

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Gabrielly Rodrigues da Silva

Combinação de Aprendizado de Máquina e Pilha ELK para Detecção Automática de Ataques Cibernéticos: Uma Abordagem Comparativa

Fortaleza – CE

11 de março de 2025

Gabrielly Rodrigues da Silva

Combinação de Aprendizado de Máquina e Pilha ELK para Detecção Automática de Ataques Cibernéticos: Uma Abordagem Comparativa

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Linha de Pesquisa: Redes de Computadores

Orientador: Prof. Dr. Reinaldo Bezerra Braga

Coorientador: Me. Bruno Marques Cavalcante de Oliveira

Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)

Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Fortaleza – CE

11 de março de 2025

Dados Internacionais de Catalogação na Publicação

Instituto Federal do Ceará - IFCE

Sistema de Bibliotecas - SIBI

Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

D111c da Silva, Gabrielly.

Combinação de Aprendizado de Máquina e Pilha ELK para Detecção Automática de Ataques Cibernéticos: Uma Abordagem Comparativa / Gabrielly da Silva. - 2025.
78 f.

Dissertação (Mestrado) - Instituto Federal do Ceará, Mestrado em Ciência da Computação, Campus Fortaleza, 2025.

Orientação: Profa. Dra. Reinaldo Bezerra Braga.

Coorientação: Profa. Ma. Bruno Marques Cavalcante de Oliveira.

1. SIEM. . 2. Inteligência Artificial.. 3. Sistema de detecção de ataque.. 4. Pilha ELK.. I.
Titulo.

CDD 004



Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)
Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Gabrielly Rodrigues da Silva

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, sendo aprovada pela Coordenação do Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará e pela banca examinadora:

Orientador: Prof. Dr. Reinaldo Bezerra Braga
Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

**Coorientador: Me. Bruno Marques Cavalcante
de Oliveira**
Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Prof. Dr. Tales Paiva Nogueira
Universidade da Integração Internacional da Lusofonia
Afro-Brasileira (UNILAB)

Prof. Dr. Fernando Parente Garcia
Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Fortaleza – CE
11 de março de 2025

*Este trabalho é dedicado a todas as pessoas que,
de alguma forma, tem contribuído para a segurança da informação,
baseado no ensino e na pesquisa científica.*

Agradecimentos

Este trabalho de dissertação representa a conclusão de uma etapa significativa da minha formação, marcada por uma rica variedade de aprendizagens, tanto pessoais quanto acadêmicas. Neste momento, sinto a necessidade de expressar minha sincera gratidão a todos que, de maneira direta ou indireta, contribuíram para que eu percorresse esse caminho.

Primeiramente, agradeço a Deus por me sustentar nos momentos difíceis dos últimos dois anos. Agradeço aos meus pais, Jessé e Januse, por me darem a oportunidade e o apoio para seguir com minha formação acadêmica. Tenho plena certeza de que todas as minhas conquistas são também suas. A minha irmã Ingrid, que pacientemente me ouviu lendo textos em voz alta e me ajudando a encontrar trechos importantes para aprimorar meu trabalho. Ao meu namorado João Pedro, por estar ao meu lado, oferecendo apoio e me ouvindo nos momentos em que considerei desistir.

Minha gratidão ao meu orientador, Dr. Reinaldo Braga, pelo conhecimento compartilhado, pelo apoio contínuo e por acreditar no meu potencial como aluna e pesquisadora desde a graduação. Agradeço também ao meu co-orientador, Me. Bruno de Oliveira, por dividir sua valiosa experiência profissional na área de segurança, que foi fundamental para a origem deste trabalho. Sou grata à Dra. Carina Oliveira, por me proporcionar um novo ponto de vista no início deste projeto, e ao Dr. Francisco Nélío, pela ajuda inestimável no aprimoramento das minhas habilidades de apresentação e na expressão das minhas ideias.

A todos, o meu mais sincero agradecimento!

“Prevention is ideal, but detection is a must.”
(James R. Chiles, autor de "Inviting Disaster".)

Resumo

Com o avanço tecnológico e o surgimento de ataques cada vez mais sofisticados, a preocupação com a proteção das redes contra ataques cibernéticos aumentou significativamente. Logo, instituições e empresas passaram a adotar ferramentas para manter a rede sob monitoramento constante, impulsionando o desenvolvimento de soluções de Gerenciamento de Informações e Eventos de Segurança, conhecidas pela sigla em inglês *Security Information and Event Management* (SIEM). No entanto, mesmo com o monitoramento contínuo, ainda é necessário analisar os dados coletados pelo sistema para detectar possíveis ataques, sendo que parte desse processo ainda é realizada manualmente. Diante desse cenário, foi conduzido um Mapeamento Sistemático (MS) para identificar os algoritmos mais utilizados na literatura. Os resultados apontaram que os mais comuns são *Decision Tree* (DT), *Random Forest* (RF), *Support Vector Machine* (SVM) e *Naive Bayes* (NB). Além disso, observou-se que os bancos de dados de ataques disponíveis na literatura geralmente contêm apenas amostras de ataques finalizados, sem considerar as técnicas empregadas para realizá-los. Para preencher essa lacuna, esta dissertação simulou e coletou amostras de nove técnicas de ataque, com o objetivo de analisá-las e detectá-las por meio de métodos manuais e automatizados. Nesse contexto, o trabalho apresenta um fluxo de detecção utilizando a *ElasticSearch*, *Logstash* and *Kibana Stack* (Pilha ELK) e Inteligência Artificial (IA). Os resultados demonstraram que o método manual proporciona uma melhor visualização do sistema, porém sua acurácia foi inferior a 39%. Já no método automatizado, foram estudados dois cenários: detecção de multiclases e detecção binária. No primeiro cenário, destacaram-se os algoritmos DT e NB, ambos com 100% de acurácia no caso de amostras balanceadas. No caso de amostras desbalanceadas, os melhores desempenhos foram obtidos por DT e RF, com acurácias de 100% e 71,4%, respectivamente. No segundo cenário, os algoritmos DT, RF e NB alcançaram 100% de acurácia, enquanto o SVM obteve 96%. Os resultados indicam que as abordagens automatizadas apresentaram um desempenho significativo na detecção de ataques. Também, observou-se que o algoritmo SVM demanda mais recursos computacionais em comparação com os demais analisados.

Palavras-chave: SIEM. Inteligência Artificial. Sistema de detecção de ataque. Pilha ELK.

Abstract

With technological evolution and growing of sophisticated attacks on the network, we observe that network protection against cyber-attacks has been also increased. Consequently, institutions and companies have used tools to maintain constant network monitoring, leading to the adoption of Security Information and Event Management (SIEM) systems. However, even with network monitoring, there is still a need for data analysis when this data is collected by the system to detect possible attacks. This process remains manual. As a result, various studies and strategies aim to use machine learning, such as Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM) and Naive Bayes (NB), to improve intrusion detection systems. In this context, this work presents a workflow of detection functioning using the SIEM system and machine learning, aiming the comparison among manual methods and automated methods for attack detection.

Keywords: SIEM. Machine Learning. Intrusion detection system. ELK Stack.

Lista de ilustrações

Figura 1 – Gráfico Temporal da Quantidade de Pesquisas por Ano	29
Figura 2 – Visão Geral entre Autores e Algoritmos	30
Figura 3 – Visão Geral entre Banco de Dados e Número de Citações	31
Figura 4 – Categorias de Classificação do IDS.	36
Figura 5 – Arquitetura Pilha ELK.	37
Figura 6 – Ilustração da Árvore de Decisão	40
Figura 7 – Ilustração da Floresta Aleatória	41
Figura 8 – Ilustração do SVM com Kernel	41
Figura 9 – Fluxograma Proposto	43
Figura 10 – Tela de Visualização dos Eventos <i>Logs</i>	51
Figura 11 – Tela de Visualização dos Alertas Emitidos	52
Figura 12 – Histograma dos Alertas Emitidos usando a Regra 1	53
Figura 13 – Histograma dos Alertas Emitidos usando a Regra 2	54
Figura 14 – Histograma dos Alertas Emitidos usando a Regra 3	54
Figura 15 – Matrizes de Confusão DT Desbalanceada	55
Figura 16 – Matrizes de Confusão DT Balanceada	56
Figura 17 – Curvas de Aprendizagem de Precisão e Erro (DT)	56
Figura 18 – Matrizes de Confusão RF Desbalanceada	57
Figura 19 – Matrizes de Confusão RF Balanceada	58
Figura 20 – Curvas de Aprendizagem de Precisão e Erro (RF)	58
Figura 21 – Matrizes de Confusão SVM Desbalanceada	59
Figura 22 – Matrizes de Confusão SVM Balanceada	59
Figura 23 – Curvas de Aprendizagem de Precisão e Erro (SVM)	60
Figura 24 – Matrizes de Confusão NB Desbalanceada	60
Figura 25 – Matrizes de Confusão NB Balanceada	61
Figura 26 – Curva de Aprendizagem de Precisão e Erro (NB)	61
Figura 27 – Matrizes de Confusão DT	62
Figura 28 – Curva de Aprendizagem de Precisão e Erro (DT)	62
Figura 29 – Matrizes de Confusão RF	63
Figura 30 – Curva de Aprendizagem de Precisão e Erro (RF)	63
Figura 31 – Matrizes de Confusão SVM	64
Figura 32 – Curva de Aprendizagem de Precisão e Erro (SVM)	64
Figura 33 – Matrizes de Confusão NB	65
Figura 34 – Curva de Aprendizagem de Precisão e Erro (NB)	65

Lista de tabelas

Tabela 1 – Tabela PICO Adaptada (SANTOS; PIMENTA; NOBRE, 2007).	27
Tabela 2 – Strings de Busca	28
Tabela 3 – Critérios de Inclusão e Exclusão	28
Tabela 4 – Trabalhos Selecionados.	30
Tabela 5 – Ataques Presentes nos Bancos de Dados e a Representação Percentual dentro da Amostra Total.	32
Tabela 6 – Ataques Executados	45
Tabela 7 – Atributos do Evento Log	47
Tabela 8 – Quantidade de Amostras por Ataques	47
Tabela 9 – Comparação da Profundidade Máxima com Acurácia Média	55
Tabela 10 – Métricas da Classificação Multiclasse Desbalanceada.	66
Tabela 11 – Métricas da Validação Cruzada	67
Tabela 12 – Métricas da Classificação Multiclasse Balanceada.	67
Tabela 13 – Métricas da Validação Cruzada	67
Tabela 14 – Comparativo do Método Manual e Automatizado	70

Lista de abreviaturas e siglas

DDoS	<i>Distributed Denial of Service</i>
DoS	<i>Denial of Service</i>
DT	<i>Decision Tree</i>
FN	Falso Negativo
FP	Falso Positivo
HIDS	<i>Host Intrusion Detection System</i>
IA	Inteligência Artificial
IDS	Sistema de Detecção de Intrusão
ML	<i>Machine Learning</i>
MS	Mapeamento Sistemático
NB	<i>Naive Bayes</i>
NIDS	<i>Network Intrusion Detection System</i>
OSSIM	Open Source Security Information Management
PICO	População, Intervenção, Comparação e <i>Outcomes</i>
Pilha ELK	<i>Elasticsearch, Logstash and Kibana Stack</i>
PLN	Processamento de Linguagem Natural
RBF	Função de Base Radial
RF	<i>Random Forest</i>
SIEM	Gerenciamento de Informações e Eventos de Segurança
SVM	<i>Support Vector Machine</i>
TF-IDF	Frequência do Termo e Frequência Inversa do Documento
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo

Sumário

1	INTRODUÇÃO	23
1.1	Objetivos Geral e Específicos	24
1.2	Produção Científica	25
1.3	Estrutura da Dissertação	25
2	MAPEAMENTO SISTEMÁTICO	27
2.1	Planejamento	27
2.2	Condução	29
2.3	Resultados	29
2.3.1	Algoritmos de IA	30
2.3.2	Bancos de dados	31
3	CONTEXTO TECNOLÓGICO	35
3.1	Sistema de Detecção de Intrusão	35
3.2	Soluções SIEM	36
3.3	Framework Mitre Attack	38
3.4	Inteligência Artificial	39
4	PROPOSTA DA DISSERTAÇÃO	43
4.1	Visão Geral	44
4.2	Atomic Red Team	44
4.3	Pré-Processamento	46
4.4	Métricas de Avaliação	48
5	RESULTADOS E DISCUSSÕES	51
5.1	Método Manual	51
5.2	Método Automatizado	54
5.2.1	Cenário 1: Classificação Multiclasse Desbalanceada e Balanceada	54
5.2.2	Cenário 2: Classificação Binária	61
5.3	Discussões	65
6	CONCLUSÃO	69
7	TRABALHOS FUTUROS	71
	REFERÊNCIAS	73

1 Introdução

A disseminação de informações passou por uma evolução significativa, migrando do meio analógico para o digital, transformando a área da comunicação (MATTOS, 2013). Inicialmente, as ameaças à segurança da informação concentravam-se principalmente na interceptação física de correspondências ou na escuta clandestina de diálogos. No entanto, com o avanço da tecnologia, a circulação de informações foi gradualmente transferida para o ambiente digital (MARCON, 2023), acompanhada pelo desenvolvimento de métodos mais sofisticados de interceptação e ataques.

Embora o avanço tecnológico tenha facilitado a troca de informações, ele também trouxe à tona novas vulnerabilidades e formas de ataque, uma vez que os modelos de ataques vêm evoluindo paralelamente. Segundo levantamento da empresa Kaspersky (2024), o sistema de detecção revelou que o número de ameaças cibernéticas cresceu 3% em 2023 em relação ao ano anterior. Em média, 411 mil arquivos maliciosos foram detectados diariamente, com o sistema operacional *Windows* sendo o principal alvo. Esses dados reforçam a urgência de implementar sistemas eficazes de coleta, monitoramento, análise e mitigação de ataques digitais, operados por profissionais especializados em segurança cibernética.

Como destacado por Ayala et al. (2021), na área de segurança, existem três fatores que impactam nas decisões: **tempo**, porque as ameaças precisam ser resolvidas rapidamente; **procedimentos manuais**, pois os sistemas de monitoramento ainda apresentam processos de identificação e detecção manual; e **decisões subjetivas**, visto que o processo para mitigar um incidente depende do conhecimento e experiência do profissional. Ou seja, os três fatores que influenciam nas tomadas de decisões são inversamente proporcionais, visto que quanto menor for o conhecimento de um profissional da área de segurança cibernética sobre o ataque em análise, maior será o tempo necessário para validar a ameaça e informar ao sistema de monitoramento se o alerta gerado é verdadeiro ou falso.

Com isso, profissionais da área de segurança são responsáveis pelo monitoramento, análise e mitigação de possíveis ataques às redes. Para realizar essas atividades, normalmente, utilizam-se de uma solução de Gerenciamento de Informações e Eventos de Segurança, do inglês (Security Information and Event Management - SIEM). De acordo com (KIMATHI, 2017), o sistema de SIEM fornece análise em tempo real dos dados coletados, ao mesmo tempo que os armazenam. Essas soluções podem apresentar informações de controle de acesso do sistema operacional, ferramentas de gerenciamento de vulnerabilidades, ferramentas de política e conformidade, banco de dados e *logs* de aplicativos.

O sistema Gerenciamento de Informações e Eventos de Segurança (SIEM), *Elasticsearch, Logstash and Kibana Stack* (Pilha ELK), tem como característica principal a coleta de eventos de todas as fontes de *logs* possíveis. Em seguida, é feita uma normalização, na qual organiza cada amostra com os respectivos atributos para depois ocorrer a indexação que facilita a pesquisa e busca rápida nos campos importantes. Assim, é possível fazer uma filtragem e criar regras de eventos para identificar eventos considerados ameaças à segurança, conseguindo gerar uma correlação das amostras coletadas com as regras, o sistema SIEM envia alertas sobre possível violação da segurança (KIMATHI, 2017). Esses alertas são então verificados individualmente pelos profissionais de cibersegurança, pois nem todo alerta emitido representa uma ameaça em potencial, podendo haver falsos positivos.

Além disso, os profissionais de segurança cibernética são acionados para verificar os alertas emitidos pelo sistema. Observa-se que esse processo exige conhecimento e precisão por parte dos profissionais para determinar se o alerta é legítimo. No entanto, à medida que as redes gerenciadas se expandem, o volume de dados e alertas recebidos aumenta proporcionalmente, sobrecarregando os responsáveis por sua análise. Como consequência, o tempo de resposta para cada alerta se prolonga, e muitos alertas que poderiam ser descartados acabam sendo analisados desnecessariamente.

Nesse contexto, esta dissertação apresenta um Mapeamento Sistemático (MS) que destaca os algoritmos *Decision Tree* (DT), *Random Forest* (RF), *Support Vector Machine* (SVM) e *Naive Bayes* (NB) como os classificadores mais utilizados na literatura. Além dos algoritmos, são enfatizados conjuntos de dados que contêm amostras de ataques, como CIC IDS 2018, UNSW-NB15, NSL-KDD e CIC IDS 2017, que, em sua maioria, incluem registros de ataques já finalizados. Por essa razão, este trabalho buscou simular e coletar amostras de técnicas de ataque que representam as etapas de um ataque real, por meio de uma biblioteca de testes mapeados para a estrutura MITRE ATT&CK¹, conhecida como *Atomic Red Team*².

Esta dissertação apresenta também, como contribuição principal, uma abordagem comparativa entre métodos manuais e métodos automatizados. Atribuindo-se o sistema da Pilha ELK ao método manual e os algoritmos destacados pelo MS ao método automatizado. Tendo como objetivo destacar as vantagens e limitações de cada abordagem.

1.1 Objetivos Geral e Específicos

Tendo em vista o aumento de ataques cibernéticos nas redes e o gerenciamento de alertas feitos de forma manual, o objetivo deste trabalho é comparar as vantagens

¹ <<https://attack.mitre.org/>>

² <<https://github.com/redcanaryco/atomic-red-team>>

e limitações do método manual proposto pela Pilha ELK e do método automatizado composto por quatro algoritmos: DT, RF, SVM e NB.

Os objetivos específicos deste trabalho são:

- Pesquisar e analisar algoritmos de aprendizagem de máquina que possam ser utilizados na detecção de ataques;
- Levantar e analisar bancos de dados de ataques presentes na literatura;
- Simular cenários de técnicas de ataques específicos do Mitre Attack;
- Coletar dados de técnicas de ataques armazenados na solução SIEM;
- Analisar e implementar regras de detecção na Pilha ELK;
- Classificar ataques simulados usando *Machine Learning* (ML);
- Analisar e avaliar quantitativamente e qualitativamente os algoritmos utilizando métricas como acurácia, f1-score, matriz de confusão, tempo de treinamento e tempo de teste;
- Comparar métodos de detecção manual e automatizado destacando vantagens e limitações.

1.2 Produção Científica

Durante este projeto de mestrado, o seguinte trabalho científico foi aceito e publicado, a saber:

- **DA SILVA, Gabrielly; OLIVEIRA, Carina; BRAGA, Reinaldo., 2023. Um Mapeamento Sistemático sobre Detecção de Ataques em Redes de Computadores. In: ESCOLA REGIONAL DE COMPUTAÇÃO DO CEARÁ, MARANHÃO E PIAUÍ (ERCEMAPI), 11. , 2023, Aracati/CE. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2023 . p. 11-20. DOI: <https://doi.org/10.5753/ercemapi.2023.236238>;**

1.3 Estrutura da Dissertação

Esta dissertação está organizada da seguinte forma:

O Capítulo 2 trata da descrição do mapeamento sistemático produzido com o levantamento bibliográfico. Consiste em um mapeamento de trabalhos relacionados à temática desta proposta, identificando o estado da arte e as oportunidades encontradas nessa linha de pesquisa e reforçando a relevância da contribuição científica contida na presente dissertação.

O Capítulo 3 aborda a fundamentação teórica das técnicas utilizadas neste trabalho. Inicialmente, há uma contextualização sobre Sistema de Detecção de Intrusão (IDS), em

seguida, as arquiteturas SIEM, *framework Mitre Attack* e Inteligência Artificial (IA) para classificação.

O Capítulo 4 apresenta o modelo proposto, assim como a metodologia desenvolvida. Inicialmente, a arquitetura proposta é dividida no fluxo geral do processo e no funcionamento interno, justificando as escolhas realizadas por meio do banco de dados. Além de demonstrar os passos do pré-processamento e o funcionamento da Pilha ELK.

O Capítulo 5 aponta os resultados e discussões, contendo os resultados obtidos dos algoritmos do DT, RF, SVM e NB. Além de uma comparação com dados balanceados e desbalanceados para criar um modelo generalizado.

Por fim, no Capítulo 6 mostra as conclusões alcançadas por meio dessa dissertação, além de contribuições e trabalhos futuros.

2 Mapeamento Sistemático

Um Mapeamento Sistemático (MS) é um estudo secundário que pretende analisar estudos primários e promover uma visão geral da área de pesquisa, além de contribuir para identificar lacunas dentro dessa área (FALBO, 2018). O Objetivo do MS é identificar, avaliar e interpretar os estudos realizados na literatura. A metodologia utilizada é baseada no trabalho de Kitchenham e Charters (2007) e encontra-se dividido em três etapas: planejamento, condução e resultado. Este mapeamento sistemático foi publicado no ERCEMAPI 2023¹.

