ANOMAD IoT
Modelos para Detecção de Ataques no Tráfego de Rede de Dispositivos IoT: Um Estudo Exploratório

Caio F. de Souza[1], Gabriel Moiseis F. Lima[1], José Antonio G. Mequelin[1], Guilherme Werneck de Oliveira[1], Gabriel Vinicius Canzi Candido[1]

[1]Instituto Federal do Paraná – Campus Pinhais (IFPR)
83330-200 – Pinhais – PR – Brasil

Resumo. *Nos últimos anos, os avanços em sensoriamento e comunicação têm permitido cada vez mais a conexão entre dispositivos por meio da Internet, especialmente no contexto da Internet das Coisas (*IoT*). Esses dispositivos, embora benéficos para a automação de tarefas, enfrentam desafios relacionados à segurança de dados devido à sua limitada capacidade computacional, o que aumenta o risco de vulnerabilidades. Em 2023, houve um aumento de 41% nos ataques a redes* IoT*, segundo a [Abranet 2023]. O presente estudo tem como objetivo comparar modelos de aprendizado de máquina para detectar ataques em redes de dispositivos* IoT *domésticos, utilizando um conjunto de dados rotulado da literatura. Foram implementados os modelos: Decision Tree, Random Forest, Support Vector Machine, K-Nearest Neighbors e Naive Bayes, além de um estudo sobre seus comportamentos sob dados de treinamento e teste normalizados e não normalizados. A principal contribuição deste estudo é auxiliar profissionais de TI na escolha e compreensão de modelos eficazes para a segurança de redes* IoT*. Além disso, o estudo busca facilitar a identificação de padrões de tráfego normais e maliciosos em redes* IoT*, utilizando o aprendizado de máquina clássico.*
Palavras-Chave: *Internet das coisas* IoT*; Segurança de dados; Vulnerabilidades; Ataques a redes* IoT*, Aprendizado de máquina; Normalização de dados; Redes domésticas* IoT.

Abstract. *In recent years, advances in sensing and communication have increasingly enabled devices to connect to each other via the Internet, especially in the context of the Internet of Things (*IoT*). These devices, while beneficial for automating tasks, face challenges related to data security due to their limited computing power, which increases the risk of vulnerabilities. In 2023, there was a 41% increase in attacks on* IoT *networks, according to [Abranet 2023]. This study aims to compare machine learning models to detect attacks on home* IoT *device networks using a labeled dataset from the literature. The Decision Tree, Random Forest, Support Vector Machine, K-Nearest Neighbors and Naive Bayes models were implemented, in addition to a study of their behaviors under normalized and non-normalized training and test data. The main contribution is to help IT professionals choose and understand effective models for* IoT *network security. Furthermore, the study seeks to facilitate the identification of normal and malicious traffic patterns in* IoT *networks, using classical machine learning.*
Keywords: *Internet of Things* IoT*; Data security; Vulnerabilities; Attacks on* IoT *networks, Machine learning; Data normalization;* IoT *home networks.*

1. Introdução

Ao longo dos anos e com o avanço da tecnologia, principalmente nas áreas de eletrônica, sistemas de sensoriamento e troca de informações expandiram horizontes e expectativas acerca de quais sistemas e dispositivos do mundo físico poderiam ser integrados à Internet. Esses sistemas e dispositivos fazem parte de uma rede conhecida como Internet das Coisas (*IoT - Internet of Things*). Estima-se que em 2025 haverá mais de 27 bilhões de dispositivos *IoT* conectados em todo o mundo [PACETE 2022].

Ainda que a *IoT* tenha o papel de facilitar o cotidiano dos usuários, seja como meio de comunicação em empresas, seja para fins domésticos, existem diversos problemas ainda não superados, relacionados à privacidade e à falta de segurança dos dispositivos em relação aos dados dos usuários. Dispositivos *IoT* são minicomputadores com funcionalidades concentradas em sensoriamento e comunicação através da rede, como câmeras, lâmpadas e sensores de movimento. Eles possuem um baixo poder de processamento, devido sua natureza física, e isso tem chamado a atenção de atacantes. Consequentemente, sua capacidade computacional limitada não embarca mecanismos de segurança fortes, causando uma carência e risco de roubo de informações e ataques em todos os contextos de negócio, inclusive nos ambientes domésticos [Butun et al. 2019, Meneghello et al. 2019].

São comuns notícias de tentativas de ataques contra dispositivos *IoT* em todo o mundo. Nos dois primeiros meses de 2023, por exemplo, semanalmente, em média 54% das organizações foram alvos dessas tentativas de ataque, com uma média de quase 60 ataques por organização por semana direcionados a dispositivos de *IoT*. Uma alta de 41% em comparação com 2022 e mais que o triplo do número de ataques se comparados aos de dois anos atrás [Abranet 2023].

Diante disso, muitas pesquisas em relação a segurança de dispositivos *IoT* têm sido realizadas por empresas, pesquisadores da área de *TI* e técnicos de cibersegurança [Xenofontos et al. 2021]. Pesquisas estas desenvolvidas com o principal objetivo de auxiliar na compreensão, no desenvolvimento e na seleção de métodos para a proteção de dados. Além do desenvolvimento de protocolos e aplicações específicas para tal cenário, como *DTLS* (*Datagram Transport Layer Security*), *CoAP* (*Constrained Application Protocol*) e *MQTT* (*Message Queuing Telemetry Transport*), outras abordagens, como o aprendizado de máquina, vêm ganhando espaço quando o intuito é combater ciberameaças.

1.1. Objetivo Geral

O objetivo deste trabalho é realizar um estudo exploratório e comparativo de modelos de aprendizado supervisionado de máquina para a detecção de ataques em redes de dispositivos *IoT* domésticos. Os modelos analisados incluem: *Decision Tree*, *Random Forest*, *Support Vector Machine*, *K-Nearest Neighbors* e *Naive Bayes*. Além disso, busca-se compreender o comportamento desses modelos com dados de treinamento e teste, tanto normalizados quanto não normalizados.

1.2. Objetivos Específicos

- Identificar, avaliar e selecionar um conjunto de dados rotulado, dos presentes na literatura, que representa o tráfego de uma rede *IoT* doméstica;

- Elaborar uma análise exploratória de dados (EDA - *Exploratory Data Analysis*) sobre o tráfego definido;
- Treinar e testar os modelos de aprendizado supervisionado de máquina, *Decision Tree*, *Random Forest*, *Support Vector Machine*, *K-Nearest Neighbors* e *Naive Bayes*, para a identificação de ataques no tráfego de rede;
- Comparar o desempenho dos modelos implementados, além de seus comportamentos sob a normalização e a não normalização dos dados de treinamento e teste.

2. Trabalhos Correlatos

Esta seção tem como objetivo apresentar estudos relacionados ao tema da detecção de ataques e anomalias em ambientes que contêm dispositivos *IoT*. Seu foco principal é abordar trabalhos que investigaram e propuseram métodos para detectar ataques, utilizando aprendizado de máquina para analisar o tráfego de dispositivos *IoT* conectados à rede. A partir da pesquisa bibliográfica realizada, foram selecionados os estudos descritos, seguidos pelas contribuições que oferecem ao desenvolvimento deste projeto.

Esses trabalhos foram escolhidos com o intuito de auxiliar no avanço deste projeto, proporcionando uma melhor compreensão sobre cibersegurança, *IoT*, conexões entre dispositivos, tráfego de informações na rede e a detecção de anomalias e ataques.

2.1. Detecção de Anomalias no Tráfego *MQTT* de Redes *IoT* Utilizando Técnicas de Aprendizado de Máquina (2021) [Silva Filho 2021]

Este trabalho propôs a implementação de um sistema de detecção de anomalias de dispositivos *IoT* de distintas infraestruturas, utilizando protocolo *MQTT*. Com o sistema fazendo uso de dois algoritmos de aprendizado de máquina: treinados *offline* e outra adaptada para *streaming* de dados. Além disso, foi explorada a interação entre técnicas de aprendizagem de máquina e sistemas de detecção de intrusão baseados em assinaturas, que monitoram o tráfego de rede de entrada e procuram padrões e sequências específicas que correspondam a assinaturas de ataques conhecidos.

O principal objetivo deste trabalho foi desenvolver uma solução de segurança para dispositivos *IoT* que utilizam o protocolo *MQTT* em suas comunicações, capaz de identificar ataques desconhecidos. Isso foi viabilizado pela integração de técnicas de aprendizado de máquina com sistemas de detecção de intrusão baseados em assinaturas.

Este trabalho apresentou uma solução modular na análise de dados de redes *IoT* operando com o protocolo *MQTT*, por meio de agentes detectores de intrusão atrelados à técnicas de aprendizado de máquina para *stream* de dados. Por meio dos resultados foi possível analisar a performance da solução proposta, que apesar de ter atingido uma acurácia geral baixa, detectou anomalias com uma taxa média de 60% de acerto.

2.2. Estudo Sobre a Segurança de Dispositivos Domésticos Conectados à Internet das Coisas (2022) [Messas and Zarpelão 2022]

Esse trabalho teve como intuito testar aparelhos selecionados, com receptores de *IPTV*, *TV Boxes* e câmeras *Wi-Fi*, com o objetivo de estudar o modo de funcionamento de testes de invasão e aplicá-los a dispositivos eletrônicos variados conectados à *Internet*, assim avaliando seus níveis de segurança.

Para cada dispositivo analisado, foi gerado um relatório com o objetivo de atrair atenção e incitar os responsáveis a corrigir as brechas de confiabilidade do equipamento. Além disso, esses relatórios tiveram a intenção de levantar questionamentos sobre o estado de dispositivos semelhantes, estimulando uma verificação mais eficaz para evitar problemas.

Nesse contexto, o documento busca sensibilizar um público mais amplo sobre a importância desses procedimentos, contribuindo para a melhoria dos padrões de qualidade dos dispositivos analisados. Esse objetivo está alinhado com a meta final de garantir a excelência na qualidade dos aparelhos em questão.

2.3. Framework Para Detecção de Ataques em Dispositivos *IoT*, Utilizando Abordagens de Aprendizado de Máquina (2023) [Oliveira 2024]

Este trabalho utilizou abordagens de aprendizado de máquina, nas quais foi desenvolvido um *framework* completo para a detecção de intrusão de ataques de Negação de Serviço (DoS - *Denial of Service*) próximo ao tempo real. Aplicando soluções relacionadas à segurança cibernética na camada de aplicação.

A metodologia deste trabalho constituiu-se em pesquisa quantitativa, baseada em números e gráficos para alcançar resultados aceitáveis com o auxílio de aprendizado de máquina. Foram realizados processos de captura de dados de pacotes *TCP, UDP* ou *ICMP*, filtragem de dados com o objetivo de reduzir a quantidade de *features* (obtendo a mínima perda de eficácia), aplicação de técnicas de correlação de variáveis e classificação para reconhecimento da qualidade na detecção do comportamento da rede.

Neste trabalho, foram utilizadas técnicas de aprendizado de máquina supervisionado, nas quais, a partir de exemplos de dados, os algoritmos aprenderam a prever resultados futuros por meio de métodos estatísticos. Por outro lado, nos modelos não supervisionados, não foram apresentados exemplos. Os algoritmos identificaram padrões nos dados para realizar previsões. Já no aprendizado por reforço, o modelo aprendeu com base no conceito de recompensa pelas ações tomadas, em que cada ação executada gerava uma reação, permitindo ao algoritmo prever resultados futuros ao maximizar as recompensas obtidas em cada decisão.

Os modelos aplicados nesse trabalho incluíram *Random Forest* e MLOPS (*MACHINE LEARNING OPERATIONS*), sendo um processo que auxilia organizações e líderes de negócios a gerar valor a longo prazo e reduzir os riscos relacionados à ciência de dados, aprendizado de máquinas e outras áreas da inteligência artificial. Como *MLFlow*, uma plataforma de código aberto para o gerenciamento do ciclo de vida de modelos de aprendizado de máquina. Foi utilizando o conceito de MLOps, *ELK - ELASTIC STACK*, *WAZUH*, uma plataforma de código aberto para a detecção e monitoramento de ameaças com regras pré-definidas. *DOCKER*, um virtualizador de *containers* de código aberto que auxilia na criação e administração de ambientes e AMQP (*ADVANCED MESSAGE QUEUING PROTOCOL*), um protocolo de mensagens corporativo desenvolvido para ser confiável, seguro, com aprovisionamento e interoperabilidade.

Este trabalho apresentou um sistema para a detecção de ataques DoS em redes *IoT*, utilizando metodologias de aprendizado de máquina para a classificação dos fluxos de dados. Foi arquitetado para ser uma solução completa, proporcionando um gerenciamento

de alarmes dos ataques ocorridos, sendo modelado desde a captura dos fluxos de dados até a visualização dos alarmes dos ataques detectados.

