UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Yasmin Marques Vieira

Detecção de Intrusão em Redes de Computadores Usando Técnicas de Detecção de Outliers

Uberlândia, Brasil

2024

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Yasmin Marques Vieira

Detecção de Intrusão em Redes de Computadores Usando Técnicas de Detecção de Outliers

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Elaine Ribeiro de Faria Paiva

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2024

Yasmin Marques Vieira

Detecção de Intrusão em Redes de Computadores Usando Técnicas de Detecção de Outliers

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 26 de abril de 2024:

Elaine Ribeiro de Faria Paiva
Orientador

Professor

Professor

Uberlândia, Brasil
2024

Agradecimentos

Agradeço primeiramente a Deus, pela minha vida, por inspirar cada decisão e por confortar meu coração nos momentos de incerteza e dificuldade.

Aos meus pais, Fernando e Gislaine, e a minha irmã Geovanna, por serem meu porto seguro e amparo em todas as situações, por me motivarem a seguir em frente em busca dos meus sonhos, e por me darem todo o apoio necessário ao longo dos anos.

Aos meus amigos, por cada risada, cada conversa e cada momento de conforto que me proporcionaram, obrigada por tornarem essa jornada mais leve e divertida.

Aos professores, por todos os ensinamentos e conselhos, que fizeram com que eu evoluísse não só intelectualmente, mas também como pessoa e profissional.

A minha orientadora Elaine Faria, por ser meu ombro amigo desde o início da graduação, pelas palavras de incentivo, que fizeram com que eu acreditasse em meu potencial, e principalmente por toda paciência, dedicação e apoio.

"Só se vê bem com o coração, o essencial é invisível aos olhos."
- Antoine de Saint-Exupéry

Resumo

A detecção de intrusão em redes de computadores é uma área de extrema importância para garantir a segurança da informação e a integridade dos sistemas. Neste trabalho, foi explorada a utilização de técnicas de aprendizado não-supervisionado para essa finalidade. A abordagem não-supervisionada é particularmente interessante, pois não requer um conjunto de dados rotulado para treinamento, tornando-a mais flexível e adaptável a novas formas de ameaças. Isso é crucial, considerando a constante evolução das ameaças cibernéticas e a necessidade de sistemas de segurança dinâmicos e adaptáveis. O estudo se concentra em avaliar comparativamente o desempenho de diferentes algoritmos, na detecção de intrusão. Foram utilizadas métricas de avaliação como precisão, revocação e F1-score para comparar o desempenho de cada um deles. Neste contexto, os experimentos foram conduzidos empregando o conjunto de dados CICIDS2017, uma referência na área de segurança cibernética. Considerando os algoritmos testados e a base CICIDS2017, nenhum dos resultados obtidos pelas técnicas não supervisionadas foram satisfatórios para o problema em questão. Dessa forma, conclui-se que mais experimentos e ajustes de parâmetros são necessários para tentar melhorar o desempenho de tais algoritmos e, assim, conseguir obter a detecção de *outliers* de forma confiável e assertiva.

Palavras-chave: Sistemas de Detecção de Intrusão, Aprendizado de Máquina, Fluxos Contínuos de Dados, Não-Supervisionado.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AM	Aprendizado de máquina
IDS	Sistema de detecção de intrusão
LOF	*Local Outlier Factor*
SVM	*Support Vector Machine*

Sumário

1 Introdução

Com o avanço da tecnologia, a Internet assume um papel importante no desenvolvimento da humanidade, o que a torna um ponto crucial na sociedade moderna, e proporciona o crescimento no acesso à rede. De acordo com dados levantados por Kemp (2023), atualmente existem mais de 5 bilhões de indivíduos usando a Internet, o que equivale a mais de metade da população mundial. Com isso, o uso de ferramentas como, *webmail*, redes sociais, *e-commerce* e *internet banking*, tornou-se rotineiro, o que alavancou o número de ataques cibernéticos, e fez com que usuários e organizações se voltassem para um elemento de grande importância: a segurança dos dados.

De acordo com dados divulgados pela Fortinet (2023), baseado nos dados de seu laboratório, o Brasil teve mais de 100 bilhões de ataques cibernéticos, o que totaliza um aumento de 16% em relação ao ano anterior (2021). Isso mostra que é necessário implementar alternativas que detectem ameaças e intrusões, a fim de garantir proteção e segurança dos dados na rede.

Uma forma de realizar o monitoramento de tráfego para detecção de possíveis ataques é utilizando um Sistema de Detecção de Intrusão (IDS, do inglês *Intrusion Detection System*). Tem-se observado o crescente uso e aplicação desses sistemas, onde dois modelos comuns e com bom funcionamento são os baseados em assinaturas e os baseados em anomalias.

Como definido por Kurose e Ross (2013), um IDS baseado em assinatura analisa cada pacote trafegado na rede, e compara-o com as assinaturas previamente armazenadas e definidas. Caso um desses pacotes seja identificado, ele será comparado com o banco de dados e se houver correspondência, resultará em um alerta. Já um IDS baseado em anomalias configura um perfil de tráfego enquanto observa o fluxo de rede em operação normal. Dessa forma, ele buscará por fluxos que fujam do comum e configurem uma anomalia na rede. Neste método é possível detectar novos ataques que não foram relacionados anteriormente, somente detectando-os como fora do perfil definido como normal (BHUYAN; BHATTACHARYYA; KA-

LITA, 2017).

Muitos trabalhos recentes da literatura Mafra et al. (2008), D'ANDRADA (2020), Yang et al. (2022), Rocha (2023), tem desenvolvido estratégias para detecção de intrusão baseado em anomalias, com o uso de técnicas de aprendizado de máquina (uma área da Inteligência Artificial) e mineração de dados. Segundo a IBM (2022), no *Cost of Data Breach* 2022, as organizações que usam Inteligência Artificial e automação tiveram um ciclo de vida de violação de 74 dias mais curto e economizaram em média US$ 3 milhões. O que evidencia a importância e impacto em se utilizar recursos atuais para constituir um bom IDS.

1.1 Objetivos

Este trabalho tem como objetivo avaliar comparativamente o desempenho preditivo de técnicas de aprendizado não-supervisionado na tarefa de detecção de intrusão em redes de computadores, usando para isso medidas de avaliação como precisão, revocação e taxa de falsos alarmes. Serão também utilizadas técnicas de aprendizado adaptativo, que conseguem atualizar o modelo aprendido quando há mudanças na forma de um ataque ocorrer.

Os objetivos específicos são:

- ❏ Analisar o desempenho preditivo de diferentes algoritmos não-supervisionados, em especial, os de detecção de *outliers*, e análisá-los comparativamente.
- ❏ Montar um cenário experimental que permita avaliar um algoritmo de detecção de *outliers* desenvolvido para fluxos contínuos de dados.
- ❏ Avaliar o uso de conjuntos de treinamento de diferentes tamanhos no desempenho do algoritmo.

1.2 Justificativa

Com o crescente uso de técnicas para monitoramento da rede, novos modelos e formas de ataques vem surgindo com o objetivo de conseguir burlar esses

sistemas. De acordo com Research (2023), os ciberataques estão aumentando em todo o mundo, registrando um aumento por semana de 38% nas redes corporativas em 2022, em comparação com 2021. Várias tendências de ameaças cibernéticas estão ocorrendo simultaneamente. Dessa forma, muitos estudos e diferentes técnicas para detecção de intrusão tem sido utilizadas, como as abordadas em Galhardi (2017) e Lima (2021).

Atualmente inúmeros trabalhos tem utilizado algoritmos de aprendizado de máquina (AM) para fluxos contínuos de dados (do inglês *data streams*). Fluxos contínuos de dados são sequências de dados geradas continuamente, na qual a distribuição que gera os dados pode mudar ao longo do tempo (GAMA, 2010). Neste caso, os modelos de decisão precisam ser constantemente atualizados a fim de tratar tais mudanças. Este é um tipico cenário de ataques em redes de computadores, pois novos ataques podem surgir e o comportamento de ataques já conhecidos podem ser modificados, o que exige uma constante atualização dos modelos já aprendidos.