2.1 Planejamento

Nesta etapa, busca-se encontrar um critério de busca para padronizar a pesquisa dentro dos repositórios de artigos como: Wiley, Scielo, IEEEExplore, ACM Digital Library, ScienceDirect e SpringerLink. Para gerar esse critério de busca, usa-se a Tabela 1 para auxiliar nesse processo. Nela, é apresentada uma tabela de População, Intervenção, Comparação e *Outcomes* (PICO), que tem como objetivo identificar as *strings* de busca que são relevantes dentro do contexto da pesquisa.

Tabela 1 – Tabela PICO Adaptada (SANTOS; PIMENTA; NOBRE, 2007).

Elemento	Descrição	Palavras-chaves
População	Todas as redes que são foco dos cibercriminosos	<i>Cyber Attack</i>
Intervenção	Deteção dos alertas falsos positivos e deteção das técnicas/táticas na rede	<i>Intrusion Detection, Anomaly, Signature</i>
Comparação	Comparar soluções de IA e <i>datasets</i> utilizados	<i>Deep Learning, Machine Learning</i>
Resultados	Verificar a efetividade dos resultados obtidos	-

Fonte: Elaborado pelos autores.

Após contextualizar a área geral do foco do MS, determinou-se as *strings* de busca para cada repositório pesquisado como mostrado na Tabela 2. Tendo os trabalhos pesquisados, faz-se necessário definir critérios de inclusão e exclusão que serão adotados durante a análise dos artigos. Com isso, os critérios de inclusão e exclusão são mostrados na Tabela 3.

Os critérios de inclusão escolhidos são para garantir que os artigos analisados apresentem algoritmos de Inteligência Artificial (IA) com foco em deteção de ataques

¹ DOI:<10.5753/ercemapi.2023.236238>

Tabela 2 – Strings de Busca

Repositórios	String de Busca
Wiley	{"Cyber Attack"anywhere and "Intrusion Detection"anywhere and "Anomaly" anywhere and "Signature"anywhere and "Deep Learning"anywhere and "Machine Learning"anywhere}
Scielo	{(Cyber Attack) AND (Intrusion Detection) OR (Anomaly) OR (Signature) AND (Deep Learning) OR (Machine Learning)}
IEEEExplore	{"All Metadata":Cyber Attack) AND ("All Metadata":Intrusion Detection) OR ("All Metadata":Anomaly) OR ("All Metadata":Signature) AND ("All Metadata":Deep Learning) OR ("All Metadata":Machine Learning)}
ACM Digital	{[All: cyber attack] AND [[All: intrusion detection] OR [All: anomaly] OR [All: signature]] AND [[All: deep learning] OR [All: machine learning]]}
ScienceDirect	{Cyber Attack and Intrusion Detection or Anomaly or Signature and Deep Learning or Machine Learning}
SpringerLing	{Cyber Attack and Intrusion Detection or Anomaly or Signature and Deep Learning or Machine Learning}

Fonte: Elaborado pelos autores.

Tabela 3 – Critérios de Inclusão e Exclusão

Inclusão	Exclusão
Abordar Algoritmos de IA	Trabalhos duplicados
Informar o banco de dados	Fora do contexto
Artigos em inglês ou português	Banco de dados indisponível
Acurácia maior que 70%	Sem análise comparativa
-	Artigos fora do período de 2017 a 2023

Fonte: Elaborado pelos autores.

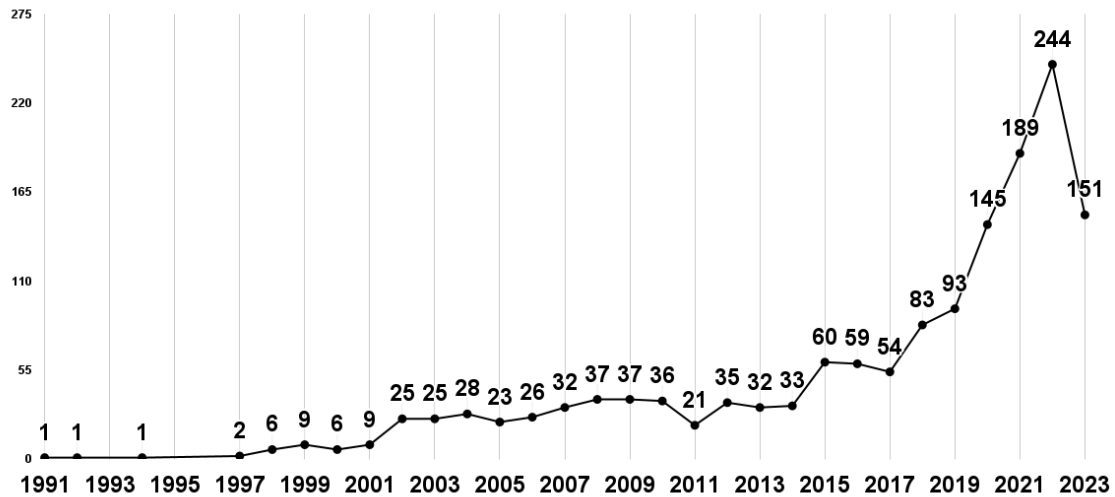
e mostre o banco de dados de ataques utilizado nos experimentos. O idioma escolhido é pensando na maioria dos artigos encontrados e na língua materna dos autores. Por fim, como o foco é detecção de ataques, prioriza-se também a acurácia encontrada nos resultados dos trabalhos, precisando ser acima de 70%, visto que a maioria dos algoritmos apresentam acurácia superior, mas há trabalhos que comparam métodos que melhoram a precisão.

Os critérios de exclusão escolhidos são usados para garantir que não permaneçam trabalhos duplicados após a importação das referências (essa varredura foi realizada com o auxílio do software Rayyan²), trabalhos fora do contexto de detecção de ataques, além de trabalhos que usam bancos de dados de ataques com acesso indisponível e sem análise comparativa. Um dos critérios essenciais dentro do mapeamento foi identificar o crescimento de estudos relacionados à detecção de ataques a partir de 2017, como observado na Figura 1, há um crescimento acelerado de pesquisas nesta área durante o COVID-19, por conta do aumento de ataques cibernéticos. Com isso, houve um corte temporal entre 2017 e 2023

² <<https://www.rayyan.ai/>>

que é onde está o foco deste MS.

Figura 1 – Gráfico Temporal da Quantidade de Pesquisas por Ano



Fonte: Elaborado pelos autores.

2.2 Condução

Esta etapa engloba cada detalhe discutido no planejamento, ou seja, os artigos são pesquisados, analisados e selecionados para o MS. Seguindo a coluna de palavras-chaves da Tabela 1 nos repositórios citados anteriormente, obteve-se 5.346 artigos. Por meio da filtragem de duplicidade e fora do contexto de detecção de ataques, chegou-se a 1.507 artigos.

Com base nessa filtragem, aplica-se o corte temporal, identificado nos critérios de exclusão, chegando-se a 534 artigos. Mesmo após o corte temporal, ainda se percebe uma quantidade inviável para analisar criteriosamente cada artigo. Com isso, usando todos os critérios de inclusão, foram selecionados 23 artigos.

2.3 Resultados

Tendo em vista os 23 artigos selecionados na Seção 2.2, a Tabela 4 traz os respectivos autores desses trabalhos, separados em IDs para facilitar a identificação dos estudos com os resultados obtidos. Além disso, foram considerados dois aspectos no resultado, o primeiro foca nos algoritmos de IA e o segundo nos bancos de dados, sendo possível visualizar os algoritmos e bancos de dados usados com mais frequência na literatura.

Tabela 4 – Trabalhos Seleccionados.

ID	Referência	ID	Referência
1	(KILINCER; ERTAM; SENGUR, 2021)	13	(KARTHIKA; MAHESWARI, 2022)
2	(DHANYA et al., 2023)	14	(HNAME; HUSSAIN, 2023)
3	(KIM et al., 2017)	15	(VISHWAKARMA; KESSWANI, 2023)
4	(SIDDIQI; PAK, 2022)	16	(AMINANTO et al., 2022)
5	(WU et al., 2022)	17	(SAYED et al., 2022)
6	(JAIN; PAWAR; MUTHALAGU, 2022)	18	(CERDA; YUAN; CHEN, 2021)
7	(SOUSA; SILVA, 2022)	19	(BENTES; FIGUEIREDO; CAMPOS, 2021)
8	(SARHAN et al., 2022)	20	(ALMSEIDIN et al., 2017)
9	(MUSHTAQ; ZAMEER; KHAN, 2022)	21	(LUCAS et al., 2021)
10	(GABER; EL-GHAMRY; HASSANIEN, 2022)	22	(KANIMOZHI; JACOB, 2021)
11	(PATGIRI et al., 2018)	23	(LEEYV et al., 2021)
12	(ATTOU et al., 2023)		

Fonte: Elaborado pelos autores.

2.3.1 Algoritmos de IA

Dentro dos estudos que abordam detecção de ataques, um tópico presente é a utilização de algoritmos de IA para auxiliar na detecção. Como pode ser visto, na Figura 2, é apresentado um conjunto de algoritmos que estão presentes nos estudos seleccionados, totalizando 36 algoritmos, assim como a quantidade de algoritmos por pesquisa e a frequência em que esses algoritmos aparecem na literatura.

Figura 2 – Visão Geral entre Autores e Algoritmos

ID	ADABOOST	AE	BILSTM	BN	BOOSTING	CATBOOST	CNN	DBN	DCNNBILSTM	DOSNET	DEEP MLP	DL	DNN	DT	ELLIPTIC	EXTRA TREE	FNN	GB	J48	KNN	LDA	LIGHTGBM	LR	LSTM	MLP	NB	NC	NLPCA	QDA	RF	RNN	RT	RTIDS	SE-IDS	SVM	XGBOOST	Total geral
1																																					3
2																																					8
3																																					1
4																																					1
5																																					5
6																																					4
7																																					9
8																																					1
9																																					9
10																																					3
11																																					2
12																																					5
13																																					2
14																																					5
15																																					10
16																																					4
17																																					6
18																																					1
19																																					2
20																																					7
21																																					4
22																																					6
23																																					7
Total geral	5	1	1	1	1	1	3	1	1	1	1	1	4	11	1	2	1	1	2	6	1	1	4	4	6	8	1	1	1	11	1	2	1	1	13	3	105

Fonte: Elaborado pelos autores.

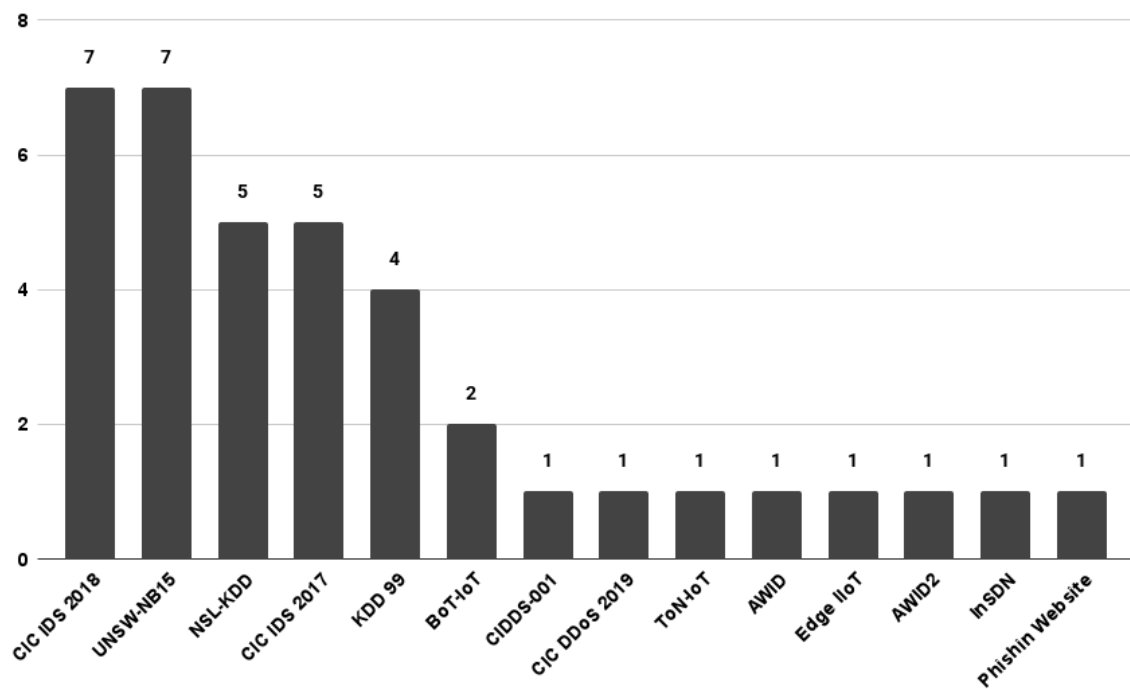
De imediato, pode-se destacar como os algoritmos de detecção mais comuns, SVM, DT e RF, tendo em vista, em que são apresentados 13, 11 e 11 vezes, respectivamente. Pode-se ainda perceber que esses algoritmos, normalmente, são apresentados em conjunto, indicando que são algoritmos padrões de comparação.

Deve-se levar em consideração também, a quantidade de algoritmos presentes em cada trabalho, representado pela última coluna da Figura 2. Normalmente, os estudos levam em consideração mais de um algoritmo para gerar comparação dos resultados obtidos. Porém, os trabalhos que apresentaram apenas um algoritmo não devem ser descartados, pois a mudança de parâmetros pode servir como um critério de comparação. A pesquisa que apresentou o maior número de algoritmos foi a de ID 15 (VISHWAKARMA; KESSWANI, 2023), com um total de 10 algoritmos.

2.3.2 Bancos de dados

Assim como os algoritmos são importantes para detectar ataques, o banco utilizado em cada estudo também, porque dependendo do banco de dados de ataques usado, pode-se retirar diferentes informações de estudo. Com isso, na Figura 3, temos uma visão de todos os bancos presentes dos estudos selecionados. Ao todo, foram encontrados 14 bancos, dentre os quais podemos destacar o CIC IDS 2018, UNSW-NB15, NSL-KDD e CIC IDS 2017, sendo estes os mais comuns na literatura.

Figura 3 – Visão Geral entre Banco de Dados e Número de Citações



Fonte: Elaborado pelos autores.

Foi realizada uma análise individual de cada banco de dados. A Tabela 5 evidencia a diversidade de ataques presentes nesses conjuntos de dados. Ao avaliar a quantidade de amostras, observa-se um volume significativo. No entanto, ao examinar a distribuição das

classes nos bancos de dados, nota-se que a maioria das amostras pertence à classe normal, em vez de classes de ataque. Em estudos que utilizam classificadores, a presença reduzida de amostras de ataques pode levar a problemas como underfitting ou viés para a classe dominante, comprometendo a eficácia da detecção de ameaças.

Tabela 5 – Ataques Presentes nos Bancos de Dados e a Representação Percentual dentro da Amostra Total.

Banco de dados	Ataques (%)	Total de Amostras
CIC IDS 2018	Benign (86%), Brute Force (2%), DoS (3,2%), DDoS (6,2%), Injection (0,0004%), Infiltration (0,8%), Botnet (1,4%)	20.253.943
UNSW-NB15	Benign (87%), Exploits (1,7%), DoS (0,64%), ShellCode (0,07%), Reconnaissance (0,55%), Fuzzers (0,95%), Generic (8,5%), Analysis (0,11%), Backdoors (0,09%), Worms (0,01%)	2.540.218
NSL-KDD	Benign (51,8%), DoS (36%), Probe (9,5%), Privilege Escalation (0,13%), Remote Access (2,6%)	148.597
CIC IDS 2017	Benign (79,5%), DDoS (4,7%), PortScan (5,8%), Botnet (0,07%), Infiltration (0,0013%), BruteForce (0,06%), XSS (0,02%), Injection (0,0008%), Patator (0,5%), DoS (9,3%), Heartbleed (0,0004%)	2.715.743
KDD 99	Benign (19,7%), DoS (79%), Probe (0,83%), Privilege Escalation (0,011%), Remote Access (0,23%)	494.020
BoT-IoT	Benign (0,013%), Reconnaissance (2,5%), DoS (44,3%), Theft (0,002%), DDoS (53,2%)	72.470.463
CIDDS-001	Benign (88,2%), Suspicious (1,4%), Unknown (0,24%), Attacker (5,2%), Victim (5%)	31.959.174
CIC DDoS 2019	Benign (0,2%), PortMap (0,4%), NetBIOS (5,4%), LDAP (4%), MSSQL (11,7%), UDP (8,6%), SYN (13,1%), DrDoS (56,7%), DDoS (0,001%), TFTP (-)	49.319.810
ToN-IoT	Benign (28,5%), Backdoor (2,4%), Password (5,2%), DDoS (20,4%), Injection (2%), Ransomware (0,28%), XSS (6,8%), Scanning (20%), DoS (14,6%), MITM (0,004%)	31.502.397
AWID	Benign (91,2%), Impersonation (2,9%), Injection (3,4%), Flooding (2,4%)	1.795.575
EDGE IIoT	Benign (53,5%), Backdoor (0,12%), DDoS (40%), MITM (0,006%), Fingerprinting (0,005%), Password (5%), Port Scanning (0,11%), Ransomware (0,05%), Injection (0,24%), Uploading (0,18%), Vulnerability Scanner (0,7%), XSS (0,08%)	20.939.622
AWID 2	Benign (89%), Impersonation (3,6%), Injection (4,3%), Flooding (3%)	1.893.518
InSDN	Benign (20%), DDoS (35,5%), Probe (28,5%), DoS (15,6%), BFA (0,41%), U2R (0,005%), Web Attack (0,06%), Botnet (0,05%)	343.889
PHISHING	Benign (55,7%), Phishing (44,3%)	11.055

Fonte: Elaborado pelos autores.

De forma geral, consegue-se ver que os ataques de *Denial of Service* (DoS) e *Distributed Denial of Service* (DDoS) são os ataques que aparecem com mais frequência dentro dos bancos de dados. Uma vantagem de vários bancos terem ataques iguais é a possibilidade de relacionar um ao outro, aumentando até mesmo a quantidade de dados.

Assim como há ataques semelhantes entre si, há também aqueles que estão presentes em apenas um banco, tais como *Worms* e *Ransomware*. Então é evidente que nenhum banco de dados sozinho abrange todos os ataques de uma rede. Além disso, há pouca atualização dos bancos de dados, o que promove a falta de dados de ataques mais sofisticados. Paralelamente, com a evolução da tecnologia, os ataques estão sendo mais silenciosos e mais difíceis de serem detectados.

No decorrer dos anos, com a tecnologia evoluindo, os ataques ficaram mais sofisticados e, com isso, houve a necessidade de novas técnicas de detecção de ataques. Portanto, o mapeamento sistemático apresentado tem como principal objetivo identificar os algoritmos de IA e os bancos de dados presentes na literatura. Como vimos, os algoritmos mais comuns são o SVM, DT e RF, levando em consideração a quantidade de vezes que foram citados dentre os trabalhos selecionados e os bancos de dados de ataques são CIC IDS 2018, UNSW-NB15, NSL-KDD E CIC IDS 2017.

Levando isso em consideração, uma forma possível de melhorar o sistema de detecção de ataques é fazendo uma comparação entre algoritmos para aproveitar as melhores características de cada algoritmo. No entanto, para implementar um algoritmo bem-sucedido na arquitetura de detecção, precisa-se de bancos de dados atualizados e com dados suficientes dos ataques estudados. Como observado nos estudos, os bancos de dados que são mais usados na literatura estão desatualizados e com desbalanceamento de classes.

Por meio desse MS, além de identificar a possibilidade de criar arquiteturas com mais de um algoritmo e melhorias nos bancos de dados como ataques atuais e dados proporcionais entre si, também foi possível identificar que o foco dos trabalhos são a detecção de um tráfego normal ou malicioso. Ou seja, isso torna possível detectar qual a técnica e tática daquele dado malicioso, permitindo a mitigação mais eficaz e rápida.

O MS possibilitou identificar os algoritmos e bancos de dados de ataques mais comuns na literatura. Contudo, não destaca soluções Gerenciamento de Informações e Eventos de Segurança (SIEM), como Prelude, Open Source Security Information Management (OSSIM) e *Elasticsearch, Logstash and Kibana Stack* (Pilha ELK). O estudo dos autores Christopher e Myers (2018) compara as três plataformas de código aberto. O Prelude oferece uma integração modular que permite a conexão de ferramentas de segurança para coleta e análise de dados de eventos, com foco principal na correlação de eventos provenientes de múltiplas fontes de dados, além de permitir a personalização de configuração e gerenciamento.

O OSSIM oferece um conjunto completo de recursos para coleta, normalização, correlação e análise de eventos de segurança. Projetado para integrar ferramentas de segurança como IDS, *firewalls* e *scanners* de vulnerabilidades, ele proporciona uma visão abrangente da segurança da rede. No entanto, apesar de sua ampla gama de funcionalidades, sua configuração e utilização são dificultadas pela complexidade da interface e das ferramentas.