2.4. Detecção de Ataques a Redes *IoT* Usando Técnicas de Aprendizado de Máquina e Aprendizado Profundo (2020) [Bochie et al. 2020]

Esse trabalho teve como objetivo avaliar o desempenho de múltiplos modelos de aprendizado de máquina, tanto tradicionais quanto profundos, para a detecção de ataques. Seguiu-se a metodologia de **coleta e agregação de dados**, na qual foram utilizadas APIs para viabilizar a coleta, e os dados coletados foram armazenados; **visualização de dados**, etapa em que foram identificados atributos de interesse, buscadas amostras defeituosas e examinadas anomalias; **tratamento de dados**, com a remoção de amostras danificadas e a seleção de atributos; **implementação e treinamento**, onde os modelos foram treinados e seus hiperparâmetros ajustados; e, por fim, a **avaliação de desempenho**, na qual amostras inéditas foram testadas e os resultados documentados e interpretados.

Os modelos utilizados incluíram redes neurais autoassociativas (*Autoencoders*) em conjunto com *decison tree*, que se mostraram eficazes em termos de capacidade de generalização e robustez na análise. O modelo de *Decision Tree* destacou-se, alcançando um desempenho superior a 90% em todas as métricas, após a seleção de atributos, tanto no conjunto de dados quanto no *BoT-IoT*. O *Random Forest* também apresentou resultados promissores, com métricas de acurácia, precisão e F1 próximas de 95%. Já as redes neurais recorrentes (RNNs) indicaram que seu desempenho pode ser prejudicado caso os dados não estejam bem organizados.

Após essa análise, os autores concluíram que modelos de aprendizado de máquina sofisticados, como redes neurais, requerem *features* sequenciais bem definidas para alcançar bom desempenho. Além disso, observaram que técnicas tradicionais de separação de conjuntos de dados podem ser prejudiciais ao desempenho de tais modelos.

2.5. Um Método para Detecção de Vulnerabilidades Através da Análise do Tráfego de Rede *IoT* (2022) [Brezolin et al. 2022]

Esse trabalho utilizou o método MANDRAKE (*Method for vulnerAbilities detectioN baseD on the IoT netwoRk pAcKEt traffic*) para a detecção de vulnerabilidades com base na análise do tráfego de rede *IoT* e em técnicas de aprendizado de máquina. Essa abordagem foi dividida em três fases: (i) captura e extração de *features* do tráfego da rede; (ii) rotulação do tráfego com base no cálculo da entropia e (iii) classificação do tráfego por meio de técnicas de aprendizado de máquina. O método foi avaliado em dois cenários: um experimental, com dispositivos *IoT* reais, outro utilizando um conjunto de dados de tráfego de uma casa inteligente.

Os resultados apresentados apontaram que o método *MANDRAKE* era altamente eficaz na detecção de vulnerabilidades em dispositivos *IoT*. Especificamente, os resultados incluíram uma precisão de até 99% na identificação de dispositivos vulneráveis, desempenho satisfatório em diferentes cenários, com a precisão da detecção de pacotes criptografados em torno de 90% a 100% para pacotes sem criptografia, além de altas taxas de acerto na classificação dos fluxos vulneráveis, com acurácia variando entre 88,5% e 99% em problemas de duas classes e entre 82,1% e 98% em problemas de múltiplas classes.

Esses resultados sugeriram que o método era uma solução eficaz para a detecção de vulnerabilidades na *IoT*, especialmente em ambientes onde a criptografia estava ausente. O método permitiu rotular e treinar uma base de dados com base na entropia, o que foi útil para identificar dispositivos vulneráveis que não utilizavam técnicas de criptografia na comunicação.

2.6. *The Internet of Things Security: A Survey Encompassing Unexplored Areas and New Insights (2022)* [Omolara et al. 2022]

Esse artigo fez uma reflexão sobre os dispositivos *IoT*, mostrando que eram uma nova onda de tecnologia que revolucionou a vida das pessoas em diversos aspectos, como saúde inteligente, casas inteligentes, cidades inteligentes, etc. Por outro lado, a segurança foi frequentemente citada como um fator crítico que permitia a adoção ou rejeição generalizada de qualquer nova tecnologia.

A segurança no nível do dispositivo era um fator chave para a operação perfeita da *IoT*. O número de dispositivos conectados continuava a aumentar geograficamente a cada dia. Portanto, a segurança da *IoT* precisava ser continuamente revista e revisitada em intervalos regulares para servir como uma medida proativa antes que os ataques acontecessem e para preparar melhores soluções futuras.

2.7. Este Trabalho

O principal diferencial deste trabalho, além do foco em redes de dispositivos *IoT* domésticos, é a realização do estudo com dados normalizados e não normalizados e, consequentemente, a compreensão do impacto que a normalização traz no desempenho e na predição dos modelos. Além disso, o estudo também visa comparar os resultados obtidos nos dois casos.

3. Materiais e Métodos

Nessa seção são abordados os materiais e métodos utilizados para o desenvolvimento do projeto *ANOMAD IoT*, descrevendo suas funções dentro da mesma seção. A Tabela 1 apresenta as tecnologias e ferramentas que foram utilizados. O diagrama em modelo cascata, ilustrado na Figura 1, foi desenvolvido a fim de auxiliar na compreensão das etapas realizadas neste projeto.

Tabela 1. Materiais e Ferramentas [Os Autores 2024]

Ferramentas		
Nome	**Descrição**	**Fonte**
IEEE DataPort	Obtenção do conjunto de dados que representa o tráfego de redes	[IEEE 2024]
Google Colab	Ambiente de desenvolvimento e disponibilização dos experimentos.	[Google-colab 2024]
Scikit-learn	Biblioteca Python para treinar e implementar modelos de aprendizado de máquina.	[Scikit-Learn 2024]
LucidChart	Um aplicativo de diagramação online que permite criar e compartilhar diagramas.	[LucidChart 2024]

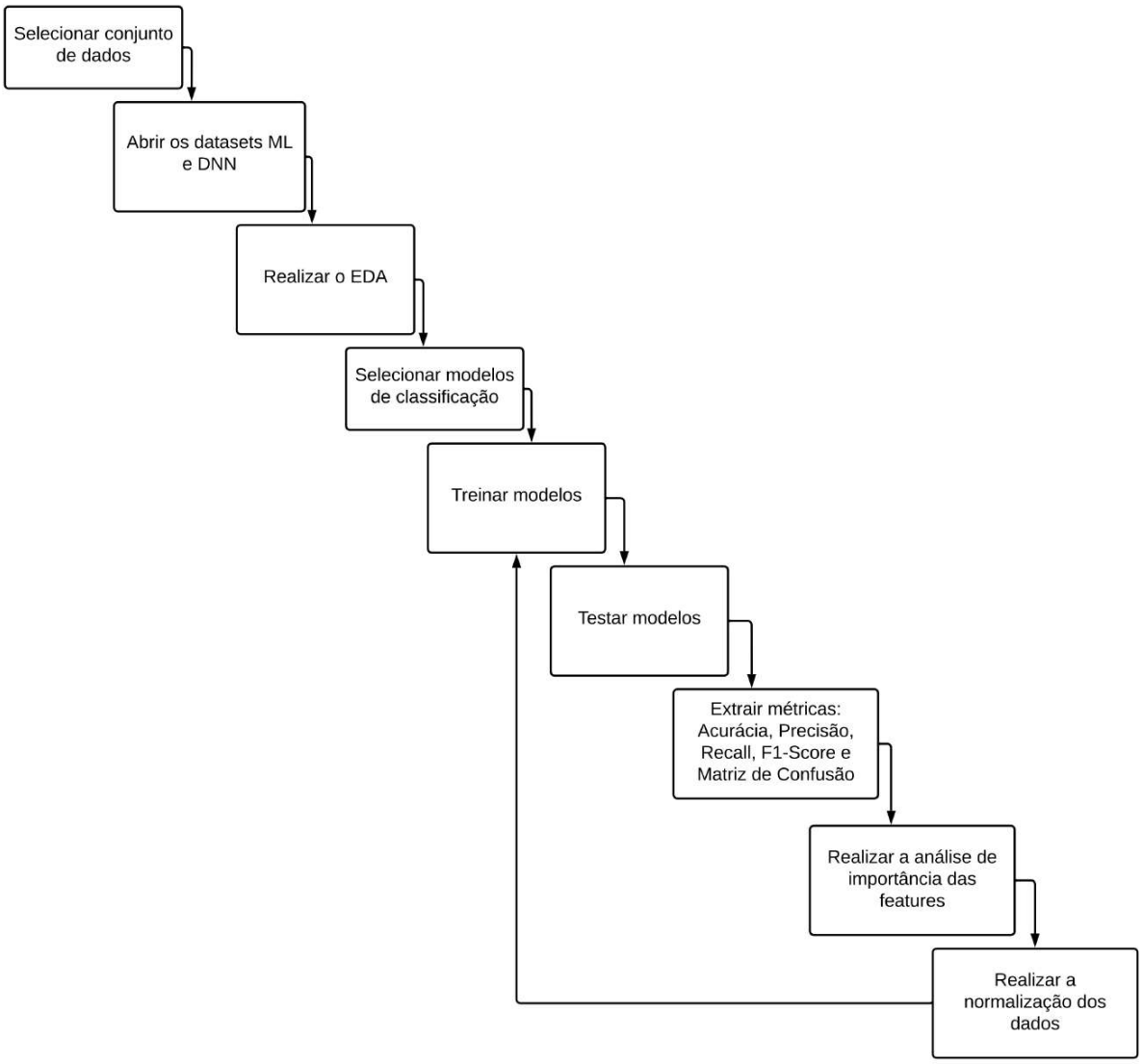

Figura 1. Diagrama em modelo cascata do desenvolvimento do projeto [Os Autores 2024]

A seleção do conjunto de dados utilizado foi concretizada por meio de uma pesquisa bibliográfica sobre conjuntos de dados presentes na literatura e disponibilizados gratuitamente no IEEE DataPort [IEEE 2024]. Foram elencados quatro conjuntos de dados possíveis para o trabalho [Koppula and L.M.I. 2024, Emeç and Özcanhan 2023, Adjei et al. 2024, Ferrag et al. 2022]. A partir disso, foi selecionado o *Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of* IoT *and* IIoT *Applications: Centralized and Federated Learning* por ser um conjunto de dados que possui uma documentação detalhada, uma grande variedade de dispositivos *IoT* domésticos, por conter arquivos em formatos ".csv"e grande variedade de dados. [Ferrag et al. 2022].

Esse conjunto de dados é composto por sistemas de detecção de intrusão baseados em aprendizado de máquina em dois modos diferentes, a saber, aprendizado centralizado e federado (o aprendizado centralizado centraliza o treinamento e armazenamento de dados, enquanto o federado descentraliza o treinamento, mantendo os dados nos dis-

positivos locais e enviando apenas atualizações do modelo para o servidor central). Especificamente, o conjunto de dados foi gerado usando um *testbed* de *IoT/IIoT* desenvolvido especificamente com um grande conjunto representativo de dispositivos, sensores, protocolos e configurações de nuvem/borda com os dados de *IoT* gerados a partir de vários dispositivos, tais como: como sensores digitais de baixo custo para detecção de temperatura e umidade, detector de incêndio, sensor ultrassônico, sensor de detecção de nível de água, medidor de sensor de pH, sensor de umidade do solo, sensor de frequência cardíaca. Além disso, foram identificados e analisados 14 ataques relacionados a protocolos de conectividade de *IoT e IIoT*, que são categorizados em cinco ameaças, incluindo ataques *DoS/DDoS*, coleta de informações, ataques *man in the middle*, ataques de injeção e ataques de *malware*.

Após a seleção do conjunto de dados para o estudo, utilizou-se a linguagem de programação Python para realizar uma análise exploratória de dados, conforme descrito na Seção 4. Posteriormente, na Seção 6, foram implementados os modelos de aprendizado supervisionados e conduzidos uma análise quantitativa com base nos resultados dos testes.

4. Análise Exploratória

A Análise Exploratória de Dados (*EDA - Exploratory Data Analysis*) é uma etapa essencial para qualquer abordagem que visa investigar e compreender comportamentos de dados. Ela é crucial para obter um entendimento básico sobre a distribuição dos dados, a identificação de valores atípicos (*outliers*) e possíveis relações, muitas vezes por meio de representações gráficas. A razão para o amplo uso de ferramentas gráficas é que, por sua própria natureza, o principal objetivo da *EDA* é explorar, e os gráficos fornecem aos analistas um grande suporte para isso [Komorowski et al. 2016]. Além de ajudar a determinar a melhor maneira de manipular diferentes fontes de dados, a *EDA* auxilia na descoberta de padrões, na identificação de anomalias e no testes de hipóteses, permitindo a criação de projetos de inteligência artificial e automação mais eficazes. Por isso, a *EDA* é de suma importância para o processo analítico. Aqui são apresentados apenas alguns dos recursos utilizados. O estudo completo pode ser consultado no *Github*. [1].

Foram identificados dois subconjuntos em formato ".csv", nomeados *ML_EdgeIIoT* (ML) e DNN_EdgeIIoT (DNN). O ML contém um conjunto de dados selecionado para utilização na avaliação de sistemas de detecção de intrusões tradicionais baseados em aprendizado de máquina. Já o DNN contém um conjunto de dados selecionado para a avaliação de sistemas de detecção de intrusão baseados em aprendizado profundo.