Assim, como nas técnicas tradicionais de AM, em fluxos contínuos de dados, o aprendizado pode ser dividido em dois grupos: o aprendizado supervisionado e o não-supervisionado. A maioria dos trabalho são focados em propostas supervisionadas, essa técnica exige um conjunto de dados rotulados para o treinamento de um modelo de decisão, o qual é usado para classificar novos dados da rede como um possível ataque ou não. No entanto, estas propostas dependem de novos dados rotulados para atualizar o modelo de decisão. Rotular novos dados e manter uma base de dados atualizada é uma tarefa difícil, pois exige um especialista de domínio para rotular tais dados.

Por outro lado, os métodos não-supervisionados foram menos explorados na tarefa de detecção de intrusão. Os métodos não-supervisionados não utilizam uma base de dados rotulada, sendo que apenas os dados e suas relações são usadas para a identificação de padrões.

Uma das técnicas não-supervisionadas é a detecção de *outliers*, que tem sido bastante retratada na literatura. No cenário de IDS, em resumo, ela consiste em analisar um conjunto de dados obtidos dos acessos a uma rede de computadores,

tendo em vista o padrão normal de acesso à rede. Ao detectar um comportamento anômalo, deve-se sinalizá-lo como um possível ataque. No cenário atual, poucos trabalhos usaram essa técnica aplicada a fluxos contínuos de dados. A comparação entre diferentes técnicas não-supervisionadas, em especial, as técnicas de detecção de *outliers* para cenários *batch* e para fluxos contínuos de dados, pode ajudar na tarefa de detecção de intrusão.

1.3 Organização do Trabalho

Esta monografia está estruturada em outros quatro capítulos além deste de introdução. Os demais capítulos estão organizados como segue.

O Capítulo 2 trata da fundamentação teórica para embasar os métodos propostos, bem como os trabalhos correlatos. O Capítulo 3 descreve detalhadamente o método utilizado para atingir os objetivos propostos. O Capítulo 4 apresenta os experimentos e a discussão dos resultados obtidos. E, por fim, o Capítulo 5 traz a conclusão desta monografia, as principais contribuições e sugestões de trabalhos futuros.

2 Fundamentação Teórica

Nesta seção, serão apresentados os principais conceitos, termos e estudos que embasam o trabalho em questão. Inicialmente, será discutida a definição e a importância da segurança em redes de computador, contextualizando-a dentro do campo de estudo específico. A seguir serão apresentados os aprendizados de máquina, que são divididos em alguns grupos, sendo dois deles amplamente estudados: supervisionado e não-supervisionado. Em sequência, será abordado sobre fluxo contínuo de dados. Seguirá com a apresentação das medidas de avaliação presentes na literatura e, por fim, as ferramentas utilizadas para detecção de *outliers*.

2.1 Sistemas de Detecção de Intrusão - *Intrusion Detection System (IDS)*

A Segurança da Informação define a forma como se deve proteger informações e dados das diversas formas de ataque, garantindo a toda informação e dado a tríade básica da segurança (KUROSE; ROSS, 2013): confidencialidade, integridade e disponibilidade. Onde a confidencialidade se refere à garantia de que as informações e dados sensíveis ou sigilosos estão protegidos contra o acesso não autorizado. A integridade visa garantir que os dados e informações permaneçam precisos e confiáveis ao longo do tempo, ela é fundamental para a confiabilidade das informações e a tomada de decisões com base nelas. E, por fim, a disponibilidade que diz respeito à garantia de que os sistemas, recursos e informações estão disponíveis e acessíveis quando necessário, isto é vital para a continuidade dos negócios e a prestação de serviços sem interrupções.

Com o passar dos anos, há um aumento significativo de informações na internet. Segundo a Cisco (2018), quase dois terços da população global terá acesso à Internet até 2023. Haverá em torno de 5,3 bilhões de usuários da Internet, que corresponde a mais de 60% da população global. E, além disso, o número de dispositivos conectados às redes IP será mais de três vezes a população global até

2023. Consequentemente, há também um crescimento no número de ataques a fim de roubar essas informações. Deste modo, a necessidade de garantir a segurança dos dados e informações que trafegam na rede também aumenta.

De acordo com a definição dada por Bace e Mell (2001), detecção de intrusão é o processo de monitoramento e analise de eventos que ocorrem em um sistema de computadores ou em uma rede. Esta analise é importante para detectar sinais de intrusões, que possam comprometer a confidencialidade, integridade, disponibilidade, ou até mesmo burlar os mecanismos de segurança da rede, o que torna possível obter êxito na invasão. Os sistemas de detecção de intrusão (IDS), são um tipo de dispositivo ou software projetado para monitorar e analisar o tráfego de rede ou os eventos do sistema em busca de atividades suspeitas ou maliciosas, e eles podem ser classificados de acordo com seu método de detecção, conforme apresentado por Effendy, Kusrini e Sudarmawan (2017), sendo dois deles amplamente utilizados na literatura: detecção baseada em assinatura, detecção baseada em anomalia. Estas duas abordagens serão descritas a seguir.

A detecção baseada em assinatura, consiste em analisar as atividades do ambiente, em busca de eventos que correspondam aos padrões pré-definidos em sua base como ataques. Como esse método necessita de uma comparação com sua base de assinaturas, ele não é apto a identificar novos eventos maliciosos que apareçam em seu ambiente, por isso, é necessário que se mantenha sua base sempre atualizada à medida que novos ataques são descobertos.

Já a detecção baseada em anomalias, ao contrário das técnicas baseadas em assinatura, compara os eventos que ocorrem no sistema com um perfil criado com base em um tráfego normal na rede. Com esse método, sempre que for detectado um evento anômalo, que não corresponde ao perfil normal, ele será classificado como um ataque.

A Tabela 1 apresenta os prós e contras dos dois métodos de detecção abordados acima.

Tabela 1 – Prós e contras dos métodos de detecção de intrusão - Adaptado de (LIAO et al., 2012)

	Baseado em Assinatura	**Baseado em Anomalia**
Prós	- Método simples e efetivo para detectar ataques conhecidos. - Análise contextual detalhada.	- Efetivo na detecção de novas e imprevistas vulnerabilidades. - Menos dependente do sistema operacional. - Facilita as detecções de abuso de privilégio.
Contras	- Ineficaz para detectar ataques desconhecidos, ataques de evasão e variantes de ataques conhecidos. - Pouco entendimento de estados e protocolos. - Difícil de manter assinaturas/padrões atualizados.	- Precisão de perfis fraca devido a eventos observados sendo constantemente alterados. - Indisponível durante a reconstrução de perfis de comportamento. - Dificuldade em disparar alertas em tempo real.

Apesar de termos alguns contras nos sistemas baseados em anomalia, muitos deles podem ser tratados com o uso de técnicas de aprendizado de máquina para fluxos contínuos de dados.

2.2 Aprendizado de máquina

Segundo a definição apresentada por Monard e Baranauskas (2003), o aprendizado de máquina é uma área de IA onde o objetivo principal é o desenvolvimento de técnicas computacionais sobre o aprendizado bem como a construção de sistemas capazes de adquirir conhecimento de forma autônoma.

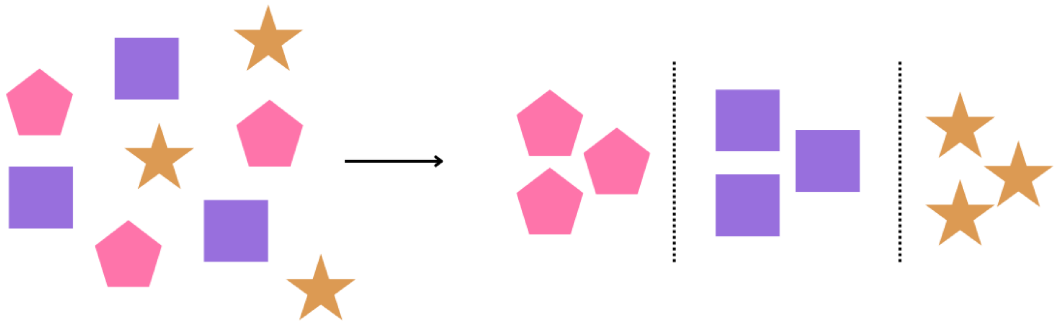
O objetivo do Aprendizado de Máquina é a construção de programas que melhorem seu desempenho por meio de exemplos (MITCHELL et al., 2007). Para isso, é preciso gerar uma grande base de dados de exemplos, a fim de produzir o conhecimento da máquina. Segundo Ludermir (2021) um dos principais métodos para derivar um novo conhecimento e predizer eventos futuros é a inferência indutiva.