A Pilha ELK é amplamente utilizada para a coleta e análise de logs de segurança, sendo frequentemente empregada no monitoramento e na visualização de dados de segurança. Ela oferece alta flexibilidade e escalabilidade, sendo ideal para grandes volumes de dados em tempo real. Embora tenha menos funcionalidades do que o OSSIM, sua grande base de usuários e comunidade ativa representam uma vantagem, facilitando o

acesso a suporte, recursos adicionais e configuração simplificada. Assim, observou-se que tanto o Prelude quanto o OSSIM têm como foco o gerenciamento de eventos de segurança, enquanto a Pilha ELK se concentra na análise de logs.

3 Contexto Tecnológico

Neste capítulo são detalhadas as tecnologias necessárias para o entendimento sobre detecção de técnicas de ataques proposta por esse trabalho. Desta forma, objetiva-se facilitar a compreensão de como as tecnologias estão interligadas.

3.1 Sistema de Detecção de Intrusão

O Sistema de Detecção de Intrusão (IDS) é aplicado em dispositivos capazes de detectar e alertar o usuário da rede sobre uma potencial violação de segurança na rede (ORIYANIO, 2017). Esse tipo de sistema busca identificar ataques por meio do tráfego da rede ou dos *logs* do sistema monitorado, gerando um alerta para possíveis intervenções automáticas ou manuais (TURCATO, 2020).

Especificamente, em redes de comunicação existem *Network Intrusion Detection System* (NIDS) e *Host Intrusion Detection System* (HIDS). O primeiro está relacionado ao monitoramento do tráfego de rede, e o segundo, está relacionado ao monitoramento de eventos locais em dispositivos específicos (ORIYANIO, 2017). Apesar do tipo do monitoramento, ambos têm como objetivo detectar atividades maliciosas.

No NIDS, há dois tipos de detecção: assinatura ou anomalia. Na baseada em assinatura, o sistema analisa o tráfego de rede, compara os pacotes recebidos com um banco de dados existente que contém os tipos de ataques conhecidos (ORIYANIO, 2017). Em seguida, se a assinatura correspondente for encontrada, o sistema identifica como possível ameaça e gera um alerta que deve ser verificado pelo profissionais de segurança. Ou seja, esse processo requer um conhecimento prévio dos ataques e normalmente são menos eficientes para identificar ataques com técnicas ainda desconhecidas.

Já no caso da detecção baseada em anomalia, o sistema analisa o tráfego por um determinado período, identificando o comportamento da rede. Assim, quando há um tráfego malicioso é observada uma anormalidade dos dados e um alerta é gerado (ORIYANIO, 2017). Nesse caso, procura-se detectar alterações no padrão do tráfego em relação ao comportamento normal, podendo gerar um número excessivo de falsos positivos.

Entre os métodos de detecção de anomalias, destacam-se aqueles baseados em análise estatística, mineração de dados, análise de sinais e, por último, em aprendizagem de máquina (TURCATO, 2020).

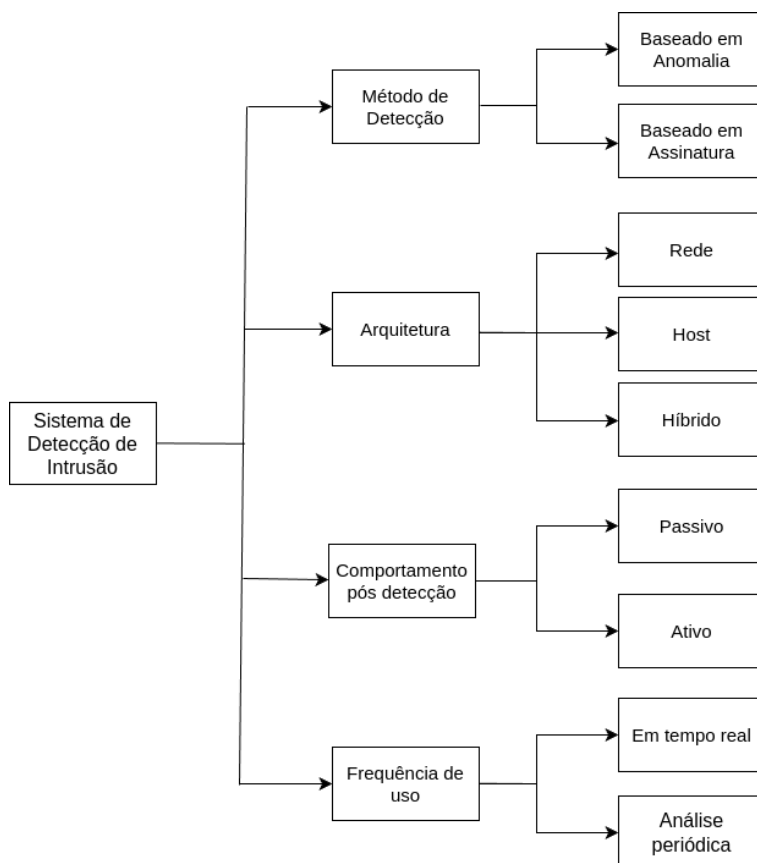
Além disso, o NIDS implementa três etapas: coleta, análise e resposta. A coleta representa a extração de dados do sistema monitorado. A análise visa identificar tentativas de ataques com base nos dados coletados. Por fim, a resposta executa ações após a

identificação de intrusão, como alertas e/ou bloqueios (TURCATO, 2020).

Na Figura 4 há um diagrama das categorias de classificação do IDS, sendo este trabalho composto por uma arquitetura de rede baseada em anomalia. A proposta é baseada na validação do monitoramento de um dispositivo específico, com potencial para ser expandido ao monitoramento do tráfego de rede.

Nesse sentido, a arquitetura visa identificar anormalidades nos dados a partir do monitoramento contínuo de dispositivos. Adicionalmente, os dados são monitorados em tempo real e operando de maneira passiva ao gerar alertas sobre possíveis invasões. No entanto, não inclui mecanismos automáticos de reação a invasões.

Figura 4 – Categorias de Classificação do IDS.



Fonte: Adaptado de (TURCATO, 2020)

3.2 Soluções SIEM

As soluções de Gerenciamento de Informações e Eventos de Segurança (SIEM) tem como objetivo reunir informações da rede e *logs* de eventos de diferentes dispositivos, possibilitando a detecção de anomalias na rede, principalmente por meio de configurações de alertas (FATEMI; GHORBANI, 2020).

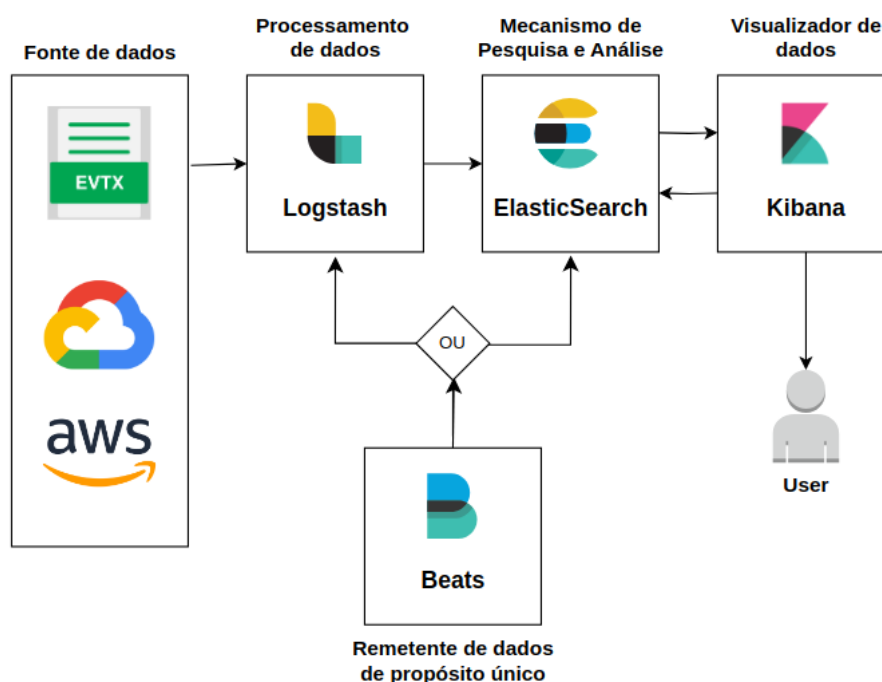
SIEM é uma arquitetura de segurança que fornece suporte às organizações, facili-

tando o reconhecimento e abordagem de possíveis ameaças e vulnerabilidades de segurança antes que a operação seja interrompida. Além disso, tem suporte para detectar anomalias e usar Inteligência Artificial (IA) para automatizar a detecção de possíveis variações anormais na rede (IBM, 2023).

Os *logs* são a principal fonte de informação analisada pelo SIEM, por serem dados brutos, havendo uma necessidade de análise para encontrar conexões significativas que forneçam dados que cooperem para identificar ameaças e vulnerabilidades cibernéticas (CHRISTOPHER; MYERS, 2018). Como há diferentes sistemas operacionais, os *logs* apresentam variações de formato de acordo com cada sistema operacional.

Uma das soluções SIEM utilizadas é a Pilha ELK. Esta arquitetura é composta por um conjunto de quatro produtos: *Elasticsearch*, *Logstash*, Kibana e *Beats*. Juntos apresentam uma plataforma capaz de coletar *logs* de fontes diferentes, transformar os dados significativos, armazenar os dados para monitoramento, análise de casos de segurança e tendências, além de criar visualizações de painéis para consultas (HOROVITS, 2018). É possível visualizar a arquitetura Pilha ELK na Figura 5.

Figura 5 – Arquitetura Pilha ELK.



Fonte: Adaptado de (TERRASTRUCT, 2023).

- *Elasticsearch*: é o mecanismo principal da Pilha ELK em que armazena os dados centralizados, possibilitando a pesquisa e análise de texto baseado no código aberto Apache Lucene, bem como análise em tempo real dos dados.
- Kibana: proporciona recursos de visualização para o usuário ser capaz de analisar os dados, criar *dashboard* e gráficos interativos, facilitando o monitoramento e análise dos dados por meio de visualizações.

- *Logstash*: é responsável por coletar os dados, executar transformações e melhorar a formatação antes de enviar ao *Elasticsearch*.
- *Beats*: é um agente opcional projetado para coletar e enviar dados de diferentes fontes para o *ElasticSearch* ou *Logstash*, há um tipo de *Beat* para cada tipo de dado ou fonte específico, como por exemplo, *Filebeat* e *Winlogbeat*.

3.3 Framework Mitre Attack

O MITRE ATT&CK é um *framework* usado como base de conhecimento no qual são apresentadas as táticas e técnicas observadas no mundo real. Normalmente, essa base é usada para desenvolver modelos e metodologias de ameaças focadas na segurança cibernética. São apresentadas 14 táticas de ataques cibernéticos, sendo elas:

Reconhecimento: envolve técnicas de coleta ativa ou passiva de informações que podem ser usadas para escolher o alvo e planejar futuras operações de ataque, essa etapa possui 10 técnicas.

Desenvolvimento de Recursos: envolve técnicas que tentam estabelecer recursos, podendo ser criadas, compradas ou roubadas sendo usadas para dar segmento ao ataque, essa etapa possui 8 técnicas.

Acesso Inicial: envolve técnicas usando vários vetores de entradas para conseguir entrar na rede, essa etapa possui 10 técnicas.

Execução: envolve técnicas que resultam na execução de códigos maliciosos de forma controlada pelo *cyber attack* em um sistema local ou remoto, essa etapa possui 14 técnicas.

Persistência: envolve técnicas que permite que o *cyber attack* mantenha o acesso ao sistema mesmo após técnicas iniciais de impedir um ataque como reinicialização ou alteração de credencial, essa etapa possui 20 técnicas.

Escalação de Privilégios: envolve técnicas para tentar obter permissões de nível superior em um sistema ou rede, para conseguir concretizar os objetivos de ataque, essa etapa possui 14 técnicas.

Evasão de Defesa: envolve técnicas que tentam evitar a detecção durante o comprometimento, normalmente tentam desativar *software* de segurança ou criptografar dados e *scripts*, essa etapa possui 43 técnicas.

Acesso a Credenciais: envolve técnicas que roubam credenciais como nome de usuários e senhas, com isso, dificultam a detecção e permite criar credenciais para se manter na rede, essa etapa possui 17 técnicas.

Descoberta: envolve técnicas para obter conhecimento sobre o sistema e a rede interna, ajudando a observar o ambiente e planejar como agir dentro do sistema, essa etapa possui 32 técnicas.

Movimento Lateral: envolve técnicas que permitem o atacante a se mover pelo ambiente, conseguindo entrar e controlar remotamente o sistema na rede, com isso, conseguindo explorar a rede para encontrar o alvo de ataque e obter acesso posteriormente, essa etapa possui 9 técnicas.

Coleção: envolve técnicas que servem para coletar informações relevantes que podem ser usadas para o objetivo final, normalmente, após a coleta há o roubo desses dados, essa etapa possui 17 técnicas.

Comando e Controle: envolve técnicas que podem ser usadas para se comunicar com o sistema sob o controle do atacante dentro da rede da vítima, normalmente, tentando simular o tráfego normal da rede para não serem detectados, essa etapa possui 17 técnicas.

Exfiltração: envolve técnicas que servem para roubar dados da rede vítima, após a coleta dos dados, são empacotados para evitar detecção enquanto são excluídos da rede original, essa etapa possui 9 técnicas.

Impacto: envolve técnicas que interrompem a disponibilidade ou comprometem a integridade da rede, esses impactos podem causar a destruição ou alteração dos dados, geralmente, essa etapa é o objetivo final do atacante, essa etapa possui 14 técnicas.

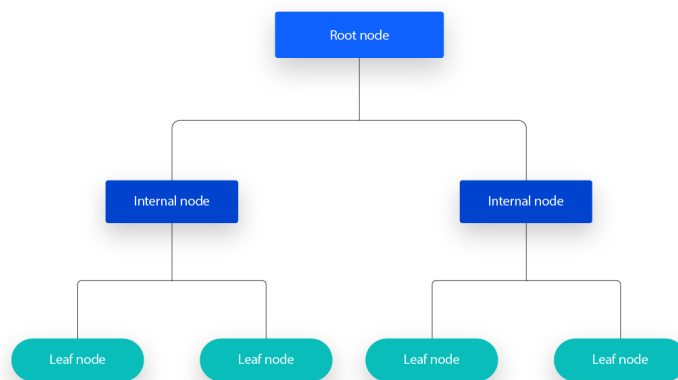
3.4 Inteligência Artificial

Como abordado no Capítulo 2, estudar estratégias de detecção de ataques usando aprendizagem de máquina, principalmente focadas em abordagens supervisionadas. Ou seja, estas abordagens têm o objetivo de classificar os dados como comportamentos normais e anormais. De acordo com o MS, em média, quatro algoritmos são necessários para gerar comparações: *Decision Tree* (DT), *Random Forest* (RF), NB e *Support Vector Machine* (SVM).

O *Decision Tree* (DT) tem uma estrutura semelhante a uma árvore, sendo composta por uma raiz, vários ramos, folhas e nós. A raiz é o início do processo, os ramos são as conexões que se iniciam na raiz até a folha e os nós são as características usadas pelo sistema. Desta forma, cada nó armazena valores de probabilidade de cada classe. Por fim, as folhas são as saídas (MUSHTAQ; ZAMEER; KHAN, 2022). Normalmente, uma amostra desconhecida entra no modelo e percorre os nós, no qual são avaliados os atributos e probabilidades, até que atinja 100% em uma classe. Após análise do caminho percorrido,

é possível encontrar o motivo do resultado. A Figura 6 é apresentada a árvore de decisão generalizada, composta por uma raiz (azul claro) que divide-se em ramos (azul escuro), este por sua vez, divide-se nas folhas (verde claro). Analisando por uma perspectiva do problema de detecção de ataques, as folhas representam as classes ou tipos de ataques, quando uma amostra é inserida, o classificador analisa cada etapa da árvore até finalizar em uma classe.

Figura 6 – Ilustração da Árvore de Decisão



O *Random Forest* (RF) é a combinação das saídas de múltiplas árvores de decisões, sendo que cada árvore do conjunto é composta por amostras do conjunto de treinamento com reposição, ou *bootstrap*. Nesta abordagem, um terço das amostras de treinamento é retirado para amostras de testes (*out-of-bag*) (MUSHTAQ; ZAMEER; KHAN, 2022). Em seguida, uma instância de aleatoriedade é injetada por meio de *feature bagging*, causando diversidade ao conjunto de dados e escolhida a classe que mais apareceu nos resultados. Por fim, as amostras *out-of-bag* são usadas para validação para confirmar o resultado previsto. A Figura 7 é uma ilustração do funcionamento do RF, composta por mais de uma árvore de decisão, em que uma amostra é analisada por cada árvore. Cada resultado é somado e um resultado médio é alcançado, assim temos uma classificação final da amostra.

O *Support Vector Machine* (SVM) é baseado em encontrar um hiperplano capaz de maximizar a distância entre os pontos de cada classe (TAHER; JISAN; RAHMAN, 2019). A distância é conhecida como margem de separação e quanto maior for a margem, melhor será a probabilidade de as classes serem classificadas corretamente. Normalmente, é usado para classificar dados que não são lineares, ou seja, não é possível separar as classes com uma linha reta. Existem três tipos de SVM com kernel: linear, polinomial e gaussiano (RBF), conforme apresentado na Figura 8. A escolha do kernel influencia a forma como as classes são separadas. No caso do kernel linear, as classes podem ser separadas por uma linha ou plano. No kernel polinomial, a separação é curvada e depende do grau do polinômio. Já no kernel gaussiano (RBF), a separação é altamente flexível, sem um formato específico.

Figura 7 – Ilustração da Floresta Aleatória

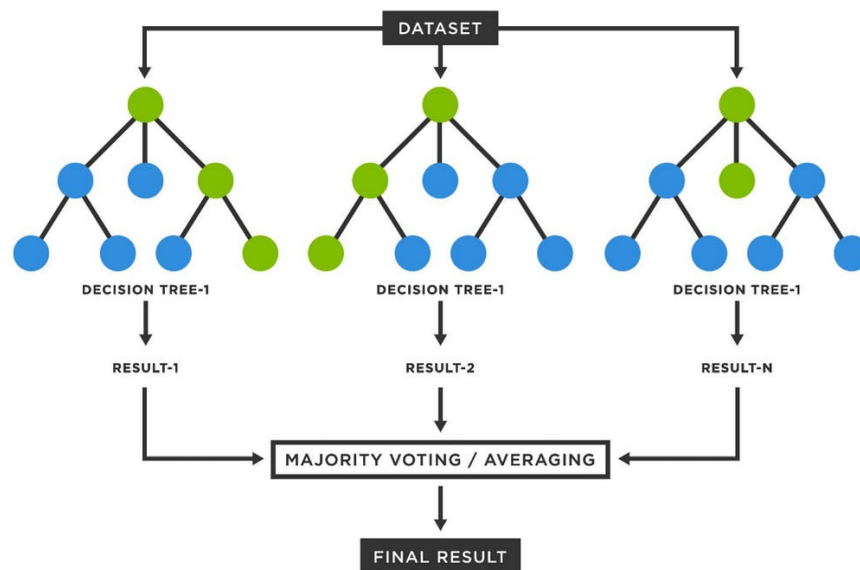
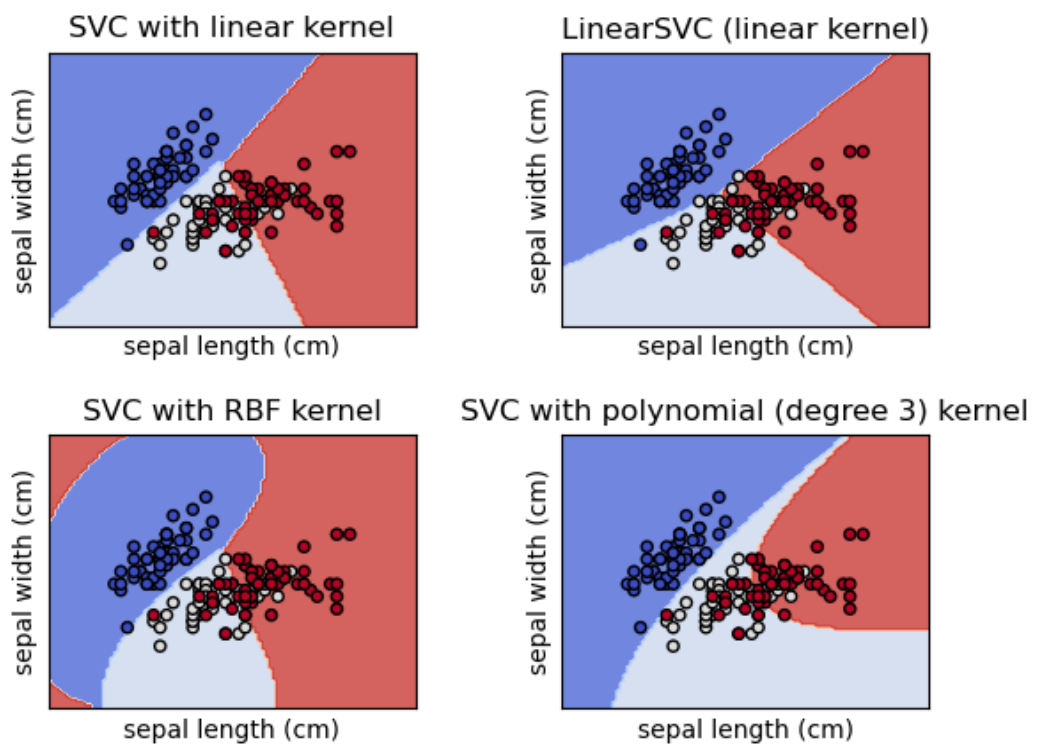


Figura 8 – Ilustração do SVM com Kernel



O *Naive Bayes* (NB) é um classificador probabilístico baseado no Teorema de Bayes, utiliza-se de probabilidade condicional. Estudos destacam simplicidade e rapidez no desempenho, sendo usado frequentemente em modelos de reconhecimento de padrões e

classificação de texto, além disso, em implementações de Processamento de Linguagem Natural (PLN) (PAJILA et al., 2023). Sendo o Teorema de Bayes definida matematicamente como:

$$P(C|X) = \frac{P(X|C) * P(C)}{P(X)}$$

Sendo $P(C|X)$, a probabilidade que a classe C aconteça dado a ocorrência do atributo X (posteriori), $P(X|C)$ é a probabilidade do atributo X ser observado dado que a classe C ocorreu (verossimilhança), $P(C)$ é a probabilidade de a classe C acontecer (priori) e $P(X)$ é a probabilidade de ocorrência do atributo X (priori).