Após essa identificação, foi selecionado o subconjunto ML e aplicada a análise exploratória de dados (EDA), na qual se constatou a existência de 63 colunas e 157.800 linhas. Em seguida, foi realizado um sumário estatístico, sendo possível a visualização de todas as *features* do conjunto de dados como mostra na Tabela 2 , seguida por sua descrição, informações, verificação de valores nulos no qual foi identificado que não existiam valores nulos. Também foram identificados através do EDA os tipos das colunas, a moda das *features* sendo 0 para todas, a média tendo variações entre 1 e 9 e a mediana tendo variações entre 0 e 2. Também foi necessária a exclusão de algumas colunas do

[1] https://github.com/GabrielMoiseis/Projeto-de-TCC

conjunto de dados por serem do tipo *string* e não contribuirem significativamente para a análise de dados e o desempenho dos modelos.

Tabela 2. Descrição das *features do* utilizadas do conjunto de dados *edge-iloTset* [Os Autores 2024]

Feature	Descrição
arp.opcode	Código de operação do *ARP* (ex.: requisição ou resposta).
arp.hw.size	Tamanho do endereço de *hardware* no protocolo *ARP*.
icmp.checksum	Soma de verificação do *ICMP* para integridade.
icmp.seq_le	Número de sequência no *ICMP*.
icmp.unused	Campos não utilizados no cabeçalho *ICMP*.
http.content_length	Comprimento do conteúdo em cabeçalhos *HTTP*.
http.response	Resposta do servidor *HTTP*.
http.tls_port	Porta de transporte segura associada ao *HTTP*.
tcp.ack	Número de reconhecimento em pacotes *TCP*.
tcp.ack_raw	Representação crua do número de reconhecimento *TCP*.
tcp.checksum	Soma de verificação *TCP* para integridade.
tcp.connection.fin	Indicador de finalização da conexão *TCP*.
tcp.connection.rst	Indicador de reinicialização da conexão *TCP*.
tcp.connection.syn	Indicador de início de conexão *TCP*.
tcp.connection.synack	Sinalização de *handshake TCP*.
tcp.dstport	Porta de destino *TCP*.
tcp.flags	Conjunto de bandeiras no cabeçalho *TCP*.
tcp.flags.ack	Bandeira de reconhecimento *TCP*.
tcp.len	Comprimento dos dados no pacote *TCP*.
udp.port	Porta associada a pacotes *UDP*.
udp.stream	Identificador único do fluxo *UDP*.
udp.time_delta	Tempo entre pacotes no fluxo *UDP*.
dns.qry.name	Nome consultado no *DNS*.
dns.qry.qu	Tipo de consulta realizada no *DNS*.
dns.qry.type	Tipo de consulta *DNS* (ex.: *A, MX, CNAME*).
dns.retransmission	Indicador de retransmissão de consultas *DNS*.
dns.retransmit_request	Solicitações retransmitidas no *DNS*.
dns.retransmit_request_in	Registro interno de retransmissões *DNS*.
mqtt.conflag.cleansess	Indicador de sessão limpa no *MQTT*.
mqtt.conflags	Bandeiras gerais do *MQTT*.
mqtt.hdrflags	Bandeiras do cabeçalho no *MQTT*.
mqtt.len	Comprimento da mensagem *MQTT*.
mqtt.msg_decoded_as	Mensagem decodificada no *MQTT*.
mqtt.msgtype	Tipo de mensagem do *MQTT*.
mqtt.proto_len	Comprimento do protocolo *MQTT*.
mqtt.topic_len	Comprimento do tópico *MQTT*.
mqtt.ver	Versão do protocolo *MQTT*.
mbtcp.len	Comprimento do*Modbus TCP*.
mbtcp.trans_id	Identificador de transação no *Modbus TCP*.
mbtcp.unit_id	Identificador de unidade no *Modbus TCP*.

É possível observar que o tipo “Normal” de tráfego aparece como o mais frequente, com pouco menos de 25.000 ocorrências, seguido por ataques do tipo *DDoS (UDP e ICMP)*, *ransomware*, injeção de *SQL*, entre outros. E os tipos de ataque menos frequentes, como *“MITM” (man-in-the-middle)* e “*Fingerprinting*”, tiveram poucas ocorrências em comparação com muitos outros como mostra a Figura 2.

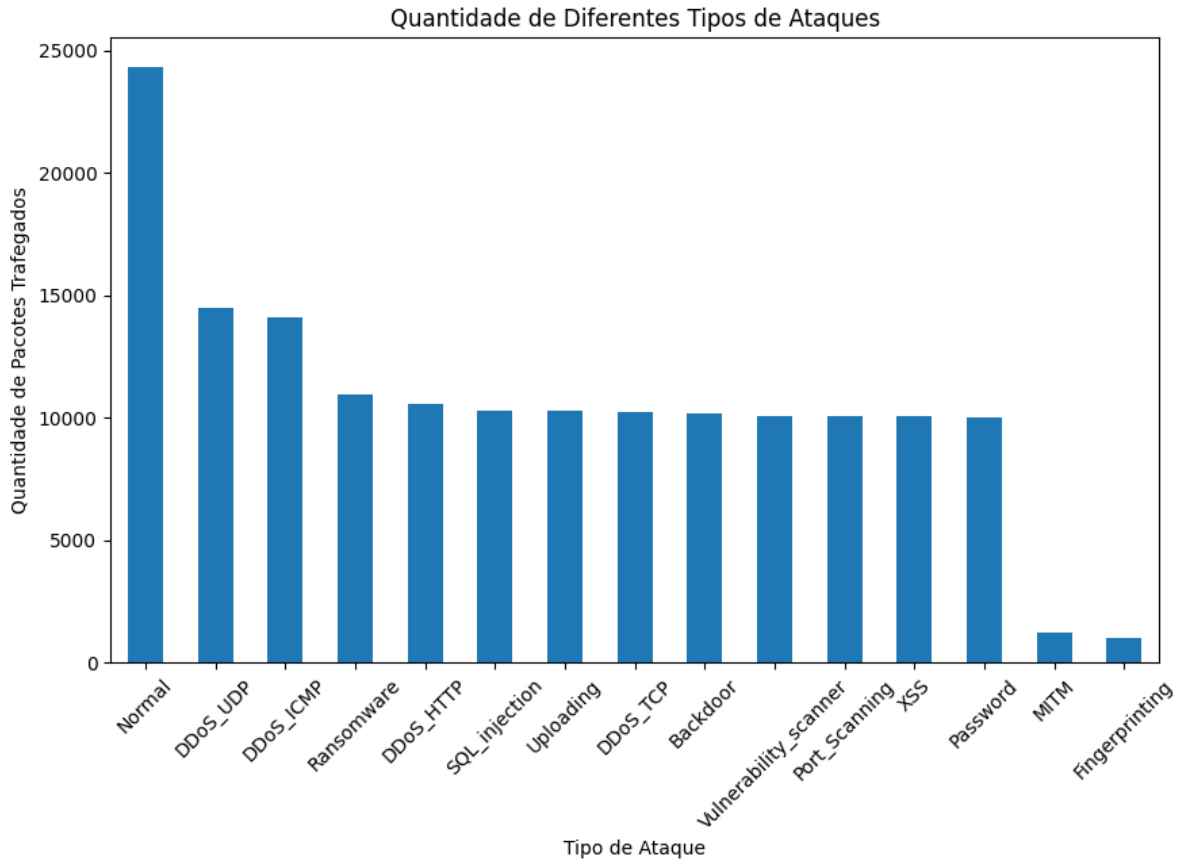

Figura 2. Pacotes trafegados na rede do ML [Os Autores 2024]

Tabela 3. Estatísticas descritivas das colunas [Os Autores 2024]

Coluna	Contagem	Média	Std	Max
arp.opcode	157800	0.014195	0.149783	2
arp.hw.size	157800	0.059848	96245	6
icmp.checksum	157800	3047.291838	11114.328203	65532
icmp.seq_le	157800	3239.979778	11406.072994	65524
icmp.transmit_timestamp	157800	40468.1604	1764075.000	77289020
icmp.unused	157800	0.0	0.0	0
http.content_length	157800	14.715520	229.659671	83655
http.response	157800	0.045748	0.208938	1
http.tls_port	157800	0.0	0.0	0

Um histograma é uma representação gráfica que organiza dados em intervalos e mostra a frequência que esses dados caem em cada intervalo. Ele foi utilizado para

auxiliar na compreensão sobre a distribuição de dados, sendo possível a visualização de valores que se desviam significativamente dos outros dados, quais valores ocorrem com mais frequência ou se exitem erros de coleta de dados ou medição. A Figura 3 ilustra um exemplo aplicado à coluna *tcp.flags*.

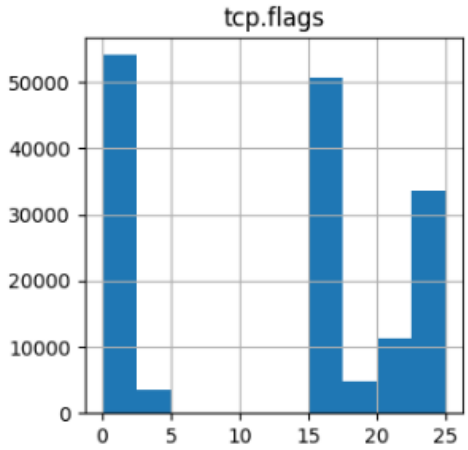

Figura 3. Histograma da coluna tcp.flags [Os Autores 2024]

Uma das coluna analisadas por meio do *boxplot*, a tcp.ack_raw, apresenta o número do *ack* da camada de transporte do *TCP/IP*, como ilustra a Figura 4. São valores relacionados ao campo de reconhecimento em pacotes *TCP*, que é usado para controlar a confiabilidade das conexões. É possível observar o valor bruto da variável, no eixo horizontal de acordo com os diferentes tipos de ataques, eixo vertical.

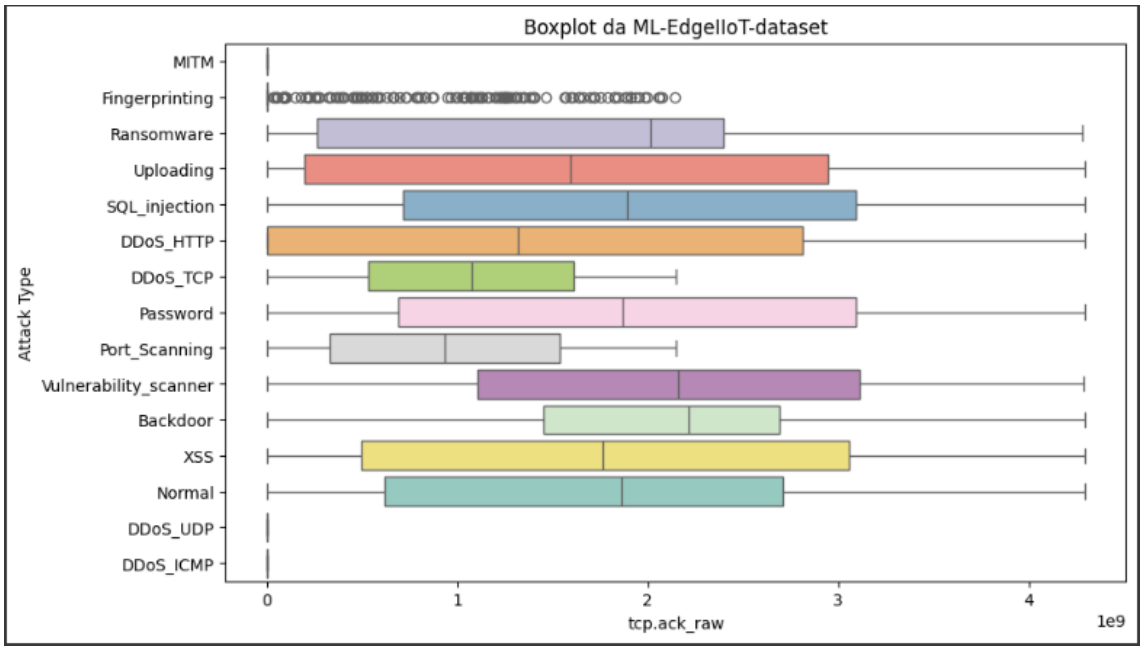

Figura 4. Distribuição de ataques na coluna tcp.ack_raw [Os Autores 2024]

Também foi desenvolvido um gráfico, baseado na técnica de mapa de calor, para

demonstrar a correlação entre as colunas do conjunto de dados. As cores indicam a força e a direção das correlações como indica na Figura 5.

A cor vermelha indica uma forte correlação diretamente proporcional (próxima de 1), ou seja, o aumento de uma variável está associado ao aumento da outra. É possível observar esse fenômeno nas variáveis *tcp.flags.ack* e *tcp.flags*. **A cor azul** representa uma correlação inversamente proporcional (próxima de -1), ou seja, o aumento de uma variável está associado à diminuição da outra. Por exemplo, as variáveis *tcp.connection.syn* e *tcp.flags.ack*. **As cores neutras** ou mais claras representam pouca ou nenhuma relação linear entre as variáveis, ou seja, correlações próximas de 0, como ilustram as variáveis *udp.port* e *tcp.len*.

A análise exploratória dos dados revelou determinados padrões. O sumário estatístico e os gráficos gerados indicam que muitas *features* apresentam uma alta frequência de valores 0, especialmente entre 50% e 75% dos dados. Isso sugere que essas *features* podem não ser relevantes para a maioria das observações, exceto em casos específicos ou em *outliers*.

