 3. Matriz de confusão média do modelo *Árvore de Decisão*

C&C). Esses ataques são difíceis devido à habilidade dos operadores de *botnets* em evitarem a detecção e dificultarem as defesas contra essas ameaças. Portanto, a detecção precisa e eficaz desses ataques é um desafio significativo para os profissionais de segurança cibernética.

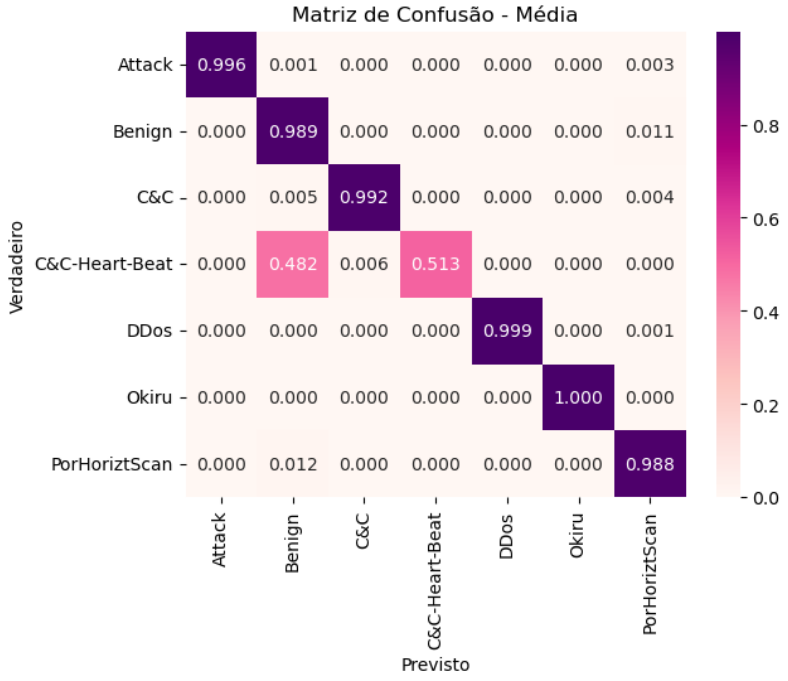

Figura 4. Matriz de confusão média do modelo *Catboost*

As matrizes de confusão também nos mostra que o IDS utilizado nos modelos *Random Forest*, Árvore de Decisão e Catboost foi capaz de acertar todas os ataques do tipo de negação de serviço distribuída (DDoS) e todas as tentativas de intrusão do *botnet Okiru*.

6.3. Avaliação de interpretabilidade

Esta seção visa apresentar os resultados dos modelos *Random Forest*, Árvore de Decisão e *CatBoost* sob a ótica da importância dos atributos no processo de classificação. Para isso, a técnica *SHAP* foi utilizada.

Conforme mencionado anteriormente, o SHAP visa explicar a saída de qualquer modelo de aprendizado de máquina, a partir da avaliação do impacto das variáveis, considerando também a sua interação com outras variáveis.

Para os algoritmos utilizados no presente projeto, assim como a Árvore de Decisão, o pré-processamento do *Random Forest* exigiu o uso das técnicas de *one-hot encoding* e *feature hashing* para codificar os atributos categóricos. Por essa razão, os atributos que passaram por esse procedimento de codificação foram alocados em subatributos e seu impacto foi verificado a partir desses componentes. O atributo original *id_resp_p* foi mapeado no subatributo *id_resp_p_14*, bem como o atributo *history* foi mapeado em *history_S, history_C, history_D*, entre outros exemplos.

A Figura 5 apresenta de forma gráfica uma análise detalhada fornecida pela técnica *SHAP* da importância dos principais atributos na classificação dos tipos de tráfego (benigno e maliciosos) considerando o modelo *Random Forest*. Pode-se interpretar esse resultado de duas formas: i) ranqueamento dos atributos mais importantes para a tarefa de classificação geral (tamanho dos somatórios das barras); e ii) identificação de quais atributos foram importantes para detectar determinado tipo de intrusão (tamanho das barras de mesma cor).