Além dos algoritmos de ML, este trabalho utilizou-se de um pré-processamento que tem como objetivo investigar e propor métodos ou sistemas de processamento computacional da linguagem humana (CASELI; NUNES, 2023), conhecida como Processamento de Linguagem Natural (PLN). Com essa técnica, buscou-se transformar os dados do tipo texto em numéricos para que fosse possível usar os dados nos algoritmos de ML.

O PLN apresenta métodos como Codificação Binária, *Counting* e Frequência do Termo e Frequência Inversa do Documento (TF-IDF) para a extração de recursos de texto. No caso deste trabalho, usou-se o TF-IDF que se utiliza tanto do método *Counting* para verificar a existência da palavra e a frequência em que ocorre quanto utiliza-se da frequência inversa para reduzir os pesos das palavras que ocorrem com mais frequência e aumentar os pesos das palavras raras por meio da fórmula:

$$TFIDF(t, d, D) = TF(t, d) * IDF(t, D)$$

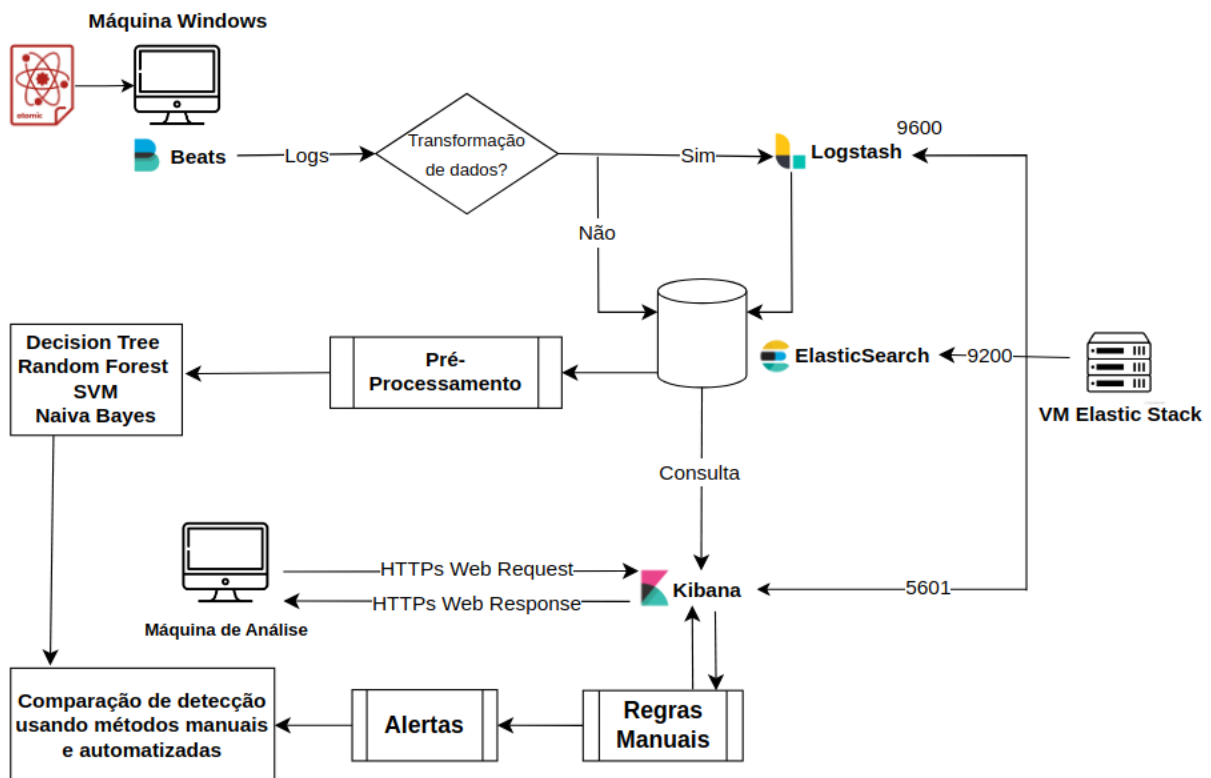
Sendo t , o termo, d é um documento onde está o termo e D é a coleção de todos os documentos.

4 Proposta da Dissertação

Este capítulo tem como objetivo estruturar a contribuição principal para a análise e comparação dos métodos manuais e automatizados apresentando o fluxograma do desenvolvimento dos cenários propostos, a criação do banco de dados com técnicas de ataque, de acordo com o *Mitre Att&ck*, e o procedimentos realizados no Kibana. Para realizar a detecção, é utilizada como arquitetura a Pilha ELK. Normalmente, ele é utilizado para receber dados, visualizar, monitorar e analisar todo o tráfego da rede.

O fluxograma proposto é ilustrado na Figura 9, a qual representa o fluxo das máquinas e a forma como se comunicam. Como pode-se observar na figura, primeiramente, são hospedados os eventos *logs* do cliente *Windows*. Em seguida, é possível analisar e visualizar os dados recebidos. Todo o monitoramento é feito por meio da Pilha ELK, juntamente com o conjunto de tecnologias presentes nessa plataforma, como o *Beats*, *Logstash*, *Elasticsearch* e Kibana. Com isso, são necessárias duas máquinas virtuais e a máquina de análise.

Figura 9 – Fluxograma Proposto



Fonte: Elaborado pelos autores.

4.1 Visão Geral

Visualizando as conexões das máquinas na Figura 9, faz-se necessário compreender os papéis desempenhados por cada máquina. Assim, a máquina *Windows 10* é o cliente de todo o experimento, sendo os *logs* gerados por essa máquina, analisados. Além disso, é o sistema que será afetado pelos ataques. Nessa máquina, são executados os comandos no *PowerShell* para simular ataques na rede.

A máquina *Elastic Stack* hospeda os serviços de *Beats*, *Logstash*, *Elasticsearch* e Kibana, que são os responsáveis pela plataforma de recebimento dos dados, análise, visualização e monitoramento da rede. Para que a máquina de análise tenha acesso aos dados coletados as portas configuradas são: 9200 (*Elasticsearch*) e 5601 (Kibana). A máquina de análise hospeda todas as máquinas virtuais e é responsável por realizar as requisições *web* para acessar o Kibana para fins de estudo da rede, monitoramento, análise manual e alertas de ataques.

Para melhor compreensão do fluxo proposto, o processo inicia-se na máquina *Windows*. Onde as técnicas de ataques do *Atomic Red Team* são executadas. Em seguida, o agente elástico instalado é responsável por enviar os *logs* gerados pela técnica de ataque implementada para a Pilha ELK, por meio do *winlogbeat*. De antemão, é necessário saber o tipo de dado que se pretende coletar. No caso desta proposta, são os *logs* referentes à aplicação, sistema, segurança, operacional e *PowerShell*.

Em seguida, é preciso verificar se há necessidade de transformar dos dados. Em caso positivo, os *logs* são reescritos em uma estrutura conhecida do sistema pelo *Logstash* e enviado ao *Elasticsearch*, mas em caso negativo, os *logs* são enviado diretamente ao *Elasticsearch*, que processa, indexa e armazena os dados para serem recuperados quando necessário.

Logo, com todos os dados coletados e armazenados no sistema, é preciso visualizar os dados por meio de requisições *web*, feitas pelo agente na máquina de análise. O Kibana é responsável por gerar uma solicitação de consulta ao armazenamento do *Elasticsearch*, retornando os dados solicitados ao agente. Assim, configura-se toda uma solução de Gerenciamento de Informações e Eventos de Segurança (SIEM).

4.2 Atomic Red Team

As técnicas de ataques são estruturados por uma biblioteca conhecida como *Atomic Red Team*¹ que possui um módulo no *PowerShell* chamado *Invoke-AtomicTest*. Nesta biblioteca, temos acesso a diferentes ataques que relacionam-se com as técnicas do *Mitre*

¹ <<https://github.com/redcanaryco/invoke-atomicredteam/wiki>>

*ATT&CK*². Oferecendo um método padronizado para gerar dados sintéticos de ataques cibernéticos.

Dentro do módulo, é possível encontrar diferentes técnicas de ataques por meio do ID, além de uma breve descrição e parâmetros necessários para executar o ataque na rede. No relatório do Red Report (2023), as principais técnicas de ataques que ocorreram no ano de 2023, com isso, 9 técnicas foram selecionadas como pode ser visto na Tabela 6. Além disso, cada ataque possui uma característica própria e ferramentas de ataque.

Tabela 6 – Ataques Executados

Tática	Técnica	ID
Defense Evasion	Obfuscated Files or Information	T1027
Defense Evasion	Rundll32	T1218.011
Defense Evasion	Rename System Utilities	T1036.003
Execution	PowerShell	T1059.001
Execution	Windows Command Shell	T1059.003
Execution	Windows Management Instrumentation	T1047
Defense Evasion, Persistence, Privilege Escalation, Initial Access	Local Accounts	T1078.003
Command and Control	Ingress Tool Transfer	T1105
Credential Access	OS Credential Dumping	T1003

Fonte: Elaborado pelos autores.

T1027 : são arquivos ou informações ofuscadas que dificultam a detecção de arquivos executáveis com códigos maliciosos ou na análise de dados criptografados. Para o experimento, usou-se a técnica que ofusca comandos do *PowerShell*.

T1218.011 : é a execução de *proxy* binário assinado com *rundll32* fazendo com que evitem as ferramentas de segurança. Para o experimento, usou-se a técnica que executa um *script* remoto usando *rundll32.exe*, após executar é aberto o *notepad.exe*.

T1036.003 : mascara ou renomeia utilitários do sistema para parecer legítimo para não ser detectado. Para o experimento, usou-se a técnica que executa o *cscript.exe* disfarçando como uma instância de *notepad.exe*.

T1059.001 : é um interpretador de comando e *script* para *PowerShell*, usada para executar ações para descobrir informações e executar códigos maliciosos. Para o experimento, usou-se a técnica que baixa um Mimikatz, permitindo que usuários visualizem e salvem credenciais de autenticação.

² <<https://attack.mitre.org/>>

- T1059.003** : é um interpretador de comando e *script* para *Windows Command Shell*, usado para obter acesso ao serviço remoto (SSH) e possibilitando alteração de permissões de usuários. Para o experimento, usou-se a técnica que executa um *script* de lote simples, é um arquivo com extensão *.bat* que contém uma lista de comandos, resultando na abertura do CMD para executar o *script* e fechando logo em seguida.
- T1047** : é uma instrumentação de gerenciamento do *Windows* responsável pelos dados de gerenciamento e operações em sistemas *Windows*, pode ser usado por um cibercriminoso para interagir com os sistemas locais e remotos para coletar informações, executar comandos e carregar *payloads* maliciosos. Para o experimento, usou-se a técnica que lista todas as contas de usuários locais.
- T1078.003** : é relacionada a contas válidas locais que são configuradas para serem usadas por usuários, suporte remoto, serviços ou administração, quando o cibercriminoso possui essa conta podem elevar privilégios e coltar credenciais. Para o experimento, usou-se a técnica que permite criar uma conta local com privilégios de administrador.
- T1105** : transfere ferramentas de entrada ou arquivos de um sistema externo para um ambiente comprometido, podem usar também de instaladores e gerenciadores de pacotes. Para o experimento, usou-se a técnica que usa o *certutil*, ferramenta que possui informações de certificados, para baixar um arquivo da *web*.
- T1003** : é o despejo de credenciais do sistema operacional para obter *login* de contas e materiais de credenciais em forma de *hash* ou texto simples que podem estar armazenadas no cache, memória ou estrutura. Para o experimento, usou-se a técnica que despeja um arquivo executável *svchost*, agrupa e coordena serviços em um único processo, para coletar credenciais do Protocolo de *Desktop* Remoto (RDP).

Considerando a máquina vítima, o sistema *Windows* 10, o módulo *Invoke-AtomicTest* da biblioteca *Atomic Red Team*, foi executado nessa máquina. Para realizar uma técnica de ataque, é necessário compreender suas funcionalidades e os parâmetros requeridos. Assim, o processo é dividido em cinco etapas fundamentais: (i) identificar se a técnica de ataque está presente no módulo e listar os tipos de testes disponíveis; (ii) verificar se há dados de pré-requisito ausentes e realizar a instalação, caso necessário; (iii) especificar os argumentos de inicialização; (iv) executar o ataque de teste; e (v) realizar a limpeza após cada teste executado. Após a conclusão dessas etapas e o término do procedimento de execução, o sistema *Windows* encaminha os eventos *logs* à Pilha ELK.

4.3 Pré-Processamento

Para analisar os atributos presentes nos *logs*, 500 amostras compostas por atividades normais e maliciosas foram analisadas, constatando-se inicialmente a presença de 213

atributos. Utilizando a biblioteca *ydata_profiling* do *Python*, verificou-se que 39 atributos eram constantes, por isso foram eliminados, visto que não contribuem na diferenciação dos dados. Além disso, 144 atributos foram descartados devido à duplicidade, categorias inválidas ou presença majoritária de caracteres como “-”. Após esse processo de filtragem, restaram 30 atributos válidos, os quais são listados na Tabela 7.

Tabela 7 – Atributos do Evento Log

Atributo	Atributo	Atributo
event.action	powershell.provider.name	user.name
event.category	winlog.activity_id	winlog.channel
event.code	process.command_line	winlog.event_data.CountOfCredentialsReturned
event.module	process.executable	winlog.event_data.Description
event.outcome	process.name	winlog.event_data.IntegrityLevel
event.provider	process.parent.args	winlog.event_data.ParentUser
event.type	process.parent.name	winlog.event_data.ProcessCreationTime
file.extension	registry.data.type	winlog.event_data.TerminalSessionId
file.name	related.user	winlog.event_data.PrivilegeList
log.level	rule.name	winlog.task

Fonte: Elaborado pelos autores.

Com base nos atributos identificados, mais de 48 mil amostras foram geradas, dividindo-se entre dados normais e maliciosos. As 9 técnicas de ataque são representadas por rótulos, conforme descrito na Tabela 8, que também apresenta a quantidade de amostras associadas a cada rótulo no banco de dados.

Tabela 8 – Quantidade de Amostras por Ataques

ID	Quantidade de amostras	Rótulo	ID	Quantidade de amostras	Rótulo
Normal	22.881	0	T1059.003	2.409	5
T1027	4.886	1	T1047	3.365	6
T1218.001	1.021	2	T1078.003	2.007	7
T1036.003	2.300	3	T1105	2.187	8
T1059.001	4.680	4	T1003	2.530	9

Fonte: Elaborado pelos autores.

Após substituir os valores *None* por "0", iniciou-se a transformação dos dados categóricos em numéricos usando técnicas de PLN. Para cada atributo categórico, as mensagens foram convertidas em uma matriz de contagens de *tokens*, em que cada coluna correspondia a uma palavra.

Em seguida, são convertidos palavras em números por meio da Frequência do Termo e Frequência Inversa do Documento (TF-IDF). A Frequência do Termo (TF) calcula a

frequência de uma palavra em cada amostra, quanto maior o número de ocorrência, mais relevante é a palavra e a Frequência Inversa do Documento (IDF) calcula a raridade da palavra, quanto maior o número de ocorrência, menos é o peso recebido. Por fim, é feito o cálculo do produto do TF e IDF para cada palavra presente na amostra. Para finalizar a transformação dos dados, é feito a soma desses produtos em cada amostra e adicionada ao novo atributo numérico.

As amostras do banco de dados foram separados em 70% para treino e 30% para teste. Nas amostras de treino, realizou-se o balanceamento para evitar viés em favor das classes dominantes, ajustando-as para que cada classe contivesse 1.021 amostras, que corresponde a menor quantidade de amostras por classe. Após a separação, os dados foram padronizados pela técnica *Z-score Standardization*, que transforma os valores para que tenham média 0 e desvio padrão 1.

Todos os algoritmos foram avaliados utilizando validação cruzada. O parâmetro *cv*, responsável por definir a estratégia de divisão na validação cruzada, foi configurado para usar a validação estratificada *K-Fold*. Essa abordagem garante que as dobras geradas preservem a proporção de amostras de cada classe, mantendo a distribuição balanceada durante o processo de validação.

4.4 Métricas de Avaliação

Para analisar o desempenho do classificador, sete medidas conhecidas foram empregadas: Acurácia, Precisão, F1-Score, Recall, Desempenho do treinamento e teste, além da curva de aprendizagem de precisão e do erro para avaliação de testes diagnósticos, cujas definições são brevemente recapituladas abaixo.

Sendo Verdadeiro Positivo (VP) a representação do número de amostras de ataques classificadas como ataque, Falso Negativo (FN) a representação do número de amostras de ataques classificadas incorretamente, Falso Positivo (FP) a representação do número de amostras normais classificadas incorretamente como ataque e Verdadeiro Negativo (VN) a representação do número de amostras normais classificadas como normais.

Acurácia (Acc) é definida como a razão entre o número de padrões corretamente classificados e o número total de amostras, como pode ser observado na expressão matemática:

$$Acc = \frac{VP + VN}{VP + VN + FP + FN}$$

Precisão é definida como a porcentagem de amostras que são classificadas positivamente, incluindo as amostras que foram classificados corretamente como ataque e as

amostras normais que foram incorretamente classificadas como ataque. Como pode ser observado na expressão matemática:

$$Precisão = \frac{VP}{VP + FP}$$

Recall é definida como a razão entre o número de ataques classificados corretamente e o número total de amostras de ataque. Como pode ser observado na expressão matemática:

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

F1-score é definido como a média harmônica entre a precisão e o recall, como pode ser observado na expressão matemática:

$$F1_score = 2 * \frac{Precisão * Recall}{Precisão + Recall}$$

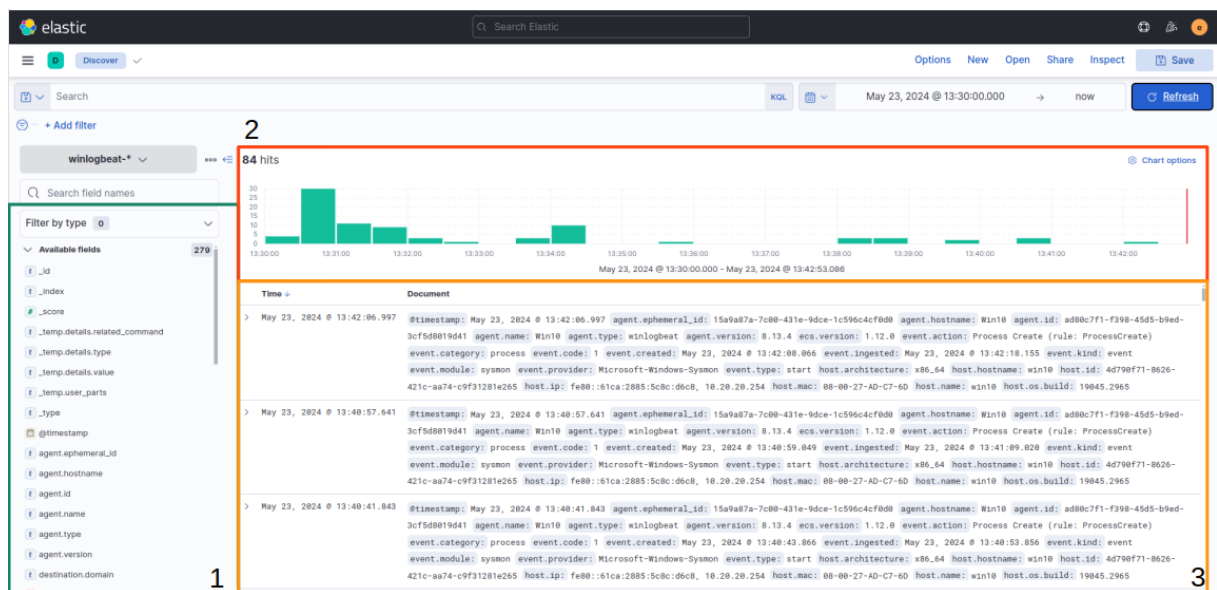
5 Resultados e Discussões

5.1 Método Manual

No método tradicional (manual), o Kibana oferece quatro tipos de serviços: pesquisa empresarial, observabilidade, segurança e análise. Nesta proposta, foram utilizados dois serviços: análise e segurança. O serviço de análise permite visualizar todos os *logs* da rede em tempo real, além de possibilitar a análise individual de cada um e suas características.