O AM pode ser classificado em três principais grupos, são eles: supervisionado, não-supervisionado e por reforço. Dentre estes, ao abordar os sistemas de detecção de intrusão, percebe-se um destaque ao aprendizado supervisionado e não-supervisionado, que são amplamente mencionado na literatura deste segmento.

2.2.1 Aprendizado supervisionado

Este método de aprendizado é amplamente utilizado na literatura de IDS, e consiste em criar um classificador que consiga rotular corretamente dados de entrada, ainda não classificados. Para a construção do classificador é apresentado diversos exemplos rotulados, ou seja, exemplos que possuem a saída desejada (classe alvo), para que com esses exemplos seja possível aprender um modelo, que será capaz de rotular novos dados que chegarem.

Figura 1 – Exemplo de funcionamento do aprendizado supervisionado - adaptado de (SAH, 2020)

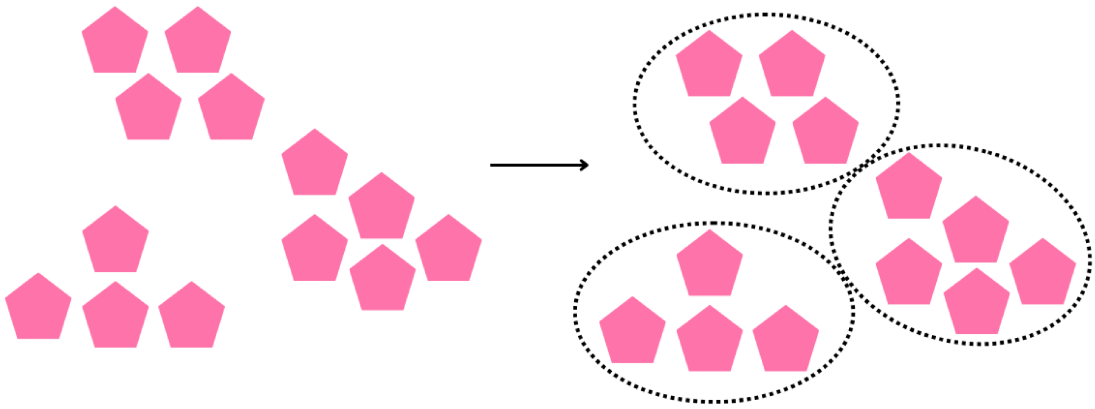


Na Figura 1, tem-se um exemplo do funcionamento do aprendizado supervisionado, onde exemplos de entrada são categorizados em um conjunto conhecido de classes, totalizando, neste caso, três classes distintas.

2.2.2 Aprendizado não-supervisionado

Diferentemente do aprendizado supervisionado, aqui os exemplos não são rotulados. Assim, para conseguir classificar os dados, o algoritmo consiste na tentativa de realizar um agrupamento por meio da semelhança dos atributos que descrevem um exemplo. Para isso, há uma análise de cada um dos exemplos a fim de encontrar algo em comum entre eles. Após a definição dos agrupamentos, é feito um estudo para determinar o que cada agrupamento significa, e se faz sentido no contexto do problema.

Figura 2 – Exemplo de funcionamento do aprendizado não supervisionado - adaptado de (SAH, 2020)



Na Figura 2, tem-se uma visão geral do aprendizado não supervisionado. As amostras de entrada são agrupadas em *clusters* com base nos padrões subjacentes.

Clustering é uma das técnica mais importantes do aprendizado de máquina não-supervisionado. Como apontado por Filho (2018), esta técnica consiste em encontrar grupos no conjunto de dados de entrada, nos quais os exemplos de um mesmo *cluster* sejam os mais similares possíveis entre si, enquanto que os exemplos em diferentes *clusters* sejam os mais distintos possíveis. Mitchell et al. (2007) mostram que essa similaridade ou diferença é dada de acordo com alguma função de similaridade/dissimilaridade adotada.

Em MILLIGAN (1996) existe uma definição de sete passos essenciais para uma análise de *cluster*, sendo eles:

- ❏ Seleção dos Exemplos de Agrupamento: devem ser escolhidos de forma a representar a estrutura de agrupamento que se acredita estar presente nos dados.
- ❏ Seleção dos Atributos de Agrupamento: representam as medidas tomadas em cada exemplo a ser agrupado.
- ❏ Padronização dos Atributos: pode ou não ser realizada.
- ❏ Medida de Associação: uma medida de similaridade ou dissimilaridade deve ser selecionada e esta deve refletir o grau de proximidade ou separação dos exemplos.
- ❏ Método de Agrupamento: um método deve ser selecionado de acordo com o tipo de agrupamento que se espera encontrar nos dados.
- ❏ Número de Clusters: determinar o número de *clusters* na solução.
- ❏ Interpretação, Teste e Replicação: interpretar os resultados no contexto do problema aplicado. Em seguida, os testes podem ser usados para determinar se existe uma estrutura de *cluster* significativa nos dados. E, por fim, a análise de replicação usada para determinar se a estrutura de *cluster* obtida pode ser replicada em uma segunda amostra.

Os algoritmos de *clustering* podem ser categorizados de diferentes formas. Xu e Tian (2015) os dividem entre algoritmos tradicionais e algoritmos modernos. Os algoritmos tradicionais são subdivididos em 9 categorias, conforme mostra a Tabela 2. Já os modernos são subdivididos em 10 categorias, como apresentado na Tabela 3.

Tabela 2 – Algoritmos tradicionais - adaptado de (XU; TIAN, 2015)

Categoria	**Algoritmos**
Baseado em partição	k-means, k-medoids, PAM, CLARA, CLARANS
Baseado em hierarquia	BIRCH, CURE, ROCK, Chameleon
Baseado na teoria de fuzzy	FCM, FCS, MM
Baseado na distribuição	DBCLASD, GMM
Baseado em densidade	DBSCAN, OPTICS, Mean-shift
Baseado em teoria dos grafos	CLICK, MST
Baseado em *grid*	STING, CLIQUE
Baseado em teoria dos fractais	FC
Baseado em modelo	COBWEB, GMM, SOM, ART

Tabela 3 – Algoritmos modernos - adaptado de (XU; TIAN, 2015)

Categoria	**Algoritmos**
Baseado em kernel	kernel k-means, kernel SOM, kernel FCM, SVC, MMC, MKC
Baseado em ensemble	CSPA, HGPA, MCLA, VM, HCE, LAC, WPCK, sCSPA, sMCLA, sHBGPA
Baseado em enxames	ACO_based(LF), PSO_based, SFLA_based, ABC_based
Baseado em mecânica quântica	QC, DQC
Baseado em teoria espectral de grafos	SM, NJW
Baseado em propagação por afinidade	AP
Baseado em densidade e distância	DD
Para dados espaciais	DBSCAN, STING, Wavecluster, CLARANS
Para stream de dados	STREAM, CluStream, HPStream, DenStream
Para dados de grande escala	K-means, BIRCH, CLARA, CURE, DBSCAN, DENCLUE, Wavecluster, FC

Outra técnica muito importante de aprendizado não-supervisionado é a baseada em detecção de *outliers* (ou anomalias), que se refere ao problema de identificar exemplos que não estejam conforme o perfil definido como normal e esperado. Esses padrões são nomeados anomalias/*outliers*. Em uma rede de computadores, um padrão anômalo sugere algum ataque à rede, colocando em risco dados e informações presentes neste meio. Em seu livro, Aggarwal (2017) apresenta

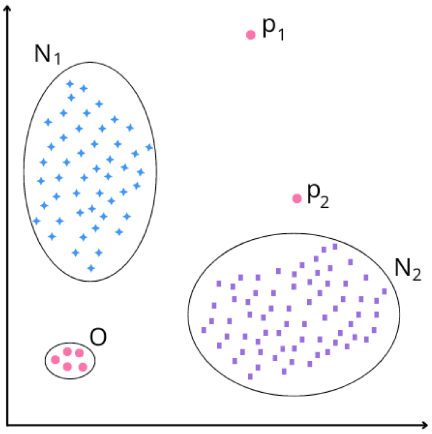
métodos de análise de outliers, conforme a Tabela 4.