No entanto, algumas colunas apresentaram valores significativos, os quais foram cruciais para a análise subsequente. A maior parte das colunas possui o tipo de dado *float64*, indicando valores numéricos contínuos. Também foram encontradas colunas do tipo *object*, que podem conter dados categóricos ou *strings*, e uma única coluna do tipo *int64*, a *Attack_label*, utilizada para identificar ataques.

É importante destacar que o conjunto de dados não apresenta valores nulos, o que indica uma boa qualidade dos dados coletados e processados. A ausência de valores faltantes facilita a análise e evita a necessidade de alterações nas colunas.

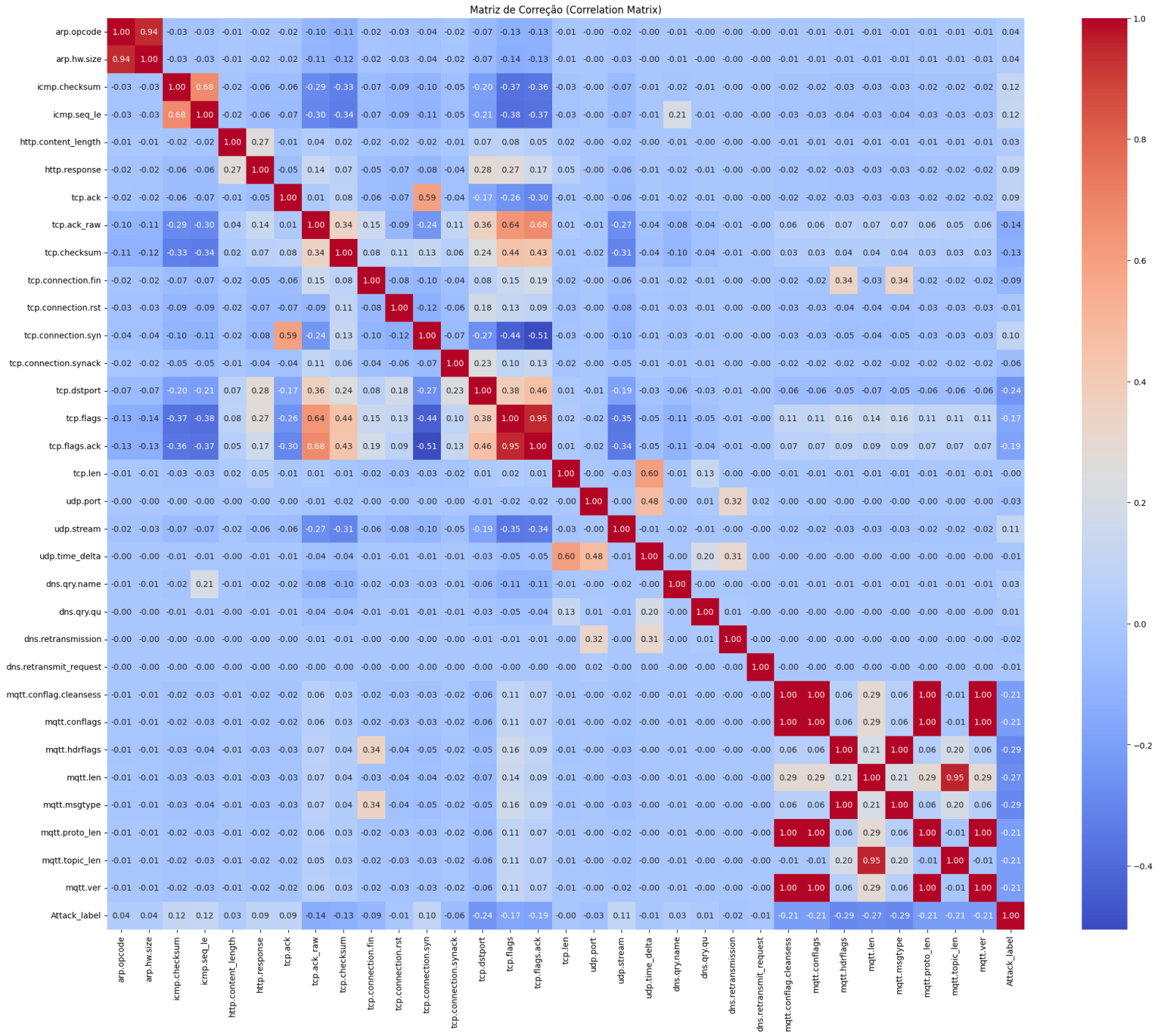

Figura 5. Mapa de calor referente à correlação entre as variáveis do conjunto de dados [Os Autores 2024]

5. Modelos de Aprendizado de Máquina e Métricas de Avaliação

Em sua essência, o aprendizado de máquina envolve o desenvolvimento de modelos matemáticos com o objetivo de compreender e prever o comportamento de dados. O "aprendizado" surge quando são definidos parâmetros ajustáveis, que podem ser adaptados aos dados observados. Assim, o modelo começa a "aprender" a partir desses dados [VanderPlas 2023], realizando ajustes em seus parâmetros internos. Dessa forma, ele consegue realizar previsões e ser ajustado conforme o contexto em que está inserido.

Na realização deste estudo foram utilizados os seguintes modelos supervisionados: *Decision Tree (DT)*, *Random Forest (RF)*, *Support Vector Machine (SVM)*, *K-Nearest Neighbors (KNN)* e *Naive Bayes (NB)*. Os dados foram divididos em 70% para treino e 30% para teste. Um ponto importante a destacar é que todos os modelos, foram treinados com dados não normalizados e depois normalizados por meio da biblioteca *Standard Scaler* [Scikit-Learn 2024] sendo utilizada para padronizar variáveis em um conjunto de dados. Transformando os dados para que tenham uma média de 0 e desvio padrão de 1. Isso é feito de forma que as variáveis sejam representadas na mesma escala, o que pode ser importante para algoritmos de aprendizado de máquina que são sensíveis a diferentes escalas de variáveis. Isso se deve ao fato de que muitos valores de *features*, quando informados em diferentes escalas, não favorecem o aprendizado do modelo que, em última instância, comprometem sua capacidade preditiva. Por exemplo, a *feature* que representa o número de porta do pacote cujo valor é 22 e a *feature* do número de pacote cujo valor é 8000, ou seja, há uma distância muito grande entre esses valores.

As métricas consideradas para a comparação dos modelos envolvem a acurácia 1, a precisão 2, o *recall* 3, o *F1-Score* 4 e a matriz de confusão, Tabela 4. Essas métricas consideram a taxa de verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN). A acurácia se refere a proporção de exatidão de todas as classificações corretas, sejam elas positivas ou negativas. A precisão estima a proporção de todas as classificações positivas que são realmente positivas. O *recall* (ou taxa de verdadeiro positivo) consiste na proporção de todos os dispositivos que foram classificados corretamente como positivos. O F1-score estabelece uma relação entre a precisão e o *recall* estimando a média harmônica. Por fim, a matriz de confusão é uma tabela com duas linhas e duas colunas que apresenta as frequências de classificação para cada classe do modelo, ela relata o numero de (VP), (VN), (FP) e (FN) como forma de comparar o desempenho do modelo de aprendizado.

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

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

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

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

Tabela 4. Exemplo de matriz de confusão [Os Autores 2024]

		Valores Previstos	
		Ataque	Não Ataque
Valores Reais	Ataque	VP	FP
	Não Ataque	FN	VN

5.1. *Decision Tree (DT)*

O modelo *DT* é um método de aprendizado supervisionado usado para classificação e regressão. O objetivo é criar um modelo capaz de realizar classificação multiclasse em um conjunto de dados [Scikit-Learn 2024].

Assim como o fluxograma o *DT* estabelece nós que possuem uma hierarquia, sendo o nó base aquele que possui os dados base e os nós folha sendo aqueles que são gerados como resposta ou resultados finais.

Foi utilizado como parâmetros para o modelo DT os parâmetros padrões, que são eles, (criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0, monotonic_cst=None).

5.2. *Random Forest* (RF)

O modelo *RF* é um algoritmo de aprendizado de máquina supervisionado que utiliza uma abordagem que combina várias *Decision Tree*. O conjunto primeiramente, utiliza um conjunto de dados de treinamento e pode ser usado para tarefa de classificação. Para cada árvore, o modelo seleciona aleatoriamente um subconjunto de dados e um subconjunto de *features*.

Para o treinamento, cada árvore o treinamento é independente, com seus próprios subconjuntos e *features*, introduzindo mais diversidade entre as árvores. O que contribui para o modelo ser mais robusto e preciso, sendo menos propenso ao *"overfitting"*, que ocorre quando um modelo aprende excessivamente os detalhes do treinamento e se ajusta tão bem a esses dados que se torna incapaz de fazer predições precisas para dados desconhecidos [Biau and Scornet 2016]. Para predição, cada árvore do modelo faz uma previsão individual e a classe final é determinada pela votação da classe mais frequente prevista.

O modelo *RF* determina a importância das *features* baseadas em quão úteis são para a criação das *Decision Tree*. Essa técnica não se baseia na eficácia de cada *feature* em reduzir a impureza das árvores.

Para o modelo *RF* foram utilizados os parâmetros (n_estimators=100, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=-1, random_state=0, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None, monotonic_cst=None). Foram utilizados os parâmetros padrão do RF exceto os parâmetros 'n_jobs' que foi alterado do padrão, de *'none'* para -1, 'random_state'que foi alterado do padrão, de *'none'* para 0.

5.3. *Support Vector Machine* (SVM)

O modelo *SVM* é um algoritmo de aprendizado supervisionado, utilizado tanto para classificação quanto para regressão. Sua principal ideia é encontrar o melhor hiperplano, que separa os dados em diferentes classes, maximizando a distância entre essas classes.

Ocorre o mapeamento dos dados em que, cada dado representa um ponto em um espaço n-dimensional onde n é o número de *features*. O *SVM* busca o melhor hiperplano (marcação de separação das classes) que maximiza a distância entre os pontos mais próximos de cada classe, chamados vetores de suporte. A distancia entre os vetores de suporte e hiperplano é chamada de margem e representa a confiança na classificação.

Para dados não separáveis linearmente, o modelo utiliza funções *kernel* que projetam os dados em um espaço de maior dimensão, tornando a separação linear possível. Os dados então são classificados de acordo com o lado em que estão posicionados em relação ao hiperplano.

Foi utilizado dentro do *SVM* o modelo *LinearSVC* que é restrito ao *kernel* linear que é eficiente para grandes conjuntos de dados, mas não pra relações complexas. Emprega a estratégia *one-vs-rest*, comparando cada classe contra todas as outras [Scikit-Learn 2024].

Para o modelo *SVM* foram utilizados os parâmetros padrão do *kernel LinearSVC* que são eles (penalty='l2', loss='squared_hinge', dual='auto', tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000).

5.4. *K-Nearest Neighbors* (KNN)

O modelo *KNN* é um algoritmo de aprendizado de máquina supervisionado utilizado para classificação, que utiliza um conjunto de dados rotulados para classificar, onde cada ponto é classificado por suas *features*.

Para o modelo fazer uma previsão ele calcula a distância entre a variável não rotulada e todos os pontos do conjunto de treinamento, atribuindo a casse mais comum entre os vizinhos mais próximos (votação da maioria) [Zhang 2016].

Ao contrário de modelos baseados em árvores ou modelos lineares, o *KNN* não possui um mecanismo interno para calcular a importância das *features*. Ele não classifica as *features* por importância inerentemente.

Para o modelo *KNN* foram utilizados os parâmetros (n_neighbors=7, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None). Foram utilizados os parâmetros padrão do KNN exceto o parâmetro 'n_neighbors', que foram alterados do padrão, de 5 para 7.

5.5. *Naive Bayes* (NB)

O algoritmo *NB* é uma técnica de classificação baseada no teorema de Bayes que assume que as *features (features)* são independentes entre si, por isso o termo *"naive"* que significa ingênuo.

Existem 3 variações de *NB*, sendo a *Gaussian* que é usado quando as *features* são contínuas e é presumido que seguem uma distribuição normal. Outra variação é o Multinominal que é geralmente usado para dados discretos, estão geralmente em problemas de classificação de texto. Por fim, a variação escolhida para esse trabalho foi a Bernoulli, que é a mais adequada para dados binários como 0 e 1, como em modelos de presença ou ausência de palavras [Scikit-Learn 2024].

No modelo *NB* a importância de uma *feature* não é calculada explicitamente como nos modelos *DT* ou *RF*, onde a importância das *features* são medidas diretamente. Porém o impacto que uma *feature* causa em uma classificação pode ser inferido indiretamente, com base em como ela influencia as probabilidades e a classificação final.

Para o modelo *NB* foram utilizados os parâmetros padrão do *kernel BernoulliNB* que são eles (*, alpha=1.0, force_alpha=True, binarize=0.0, fit_prior=True, class_prior=None).

6. Experimentos e Resultados

Após realização da análise exploratória, foi iniciada a etapa de implementação dos modelos, na qual os dados foram divididos entre treino e teste seguido de seus treinamentos e testes, sendo os primeiros testes sem normalizar os dados e o segundo com os dados normalizados. O experimento completo pode ser consultado no *Github*. [2].