A Figura 10 ilustra a interface de monitoramento e visualização dos eventos, sendo composta por três blocos principais: o primeiro sinalizado em verde exibe as características presentes nos *logs*, permitindo a aplicação de filtros por meio de *strings*; o segundo sinalizado em vermelho, apresenta um histograma que ilustra a quantidade de *logs* em um determinado período; e o terceiro sinalizado em amarelo, detalha os *logs* coletados do sistema, sendo possível visualizar no formato de tabela ou *json*.

Figura 10 – Tela de Visualização dos Eventos *Logs*



Fonte: Elaborado pelos autores.

No serviço de segurança, foram utilizados dois recursos principais: regras e alertas. As regras permitem criar critérios de filtragem na rede, enquanto os alertas são emitidos quando os critérios definidos pelas regras são identificados nos eventos *logs* coletados. A combinação desses dois recursos reduz significativamente o volume de dados a serem analisados. Para criar uma regra, é necessário seguir três etapas: (1) definir uma *string* de consulta usando a sintaxe *Kusto Query Language* (KQL); (2) descrever a regra, selecionar o nível de severidade e atribuir um escore de risco; e (3) programar o intervalo de tempo

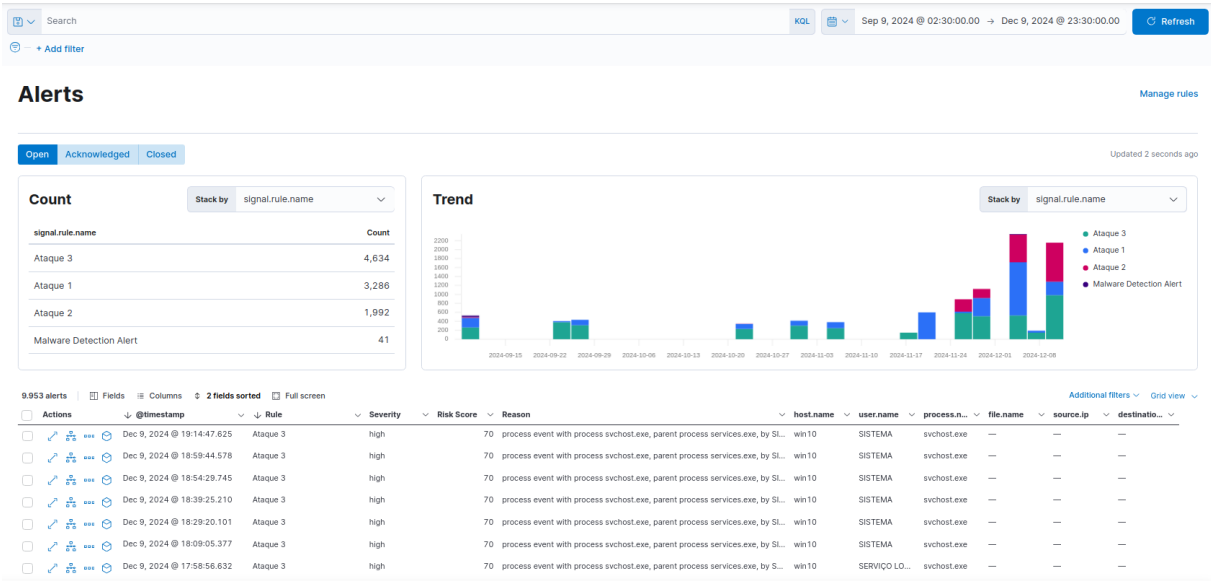
para a execução da regra. Na proposta deste trabalho, foram configuradas 3 regras para detectar os ataques realizados.

A primeira regra utiliza a *string winlog.task* para filtrar as *tasks* "Ciclo de Vida do Provedor", "Ciclo de Vida do Mecanismo" e "Executar um Comando Remoto". A segunda regra está relacionada à *string process.name* que é o nome do programa executado e filtra mais de 14 tipos de programas executáveis, como *net.exe*, *reg.exe*, *rundll32.exe*, sendo que a presença desses tipos de arquivos é considerada suspeita. Por fim, a terceira regra estabelece uma relação entre os arquivos *cmd.exe* e *powershell.exe*. Como esses arquivos são mais comuns, eles não foram incluídos na segunda regra para evitar um alto número de falsos positivos. Em vez disso, adotou-se uma estratégia com o operador AND, onde o *process.name* recebe um desses arquivos enquanto o *process.parent.name* recebe o outro.

Após a criação e ativação das regras, o sistema inicia o monitoramento dos eventos *logs*. Sempre que uma regra é acionada, um alerta é gerado para cada evento correspondente à *string* definida na regra. Em seguida, cabe ao analista de segurança investigar os alertas emitidos, avaliando se correspondem a alertas positivos ou a falsos positivos.

Além disso, os alertas podem ser filtrados com base em diversos critérios, como pontuação, gravidade, tipo, nome da tática, endereço IP de origem ou destino, ação ou categoria do evento, nome de *host* ou usuário. Neste experimento, a filtragem foi realizada com base na categoria do evento (*signal.rule.name*). Como resultado, um total de 9.953 alertas foi emitido.

Figura 11 – Tela de Visualização dos Alertas Emitidos



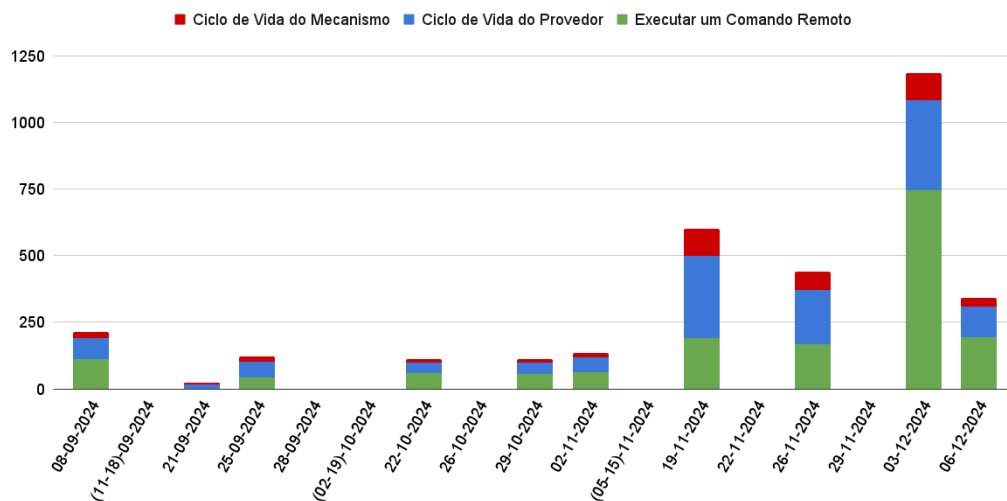
Fonte: Elaborado pelos autores.

A Figura 11 ilustra a tela do Kibana onde encontram-se os alertas emitidos, sendo composto por três blocos principais: contagem, tendência e alertas individuais. O bloco

da contagem exibe a quantidade de ataques emitidos relacionado a cada regra criada. Mesmo apresentando regras diferentes, foi considerada a junção de todas elas. No bloco de tendência, exibe o histograma da quantidade de alertas relacionada com a linha do tempo. Como o foco é visualizar a eficiência das regras, escolheu-se observar os alertas por meio do *signal.rule.name* para identificar se as regras ativadas estavam capturando os dados de ataque. Por fim, no último bloco, apresenta os alertas individuais que devem ser analisados por especialistas.

A quantidade de alertas gerados por cada regra, bem como os eventos logs filtrados, são apresentados em histogramas. A Figura 12 ilustra o histograma de alertas relacionado à regra 1. O histograma foi filtrado usando a mesma *string* de consulta, *event.action*. Assim, é possível observar como os alertas relacionados ao nome do evento estão distribuídos.

Figura 12 – Histograma dos Alertas Emitidos usando a Regra 1

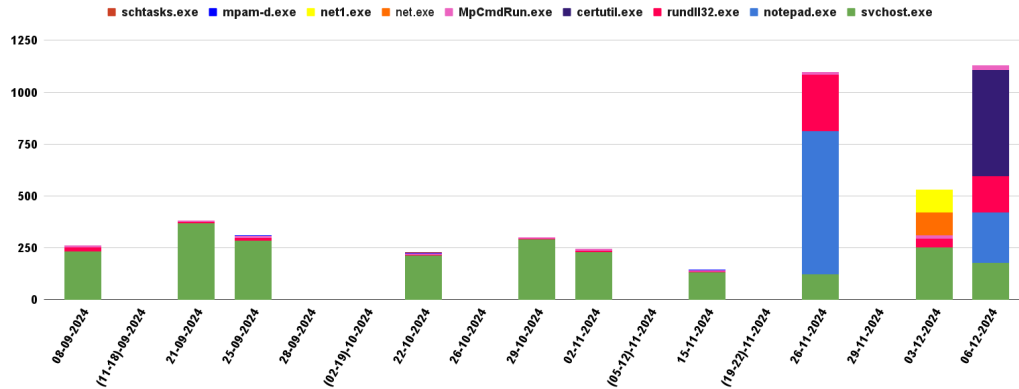


Fonte: Elaborado pelos autores.

Na Figura 13, o histograma de alertas relacionado à regra 2 foi filtrado utilizando o *process.name*, com o objetivo de visualizar cada momento em que o alerta foi emitido, de acordo com o programa em execução. Observa-se uma variedade de programas executados ao longo da linha do tempo.

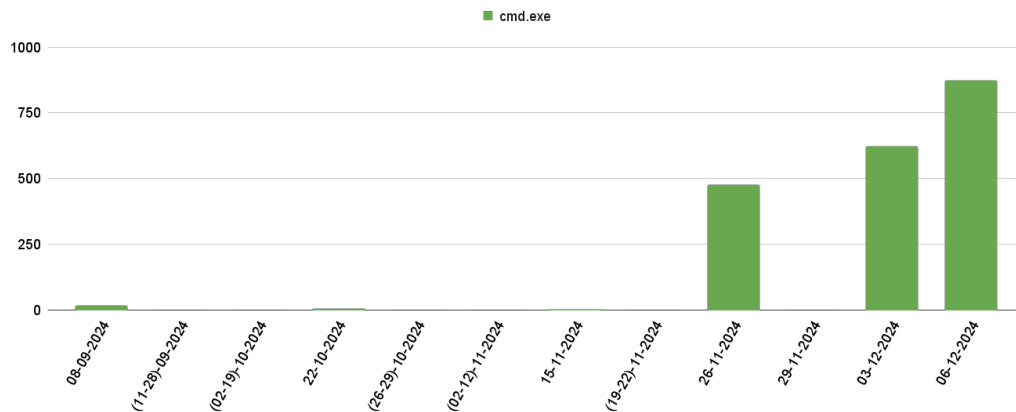
Por fim, a Figura 14 apresenta o histograma de alertas relacionado à regra 3. Diferentemente das filtrações anteriores, que estavam totalmente alinhadas às *strings* de consulta, nesta filtração utilizou-se o nome do programa executado, uma vez que não era possível exibir as demais características previstas na *string*. Assim, é possível observar que as regras criadas apresentaram um retorno positivo com base na análise dos alertas emitidos.

Figura 13 – Histograma dos Alertas Emitidos usando a Regra 2



Fonte: Elaborado pelos autores.

Figura 14 – Histograma dos Alertas Emitidos usando a Regra 3



Fonte: Elaborado pelos autores.

5.2 Método Automatizado

O método automatizado utiliza-se dos algoritmos *Decision Tree* (DT), *Random Forest* (RF), *Support Vector Machine* (SVM) e *Naive Bayes* (NB). O experimento foi conduzido em dois cenários distintos: classificação multiclasse desbalanceada e balanceada e classificação binária.

5.2.1 Cenário 1: Classificação Multiclasse Desbalanceada e Balanceada

Para configurar os parâmetros do DT, foi selecionado o critério de entropia. No entanto, para definir a profundidade máxima da árvore, gerou-se um código que testou valores entre 1 e 6, com o objetivo de identificar a profundidade que apresentasse os melhores resultados, conforme ilustrado na Tabela 9.

Com o objetivo de evitar a ocorrência de *overfitting*, optou-se por usar uma

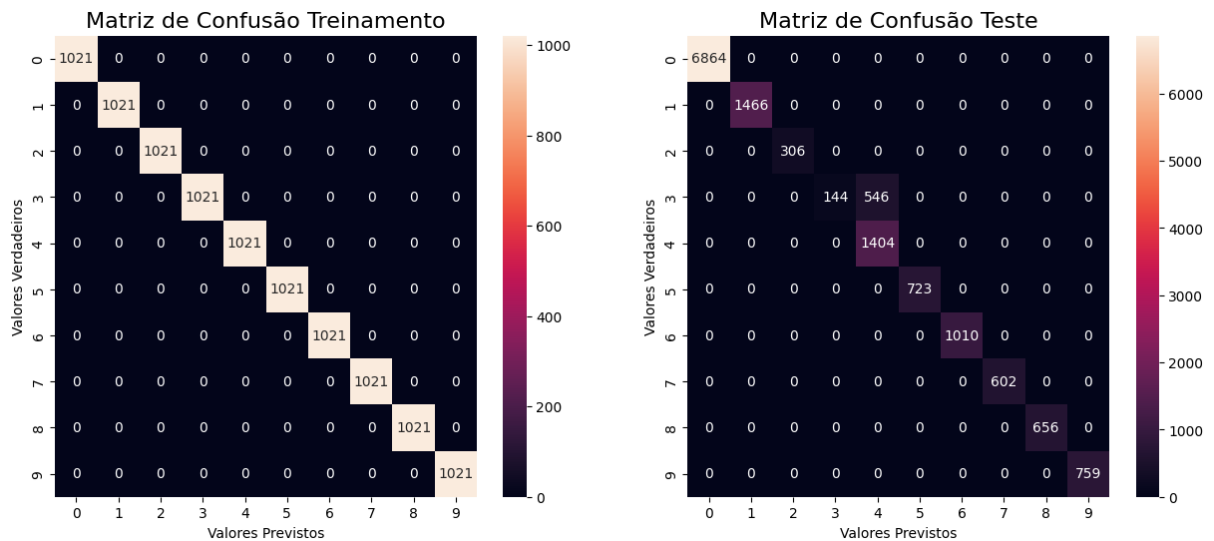
Tabela 9 – Comparação da Profundidade Máxima com Acurácia Média

Profundidade Máxima	Acurácia Média Desbalanceada (%)	Acurácia Média Balanceada (%)
1	19,98	20,01
2	39,96	40,02
3	79,92	79,99
4	100	100
5	100	100
6	100	100

Fonte: Elaborado pelos autores.

profundidade igual a 4, que foi a primeira a alcançar o melhor resultado. Com essa configuração, obteve-se uma acurácia de treinamento igual a 100% e de teste desbalanceado igual a 96,23%. Na Figura 15 observa-se a matriz de confusão, onde não há erros de classificação no treinamento. Contudo, no teste, foi identificada uma dificuldade em classificar a classe 3, frequentemente confundida com a classe 4.

Figura 15 – Matrizes de Confusão DT Desbalanceada

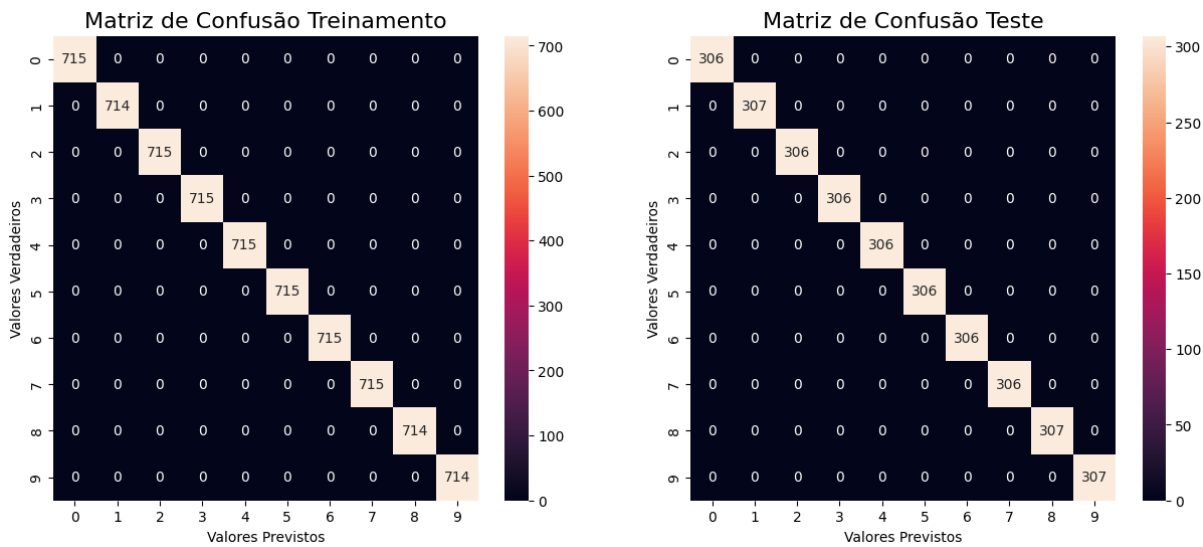


Fonte: Elaborado pelos autores.

Além disso, no teste balanceado, obteve-se 100% de acurácia no treinamento e no teste. Na Figura 16, apresenta a matriz de confusão com todos os dados classificados corretamente de acordo com suas respectivas classes. Tendo esses resultados, pode-se observar que há pouca diferença quando utiliza-se os métodos distintos, tanto na quantidade de acertos quanto no desempenho computacional.

Para avaliar o modelo do algoritmo DT, foi aplicado a validação cruzada e as curvas de aprendizagem de precisão e erro. Como trata-se do mesmo algoritmo, a diferença

Figura 16 – Matrizes de Confusão DT Balanceada

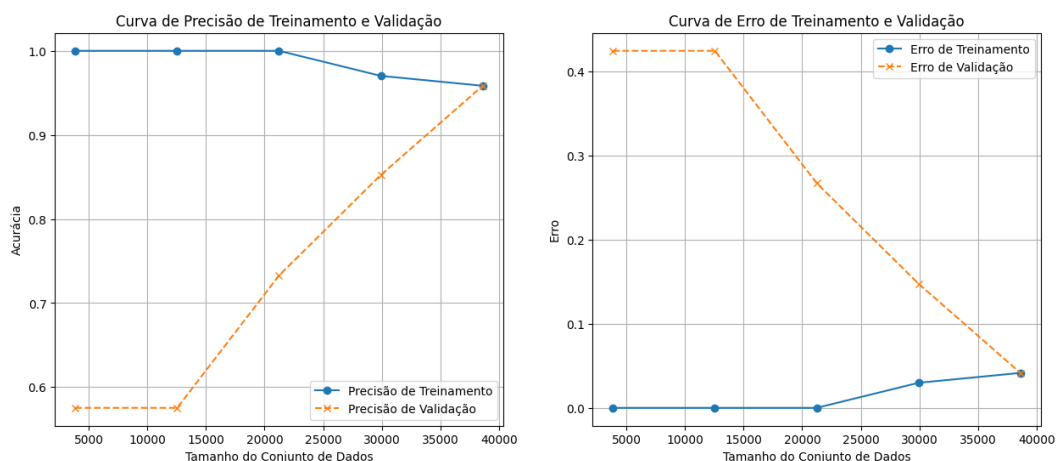


Fonte: Elaborado pelos autores.

de método balanceado ou desbalanceado não interfere nessa avaliação, sendo assim, os resultados obtidos são referentes a ambos. Na Figura 17 observa-se que a curva de treinamento apresenta um leve declínio à medida que o número de amostras aumenta, enquanto a curva de validação é crescente, indicando uma possibilidade de ter ocorrido uma boa generalização do modelo.

A curva de erro, por sua vez, apresenta o comportamento inverso ao da precisão, quando o modelo está bem ajustado, a curva de validação tende a diminuir, porque, à medida que a capacidade de classificação do modelo melhora, menos erros são cometidos. Já, na validação cruzada, o modelo alcançou uma acurácia média de 95,8% e um desvio padrão de 5,14e-05.