Tabela 4 – Métodos e seus análogos não supervisionados na análise de outliers - adaptado de (AGGARWAL, 2017)

Supervisionado	**Não Supervisionado**
KNN	KNN, LOF, LOCI
Árvore de decisão	Isolation Trees
Florestas aleatórias	Isolation Forests
SVM	One-class SVM
Naive Bayes	Baseado em densidade, DBSCAN

A Figura 3 representa um exemplo de anomalias em um conjuntos de dados. Nela existem duas regiões normais, N1 e N2. Pontos que estão distantes dessas regiões (p1 e p2), e também do conjunto de pontos na região O, são categorizados como anomalias.

Figura 3 – Anomalias em um conjunto de dados bidimensionais - adaptado de (PRASAD; ALMANZA-GARCIA; LU, 2009)



2.3 Fluxos Contínuos de Dados

Segundo Santos, Barros e Gonçalves (2016), os fluxos contínuos de dados são ambientes onde os dados fluem continuamente, rapidamente e em grandes quantidades. Um fluxo de dados consiste em um conjunto de registros multidimensionais $X_1...X_k...$, chegando nos momentos de tempo $T_1...T_k....$ Cada X_i é um registro multidimensional contendo d-dimensões que é denotado por $X_i = (x_i^1...x_i^d)$ (AGGARWAL et al., 2003).

Como apontado por MONTEIRO (2022), um fluxo de dados apresenta as seguintes características:

- A quantidade de dados é potencialmente infinita, o que impossibilita armazenar, sendo possível apenas armazenar um pequeno resumo dos dados.
- Devido à velocidade que os dados chegam, eles devem ser processados em tempo real e logo após descartados.
- Com o passar do tempo, a distribuição dos dados pode sofrer alterações, causando a mudança de conceito e tornando os dados irrelevantes ou prejudicar a qualidade/precisão do modelo.

Vários trabalhos da literatura tem usado algoritmos para fluxos contínuos de dados no problema de detecção de intrusão em redes de computadores. A maioria desses trabalhos usam algoritmos de classificação, mas também os algoritmos de detecção de *outliers* têm sido usados.

2.4 Medidas de avaliação

As métricas escolhidas para avaliar os algoritmos de aprendizado de máquina são muito importantes, pois para um dado problemas existem diferentes algoritmos capazes de solucionar, porém precisamos escolher o que tem melhor desempenho e, para isso, utilizamos algumas medidas para realizar a avaliação que torne possível a comparação e seleção do melhor. Para tanto, nesta seção são abor-

dadas algumas das que são frequentemente utilizadas pela comunidade de AM em relação a IDS.

2.4.1 Matriz de confusão

Segundo Faceli et al. (2011), a matriz de confusão ilustra o número de predições corretas e incorretas em cada classe. Dessa forma, uma célula que apresenta o mesmo valor na linha e coluna, representa um acerto do algoritmo, caso contrário um erro.

Tabela 5 – Matriz de confusão

		Classe Predita	
		Positivo	**Negativo**
Classe Real	**Positivo**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Negativo	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Em muitos problemas binários, como classificação de dados, a classe minoritária é tipicamente designada como a classe positiva, enquanto a classe majoritária é considerada a classe negativa. No contexto de Sistemas de Detecção de Intrusões (IDS), essa convenção tem sido adotada. Aqui, o evento de intrusão ou ataque é geralmente designado como a classe positiva, enquanto o tráfego de rede normal é considerado a classe negativa. Essa distinção é essencial, pois o foco principal do IDS é identificar atividades maliciosas ou intrusivas, tornando os ataques o alvo de interesse e, portanto, a classe positiva na maioria dos casos de detecção de intrusões. No entanto é comum também, em problemas multiclasse, considerar cada uma das classes do problema como positiva e as demais como negativas, sendo que para cada classe, uma medida é calculada. Ao fim, a média considerando as n classes do problema é calculada.

Como apresentado na Tabela 5, para cenários de classificação binária, existe também uma nomenclatura usual para cada caso:

- ❏ Se um exemplo da classe ataque é identificado como um ataque, tem-se um **verdadeiro positivo**;
- ❏ No caso de um exemplo da classe normal ser identificado corretamente como um normal, tem-se um **verdadeiro negativo**;
- ❏ Se um exemplo da classe ataque não é identificado como um ataque, tem-se um **falso negativo**;
- ❏ No caso de um exemplo da classe normal ser identificado de forma errada é denominado **falso positivo**.

2.4.2 Métricas de Avaliação

As medidas mais comuns vistas na literatura sobre IDSs são: a precisão, a revocação, e F1-score.

- Precisão: dentre todas as classificações de classe positivo que o modelo fez, quantas estão corretas.

$$Prec = \frac{VP}{VP + FP} \tag{2.1}$$

- *Recall*/Revocação: dentre todas as situações de classe positivo como valor esperado, quantas estão corretas.

$$Revoc = \frac{VP}{VP + FN} \tag{2.2}$$

- F1-Score: média harmônica entre precisão e revocação.

$$F1 = 2 \times \frac{Prec \times Revoc}{Prec + Revoc} \tag{2.3}$$

2.5 Ferramentas para detecção de outlier

2.5.1 MOA *framework*

Abordado em Bifet et al. (2010), o *Massive Online Analysis* (MOA) é um *framework open source* para lidar com fluxos de dados massivos, potencialmente

infinitos e em evolução. Este framework traz o benefício de conseguir comparar diversas bases de dados de cenários do mundo real ou artificial, tendo uma vasta gama de algoritmos, medidas de avaliação e geradores de dados (MONTEIRO, 2022).

Para classificar temos as seguintes implementações: *boosting*, *bagging*, e *Hoeffding Trees*, todos com ou sem o *Naive Bayes* como classificador nas folhas. Já para agrupar, ele implementa: *StreamKM++*, *CluStream*, *ClusTree*, *Den-Stream*, *D-Stream* e *CobWeb*. Para detecção de *outlier*, que é o foco deste trabalho, o suporte em sua biblioteca principal não é tão extenso ou direto, sendo necessário adaptar e explorar alguns algoritmos, ou buscar extensões e contribuições da comunidade. Por fim, o MOA suporta uma interação bidirecional com WEKA, e é lançado sob a licença GNU GPL.

2.5.2 PyOD

O PyOD, abreviação para *Python Outlier Detection*, é uma biblioteca de código aberto que desempenha um papel crucial na identificação de *outliers*. A sua utilidade no contexto deste estudo reside na capacidade de disponibilizar uma ampla gama de algoritmos e técnicas eficazes para detectar essas anomalias. Como destacado em Zhao, Nasrullah e Li (2019a) esta biblioteca contém mais de 20 algoritmos que abrangem tanto técnicas clássicas, como o fator local de anomalia, quanto arquiteturas de redes neurais recentes, como autoencoders ou modelos adversariais. Dessa forma, o PyOD simplifica a implementação e experimentação de tais métodos, tornando a detecção uma tarefa mais acessível e eficaz, o que é de suma importância para a análise de dados e fornecimento de informações cruciais. A aplicação do PyOD contribui significativamente para a qualidade e a precisão da pesquisa e, consequentemente, para a tomada de decisões embasadas em dados confiáveis.

2.5.3 PySAD

PySAD é um framework de código aberto em Python para a detecção de anomalias em dados em fluxo contínuo. É uma ferramenta valiosa para lidar com

dados em tempo real e identificar anomalias com eficácia. Em seu trabalho Yilmaz e Kozat (2020a) abordam vantagens ao utilizar este *framework*, seus componentes modulares permitem a criação de pipelines flexíveis, incluindo pré-processadores, projetores, modelos de detecção de anomalias e pós-processadores. Além disso, calibradores de probabilidade convertem as pontuações dos modelos em estimativas de probabilidade de anomalia. Essa abordagem modular oferece uma solução completa para a detecção eficaz de anomalias em fluxos contínuos de dados, permitindo a personalização de cada etapa do processo conforme necessário.

2.6 Trabalhos correlatos

Em seu trabalho, Galhardi (2017) destaca a elaboração de um modelo de detecção de anomalias em redes de computadores utilizando uma combinação de métodos não-supervisionados. Nele, as amostras de ataque das bases NSL-KDD e ACME-IDE foram escolhidas de maneira aleatória e, no pré-processamento, os atributos das bases foram normalizados. Foi possível concluir que o DBSCAN utilizado para o agrupamento, obteve uma alta taxa de detecção porém apresentou também uma taxa elevada de falsos positivos, e com isso, foi realizada a combinação de DBSCAN com *one-class* SVM e DBSCAN com *k-means*. Os resultados mostraram que o melhor resultado obtido deriva-se da segunda combinação citada anteriormente.