Analisando ainda a Figura 5, pode-se perceber que os 5 atributos mais relevantes na classificação geral foram: *id_resp_p*, *id_orig_p*, *orig_ip_bytes*, *orig_pkts* e *duration*, que são descritos na Tabela 3. Em especial, pode-se ressaltar o atributo *id_resp_p* referente à porta de destino da conexão, o qual foi apontado como o mais importante na detecção de intrusão no geral, e em especial para os ataques *PartOfAHorizontalPortScan* e *Okiru*. Como o ataque *PartOfAHorizontalPortScan* faz uma varredura de portas na máquina alvo, era de se esperar que os atributos relacionados às portas de origem e destino tivessem um papel importante na detecção do ataque, o que foi confirmado na presente Figura 5.

Tabela 3. Descrição dos atributos mais relevantes

Atributos	Descrição
id_resp_p	Porta usada para a resposta do dispositivo onde ocorreu a captura.
id_orig_p	Porta usada pelo respondente.
orig_ip_bytes	Números de bytes sendo enviados para o dispositivo.
orig_pkts	Número de pacotes sendo enviados para o dispositivos.
duration	Quantidade de tempo em que os dados foram negociados entre o dispositivo e o invasor.

Em [Singh 2020], ao analisar o atributo *history*, é possível verificar que o tipo D

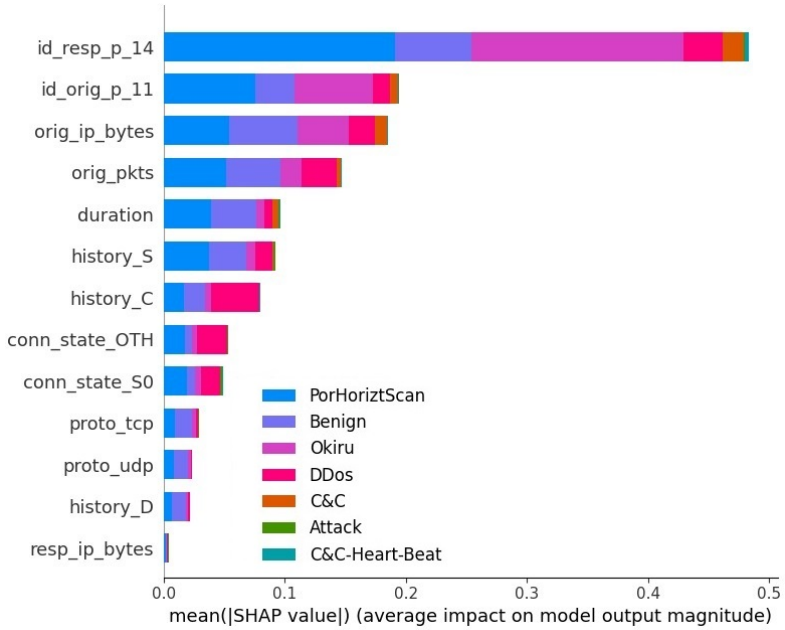

Figura 5. Interpretabilidade do algoritmo *Random Forest* utilizando a técnica *SHAP*.

corresponde a um pacote com carga, enquanto o tipo C significa um pacote com erro no *checksum*. Dessa forma, pode-se observar na Figura 5 que o subatributo *history_C*, que aponta para um erro de *checksum*, tem grande relevância para a detecção de *DDoS*, enquanto o subatributo *history_D*, possui maior importância na detecção de tráfego benigno (classe *Benign*).

Ao analisar a Figura 6 que é referente à interpretabilidade do modelo a partir da Árvore de Decisão, é possível observar um cenário parecido, com o mesmo subatributo em primeiro lugar no ranking, bem como a boa classificação também dos atributos *id_resp_p*, *id_orig_p*, *orig_pkts*, *duration*, com o acréscimo de *history* dentre os 5 mais relevantes no ranqueamento.

Por fim, a Figura 7 representa a interpretabilidade para o modelo CatBoost. Para este caso, é importante destacar que pelo seu pré-processamento distinto, os atributos não precisaram ser mapeados em subatributos. Portanto, os atributos permaneceram com o mesmo título original. Nele, também é possível observar que as portas foram os atributos mais importantes no ranqueamento, tanto para a contribuição geral da classificação dos rótulos, quanto na sua contribuição para classificação individual de cada um deles. No entanto, é interessante notar que o seu impacto diverge dos outros modelos quanto ao tipo de rótulo. A porta possui grande impacto em rótulos como *Okiru, DDoS, C&C* e outros, e um impacto menor no *PartOfAHorizontalPortScan*.