Figura 17 – Curvas de Aprendizagem de Precisão e Erro (DT)

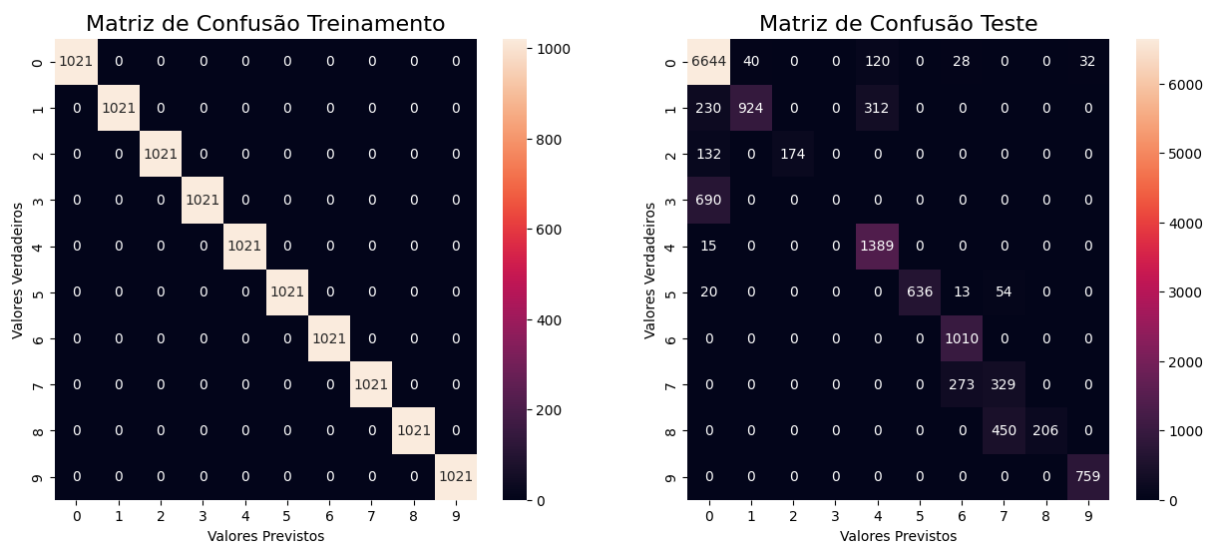


Fonte: Elaborado pelos autores.

Para definir os parâmetros do RF, gerou-se um código que testou valores entre 2 e 30 para o número de árvores e valores entre 1 e 20 para a profundidade máxima das árvores. Com isso, obteve-se os valores 25 e 8, respectivamente.

Com esses parâmetros, obteve-se uma acurácia de treinamento de 99,9% e de teste desbalanceados, 71,45%. Observa-se, na Figura 18, que a matriz de confusão de teste desbalanceado apresenta várias ocorrências de falsos negativos, classificando amostras de ataques como normais. Há também ocorrências de falsos positivos, em que há a classificação de amostras normais são classificadas como ataques.

Figura 18 – Matrizes de Confusão RF Desbalanceada



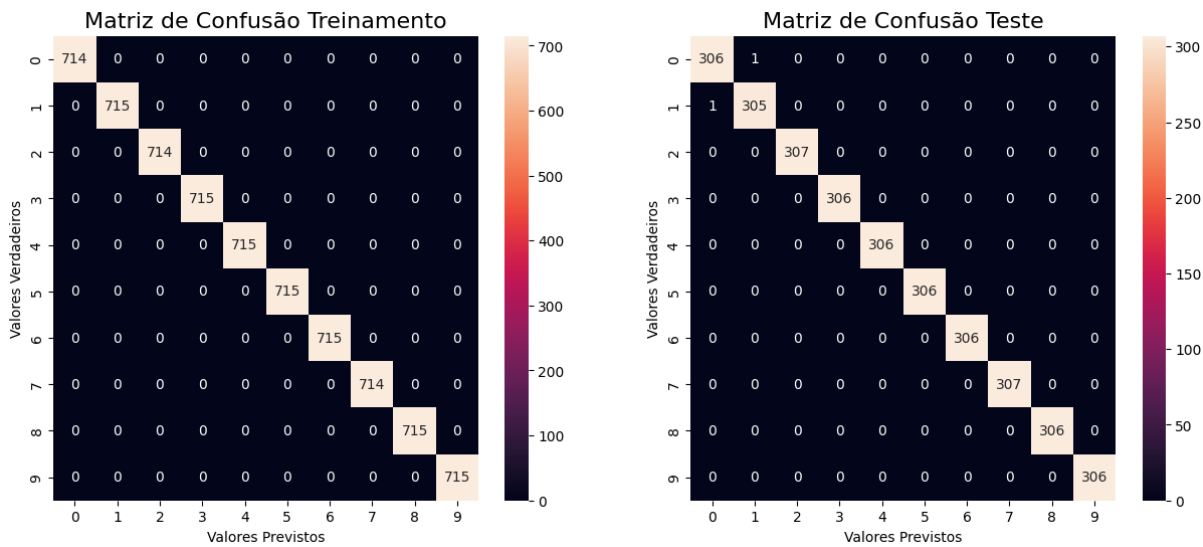
Fonte: Elaborado pelos autores.

Por outro lado, nos teste com amostras balanceadas, o modelo apresentou uma acurácia de 99% no treinamento e no teste. Conforme mostrado na Figura 19, os falsos negativos deixam de ocorrer, e há uma redução significativa de falsos positivos. Contudo, ainda ocorre confusão na determinação da classe respectiva.

Para avaliar o modelo do algoritmo RF, foi aplicado a validação cruzada e a plotagem das curvas de aprendizagem de precisão e erro. Na Figura 20, observa-se que a curva de precisão da validação é crescente, enquanto a de treinamento permanece constante, indicando uma possível generalização adequada do modelo. A curva de erro também apresenta o comportamento esperado. Além disso, na validação cruzada, o modelo alcançou uma acurácia média de 99,97% com um desvio padrão de $2,4e-4$.

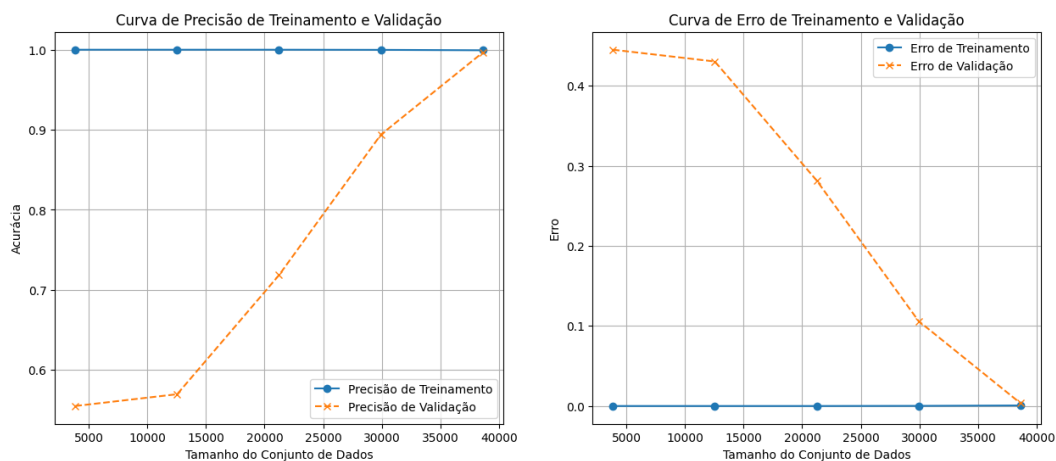
Para configurar os parâmetros do SVM, foi definido o kernel de Função de Base Radial (RBF) e o parâmetro de regularização (C) igual a 1. Com esses parâmetros, obteve-se uma acurácia de treinamento igual a 99,5% e de teste desbalanceado, 54,1%. Na Figura 21, observa-se que a matriz de confusão do treinamento apresenta uma baixa ocorrência de falsos positivos e negativos, mas há erros de classificação nas classes 2 e 4. No entanto,

Figura 19 – Matrizes de Confusão RF Balanceada



Fonte: Elaborado pelos autores.

Figura 20 – Curvas de Aprendizagem de Precisão e Erro (RF)



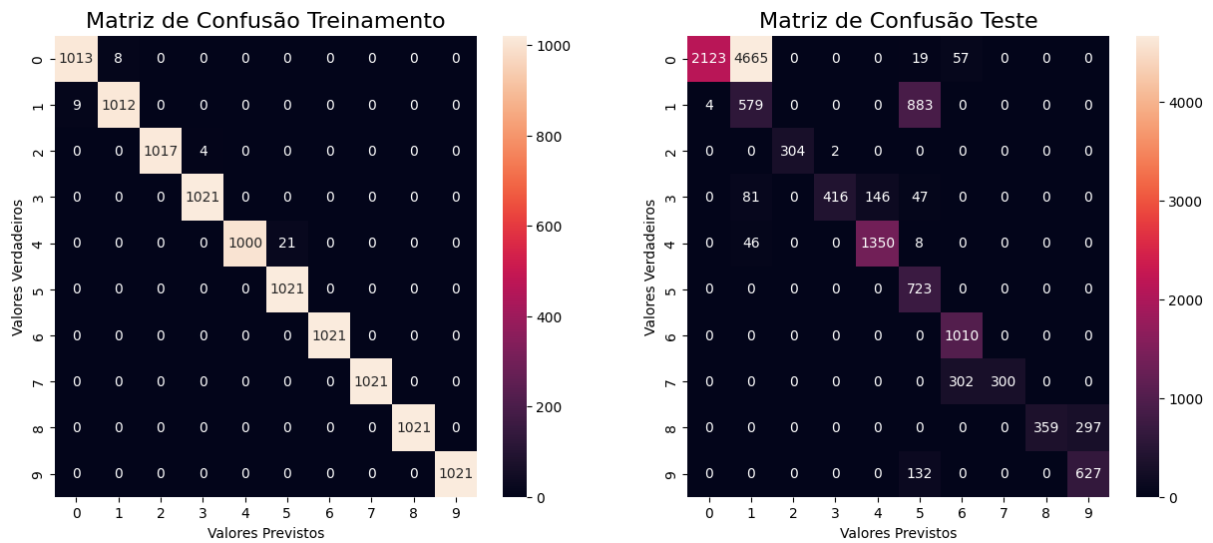
Fonte: Elaborado pelos autores.

a matriz de confusão do teste exibe uma alta ocorrência de falsos positivos, com amostras normais sendo erroneamente classificadas como pertencentes à classe 1.

Em contrapartida, nos testes com amostras balanceadas, a acurácia de treinamento foi mantida em 99,5%, enquanto a acurácia de teste aumentou para 99,7%. Na Figura 22, nota-se que a matriz de confusão do treinamento é semelhante à do cenário anterior. No entanto, a matriz de confusão do teste apresenta uma melhora significativa, com redução tanto de falsos positivos quanto de falsos negativos, além de poucos erros de classificação relacionados a ataques.

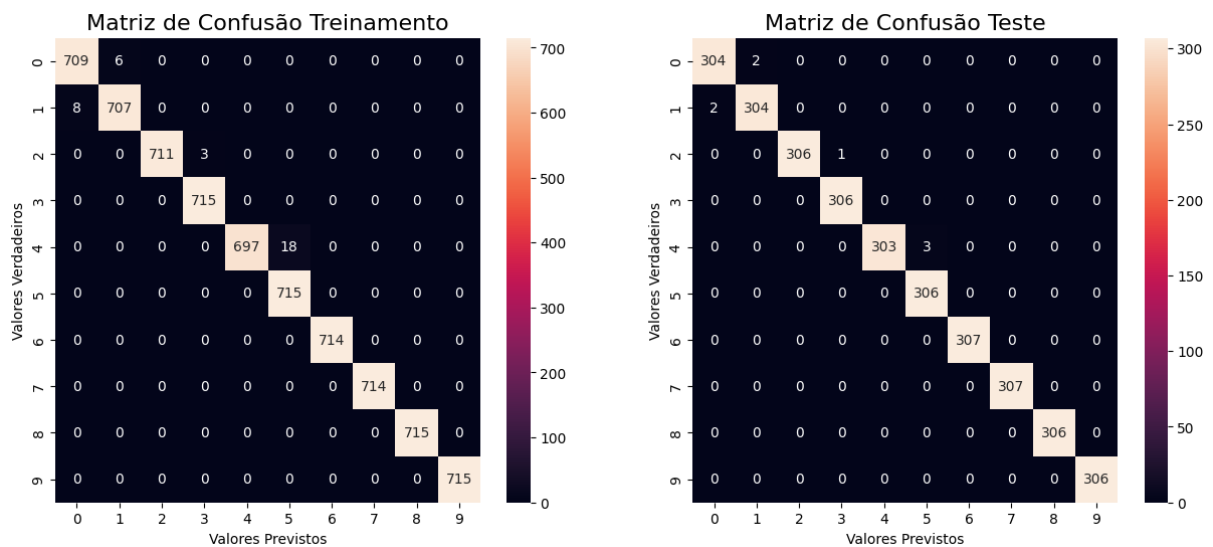
Para avaliar o modelo do algoritmo SVM, foi aplicado a validação cruzada e a plotagem das curvas de aprendizagem de precisão e erro. Na Figura 23, observa-se uma

Figura 21 – Matrizes de Confusão SVM Desbalanceada



Fonte: Elaborado pelos autores.

Figura 22 – Matrizes de Confusão SVM Balanceada

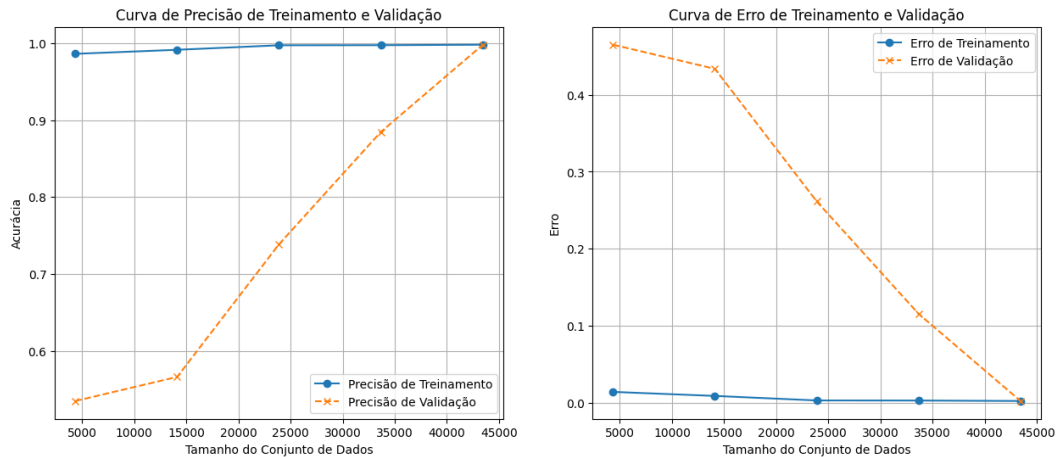


Fonte: Elaborado pelos autores.

curva de precisão da validação é crescente, como esperado para um modelo com boa capacidade de generalização. Da mesma forma, a curva de erro da validação é decrescente. Além disso, na validação cruzada, o modelo alcançou uma acurácia média de 99,7% com um desvio padrão igual a $5,33e-4$.

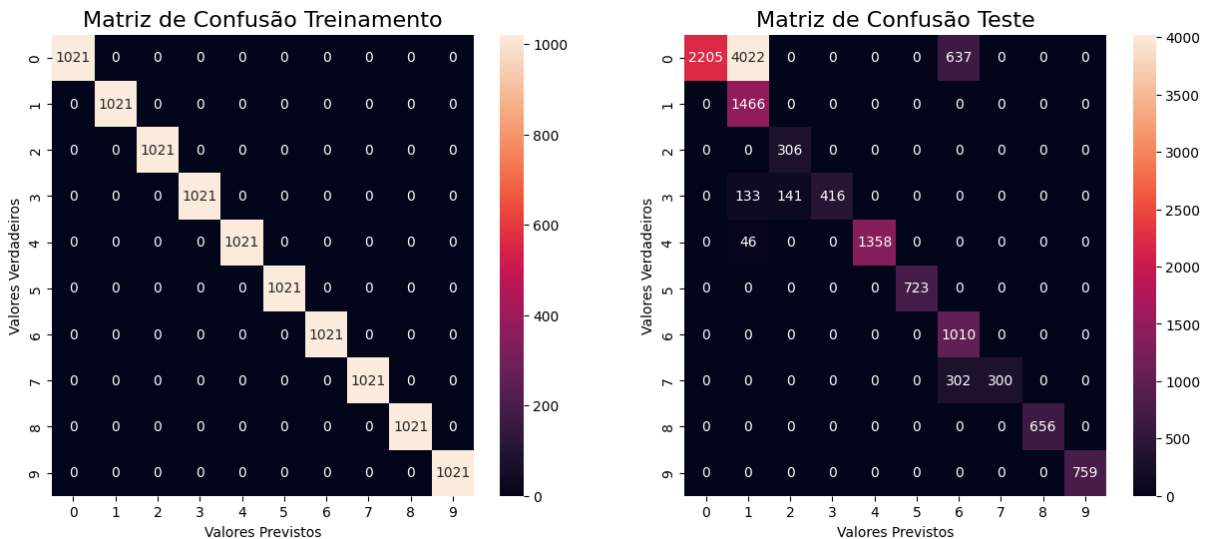
Os parâmetros do NB foram mantidos nos padrões da biblioteca, sem a necessidade de ajustes. Com esses ajustes, obteve-se acurácia de treinamento igual a 100% e de teste desbalanceado, 63,5%. Na Figura 24, observa-se que a matriz de confusão de teste exibe uma ocorrência alta de falsos positivos além de muitas classificações errôneas das classes 2, 3, 6 e 7.

Figura 23 – Curvas de Aprendizagem de Precisão e Erro (SVM)



Fonte: Elaborado pelos autores.

Figura 24 – Matrizes de Confusão NB Desbalanceada

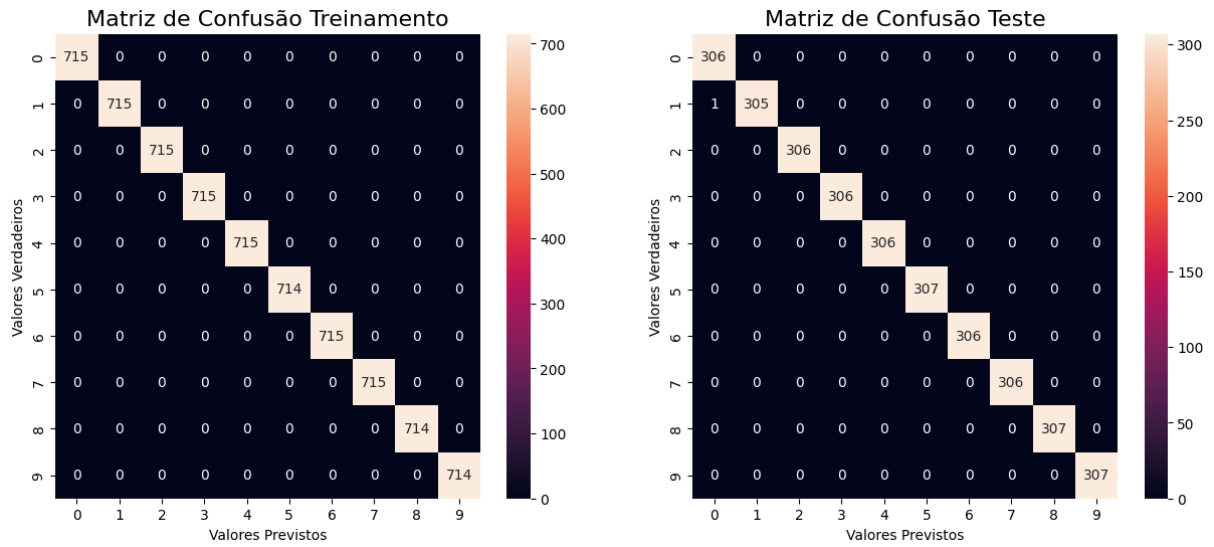


Fonte: Elaborado pelos autores.

Por outro lado, nos testes com amostras balanceadas, a acurácia de treinamento permaneceu em 100%, enquanto a acurácia de teste aumentou significativamente para 99,96%. Na Figura 25, observa-se que a matriz de confusão de teste apresenta uma melhora expressiva, apresentando apenas um falso negativo, em que uma amostra da classe 1 de ataque como normal.

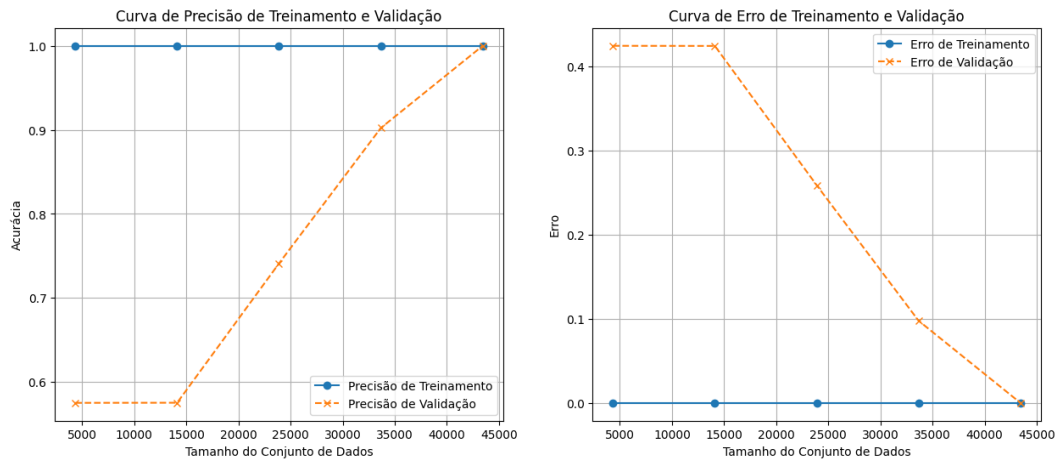
Para avaliar o modelo do algoritmo NB, foi aplicado a validação cruzada e a plotagem das curvas de aprendizagem de precisão e erro. Na Figura 26, observa-se que a curva de precisão da validação é crescente, conforme esperado para um modelo com boa capacidade de generalização. Da mesma forma, a curva de erro da validação é decrescente. Além disso, na validação cruzada, o modelo alcançou uma acurácia média de 99,99% com um desvio padrão de $6,21e-05$.