O trabalho de Lima (2021) avalia o uso do algoritmo de detecção de anomalia *Local Outlier Factor* (LOF) na identificação de ataques em redes privadas, como as redes *LAN*, que expõem serviços em uma rede *WAN* ou a própria Internet. A base de dados utilizada no trabalho foi a NSL-KDD adaptada de KDD'99 e no pré-processamento foram selecionados os atributos não numéricos para torná-los numéricos. Como medida de avaliação foram utilizadas F1-*score*, precisão e revocação. Os resultados apontaram bons resultados para detecção de intrusões na rede e que com seu uso é possível detectar ataques ainda não compartilhados na comunidade.

Em Vieira (2022), temos o uso do algoritmo DBSCAN a fim de detectar ataques a redes IoT, a partir da análise dos cabeçalhos dos pacotes coletados na rede.

Foi utilizado também o teste *Page-Hinkley* para a detecção de mudanças abruptas no comportamento do tráfego de rede. As bases de dados utilizadas foram públicas, que continham pacotes provenientes de uma rede IoT com diversos dispositivos domésticos. No pré-processamento, foram extraídos os campos mais significativos dos cabeçalhos dos pacotes de cada protocolo. Como medida de avaliação foram utilizadas F1-*score*, precisão e revocação. Por fim, a partir dos resultados foi observado que há uma alta detecção de ataque e uma baixa taxa de falsos positivos.

Em Zhang et al. (2015), a fim de detectar anomalias de grandes dados de tráfego de rede usando adaptação, há o desenvolvimento de uma nova técnica nomeada A-SPOT. A base utilizada no trabalho foi KDD CUP 1999. Os experimentos mostram que o A-SPOT não é apenas mais eficaz na detecção de anomalias ocultas nos subespaços de alta dimensão de fluxos de dados, mas também produz um nível significativamente mais baixo de falsos positivos do que os métodos existentes de detecção de anomalias. A avaliação do experimento também mostra que o A-SPOT é muito eficiente e escalável para fluxos de dados grandes e de alta dimensão.

Em seu trabalho, Bellaachia e Bhatt (2005) apresentam um sistema de agrupamento em tempo real, chamado *Enhanced Stream Mining* (ESM), que pode analisar informações de pacotes (cabeçalhos e dados) para determinar intrusões. A base utilizada foi KDD CUP 99, e no pré-processamento, foi atribuído uma pontuação ao pacote. Dessa forma, cada pacote foi tratado como um vetor n-dimensional. Como medida de avaliação temos a matriz de confusão. Os resultados destacam a importância de ajustar e otimizar os parâmetros do modelo, como a escolha da mediana, tamanho da janela, etc. Também foi destacado que é fundamental realizar experimentos e ajustes para encontrar o equilíbrio certo entre detecção precisa e tempo de resposta eficiente.

Em Ribeiro, Santos e Nascimento (2021), o objetivo era superar os principais desafios que surgem ao desenvolver um sistema baseado em anomalias usando algoritmos de aprendizado de máquina. Foi utilizado o algoritmo *OutlierDenStream*. No pré-processamento, houve um ajuste manual dos parâmetros lambda e beta, que são parâmetros importantes para o algoritmo. Como medida de avaliação, destaca-se a precisão e revocação. A partir dos experimentos, foi possível

observar que a solução obteve 97,83% de precisão, 99% de revocação, 80% de precisão e 2,3% de taxa de falsos positivos para 10% de ataques DDoS no tráfego normal. Esses resultados mostram que a técnica proposta é eficaz.

No trabalho de Zhu, Wang e Zhu (2019), os autores buscam solucionar a tarefa de detecção de intrusões. Foi utilizado o algoritmo ASWD (*Stream algorithm*), que é um algoritmo de agrupamento de fluxo de dados baseado em janela deslizante atenuada. A base utilizada foi KDD CUP 1999 e as medidas de avaliação foram: acurácia, taxa de informações erradas e taxa de relatórios equivocados. Os resultados da simulação mostram a viabilidade do algoritmo de agrupamento de fluxo de dados baseado em janela deslizante atenuada. Foi destacado também que ele resolve o problema de que a extração de padrões de características do sistema de detecção de intrusões tradicional não consegue se adaptar à detecção de sistemas dinâmicos e em tempo real.

Em seu trabalho Yin et al. (2019) busca atender às necessidades do ambiente de rede dinâmico. A base utilizada foi KDD CUP 99, e no pré-processamento, os atributos categóricos foram removidos para simplificar o cálculo, e também houve a normalização dos atributos. O algoritmo utilizado foi de clusterização. Como medida de avaliação destaca-se a taxa de detecção e taxa de falsos positivos. Por fim, como resultado, foi apresentado um algoritmo aprimorado de agrupamento de fluxo de dados. O modelo pode ser modificado com a mudança do fluxo de dados e detectar comportamentos anômalos em tempo hábil.

A Tabela 6 resume os trabalhos analisados nesta seção:

Tabela 6 – Trabalhos correlatos

Trabalhos	**Base de dados**	**Algoritmos usados**	**Medida de avaliação**	**Conclusões**
Galhardi (2017)	NSL-KDD, ACME-IDE	DBSCAN, DBSCAN e one-class SVM, DBSCAN e K-médias	-	-
Lima (2021)	NSL-KDD, adaptado de KDD'99	Local Outlier Factor (LOF)	F1 Score (precisão e recall)	Possível usar LOF como IDS, mesmo não sendo o melhor modelo.
Vieira (2022)	Dados de IoT domésticos	DBSCAN	F1 Score (precisão e recall)	Alta detecção de ataques, baixa taxa de falsos positivos.
Zhang et al. (2015)	1999 KDD CUP	A-SPOT	-	A-SPOT eficaz, baixo nível de falsos positivos.
Bellaachia e Bhatt (2005)	1999 KDD CUP	-	-	Otimizar parâmetros para equilíbrio entre detecção e tempo de resposta.
Ribeiro, Santos e Nascimento (2021)	-	OutlierDenStream	Recall, Precisão, Acurácia, FPR	Eficácia da técnica proposta.
Zhu, Wang e Zhu (2019)	1999 KDD CUP	ASWD Stream, algoritmo de janela deslizante atenuada	Taxa de detecção, baixas taxas de desinformação e relato incorreto	Viabilidade do ASWD para sistemas dinâmicos e em tempo real.
Yin et al. (2019)	1999 KDD CUP	Clustering Algorithm	Detect Rate and False Positive Rate	Detecção em tempo hábil de comportamentos anômalos.

2.7 Considerações Finais

Neste capítulo foram apresentados os principais trabalhos presentes na literatura, que estão relacionados com o tema principal dessa monografia. Este capítulo também oferece uma visão geral sobre o campo da detecção de anomalias

e identificação de intrusões em redes, e mostra uma visão geral dos principais conceitos de aprendizado de máquina usados neste trabalho. Por meio desta revisão, é possível identificar tanto a diversidade de abordagens e algoritmos aplicados quanto os variados contextos de aplicação, desde redes de computadores tradicionais até ambientes específicos como redes IoT (Internet das Coisas) e sistemas dinâmicos em tempo real.

Alguns trabalhos destacam a importância de algoritmos capazes de se adaptar a ambientes de rede dinâmicos e em tempo real. A capacidade de ajustar modelos em função do fluxo contínuo de dados é essencial para manter a eficácia da detecção de anomalias frente a novos tipos de ataques.