Os resultados apresentados na Tabela 5 são referentes aos desempenhos dos modelos quando treinados e testados sem a normalização e após a normalização por meio da biblioteca *Standard Scaler*. Para a maioria dos algoritmos, a normalização dos dados resultou em uma ligeira melhora no desempenho, especialmente em termos de precisão e *F1-Score*. A *Decision Tree* e o *Random Forest* apresentaram os melhores resultados de forma geral, sem a normalização de dados, com alta acurácia, precisão, *recall* e *F1-Score*. Isso indica que esses algoritmos foram capazes de aprender as relações entre as *features* do conjunto de dados em seu estado original. O *Support Vector Machine* e o *K-Nearest Neighbors* obtiveram um desempenho um pouco inferior em relação aos dois modelos anteriores. Mesmo assim, eles apresentaram bons resultados, ainda mais quando os dados foram normalizados. Já o *Naive Bayes* apresentou os piores resultados entre os modelos testados, especialmente em termos de acurácia. Isso pode indicar que as hipóteses de independência condicional desse modelo não se ajustam bem aos dados, tanto em seu estado original quanto normalizados.

Tabela 5. Desempenho dos modelos de aprendizado na detecção de ataques [Os Autores 2024]

Classificador	Normalizado	Acurácia	Precisão	Recall	F1-Score
Decision Tree	Não	**99,1**	**99,5**	**99,4**	**99,4**
	Sim	98,3	99,1	98,9	99
Random Forest	Não	**99,2**	**99,5**	**99,5**	**99,5**
	Sim	98,5	99,3	98,8	99,1
Support Vector Machine	Não	84,5	**100**	**88,5**	91,6
	Sim	**87,7**	99,9	87,3	**93,2**
K-Nearest Neighbors	Não	88,1	97,6	89,3	93,3
	Sim	**95,8**	**98,2**	**96,9**	**97,5**
Naive Bayes	Não	83.1	92,4	88,1	90,2
	Sim	**85,5**	**95,6**	**88,2**	**91,8**

As matrizes de confusão de cada modelo foram geradas com o objetivo de avaliar e comparar, de forma detalhada, o desempenho de cada abordagem. Nos casos de

[2]https://github.com/GabrielMoiseis/Projeto-de-TCC

tráfego normal, o valor correspondente é 0, enquanto que, para os casos de ataque, o valor correspondente é 1.

No caso da *Decision Tree*, foram construídas duas matrizes: uma utilizando os dados sem normalização, conforme mostrado na Figura 6, e outra com os dados após o processo de normalização, como apresentado na Figura 7. Essa análise foi conduzida para todos os modelos de aprendizado de máquina utilizados no desenvolvimento do trabalho. Por exemplo, as Figuras 18 e 19 exibem os resultados para o *Random Forest*, as Figuras 20 e 21 para o SVM, as Figuras 12 e 13 para o KNN, e as Figuras 22 e 15 para o *Naive Bayes*.

Essa comparação entre as matrizes permite observar como a normalização dos dados impacta a eficácia dos modelos, destacando possíveis melhorias ou variações nos resultados das previsões.

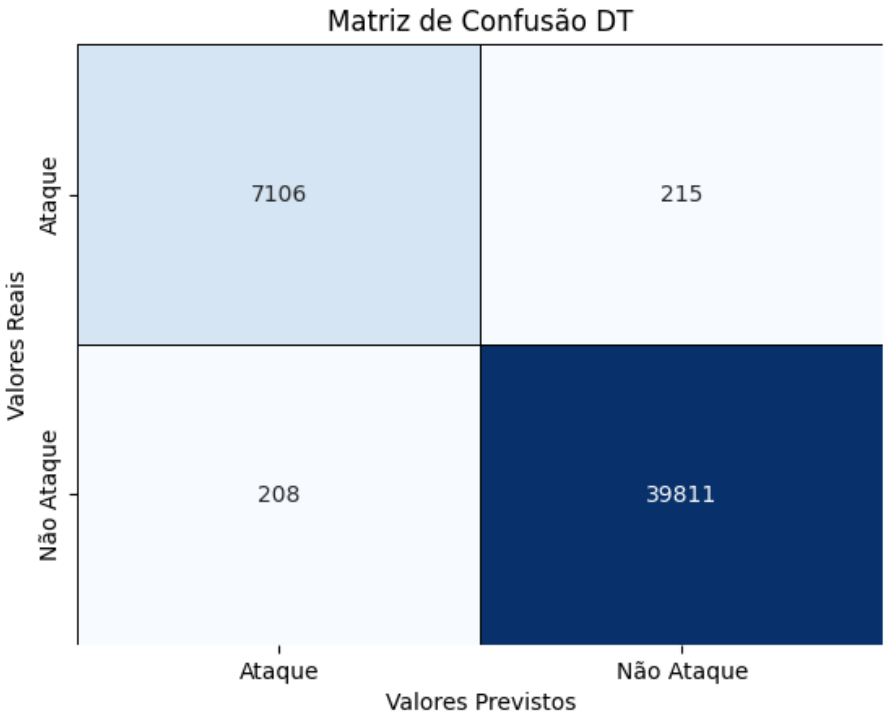

Figura 6. Matriz de confusão da *Decision Tree* [Os Autores 2024]

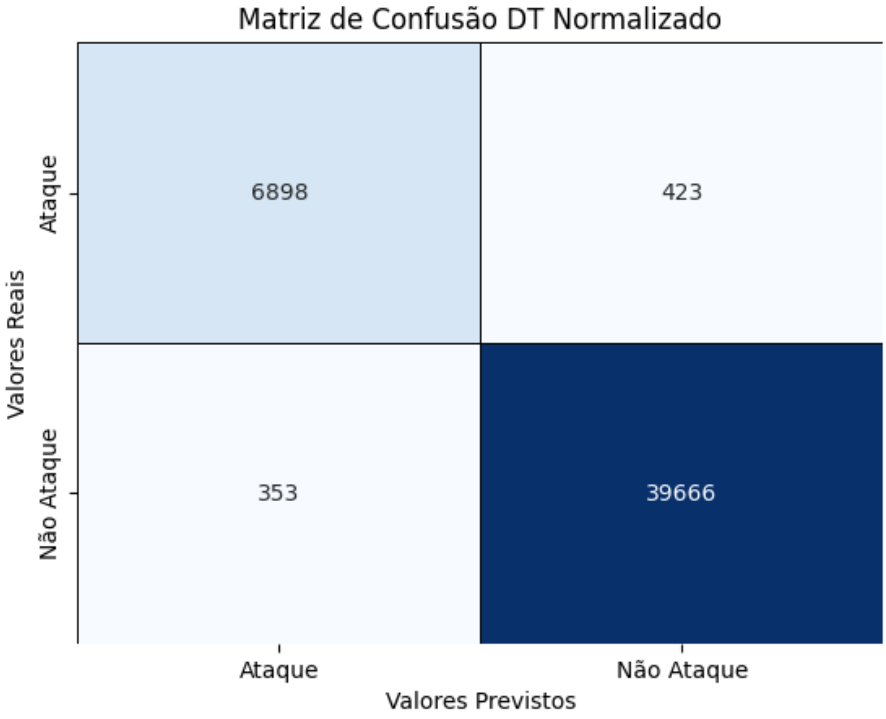

Figura 7. Matriz de confusão da *Decision Tree* com os dados normalizados [Os Autores 2024]

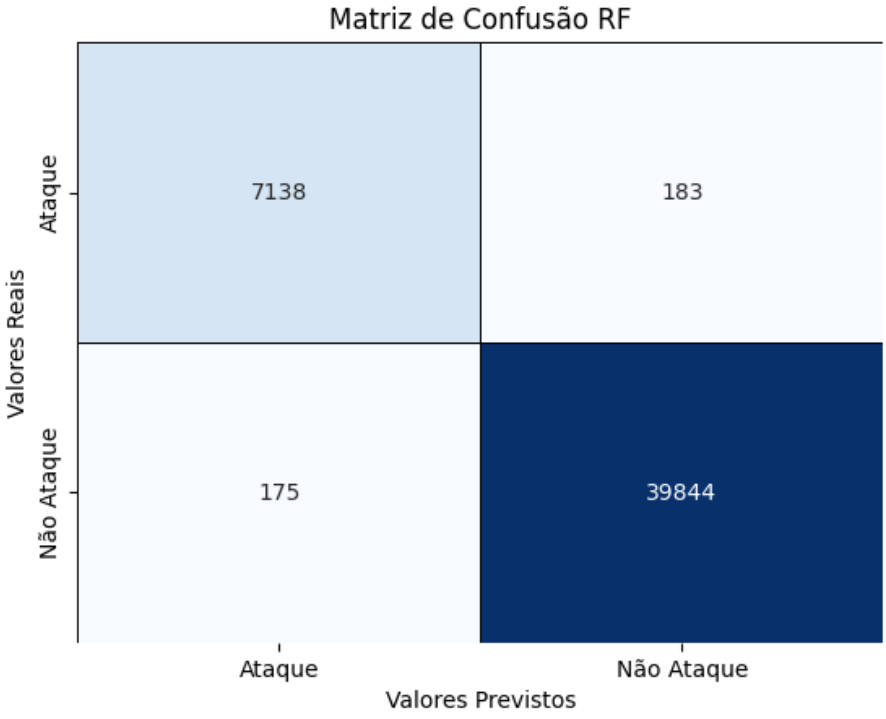

Figura 8. Matriz de confusão do *Random Forest* [Os Autores 2024]

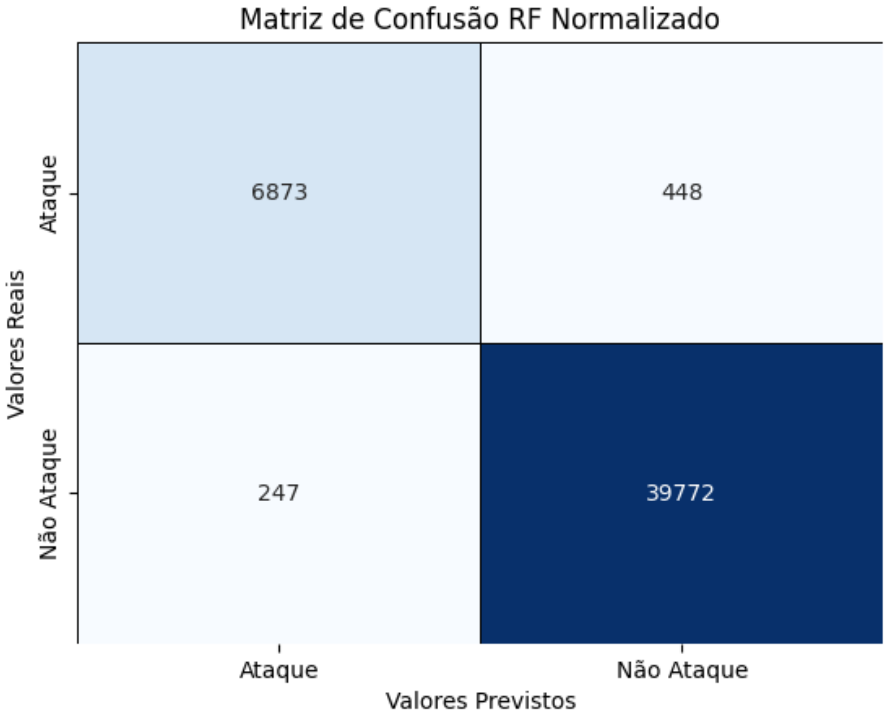

Figura 9. Matriz de confusão do *Random Forest* com os dados normalizados [Os Autores 2024]

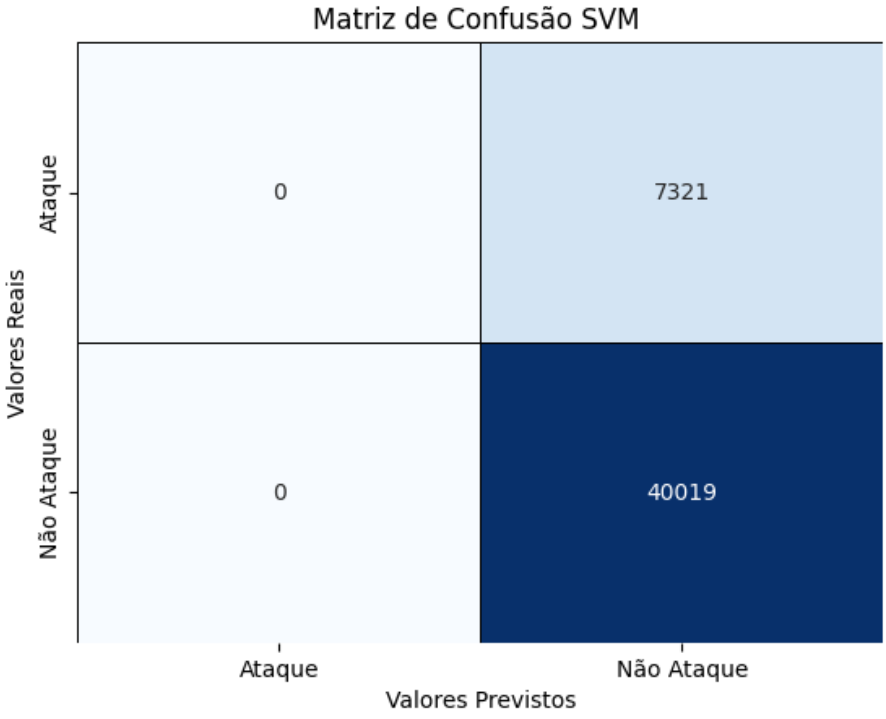

Figura 10. Matriz de confusão do *Support Vector Machine* [Os Autores 2024]