Figura 25 – Matrizes de Confusão NB Balanceada



Fonte: Elaborado pelos autores.

Figura 26 – Curva de Aprendizagem de Precisão e Erro (NB)



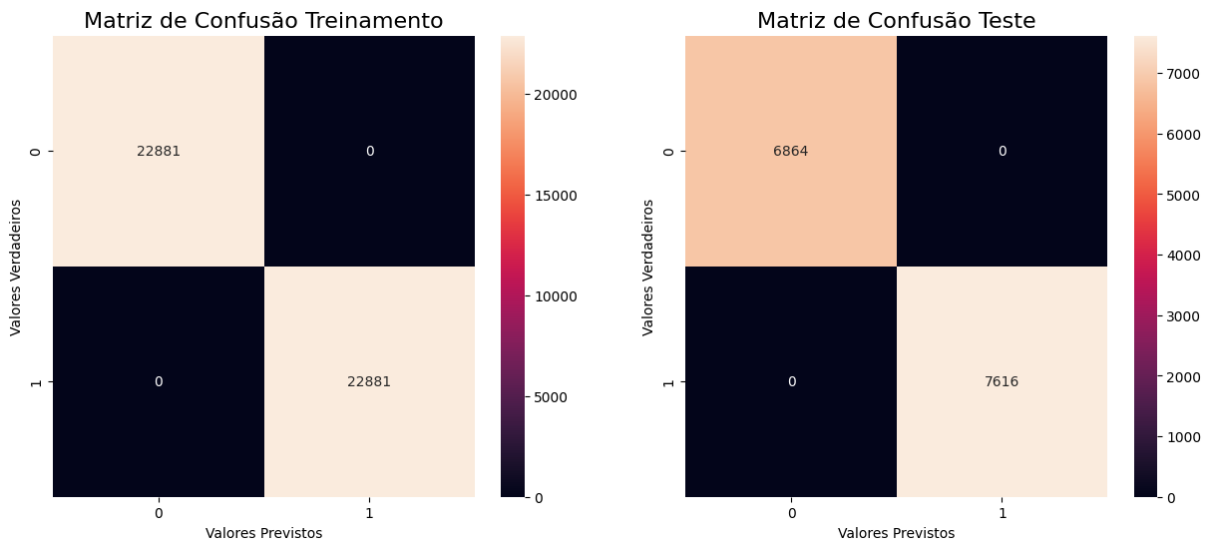
Fonte: Elaborado pelos autores.

5.2.2 Cenário 2: Classificação Binária

Para configurar os parâmetros do DT, foi selecionado o critério de entropia. No entanto, para definir a profundidade máxima da árvore, gerou-se um código com testou valores entre 1 e 6, com o objetivo de identificar a profundidade que apresentasse os melhores resultados. Optou-se por usar uma profundidade igual a 1, visto que em todos os testes foram obtidos acurácia de 100%. Com essa configuração, obteve-se 100% de acurácia no treinamento e no teste. Na figura 27, observa-se as matrizes de confusão de treinamento e teste com as classificações exatas das amostras normais e de ataque.

Para avaliar o modelo do algoritmo DT, foi aplicado a validação cruzada e a plotagem das curvas de aprendizagem de precisão e erro. Na Figura 28, observa-se que a

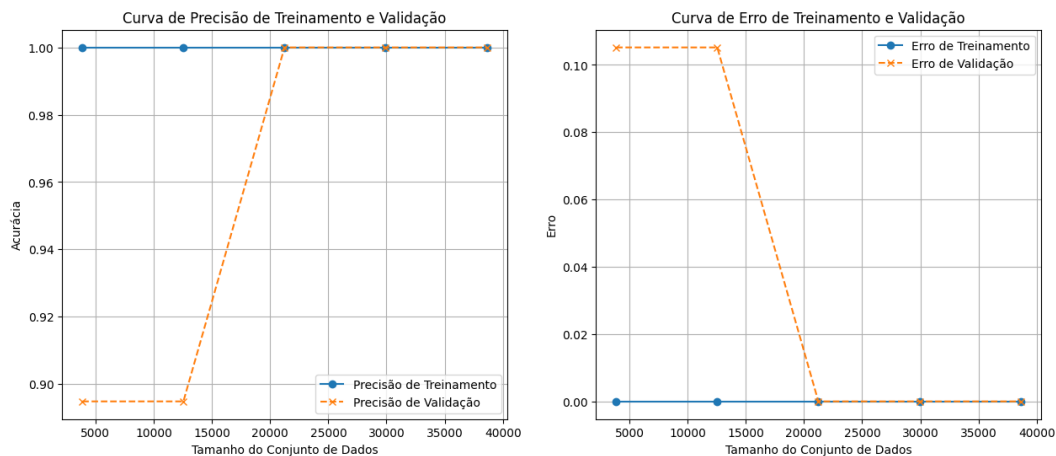
Figura 27 – Matrizes de Confusão DT



Fonte: Elaborado pelos autores.

curva de precisão da validação é crescente, conforme esperado para um modelo com boa capacidade de generalização. Da mesma forma, a curva de erro da validação é decrescente. Além disso, na validação cruzada, o modelo alcançou uma acurácia média de 100% com um desvio padrão nulo.

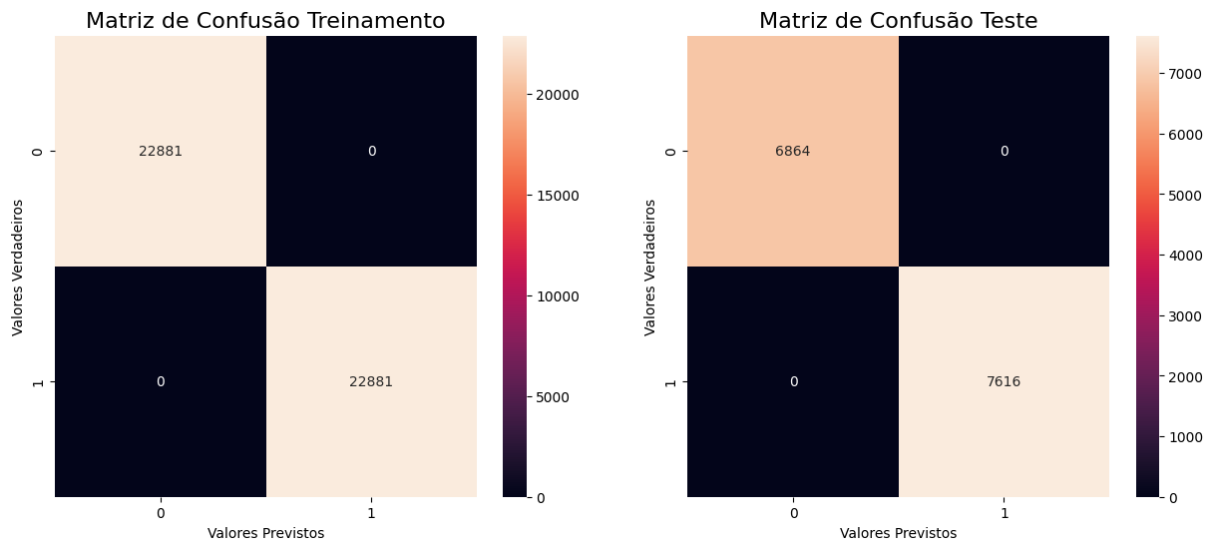
Figura 28 – Curva de Aprendizagem de Precisão e Erro (DT)



Fonte: Elaborado pelos autores.

Para configurar os parâmetros do RF, gerou-se um código que testou valores entre 2 e 30 para o número de árvores e valores entre 1 a 20 para a profundidade máxima das árvores. Com isso, obteve-se os valores 3 e 19, respectivamente. Com esses parâmetros, obteve-se uma acurácia igual a 100% no treinamento e no teste. Na Figura 29, observa-se as matrizes de confusão de treinamento e teste com as classificações exatas das amostras normais e de ataque.

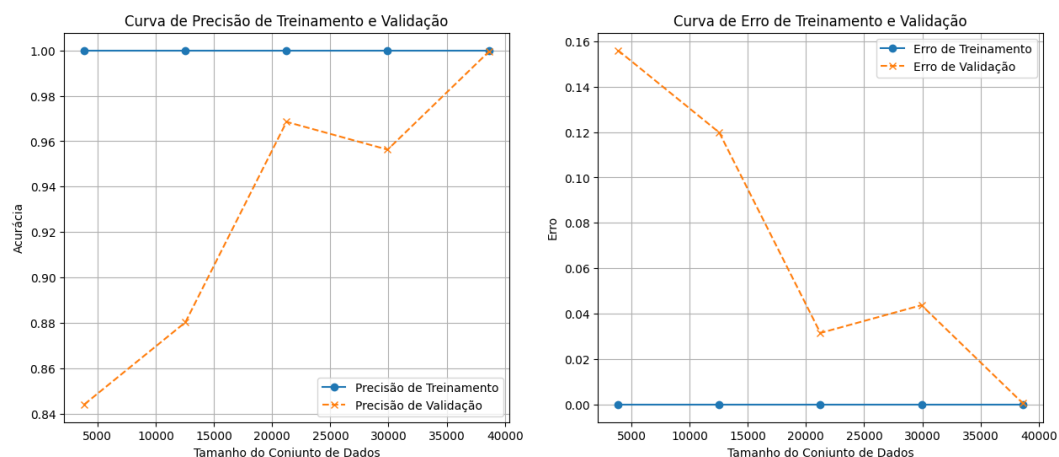
Figura 29 – Matrizes de Confusão RF



Fonte: Elaborado pelos autores.

Para avaliar o modelo do algoritmo RF, foi aplicado a validação cruzada e plotagem das curvas de aprendizagem de precisão e erro. Na Figura 30, observa-se que a curva de precisão da validação é crescente, mesmo ocorrendo variações, enquanto a de treinamento permanece constante, indicando uma possível generalização adequada do modelo. A curva de erro também apresenta o comportamento esperado. Na validação cruzada, o modelo alcançou uma acurácia média de 99,98% e um desvio padrão igual a $1,2e-4$.

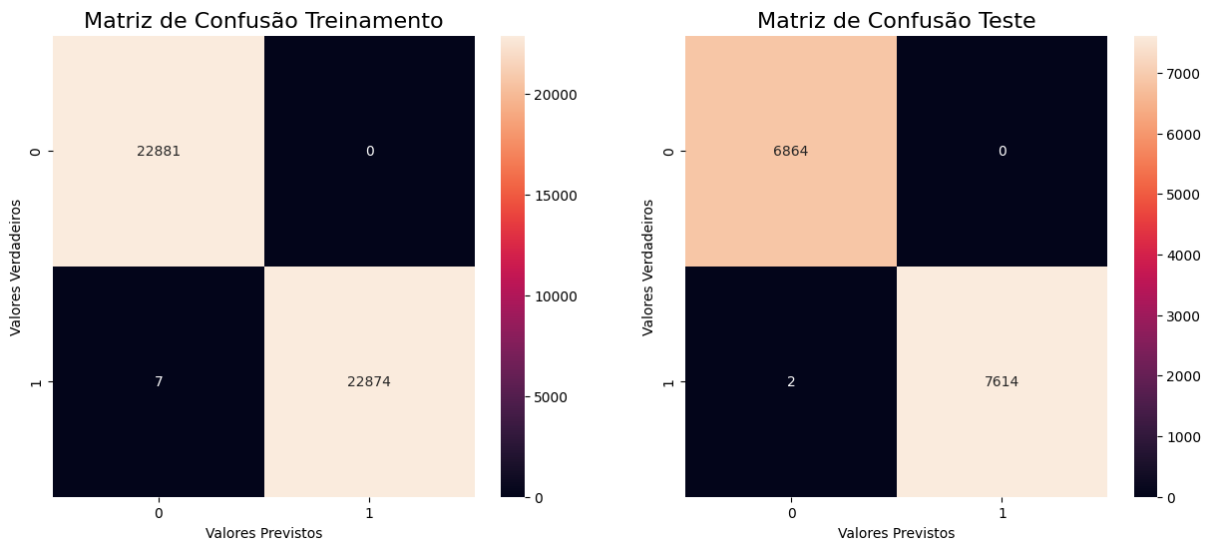
Figura 30 – Curva de Aprendizagem de Precisão e Erro (RF)



Fonte: Elaborado pelos autores.

Para configurar os parâmetros do SVM, foi definido o kernel de RBF e o parâmetro de regularização (C) igual a 1. Com esses parâmetros, obteve-se uma acurácia igual a 99,98% no treinamento e no teste. Na Figura 31, observa-se que tanto na matriz de confusão de treinamento e teste, exibe ocorrências de falsos negativos. Ou seja, amostras de ataques são classificadas erroneamente como amostras normais.

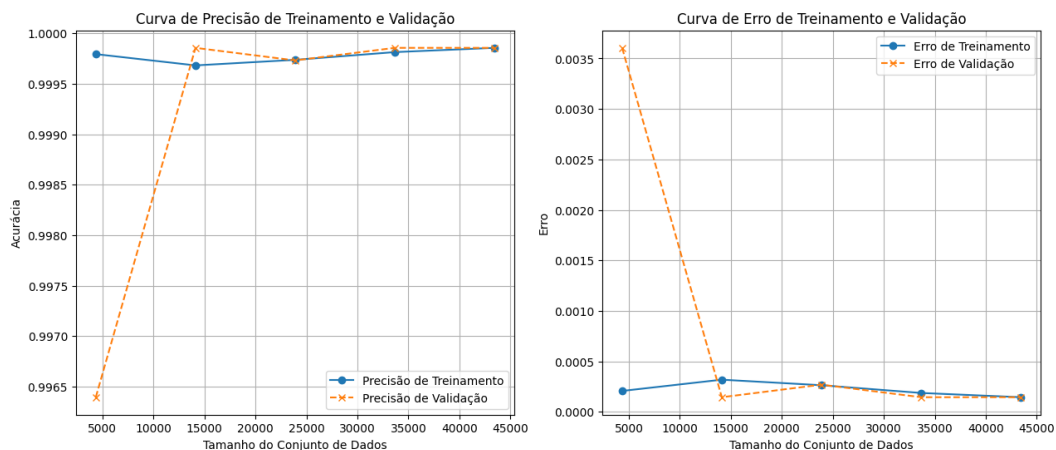
Figura 31 – Matrizes de Confusão SVM



Fonte: Elaborado pelos autores.

Para avaliar o modelo do algoritmo SVM, foi aplicado a validação cruzada e a plotagem das curvas de aprendizagem de precisão e erro. Na Figura 32, observa-se uma curva de erro de treinamento com uma crescente leve, mas o erro tende a zero como previsto. No entanto, a curva de erro de teste, decresce como deve ser, mas há variações no decorrer do aumento das amostras. Na validação cruzada, o modelo alcançou uma acurácia média de 99,98% com um desvio padrão igual a $1,05e-4$.

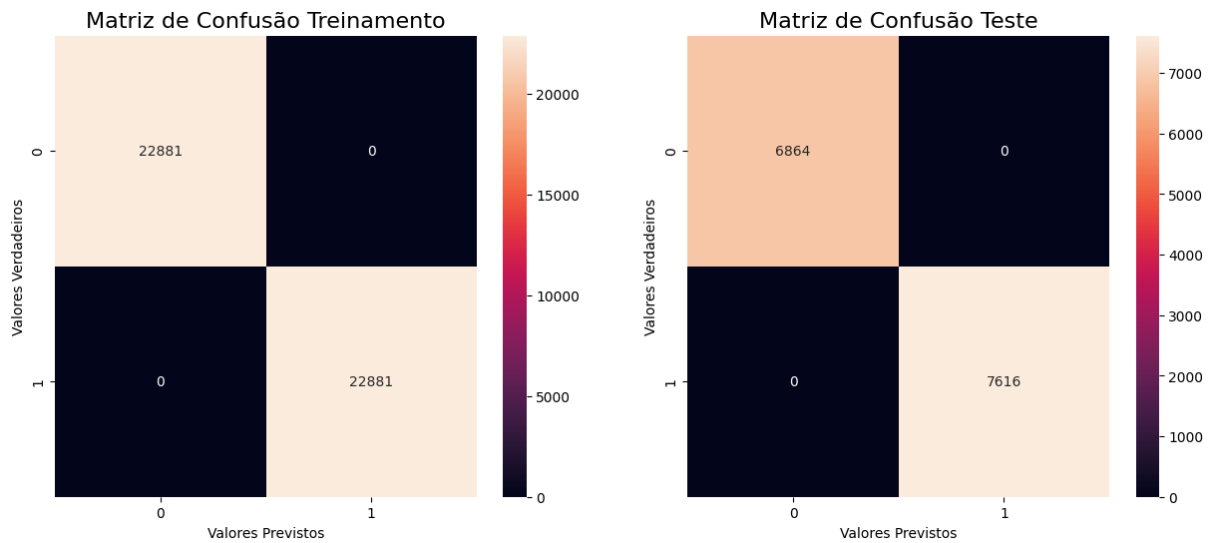
Figura 32 – Curva de Aprendizagem de Precisão e Erro (SVM)



Fonte: Elaborado pelos autores.

Os parâmetros do NB foram mantidos nos padrões da biblioteca, sem a necessidade de ajustes. Com esses ajustes, obteve-se uma acurácia igual a 100% no treinamento e no teste. Na Figura 33, observa-se as matrizes de confusão de treinamento e teste com as classificações exatas das amostras normais e de ataque.

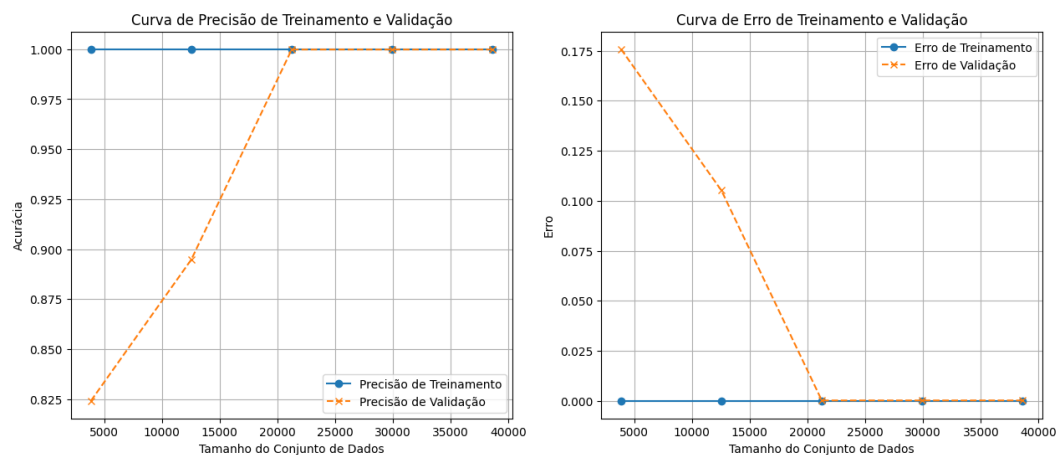
Figura 33 – Matrizes de Confusão NB



Fonte: Elaborado pelos autores.

Para avaliar o modelo do algoritmo NB, foi aplicado a validação cruzada e a plotagem das curvas de aprendizagem de precisão e erro. Na Figura 26, observa-se que a curva de precisão de validação é crescente, conforme esperado para um modelo com boa capacidade de generalização. Da mesma forma, a curva de erro da validação é decrescente. Na validação cruzada, o modelo alcançou uma acurácia média de 99,98% com um desvio padrão igual a $9,26 \times 10^{-5}$.

Figura 34 – Curva de Aprendizagem de Precisão e Erro (NB)



Fonte: Elaborado pelos autores.

5.3 Discussões

Considerando os resultados apresentados na Seção 5, a Tabela 10 compila as métricas extraídas dos modelos de algoritmos testados para classificação de multiclases e

exibe os tempos de treinamento e teste que os algoritmos processam os dados.

Tabela 10 – Métricas da Classificação Multiclasse Desbalanceada.