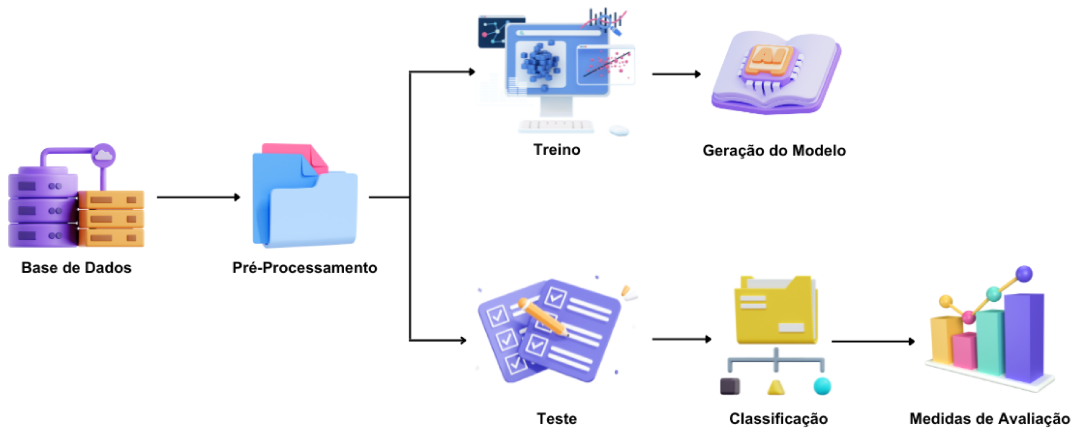
O Capítulo 3 apresentará o método experimental proposto para detecção de intrusão em redes, utilizando o conjunto de dados CICIDS2017. Além disso, são detalhadas as etapas necessárias para a realização dos experimentos e análise dos resultados.

3 Método para Detecção de Intrusão em Redes de Computadores

Conforme apresentado anteriormente, o objetivo principal deste trabalho é avaliar comparativamente o desempenho preditivo de técnicas de aprendizado não-supervisionado, em especial técnicas de detecção de *outliers*, na tarefa de detecção de intrusão em redes de computadores. Este capítulo visa apresentar o método proposto para detecção de intrusão em redes de computadores usando algoritmos de detecção de outliers.

A Figura 4, mostra a visão geral da proposta. Cada uma das etapas da proposta será apresentada em mais detalhes nas seções subsequentes.

Figura 4 – Visão geral do método proposto para detecção de intrusão



A Seção 3.1 aborda detalhadamente as característica da base de dados CICIDS2017, utilizada nos experimentos deste trabalho. A Seção 3.2 descreve o processo de pré-processamento aplicado à base de dados selecionada. A Seção 3.3 descreve os algoritmos utilizados bem como as características do *framework* MOA e do Python. A Seção 3.4 descreve o método utilizado para separação em treino e

teste. Por fim, a Seção 3.5 descreve as medidas de avaliação utilizadas para avaliar os experimentos.

3.1 Seleção da Base de Dados

A escolha da base de dados é de suma importância para que os experimentos simulem as condições mais próximas à realidade operacional. Para atingir esse objetivo, se tornou crucial à busca por uma base que refletisse de forma abrangente o cenário de ameaças, contendo uma diversidade de ataques e padrões de tráfego normal, e que apresentasse os rótulos de suas instâncias. Após análise de diversas bases, a CICIDS2017 foi escolhida como a base de dados principal para todos os experimentos, devido à sua integralidade em atender às exigências específicas para os experimentos, além de ser amplamente utilizada em trabalhos relacionados a IDS. Esta base foi concebida para abranger uma variedade de ameaças cibernéticas, oferecendo um conjunto de dados abrangente e representativo, disponibilizados em um arquivo *Comma-Separated Values* (CSV). Além disso, a base contém informações sobre fluxos de rede, definidos como sequências de pacotes que compartilham características similares, tais como endereços IP de origem e destino, portas e protocolos.

Capturado ao longo de cinco dias (3 a 7 de Julho de 2017), o conjunto de dados inclui tanto tráfego normal quanto de ataques, abrangendo instâncias de Ataque de Força Bruta FTP e SSH, Ataque de Negação de Serviço, Ataque *Heartbleed*, Ataque Web, Ataque de Infiltração, *Botnet* e Ataque de Negação de Serviço Distribuído. O primeiro dia de captura foi excluído das análises realizadas neste trabalho, pois não possuía dados do tipo Ataque, somente Normal. Nos demais dias da semana, foram observados ataques, distribuídos em dois períodos distintos: pela manhã e à tarde. Os detalhes sobre a distribuição desses ataques podem ser encontrados na Tabela 7.

Tabela 7 – Distribuição dos ataques por dia da semana e período do dia

Dia da Semana	**Período**	**Ataque**
Terça-feira	Manhã	Força Bruta
	Tarde	Força Bruta
Quarta-feira	Manhã	DoS
	Tarde	HeartBleed
Quinta-feira	Manhã	Web
	Tarde	Infiltração
Sexta-feira	Manhã	BotNet
	Tarde	DDoS e Escaneamento de Portas

Para a realização dos experimentos, foram selecionados os dados coletados na Terça-Feira e na Sexta-Feira. Estes dias foram escolhidos pelo fato de serem distintos entre si, onde na Terça-Feira tem-se somente um tipo de ataque, e na Sexta-Feira tem-se dois distintos. Além disso, eles foram escolhidos também pelo espaçamento temporal de três dias entre eles e pela dificuldade de classificação das bases. A base de Sexta-feira é considerada uma base mais fácil de classificar do que a base de Terça-feira devido ao menor desbalanceamento de classes presente nela.

3.2 Pré-Processamento

A etapa de pré-processamento desempenha um papel crucial no ciclo de Descoberta de Conhecimento em Bancos de Dados (KDD), sendo essencial para melhorar a qualidade e utilidade dos dados antes da aplicação de técnicas de análise ou modelagem. Durante essa fase, diversas atividades podem ser realizadas para garantir que os dados estejam prontos para serem utilizados nas etapas subsequentes do processo. O pré-processamento foi realizado da seguinte forma:

- ❏ Remoção de um atributo duplicado *Fwd Header Length* na base;

- ❏ Remoção dos atributos *Destination Port e CWE Flag Count*;
- ❏ Soma dos atributos *Total Fwd Packets e Total Backward Packets*;
- ❏ Soma dos atributos *Total Length of Fwd Packets e Total Length of Bwd Packets*;
- ❏ Remoção de todas as entradas que continham algum atributo com valor *null*;
- ❏ Remoção de todas as entradas que continham algum atributo com valor infinito;
- ❏ Conversão do atributo 'Label' do tipo string para numérico.

A Tabela 8 mostra o nome e o tipo de cada um dos atributos usados na etapa de detecção de *outliers*.

Tabela 8 – Atributos da Base de Dados

Índice	**Atributo (Tipo)**
1	Flow Duration (numeric)
2	Fwd Packet Length Max (numeric)
3	Fwd Packet Length Min (numeric)
4	Fwd Packet Length Mean (numeric)
5	Fwd Packet Length Std (numeric)
6	Bwd Packet Length Max (numeric)
7	Bwd Packet Length Min (numeric)
8	Bwd Packet Length Mean (numeric)
9	Bwd Packet Length Std (numeric)
10	Flow Bytes/s (numeric)
11	Flow Packets/s (numeric)
12	Flow IAT Mean (numeric)
13	Flow IAT Std (numeric)
14	Flow IAT Max (numeric)
15	Flow IAT Min (numeric)
16	Fwd IAT Total (numeric)

Índice	**Atributo (Tipo)**
17	Fwd IAT Mean (numeric)
18	Fwd IAT Std (numeric)
19	Fwd IAT Max (numeric)
20	Fwd IAT Min (numeric)
21	Bwd IAT Total (numeric)
22	Bwd IAT Mean (numeric)
23	Bwd IAT Std (numeric)
24	Bwd IAT Max (numeric)
25	Bwd IAT Min (numeric)
26	Fwd PSH Flags (numeric)
27	Bwd PSH Flags (numeric)
28	Fwd URG Flags (numeric)
29	Bwd URG Flags (numeric)
30	Fwd Header Length (numeric)
31	Bwd Header Length (numeric)
32	Fwd Packets/s (numeric)
33	Bwd Packets/s (numeric)
34	Min Packet Length (numeric)
35	Max Packet Length (numeric)
36	Packet Length Mean (numeric)
37	Packet Length Std (numeric)
38	Packet Length Variance (numeric)
39	FIN Flag Count (numeric)
40	SYN Flag Count (numeric)
41	RST Flag Count (numeric)
42	PSH Flag Count (numeric)
43	ACK Flag Count (numeric)
44	URG Flag Count (numeric)
45	ECE Flag Count (numeric)
46	Down/Up Ratio (numeric)
47	Average Packet Size (numeric)