7. Considerações Finais e Trabalhos Futuros

Este trabalho propôs um *pipeline* eficiente para detecção de intrusão em redes IoT utilizando aprendizado de máquina. Utilizando a base de dados *IoT-23*, três algoritmos de aprendizado de máquina baseados em árvores foram testados. Dentre eles, o algoritmo *Random Forest* se destacou, alcançando uma precisão de 98,8% e um *recall* de 99,5% na

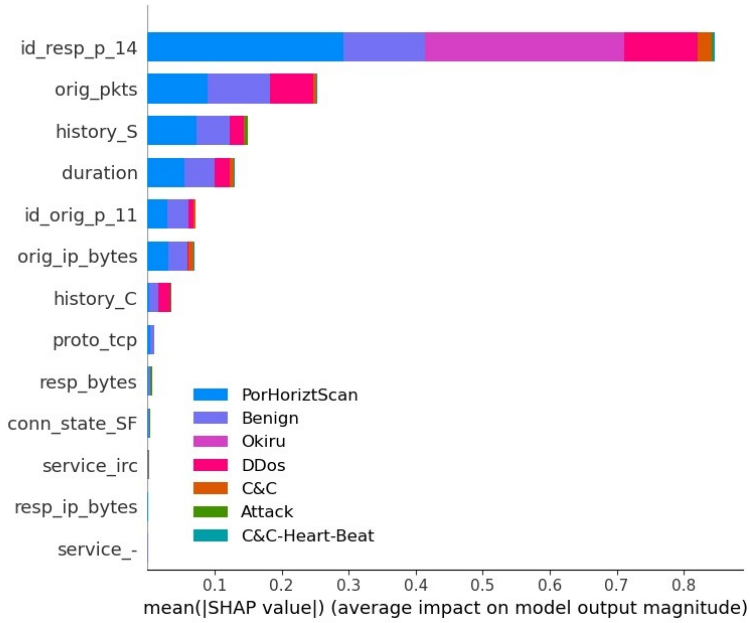

Figura 6. Interpretabilidade do algoritmo Árvore de Decisão utilizando a técnica *SHAP*.

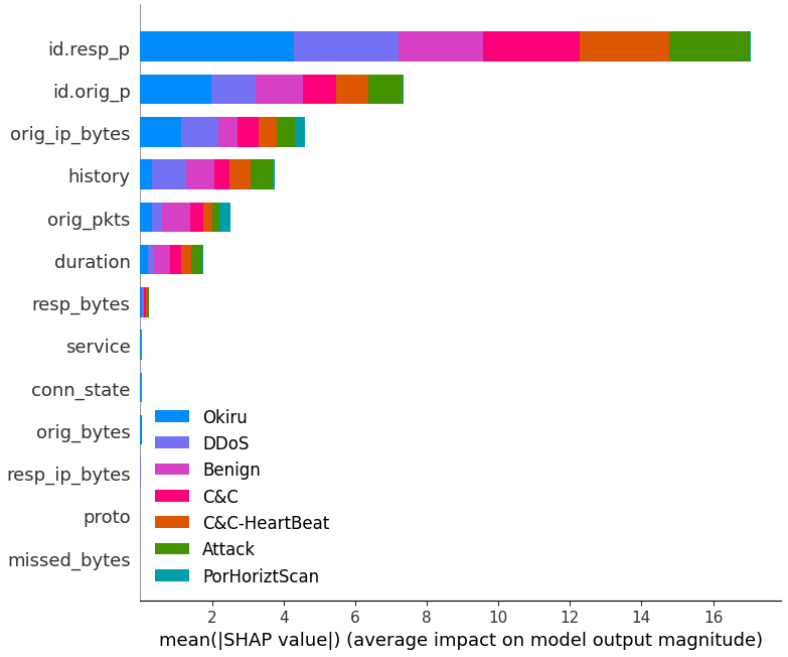

Figura 7. Interpretabilidade do algoritmo *CatBoost* utilizando a técnica *SHAP*.

tarefa de classificação entre tráfego benigno e 6 tipos de ataques. É interessante observar que o algoritmo *CatBoost* atingiu excelentes métricas sem precisar utilizar as técnicas de pré-processamento exigidas pelos outros algoritmos.