Treinamento					
Algoritmo	Acc(%)	Precisão(%)	Recall(%)	F1	Tempo(s)
DT Balanceado	100	100	100	1	0,0211
RF Balanceado	99,97	99,97	99,97	0,9997	0,4572
SVM Balanceado	99,51	99,51	99,51	0,9951	0,2623
NB Balanceado	100	100	100	1	0,0130
DT Desbalanceado	100	100	100	1	0,0194
RF Desbalanceado	99,99	99,99	99,99	0,9999	0,1147
SVM Desbalanceado	99,58	99,59	99,58	0,9958	0,5644
NB Desbalanceado	100	100	100	1	0,0379
Teste					
Algoritmo	Acc(%)	Precisão(%)	Recall(%)	F1	Tempo(s)
DT Balanceado	100	100	100	1	0,0037
RF Balanceado	99,93	99,93	99,93	0,9993	0,0306
SVM Balanceado	99,73	99,73	99,73	0,9973	0,3379
NB Balanceado	99,96	99,96	99,96	0,9996	0,0079
DT Desbalanceado	96,23	97,20	92,80	0,9182	0,0033
RF Desbalanceado	71,45	85,65	84,44	0,8067	0,0163
SVM Desbalanceado	54,64	78,35	71,80	0,6894	1,67
NB Desbalanceado	64,03	84,94	84,13	0,7841	0,1426

Fonte: Elaborado pelos autores.

Observou-se que a diferença nas métricas entre os dados balanceados e desbalanceados no DT é pequena. Durante o treinamento, os valores obtidos foram equivalentes, enquanto, nos testes, os dados balanceados apresentaram uma leve melhora, embora não significativa.

Dado que as métricas são tão próximas, realizou-se uma análise de desempenho computacional por meio do tempo de execução, que também revelou valores semelhantes. Assim, considerando que cenários reais geralmente lidam com dados desbalanceados, o algoritmo de DT demonstrou resultados satisfatórios.

Já os algoritmos RF, SVM e NB apresentaram diferenças significativas de desempenho dependendo do cenário analisado. Durante o treinamento, as métricas permaneceram constantes ou apresentaram leves variações; entretanto, no teste, houve uma melhora significativa nas métricas com o uso de amostras balanceadas. Isso sugere a hipótese de que é necessário adicionar mais amostras das classes relacionadas a técnicas de ataque para melhorar a capacidade de generalização do modelo.

Ao analisar o desempenho computacional, observou-se que o SVM demandou muitos recursos para realizar os testes com amostras desbalanceadas. Em um cenário com dados reais, onde predominam amostras desbalanceadas, isso indica a necessidade de recursos computacionais mais avançados e robustos. Além dos resultados obtidos, as métricas da validação cruzada apresentadas na Tabela 11 sugere que quando os algoritmos recebem amostras balanceadas, resultam em acurácias mais próximas.

Tabela 11 – Métricas da Validação Cruzada

Algoritmo	Desvio Padrão	Acc Média(%)
DT	5,14e-05	95,8
RF	2,4e-04	99,97
SVM	5,33e-04	99,7
NB	6,21e-05	99,99

Fonte: Elaborado pelos autores.

Considerando os resultados da Tabela 12, que apresenta as métricas extraídas dos modelos de algoritmos testados para classificação binária e exibe os tempos de treinamento e teste que os algoritmos processam os dados.

Tabela 12 – Métricas da Classificação Multiclasse Balanceada.

Treinamento					
Algoritmo	Acc(%)	Precisão(%)	Recall(%)	F1	Tempo(s)
DT	100	100	100	1	0,0404
RF	100	100	100	1	0,8903
SVM	96,48	94,72	98,45	0,9655	12,279
NB	100	100	100	1	0,0278
Teste					
Algoritmo	Acc(%)	Precisão(%)	Recall(%)	F1	Tempo(s)
DT	100	100	100	1	0,0029
RF	100	100	100	1	0,0263
SVM	96,26	95,04	98	0,965	2,246
NB	100	100	100	1	0,0079

Fonte: Elaborado pelos autores.

No experimento com classificação binária, não foi necessário estudar dois cenários, pois a quantidade de amostras normais e de ataques era semelhantes. Assim, optou-se por utilizar o banco de dados de forma desbalanceada. Em relação às métricas alcançadas pelos algoritmos DT, RF e NB, foi observada uma acurácia de 100% no treinamento e no teste. Dessa forma, ao analisar o tempo de execução necessário para cada modelo, conclui-se que o DT e o NB apresentaram resultados mais satisfatórios. Por outro lado, o algoritmo SVM alcançou uma acurácia elevada de 99,96% nos testes, mas exigindo mais recursos computacionais em comparação aos demais algoritmos.

Além dos resultados obtidos, as métricas da validação cruzada apresentadas na Tabela 13 sugere que os algoritmos alcançaram resultados satisfatórios e esperados pelo modelo.

Tabela 13 – Métricas da Validação Cruzada

Algoritmo	Desvio Padrão	Acc Média(%)
DT	0	100
RF	1,2e-04	99,98
SVM	1,05e-04	99,98
NB	9,26e-05	99,98

Fonte: Elaborado pelos autores.

6 Conclusão

O objetivo principal deste trabalho foi analisar a eficiência dos métodos manuais e automatizados na detecção de ataques, por meio de uma análise comparativa. Para isso, o método manual foi representado pela solução SIEM, implementada com o *Elasticsearch, Logstash and Kibana Stack* (Pilha ELK), enquanto o método automatizado utilizou algoritmos de aprendizagem de máquina supervisionada, *Decision Tree* (DT), *Random Forest* (RF), *Support Vector Machine* (SVM) e *Naive Bayes* (NB).

Para alcançar o objetivo principal, foi realizado um Mapeamento Sistemático (MS) da literatura, que incluiu estudos focados em algoritmos de aprendizagem de máquina aplicados à detecção de ataques, com bancos de dados previamente definidos. Esse mapeamento foi essencial para identificar os principais algoritmos e bancos de dados empregados nas pesquisas. Entre os algoritmos destacados estão DT, RF, SVM e NB. Quanto aos bancos de dados mais utilizados, foram identificados CID-IDS 2018, UNSW-NB15, NSL-KDD e CID-IDS 2017.

Observou-se que os bancos de dados encontrados na literatura geralmente apresentam amostras correspondentes a tipos de *malwares*. Diante disso, optou-se por simular técnicas de ataques descritas no *Mitre Attack*, utilizando a ferramenta *Atomic Red Team* para gerar um banco de dados sintético com mais de 48 mil amostras. Diferentemente dos bancos de dados clássicos, que focam no ataque final, este trabalho analisou as etapas intermediárias que um ataque pode executar em uma rede para alcançar o sucesso.

Para este estudo, criou-se um cenário composto por uma solução SIEM: baseada na Pilha ELK, que monitora, analisa e detecta dados de técnicas de ataque simuladas na rede. Para o monitoramento, o Kibana oferece uma interface de visualização dos *logs* gerados pelo sistema em análise. A partir desses *logs*, são criadas regras de decisão que analisam o tráfego em busca de *strings* específicas. Quando essas *strings* são encontradas, alertas de detecção são gerados e precisam ser analisados por especialistas para confirmar se representam ou não um ataque iminente. Utilizando esse método manual, foram detectadas 9.953 amostras classificadas como ataque.

As regras criadas no Kibana demonstraram eficácia na análise do experimento proposto. O Kibana é uma ferramenta de visualização que oferece diversas funcionalidades, como filtragem por *strings*, datas e outros critérios, permitindo a geração de relatórios e a criação de *dashboards* com resumos dos *logs* coletados. Essas funcionalidades aprimoram a visualização e auxiliam na apresentação e interpretação dos dados.

No método automatizado, foram utilizados algoritmos clássicos amplamente descritos na literatura. A análise foi realizada em dois cenários: classificação multiclasse

desbalanceada e balanceada; e classificação binária. Essa abordagem, que utiliza dados balanceados e desbalanceados, permitiu simular tanto um banco de dados ideal, com a mesma quantidade de amostras para todas as classes, quanto um conjunto de dados real gerados de forma aleatória. Observou-se que os melhores resultados na classificação multiclasse foram obtidos utilizando uma base de dados balanceada no treinamento e no teste. Especificamente, o algoritmo DT alcançou 100%, o RF obteve 99,93%, o SVM atingiu 99,73% e o NB registrou 99,96% nas amostras de teste. Nas classificação binária, os resultados mantiveram-se satisfatórios, com o DT, RF e NB alcançando 100% de acurácia, enquanto o SVM obteve 96,26%.

Embora os resultados tenham sido satisfatórios, é essencial compreender que cada algoritmo possui suas vantagens e limitações. O *DT* é intuitivo e fácil de interpretar; no entanto, tende ao *overfitting*, especialmente em dados complexos. O *RF* é robusto, apresenta alta precisão e reduz o *overfitting* ao combinar várias árvores e amostragens, mas pode demandar mais recursos computacionais. O *SVM* é menos suscetível ao *overfitting* e eficiente para dados de alta dimensionalidade, porém requer maior complexidade computacional em comparação com os demais algoritmos. Por sua vez, o NB é escalável e eficiente para grandes volumes de dados, mas apresenta desempenho moderado quando comparado a modelos mais sofisticados.

Em resumo, conforme indicado na Tabela 14, o método manual obteve uma acurácia significativamente inferior em comparação aos métodos automatizados, mas oferece a vantagem de detecção em tempo real. Além disso, quando uma nova regra é inserida no sistema, ele imediatamente começa a investigar e gerar novos alertas. Por outro lado, os métodos automatizados, mesmo com dois cenários diferentes, alcançaram resultados superiores a 90%. Contudo, a etapa de classificação das amostras é realizada manualmente e revisada periodicamente, resultando em um atraso na detecção de ameaças. Em suma, ambos os métodos apresentam possibilidades de aprimoramento, mas, se combinados, podem se complementar de forma eficaz.

Tabela 14 – Comparativo do Método Manual e Automatizado

Métodos	Ferramenta	Resultado	Etapa Manual	Etapa automatizada
Manual	Pilha ELK	39%	Criação das regras e análise dos alertas	Coleta dos dados da rede e produção de alertas
Automatizado Multiclasse	DT, RF, SVM, NB	DT (100%), RF (99,93%), NB (99,96%), SVM (99,73%)	Classificação das amostras coletadas	Aprendizagem da máquina e detecção
Automatizado Binário	DT, RF, SVM, NB	DT, RF, NB (100%), SVM (96,26%)	Classificação das amostras coletadas	Aprendizagem da máquina e detecção

Fonte: Elaborado pelos autores.

7 Trabalhos Futuros

Visto que o método manual é embasado em regras criados pelos profissionais da área de segurança cibernética, observou-se que em cenários maiores e mais complexos, é necessário avaliar se essas mesmas regras são adequadas à realidade dos dados inseridos e às variações existentes, configurando uma abordagem com potencial para ser explorada. Pode-se explorar também, a utilização do Kibana para gerar visualizações interativas de *dashboards*.

Já no método automatizado, observou-se que o experimento proposto foi baseado em um banco de dados relativamente pequeno em comparação à quantidade de informações geradas atualmente, os algoritmos demonstraram um desempenho eficaz. No entanto, em cenários mais robustos e complexos, torna-se necessária a adoção de algoritmos de aprendizado de máquina mais sofisticados. Este trabalho não contemplou a aplicação de redes neurais ou *deep learning*, o que representa um direcionamento promissor para pesquisas futuras.

Referências

ALMSEIDIN, M. et al. Evaluation of machine learning algorithms for intrusion detection system. *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, p. 000277–000282, 2017. Disponível em: <<https://api.semanticscholar.org/CorpusID:19772255>>. Citado na página 30.

AMINANTO, M. E. et al. Multi-class intrusion detection using two-channel color mapping in iee 802.11 wireless network. *IEEE Access*, v. 10, p. 36791–36801, 2022. Citado na página 30.

ATTOU, H. et al. Cloud-based intrusion detection approach using machine learning techniques. *Big Data Mining and Analytics*, v. 6, n. 3, p. 311–320, 2023. Citado na página 30.

AYALA, C. et al. A hybrid recommender system for cybersecurity based on a rating approach. In: DAIMI, K. et al. (Ed.). *Advances in Security, Networks, and Internet of Things*. Cham: Springer International Publishing, 2021. p. 397–409. ISBN 978-3-030-71017-0. Citado na página 23.

BENTES, E.; FIGUEIREDO, Y.; CAMPOS, L. Aplicação de algoritmos de aprendizado de máquina para detecção de intrusão. In: *Anais Estendidos do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2021. p. 209–216. ISSN 2177-9384. Disponível em: <https://sol.sbc.org.br/index.php/sbrc_estendido/article/view/17173>. Citado na página 30.

CASELI, H. de M.; NUNES, M. das G. V. *Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português*. first. Brasileiras em Processamento de Linguagem Natural, 2023. Disponível em: <<https://brasileiraspln.com/livro-pln/1a-edicao/>>. Citado na página 42.

CERDA, B. M.; YUAN, S.; CHEN, L. Phishing detection using deep learning. In: DAIMI, K. et al. (Ed.). *Advances in Security, Networks, and Internet of Things*. Cham: Springer International Publishing, 2021. p. 117–128. ISBN 978-3-030-71017-0. Citado na página 30.

CHRISTOPHER, F. E.; MYERS, K. J. *Siem-Enabled Cyber Event Correlation (What and How)*. Monterey, Califórnia, 2018. Citado 2 vezes nas páginas 33 e 37.

DHANYA, K. et al. Detection of network attacks using machine learning and deep learning models. *Procedia Computer Science*, v. 218, p. 57–66, 2023. ISSN 18770509. Citado na página 30.

FALBO, R. de A. Mapeamento sistemático. *Retrieved October*, v. 7, 2018. Citado na página 27.

FATEMI, M. R.; GHORBANI, A. A. *Threat Hunting in Windows Using Big Security Log Data*. Security, Privacy, and Forensics Issues in Big Data, 2020. 168-188 p. Disponível em:

<<http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-5225-9742-1.ch007>>. Citado na página 36.

GABER, T.; EL-GHAMRY, A.; HASSANIEN, A. E. Injection attack detection using machine learning for smart iot applications. *Physical Communication*, v. 52, p. 101685, 2022. ISSN 1874-4907. Citado na página 30.

HNAMTE, V.; HUSSAIN, J. Dcnnbilstm: An efficient hybrid deep learning-based intrusion detection system. *Telematics and Informatics Reports*, v. 10, p. 100053, 2023. ISSN 2772-5030. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2772503023000130>>. Citado na página 30.

HOROVITS, D. *The Complete Guide to the ELK Stack*. 2018. Disponível em: <<https://logz.io/learn/complete-guide-elk-stack/#what-elk-stack>>. Citado na página 37.

IBM. *O que é SIEM?* 2023. Disponível em: <<https://www.ibm.com/br-pt/topics/siem>>. Citado na página 37.

JAIN, S.; PAWAR, P. M.; MUTHALAGU, R. Hybrid intelligent intrusion detection system for internet of things. *Telematics and Informatics Reports*, v. 8, p. 100030, 2022. ISSN 2772-5030. Citado na página 30.

KANIMOZHI, V.; JACOB, T. P. Artificial intelligence outflanks all other machine learning classifiers in network intrusion detection system on the realistic cyber dataset cse-cic-ids2018 using cloud computing. *ICT Express*, v. 7, n. 3, p. 366–370, 2021. ISSN 2405-9595. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2405959520304926>>. Citado na página 30.

KARTHIKA, R.; MAHESWARI, M. Detection analysis of malicious cyber attacks using machine learning algorithms. *Materials Today: Proceedings*, v. 68, p. 26–34, 2022. ISSN 2214-7853. 6th International Conference on Recent Advances in Material Chemistry. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2214785322038068>>. Citado na página 30.

KASPERSKY. *Kaspersky: mais de 400 mil malware foram descobertos por dia em 2023*. 2024. Disponível em: <https://www.kaspersky.com.br/about/press-releases/2024_kaspersky-mais-de-400-mil-malware-foram-descobertos-por-dia-em-2023>. Citado na página 23.

KILINCER, I. F.; ERTAM, F.; SENGUR, A. Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Computer Networks*, v. 188, p. 107840, 4 2021. ISSN 13891286. Citado na página 30.

KIM, J. et al. Method of intrusion detection using deep neural network. In: *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*. Jeju: IEEE, 2017. p. 313–316. ISBN 978-1-5090-3015-6. Citado na página 30.

KIMATHI, C. C. *A Platform for monitoring of security and audit events: a test case with windows systems*. Dissertação (Mestrado) — Strathmore University, 2017. Citado 2 vezes nas páginas 23 e 24.

KITCHENHAM, B. A.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering (EBSE 2007-001)*. Keele University and Durham University Joint Report, 2007. Citado na página 27.

LEEVY, J. L. et al. Detecting cybersecurity attacks across different network features and learners. *Journal of Big Data*, v. 8, n. 1, p. 38, Feb 2021. ISSN 2196-1115. Disponível em: <<https://doi.org/10.1186/s40537-021-00426-w>>. Citado na página 30.

LUCAS, T. et al. Stacking-based committees para detecção de ataques em redes de computadores - uma abordagem por exaustão. In: *Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2021. p. 644–657. ISSN 2177-9384. Disponível em: <<https://sol.sbc.org.br/index.php/sbrc/article/view/16753>>. Citado na página 30.

MARCON, D. V. Seguro contra riscos cibernéticos: desafios para delimitar a garantia e promover a cibersegurança na era digital. 2023. Citado na página 23.

MATTOS, S. A. S. *A revolução digital e os desafios da comunicação*. [S.l.]: Editora UFRB, 2013. Citado na página 23.

MUSHTAQ, E.; ZAMEER, A.; KHAN, A. A two-stage stacked ensemble intrusion detection system using five base classifiers and mlp with optimal feature selection. *Microprocessors and Microsystems*, v. 94, p. 104660, 2022. ISSN 0141-9331. Citado 3 vezes nas páginas 30, 39 e 40.

ORIYANIO, S.-P. Working with defensive and detection systems. In: *Penetration Testing Essentials*. John Wiley Sons, Ltd, 2017. p. 171–191. ISBN 9781119419358. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119419358.ch11>>. Citado na página 35.

PAJILA, P. et al. A comprehensive survey on naive bayes algorithm: Advantages, limitations and applications. In: *2023 4th International Conference on Smart Electronics and Communication (ICOSEC)*. [S.l.: s.n.], 2023. p. 1228–1234. Citado na página 42.

PATGIRI, R. et al. An investigation on intrusion detection system using machine learning. In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. [S.l.: s.n.], 2018. p. 1684–1691. Citado na página 30.

REPORT, S. *Ataques cibernéticos no trabalho remoto mais que triplicaram durante a pandemia*. 2023. Disponível em: <<https://www.securityreport.com.br/ataques-ciberneticos-no-trabalho-remoto-mais-que-triplicam-durante-a-pandemia/>>. Citado na página 45.

SANTOS, C. M. da C.; PIMENTA, C. A. de M.; NOBRE, M. R. C. The pico strategy for the research question construction and evidence search. *Revista Latino-Americana de Enfermagem*, Escola de Enfermagem de Ribeirão Preto / Universidade de São Paulo, v. 15, p. 508–511, 2007. Citado 2 vezes nas páginas 17 e 27.

SARHAN, M. et al. Towards a standard feature set of nids datasets. *Mobile Networks and Applications*, 2022. Citado na página 30.

SAYED, M. S. E. et al. A flow-based anomaly detection approach with feature selection method against ddos attacks in sdns. *IEEE Transactions on Cognitive Communications and Networking*, v. 8, n. 4, p. 1862–1880, 2022. Citado na página 30.

SIDDIQI, M. A.; PAK, W. Tier-based optimization for synthesized network intrusion detection system. *IEEE Access*, v. 10, p. 108530–108544, 2022. Citado na página 30.

SOUSA, W. T. M.; SILVA, C. A. Análise de desempenho em algoritmos de aprendizagem de máquina na detecção de intrusão baseada em fluxo de rede usando o conjunto de dados unsw-nb15. *Revista de Sistemas e Computação*, v. 12, n. 2, p. 51–57, 2022. Citado na página 30.

TAHER, K. A.; JISAN, B. M. Y.; RAHMAN, M. M. Network intrusion detection using supervised machine learning technique with feature selection. *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, p. 643–646, 2019. Disponível em: <<https://api.semanticscholar.org/CorpusID:67871296>>. Citado na página 40.

TERRASTRUCT. *Elastic Stack (ELK) Architecture Diagram*. 2023. Disponível em: <<https://terrastruct.com/examples/post/elastic-stack-elk-architecture-diagram/>>. Citado na página 37.

TURCATO, A. C. *Desenvolvimento de método para detecção de intrusão em redes PROFINET baseado em técnicas de Aprendizado de Máquina*. Tese (Doutorado em Sistemas Dinâmicos) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020. Citado 2 vezes nas páginas 35 e 36.

VISHWAKARMA, M.; KESSWANI, N. A new two-phase intrusion detection system with naïve bayes machine learning for data classification and elliptic envelop method for anomaly detection. *Decision Analytics Journal*, v. 7, p. 100233, 2023. ISSN 2772-6622. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2772662223000735>>. Citado 2 vezes nas páginas 30 e 31.

WU, Z. et al. Rtids: A robust transformer-based approach for intrusion detection system. *IEEE Access*, v. 10, p. 64375–64387, 2022. Citado na página 30.