Índice	**Atributo (Tipo)**
48	Avg Fwd Segment Size (numeric)
49	Avg Bwd Segment Size (numeric)
50	Fwd Avg Bytes/Bulk (numeric)
51	Fwd Avg Packets/Bulk (numeric)
52	Fwd Avg Bulk Rate (numeric)
53	Bwd Avg Bytes/Bulk (numeric)
54	Bwd Avg Packets/Bulk (numeric)
55	Bwd Avg Bulk Rate (numeric)
56	Subflow Fwd Packets (numeric)
57	Subflow Fwd Bytes (numeric)
58	Subflow Bwd Packets (numeric)
59	Subflow Bwd Bytes (numeric)
60	Init_Win_bytes_forward (numeric)
61	Init_Win_bytes_backward (numeric)
62	act_data_pkt_fwd (numeric)
63	min_seg_size_forward (numeric)
64	Active Mean (numeric)
65	Active Std (numeric)
66	Active Max (numeric)
67	Active Min (numeric)
68	Idle Mean (numeric)
69	Idle Std (numeric)
70	Idle Max (numeric)
71	Idle Min (numeric)
72	Soma Fwd Packets (numeric)
73	Soma Length of Fwd Packets (numeric)
74	Label (numeric)

3.3 Algoritmos de Detecção de *Outliers*

Existem vários algoritmos eficazes para detecção de *outliers*, propostos na literatura. Cada algoritmo possui abordagens distintas para identificar e caracterizar *outliers*, sendo escolhido com base na natureza dos dados e nos objetivos da análise. Muitos algoritmos tradicionais foram inicialmente desenvolvidos com foco no processamento em lote (*batch processing*), dentre eles, foram selecionados devido sua forte presença na literatura, os seguintes:

- Isolation Forest (LIU; TING; ZHOU, 2008)
- Local Outlier Factor (LOF) (BREUNIQ et al., 2000)
- One-Class SVM (LI et al., 2003)

Assim como mencionado por Pereira (2017), alguns exemplos de algoritmos específicos para detecção de *outliers* em fluxo contínuo de dados propostos na literatura são: iLOF (SALEHI et al., 2015), Abstract-C (YANG; RUNDENSTEINER; WARD, 2009), STORM (ANGIULLI; FASSETTI, 2010) e MCOD (KONTAKI et al., 2011).

O desenvolvimento de aplicações para processamento em fluxo contínuo de dados apresenta desafios significativos que vão além da simples implementação de algoritmos. A aplicação prática de algoritmos específicos para tarefas de detecção de *outliers* em fluxos contínuos esbarra na falta de ferramentas especializadas ou integração direta em certas plataformas. Neste trabalho, inicialmente foi utilizado o *framework* MOA, que contém os seguinte algoritmos em seu sistema: *AbstractC, ApproxSTORM, ExactSTORM, AnyOut, MCOD* e *SimpleCOD*. Porém, devido a falta de medidas de avaliação necessárias para avaliar a qualidade dos algoritmos, não foi possível continuar utilizando o MOA.

Com isso, uma nova ferramenta foi selecionada para os experimentos: Python. Segundo Robinson (2017), a linguagem de programação Python está ganhando uma popularidade tremenda entre cientistas de dados e desenvolvedores de software. É uma escolha popular, graças à sua natureza interativa e ao ecossistema

crescente de bibliotecas científicas. Neste contexto, foram empregadas, principalmente, as seguintes: *Pandas, NumPy, Matplotlib* e *Scikit-Learn*.

O *Scikit-learn* fornece implementações de muitos algoritmos conhecidos de aprendizado de máquina, ao mesmo tempo em que mantém uma interface fácil de usar integrada de forma coesa com a linguagem Python (PEDREGOSA et al., 2011).

Para os experimentos com fluxo de dados, foi utilizado *PySAD*, que incorpora diversos métodos de ponta para detecção de anomalias em fluxos de dados. O *framework* oferece um conjunto completo de ferramentas para projetar experimentos de detecção de anomalias, abrangendo desde projetores até calibradores de probabilidade (YILMAZ; KOZAT, 2020b). Além disso, também foi usado o *PyOD*, que oferece acesso a uma ampla variedade de algoritmos de detecção de *outliers* (ZHAO; NASRULLAH; LI, 2019b). O algoritmo escolhido para os experimentos *stream* foi o *Local Outlier Factor* (LOF). No caso deste modelo, foi necessário configurar um parâmetro *threshold*, que é um limiar para separar normal de ataque, pois neste modelo temos um retorno que representa o score de anomalia do objeto em análise.

3.4 Método para Separação dos Dados em Treino e Teste

No cenário *batch*, a separação da base de dados em treino e teste foi feita da seguinte forma:

- Divisão da base em dois vetores, sendo um de exemplos normais e outro de ataques;
- Dos normais, 70% (aleatório) foi reservado para treino;
- Para o teste, 30% (aleatório) do vetor de normais, concatenado com 100% do vetor de ataques;

No cenário *stream*, o modelo inicial foi treinado com os cem primeiros exemplos do vetor de treino.

3.5 Avaliação

A avaliação rigorosa de modelos é um componente crucial na condução de pesquisas em aprendizado de máquina, desempenhando um papel significativo na compreensão e otimização de algoritmos. Dentre as métricas, a precisão, revocação e *F1-score* são medidas amplamente utilizadas na literatura científica.

A precisão é uma medida essencial que avalia a proporção de predições corretas, tanto positivas quanto negativas, em relação ao total de predições. Esta métrica é especialmente relevante em contextos onde a minimização de falsos positivos é crítica, como em diagnósticos médicos ou sistemas de segurança, sendo vital para garantir resultados confiáveis.

Por sua vez, a revocação emerge como uma métrica essencial ao focar na capacidade do modelo em identificar corretamente todas as instâncias positivas. Ao avaliar a proporção de predições positivas corretas em relação ao total de instâncias positivas, a revocação oferece uma perspectiva valiosa sobre a sensibilidade do modelo. Em sistemas de detecção de fraudes, a revocação desempenha um papel crucial na avaliação do desempenho do modelo, garantindo abordagens robustas e confiáveis.

O *F1-score*, é uma métrica que combina precisão e revocação, fornecendo uma medida equilibrada que é particularmente relevante em situações em que é desejável um compromisso entre a minimização de falsos positivos e falsos negativos. É uma métrica valiosa para avaliar o desempenho global do modelo.

3.6 Considerações Finais

Neste capítulo, foi apresentada a base de dados utilizada no trabalho (CICIDS2017), selecionada por sua variedade de ataques e representatividade de tráfego real. Este capítulo também descreve as etapas de pré-processamento e detalha a seleção de algoritmos, como *Isolation Forest*, LOF e *One-Class SVM*, a fim de evidenciar a preparação e ajustes dos dados para uma análise eficaz.

O Capítulo 4 apresentará os resultados dos experimentos realizados, utilizando o conjunto de dados CICIDS2017 após a aplicação dos pré-processamentos e

dos algoritmos que foram apresentados neste capítulo. Além disso, será detalhada a análise de desempenho de cada um deles.

4 Resultados dos Experimentos para Detecção de Intrusão

Os resultados dos experimentos para avaliar o método proposto são apresentados e discutidos neste capítulo. A Seção 4.1 mostra um resumo da composição da base de dados selecionada (CICIDS2017). Na Seção 4.2 é apresentada a base de dados utilizada nos experimentos após o pré-processamento. A Seção 4.3 apresenta os resultados divididos por algoritmo e classe de interesse, e também as medidas de avaliação e seus respectivos valores com uma discussão sobre os resultados obtidos.

4.1 Base de Dados

A Tabela 9 resume os detalhes da base selecionada conforme mencionado anteriormente, na Seção 3.1. Nela encontra-se as seguintes informações: qual o arquivo de dados, o número total de instâncias deste, o número de instâncias que são Ataques e as que são Normais, e, por fim, o número de atributos total desta base antes de qualquer tipo de tratamento.