Foi apresentada também uma análise de interpretabilidade do modelo *Random Forest*, indicando aqueles atributos que mais contribuíram para a identificação dos tipos de ataques. Os atributos relacionados às portas utilizadas nas conexões se sobressaíram

como os mais importantes na classificação, em especial para os ataques de varredura de portas e do *botnet Okiru*.

Como trabalhos futuros, sugere-se a exploração de outros algoritmos de aprendizado de máquina ou outras técnicas de *ensemble* para melhorar ainda mais o desempenho do sistema de detecção de intrusão. Pode-se investigar a inclusão de seleção de atributos mais relevantes para aumentar a capacidade de classificação entre atividades benignas e maliciosas. Além disso, é importante considerar a expansão do estudo para outras bases de dados IoT, a fim de avaliar a generalização do *pipeline* e sua eficácia em diferentes contextos e cenários, fornecendo *insights* valiosos para futuras pesquisas nessa área e contribuindo para melhorar a proteção contra ameaças cibernéticas.

Referências

Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., and Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1):e4150.

Aragão, M. V. C., Mafra, S., and Figueiredo, F. A. P. d. (2022). Análise de tráfego de rede com machine learning para identificação de ameaças a dispositivos IoT. In *Anais do XL Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*. Sociedade Brasileira de Telecomunicações.

Boeckl, K., Fagan, M., Fisher, W., Lefkovitz, N., Megas, K. N., Nadeau, E., O'Rourke, D. G., Piccarreta, B., and Scarfone, K. (2019). Considerações para gerenciar riscos de privacidade e segurança cibernética na internet das coisas (iot).

Booij, T. M., Chiscop, I., Meeuwissen, E., Moustafa, N., and den Hartog, F. T. (2021). Ton_iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets. *IEEE Internet of Things Journal*, 9(1):485–496.

Carvalho, A. F. A. d., Santos, C. M. L., and Gonçalves, L. V. (2022). Segurança em iot.

Chang, W. and Grady, N. (2019). Nist big data interoperability framework: Volume 1, definitions.

Chunduri, H., Gireesh Kumar, T., and Charan, P. V. S. (2021). A multi class classification for detection of iot botnet malware. In Chaubey, N., Parikh, S., and Amin, K., editors, *Computing Science, Communication and Security*, pages 17–29, Cham. Springer International Publishing.

Colombo, A. G., Keller, V. S., and Viegas, S. C. (2018). Internet of things e linguagem de programação. *REFAQI-REVISTA DE GESTÃO EDUCAÇÃO EE TECNOLOGIA*, 4(2):5–5.

Grus, J. (2016). *Data Science do Zero*. Alta Books.

Harrison, M. (2019). *Machine Learning – Guia de Referência Rápida: Trabalhando com dados estruturados em Python*. Novatec Editora.

Hegde, M., Kepnang, G., Al Mazroei, M., Chavis, J. S., and Watkins, L. (2020). *Identification of Botnet Activity in IoT Network Traffic Using Machine Learning*.

Kozik, R., Pawlicki, M., and Choraś, M. (2021). A new method of hybrid time window embedding with transformer-based traffic data classification in iot-networked environment. *Pattern Analysis and Applications*, 24(4):1441–1449.

Liang, Y. and Vankayalapati, N. (2022). Machine learning and deep learning methods for better anomaly detection in iot-23 dataset cybersecurity. *Preprint. Available online: https://github. com/yliang725/Anomaly-Detection-IoT23 (accessed on 22 December 2022)*.

Marquês, F. (2022). *Aprendizagem Da Máquina. Aprendizagem Supervisionada Através de R*. Amazon Digital Services LLC - Kdp.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill Education, 1st edition.

Sarker, I. H., Kayes, A. S. M., Badsha, S., Alqahtani, H., Watters, P., and Ng, A. (2020). Cybersecurity data science: an overview from machine learning perspective. *Journal of Big Data*, 7(1).

Singh, A. (2020). Use of machine learning for securing IoT. In *ERA: Education and Research Archive*. University of Alberta Libraries.

SOUSA, B. F. L. M. et al. (2016). Um sistema de detecção de intrusão para detecção de ataques de negação de serviço na internet das coisas.

Stoian, N.-A. (2020). Machine learning for anomaly detection in iot networks: Malware analysis on the iot-23 data set. *University of Twente*, (B.S. thesis).