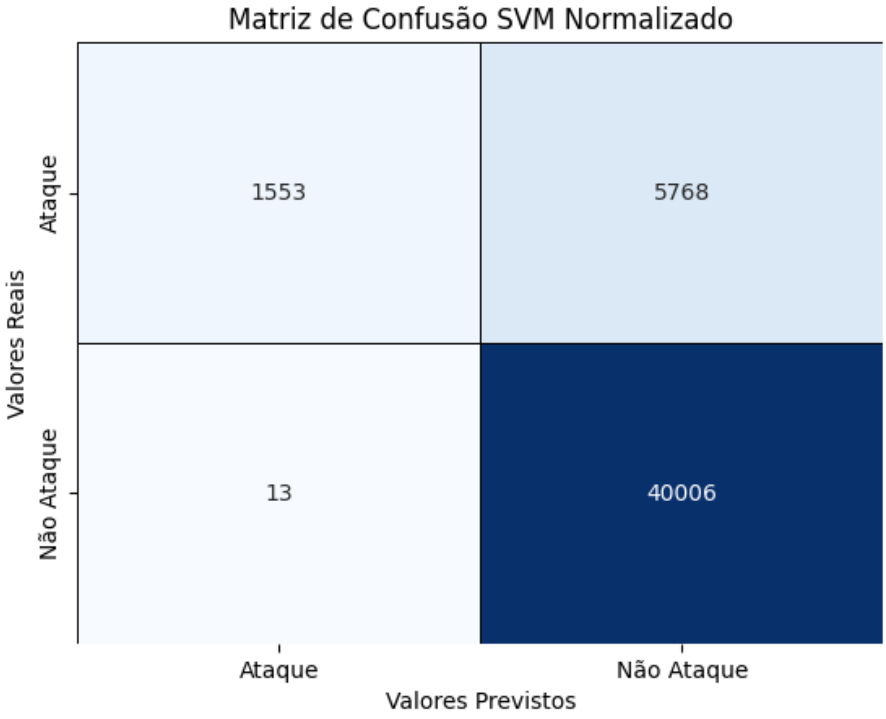

Figura 11. Matriz de confusão do *Support Vector Machine* com os dados normalizados [Os Autores 2024]

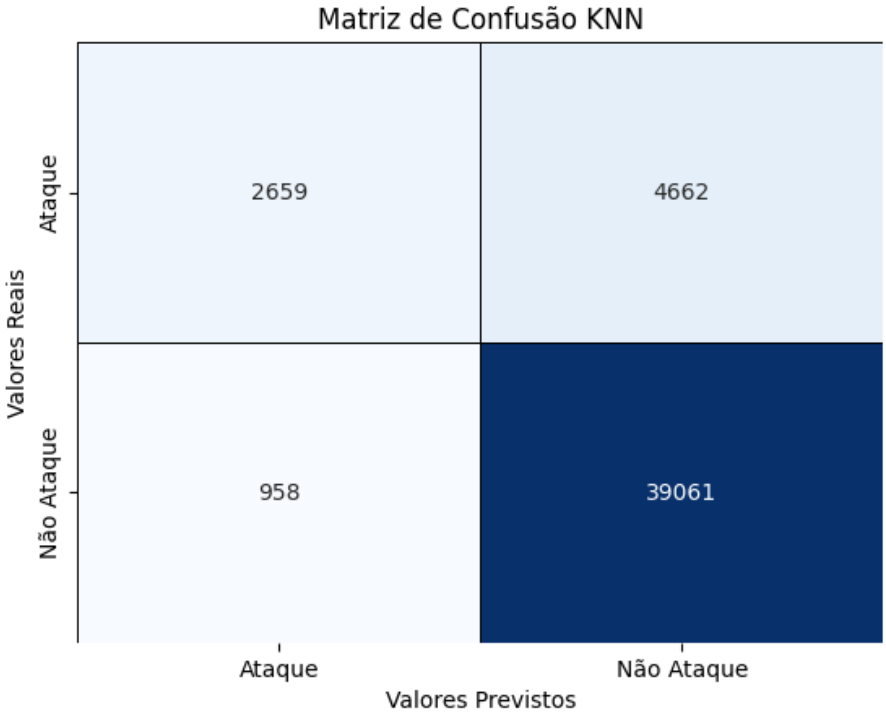

Figura 12. Matriz de confusão do KNN [Os Autores 2024]

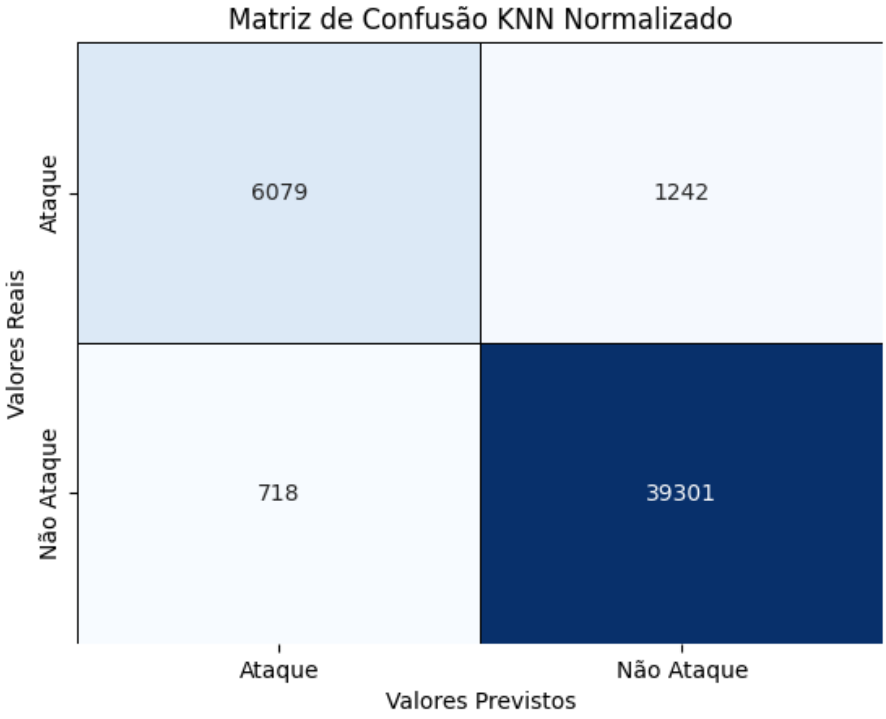

Figura 13. Matriz de confusão do KNN com os dados normalizados [Os Autores 2024]

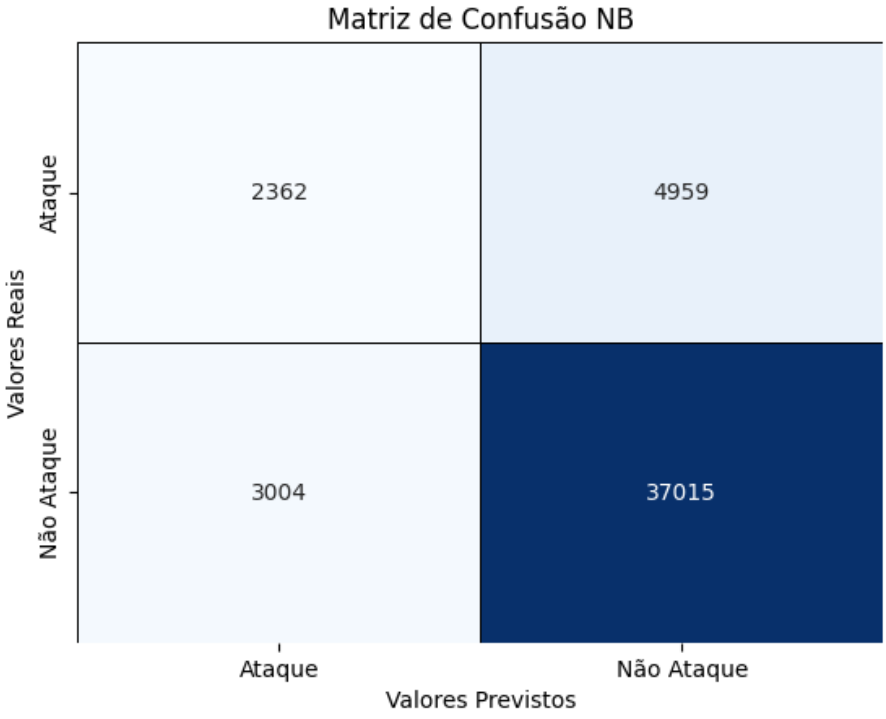

Figura 14. Matriz de confusão do *Naive Bayes* [Os Autores 2024]

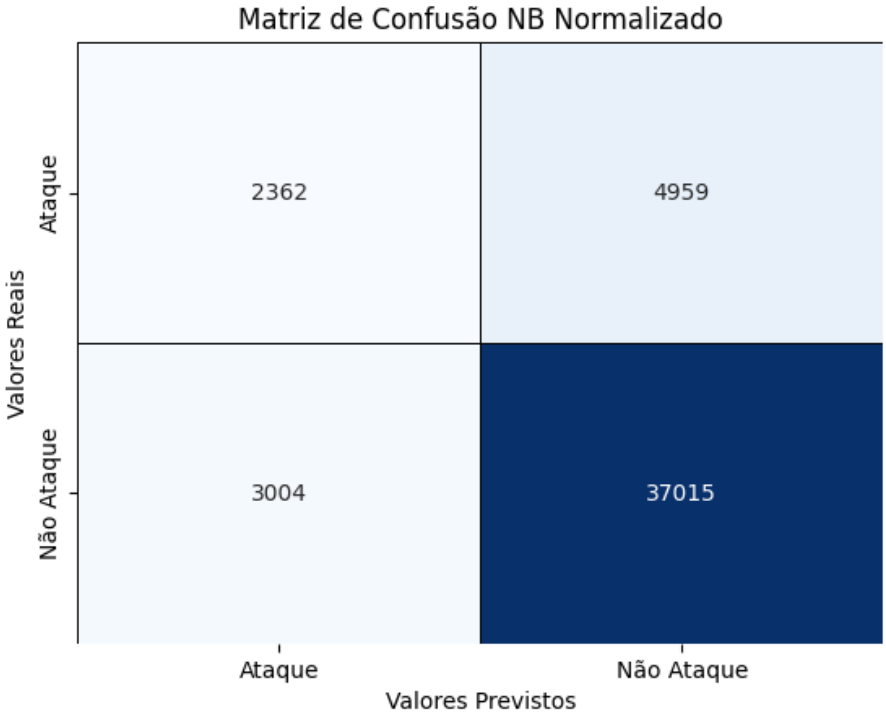

Figura 15. Matriz de confusão do *Naive Bayes* com os dados normalizados [Os Autores 2024]

Também foi realizada uma análise da importância das *features* para cada modelo. Para os dados não normalizados, os resultados estão apresentados nas seguintes figuras: para a *Decision Tree*, na Figura 16; para o *Random Forest*, na Figura 18; para o SVM, na Figura 20; e para o *Naive Bayes*, na Figura 22.

Já para os dados normalizados, as respectivas figuras são: para a *Decision Tree*, a Figura 17; para o *Random Forest*, a Figura 19; para o SVM, a Figura 21; e para o *Naive Bayes*, a Figura 23.

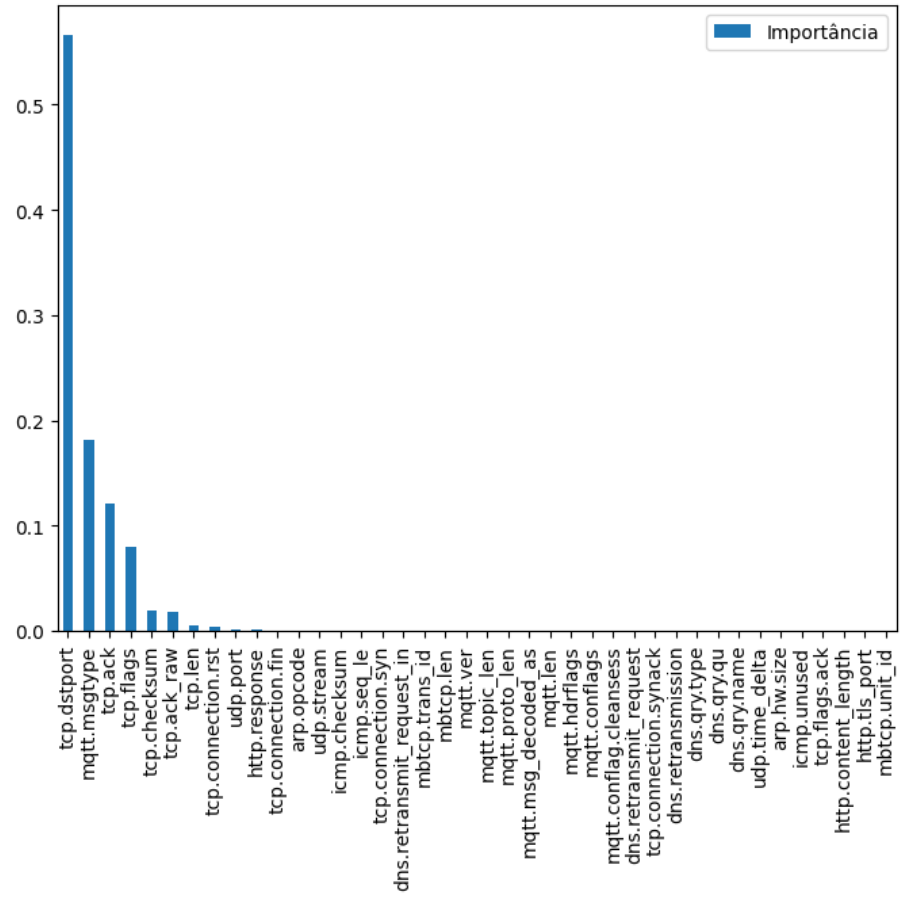

Figura 16. Gráfico de importância das *features* do modelo *Decision Tree* [Os Autores 2024]

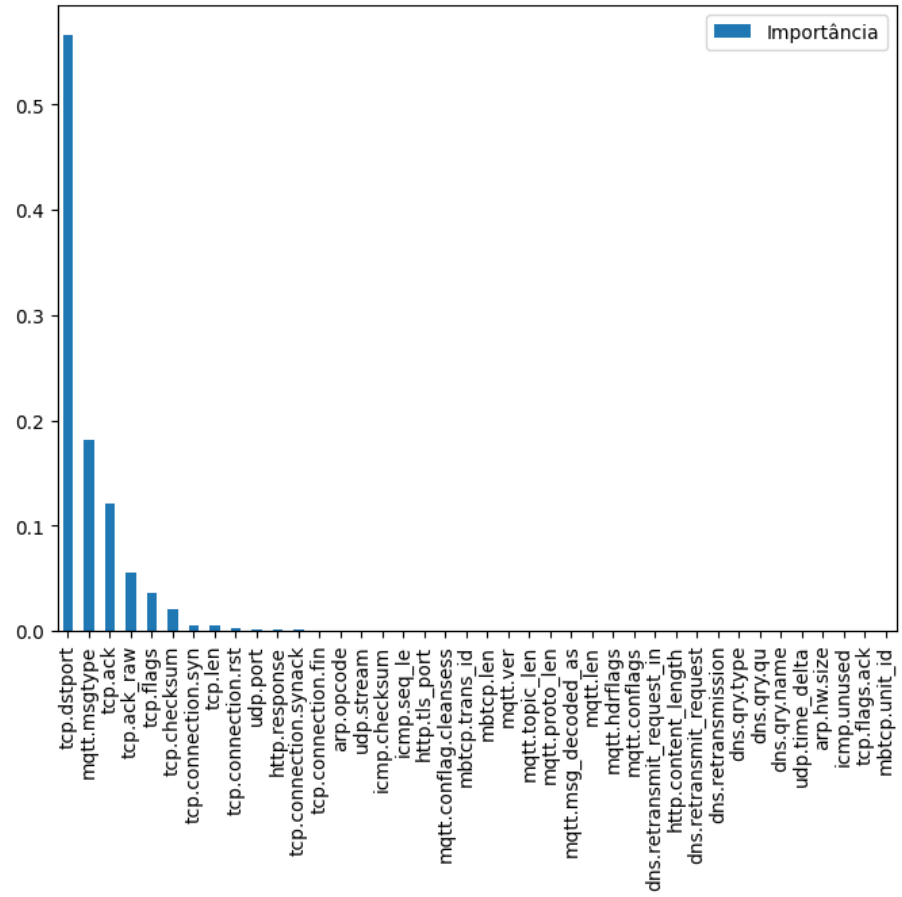

Figura 17. Gráfico de importância das *features* do modelo *Decision Tree* com os dados normalizados [Os Autores 2024]

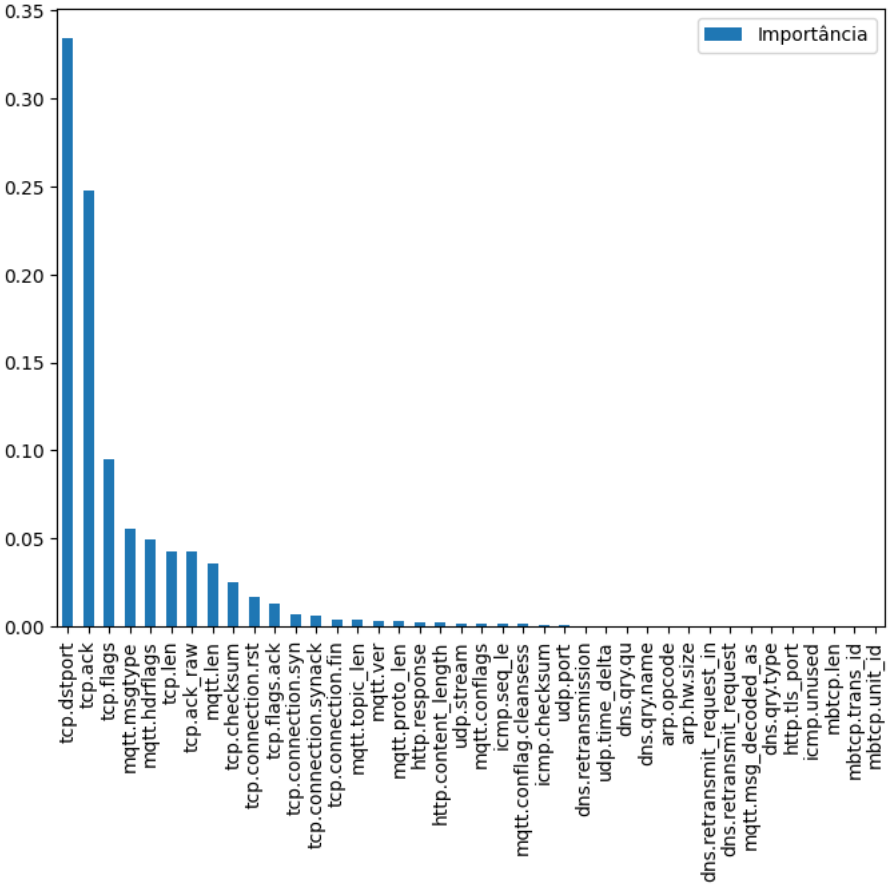

Figura 18. Gráfico de importância das *features* do modelo *Random Forest* [Os Autores 2024]

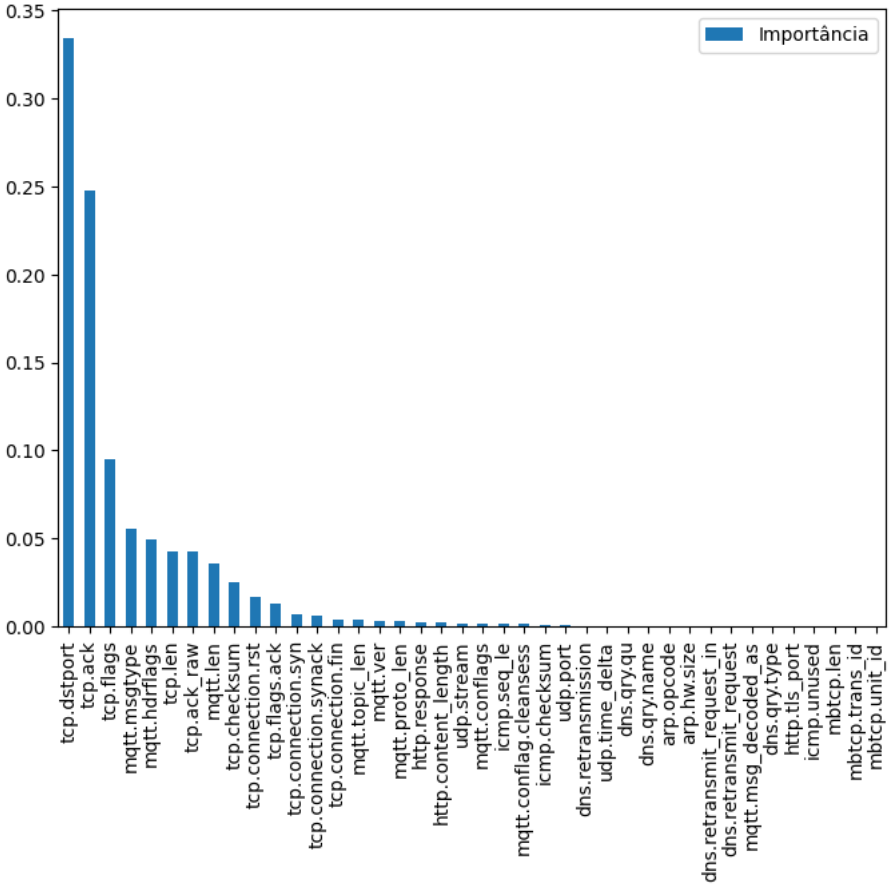

Figura 19. Gráfico de importância das *features* do modelo *Random Forest* com os dados normalizados [Os Autores 2024]

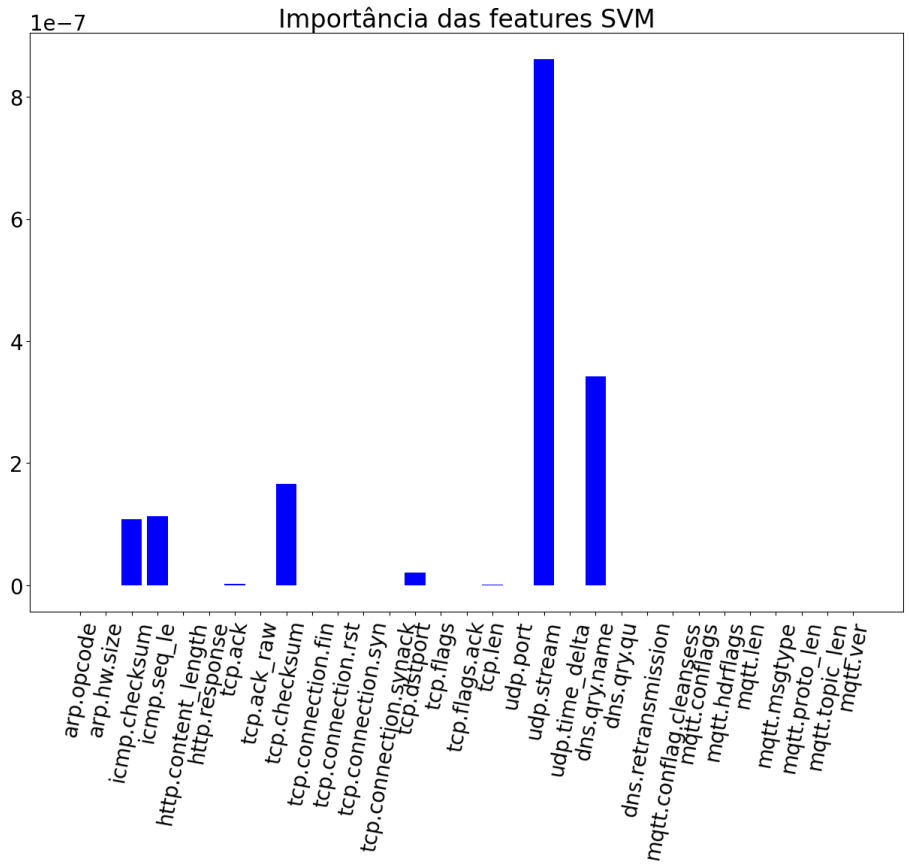

Figura 20. Gráfico de importância das *features* do modelo *Support Vector Machine* [Os Autores 2024]

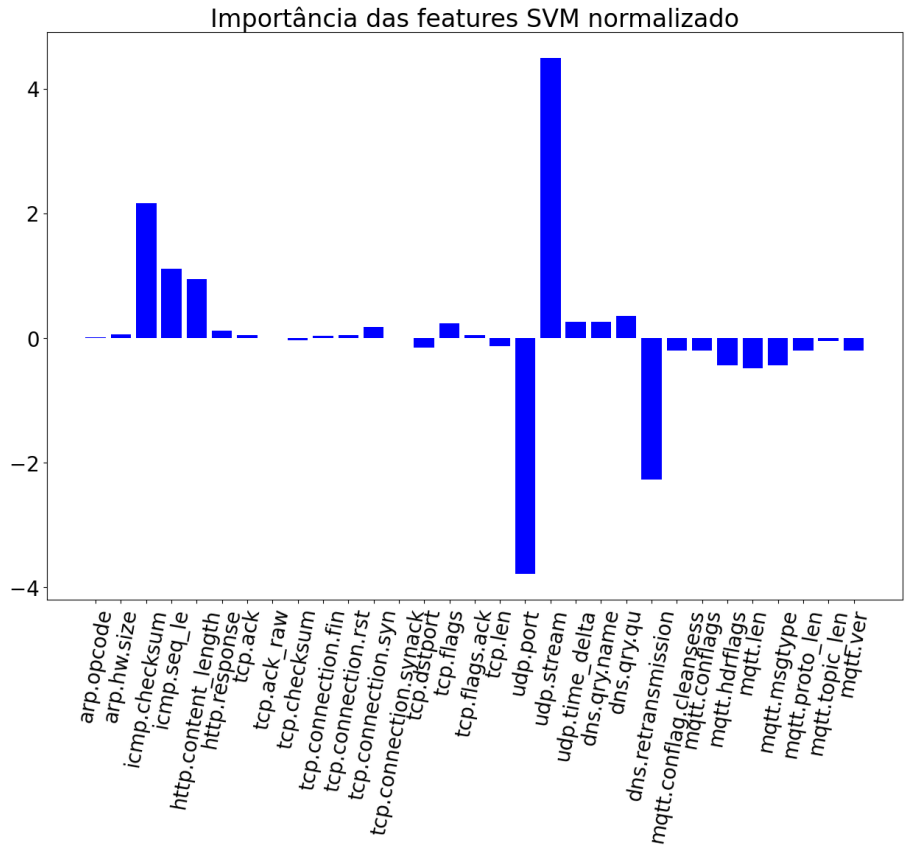

Figura 21. Gráfico de importância das *features* do modelo *Support Vector Machine* com os dados normalizados [Os Autores 2024]

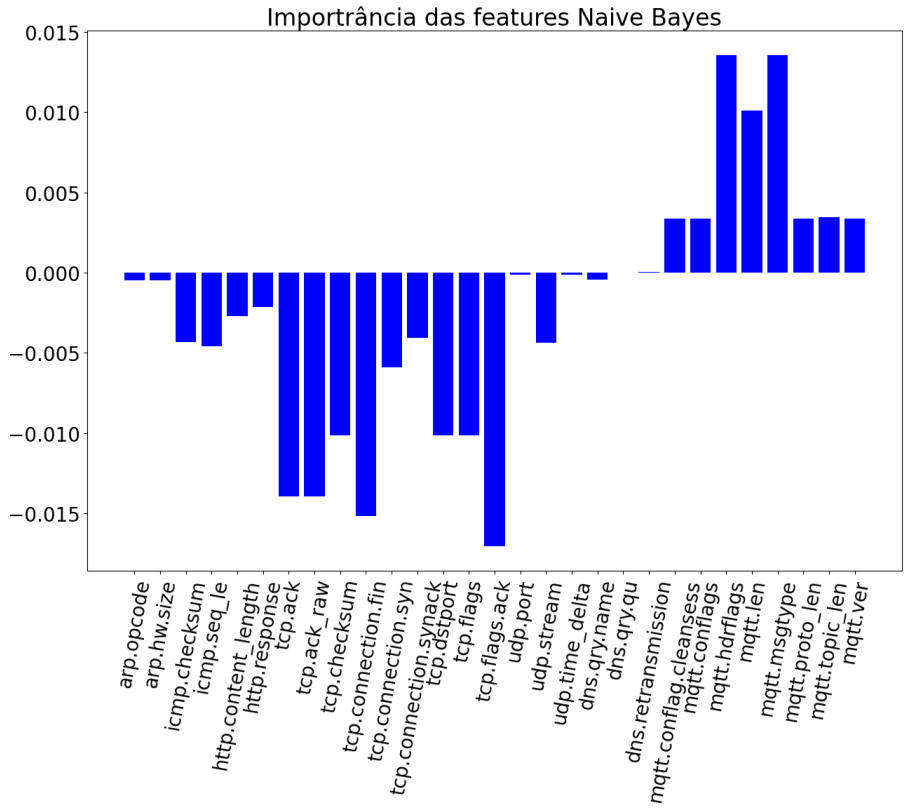

Figura 22. Gráfico de importância das *features* do modelo *Naive Bayes* [Os Autores 2024]

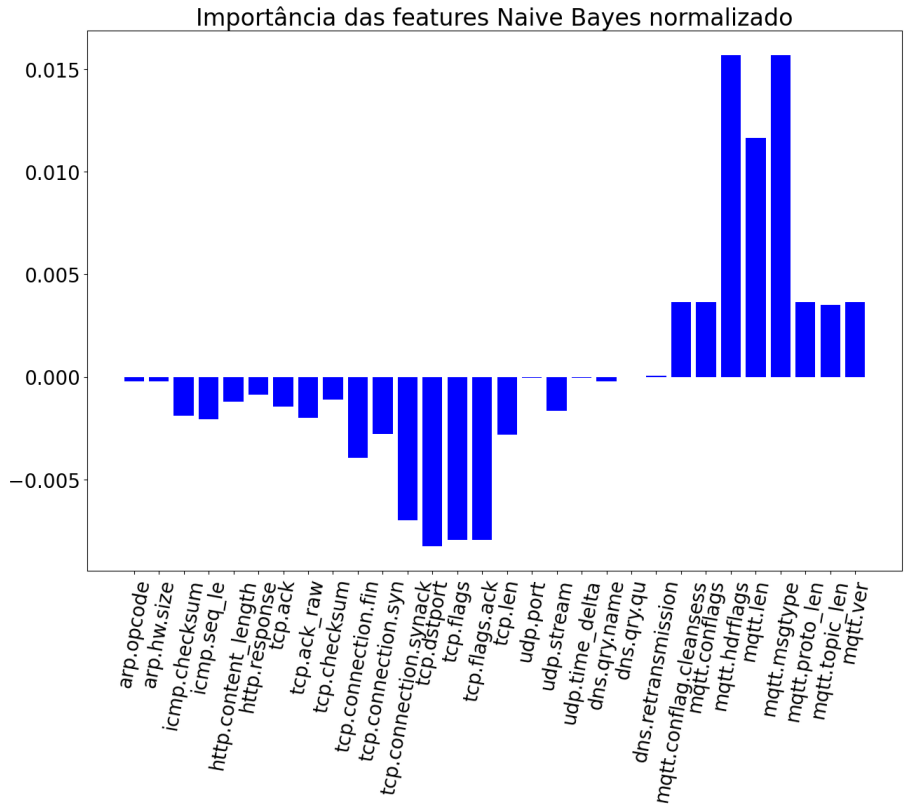

Figura 23. Gráfico de importância das *features* do modelo *Naive Bayes* com os dados normalizados [Os Autores 2024]

7. Conclusão

Este trabalho teve como objetivo realizar um estudo exploratório comparativo de modelos de aprendizado de máquina supervisionados para a detecção de ataques em tráfego de rede de dispositivos *IoT* domésticos. A primeira etapa do desenvolvimento do estudo consistiu na pesquisa de trabalhos similares ao objetivo do presente trabalho, bem como na busca por conjuntos de dados que representassem o tráfego de redes *IoT* domésticas, resultando na seleção do *Edge_IIoT_Dataset*.

Após a compreensão dos trabalhos correlatos e a seleção do conjunto de dados, foi iniciada a análise exploratória de dados (*EDA*) para compreender os diferentes tipos de tráfego de rede, como os normais e os padrões associados a ataques. Para isso, foi elaborado um sumário estatístico, permitindo uma melhor visualização da contagem de valores, descrição das colunas, verificação de dados ausentes, entre outros aspectos. Além disso, foram realizadas análises das distribuições de cada coluna, utilizando gráficos como o *boxplot* para apresentar os principais parâmetros, como a mediana, os quartis e os valores extremos. E por fim, foi realizada uma análise de correlação entre as *features* do conjunto de dados utilizando o mapa de calor.

Com a compreensão e visualização dos dados, foi realizada a etapa de teste e treinamento dos modelos de aprendizado de máquina supervisionados. Nesta etapa, os dados foram analisados tanto em sua forma normalizada quanto não normalizada, a fim de comparar o desempenho dos modelos. Os resultados indicaram que *Random Forest*

e *Decision Tree* apresentaram os melhores desempenhos com dados não normalizados, enquanto *K-Nearest Neighbors* e *Naive Bayes* demonstraram melhores resultados com dados normalizados.

Durante o desenvolvimento do trabalho de conclusão de curso, surgiram algumas dificuldades. Durante a análise exploratória, algumas colunas do subconjunto de dados ML apresentavam valores no formato *string*, o que demandou mais tempo para a extração de informações. Por conta disso, foi necessário excluir essas colunas. Além disso, o trabalho atravessou um período de greve, o que prejudicou o andamento no início deste trabalho.

Apesar das dificuldades, foi possível alcançar resultados satisfatórios. Espera-se que a publicação deste estudo exploratório possa contribuir com a comunidade acadêmica e ajudar as pessoas a compreender os riscos e o funcionamento dos dispositivos *IoT* e suas redes. Essa é uma tecnologia em expansão, tornando-se cada vez mais presente no cotidiano das pessoas com acesso a ela.

Referências

[Abranet 2023] Abranet (2023). Ataques a dispositivos da internet das coisas (iot) crescem 41%. `https://www.abranet.org.br/Noticias/Ataques-a-dispositivos-da-internet-das-coisas-%28IoT%29-crescem-41%25-4300.html`.

[Adjei et al. 2024] Adjei, J., Zincir-Heywood, N., Nandy, B., and Seddigh, N. (2024). Dalhousie nims lab iot 2024 dataset.

[Biau and Scornet 2016] Biau, G. and Scornet, E. (2016). A random forest guided tour. *Test*, 25:197–227.

[Bochie et al. 2020] Bochie, K., Gonzalez, E. R., Giserman, L. F., Campista, M. E. M., and Costa, L. H. M. (2020). Detecção de ataques a redes iot usando técnicas de aprendizado de máquina e aprendizado profundo. In *Anais do XX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 257–270. SBC.

[Brezolin et al. 2022] Brezolin, U. Q., Prates Jr, N. G., Vergütz, A., and Nogueira, M. (2022). Um método para detecçao de vulnerabilidades através da análise do tráfego de rede iot. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 447–460. SBC.

[Butun et al. 2019] Butun, I., Österberg, P., and Song, H. (2019). Security of the internet of things: Vulnerabilities, attacks, and countermeasures. *IEEE Communications Surveys & Tutorials*, 22(1):616–644.

[Emeç and Özcanhan 2023] Emeç, M. and Özcanhan, M. H. (2023). Rout-4-2023: Rpl based routing attack dataset for iot.

[Ferrag et al. 2022] Ferrag, M. A., Friha, O., Hamouda, D., Maglaras, L., and Janicke, H. (2022). Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications: Centralized and federated learning.

[Google-colab 2024] Google-colab (2024). Google colab. Disponível em: https://colab.google/.

[IEEE 2024] IEEE (2024). Ieee dataport. Disponível em: https://ieee-dataport.org/.

[Komorowski et al. 2016] Komorowski, M., Marshall, D. C., Salciccioli, J. D., and Crutain, Y. (2016). *Exploratory Data Analysis*, pages 185–203. Springer International Publishing, Cham.

[Koppula and L.M.I. 2024] Koppula, M. and L.M.I., L. J. (2024). A real time dataset "idsiot2024".

[LucidChart 2024] LucidChart (2024). Lucidchart. Disponível em: https://www.lucidchart.com/.

[Meneghello et al. 2019] Meneghello, F., Calore, M., Zucchetto, D., Polese, M., and Zanella, A. (2019). Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices. *IEEE Internet of Things Journal*, 6(5):8182–8201.

[Messas and Zarpelão 2022] Messas, G. E. and Zarpelão, B. B. (2022). Estudo sobre a segurança de dispositivos domésticos conectados à internet das coisas.

[Oliveira 2024] Oliveira, F. B. d. (2024). Framework para detecção de ataques dos em dispositivos iot, utilizando abordagens de aprendizado de máquinas.

[Omolara et al. 2022] Omolara, A. E., Alabdulatif, A., Abiodun, O. I., Alawida, M., Alabdulatif, A., and Arshad, H. (2022). The internet of things security: A survey encompassing unexplored areas and new insights. *Computers & Security*, 112:102494.

[PACETE 2022] PACETE, L. (2022). Iot: até 2025, mais de 27 bilhões de dispositivos estarão conectados.

[Scikit-Learn 2024] Scikit-Learn (2024). Scikit-learn. Disponível em: https://scikit-learn.org/.

[Silva Filho 2021] Silva Filho, J. G. d. (2021). Detecção de anomalias no tráfego mqtt de redes iot utilizando técnicas de aprendizado de máquina.

[VanderPlas 2023] VanderPlas, J. (2023). *Python Data Science Handbook*. O'Reilly, 2nd edition.

[Xenofontos et al. 2021] Xenofontos, C., Zografopoulos, I., Konstantinou, C., Jolfaei, A., Khan, M. K., and Choo, K.-K. R. (2021). Consumer, commercial, and industrial iot (in) security: Attack taxonomy and case studies. *IEEE Internet of Things Journal*, 9(1):199–221.

[Zhang 2016] Zhang, Z. (2016). Introduction to machine learning: k-nearest neighbors. *Annals of translational medicine*, 4(11).