Tabela 9 – Resumo da Base CICIDS2017

Dia da semana/ Período	**Total**	**Ataques**	**Normais**	**Nro. de Atributos**
Terça-Feira/ Dia todo	445909	13835	432074	79
Sexta-Feira/ Dia todo	703245	288923	414322	79

4.2 Resultado do Pré-Processamento

Na Tabela 10, tem-se as mesmas informações da tabela anterior, porém nesta tabela os valores sofrem alterações em razão da aplicação do pré-processamento

na base. É possível observar que houve uma redução de duzentos e sessenta e quatro instâncias na base de terça-feira, e de quinhentos e vinte e sete na de sexta-feira. Isso se deve ao fato de que no pré-processamento foi aplicado uma exclusão de instâncias que continham atributos nulos e/ou infinitos. Além disso, o número de atributos foi reduzido em cinco unidades, pois conforme explicado na Seção 3.2, um atributo estava duplicado na base, e, por isso, foi removido, dois atributos não necessários também foram removidos e, por fim, houve a junção (soma) de dois pares de atributos.

Tabela 10 – Sumário da Base CICIDS2017 após Pré-Processamento

Dia da semana/ Período	**Total**	**Ataques**	**Normais**	**Nro. de Atributos**
Terça-Feira/ Dia todo	445645	13832	431813	74
Sexta-Feira/ Dia todo	702718	288785	413933	74

4.3 Análise de desempenho dos algoritmos de detecção de *outliers*

A Tabela 11 apresenta um resumo da divisão aplicada as instâncias para treino e teste. Considerando o número total de instâncias Normais, 70% foi destinado ao treino, e o teste abriga os 30% restantes. No teste são também acrescentadas todas as instâncias do tipo Ataque. Assim, durante o treinamento, o modelo não tem acesso a nenhuma instância do tipo ataque.

Tabela 11 – Número de instâncias de treino e teste

	Dia da semana/ Período	**Treino**	**Teste**
Nro. de instâncias	Terça-feira/ Dia todo	302269	143376
	Sexta-feira/ Dia todo	289753	412965

4.3.1 Resultados da Base de Terça-Feira

A Tabela 12 apresenta os resultados dos experimentos de quatro métodos diferentes de detecção de *outliers* efetuados usando a base de dados de terça-feira. Nesta tabela, primeiramente a classe normal foi considerada a positiva e as medidas precisão, revocação e F1-score foram calculadas. A seguir, a classe ataque (minoritária), foi considerada a positiva e as mesmas medidas foram calculadas. Este segundo cenário é o comumente usado na literatura de IDS. Além disso a matriz de confusão de cada experimento pode ser vista nos apêndices A.1, B.1, C.1 e D.1.

Tabela 12 – Resultados base de terça-feira

	Classe Normal			**Classe Ataque**		
	Precisão	**Revocação**	**F1**	**Precisão**	**Revocação**	**F1**
Isolation Forest	0,96	0,55	0,70	0,16	0,79	0,26
LOF - Novelty	0,92	0,83	0,87	0,17	0,31	0,22
SVM - One-Class	0,95	0,90	0,92	0,36	0,52	0,43
LOF Stream	0,91	0,76	0,83	0,12	0,31	0,17

Ao analisar os resultados da classe ataque, pode-se destacar:

O ***Isolation Forest*** tem a maior revocação, mostrando que ele é bom em identificar ataques. Porém sua precisão é baixa, o que indica que possui um grande número de FP, ou seja, muitas instâncias da classe normal estão sendo incorretamente classificadas como ataque. Aqui vê-se que o algoritmo não aprendeu bem a classe normal.

O ***LOF - Novelty*** e o ***LOF Stream*** apresentam resultados semelhantes, as três medidas para a classe ataque são baixas, o que mostra que não são tão eficazes para a classe em questão. Como a precisão é apenas 0,31, significa que apenas 31% dos ataques foram classificados como tal, sendo que os demais foram confundidos com a classe normal. Por outro lado, as medidas são boas para a classe normal, o que mostra que ele se especializou na classe normal. Aqui é importante destacar que o modelo inicial do *LOF Stream* é treinado com apenas 100 instâncias, mas o modelo é atualizado ao longo do fluxo.

O ***SVM - One-Class*** apresenta um desempenho mais equilibrado, sua precisão é a mais alta dentre os métodos, tem uma revocação que indica que ele consegue identificar mais de metade dos ataques reais. E com isso, um F1-score maior quando comparado aos demais algoritmos. A precisão e revocação para a classe normal são melhores que a dos outros algoritmos.

4.3.2 Resultados da Base de Sexta-Feira

A Tabela 13 apresenta os resultados de desempenho dos mesmos quatro métodos de detecção de anomalias usados no experimento anterior, mas com um conjunto de dados diferente, referente à sexta-feira. Aqui é importante destacar que a base de sexta-feira tem muito mais ataques que a base de terça. Enquanto na base de terça existem 13835 ataques, a base de sexta tem 288923.

A matriz de confusão de cada experimento pode ser vista nos apêndices A.2, B.2, C.2 e D.2.

A mesma estratégia de avaliação foi utilizada. Ao analisar os resultados da classe ataque, destaca-se que:

Tabela 13 – Resultados base de sexta-feira

	Classe Normal			**Classe Ataque**		
	Precisão	**Revocação**	**F1**	**Precisão**	**Revocação**	**F1**
Isolation Forest	0,28	0,56	0,37	0,67	0,39	0,50
LOF - Novelty	0,38	0,83	0,52	0,85	0,41	0,55
SVM - One-Class	0,32	0,90	0,48	0,82	0,20	0,32
LOF - Stream	0,32	0,75	0,45	0,75	0,33	0,45

O ***Isolation Forest*** apresenta valores medianos para as três medidas avaliadas considerando a classe ataque como positiva. Neste caso, a precisão é maior que a revocação, e tudo isso origina um F1-Score de 0,50. No entanto, os valores para a classe normal estão bem ruins, ou seja, esta classe não foi bem aprendida.

O ***LOF - Novelty*** se destaca como o algoritmo que alcançou a maior precisão, com um valor de 0,85. Além disso, apesar de baixa, a revocação apresentada por ele é a maior dentre os quatro algoritmos. Este modelo apresenta o mais alto F1-Score tanto para a classe positiva quanto para a negativa, embora o valor obtido não seja considerado bom para o problema, já que fica em torno de 0,5.

O ***SVM - One-Class*** mostrou uma precisão alta, o que significa que a maioria das instâncias classificadas como ataques eram realmente composta por ataques. No entanto, sua revocação foi baixa, o que indica que embora seja preciso, ele perde muitos ataques verdadeiros.

O ***LOF - Stream*** apresenta resultados similares ao *LOF - Novelty*.

4.3.3 Análise geral dos experimentos

Os resultados dos experimentos indicam que não houve um algoritmo que obteve melhores resultados com as duas bases de dados. O *SVM - One-class* se mostrou melhor na base de terça-feira, mas não obteve o melhor desempenho na base de sexta-feira. Considerando a classe ataque como a positiva, pode-se concluir que os resultados não são satisfatórios, já que o melhor resultado de F1-score foi 0,5. Além disso, os resultados de revocação estão bem baixos, indicando que a maioria dos ataques não são identificados como tal. Nos poucos cenários em que obtêm-se valores altos de revocação, percebe-se que a classificação das instâncias normais fica prejudicada, o que geraria uma grande quantidade de falsos alarmes. Nesse caso, muitas instâncias normal seriam incorretamente classificadas como ataque.

Considerando as características dos algoritmos, é importante destacar que o algoritmo de fluxos contínuos de dados não gerou resultados melhores que o seus competidores do cenário *batch*. No entanto, o número de exemplos usados para a geração do modelo inicial no algoritmo de fluxos é bem menor que do *batch*.

Algumas das possíveis motivações para o baixo desempenho dos algorit-

mos é que a base de dados possui uma alta dimensionalidade. Vários trabalhos da literatura tentam eliminar parte dos atributos da base a fim de reduzir a dimensionalidade da mesma. Outro ponto é que os parâmetros dos algoritmos não foram explorados, sendo que os valores padrão foram usados. Por fim, uma análise do impacto das instâncias escolhidas no treinamento e o desempenho do modelo poderia ajudar na construção de modelos mais generalistas. Também seria interessante estudar os demais dias da semana da base CICIDS2017, bem como outras bases de dados da literatura.

5 Conclusão

Neste trabalho, foi apresentada uma avaliação de desempenho para métodos de detecção de intrusão em redes de computadores. Para isso, foi analisado o desempenho preditivo de quatro algoritmos, utilizando diferentes medidas de avaliação. Dentre os quatro, três deles são algoritmos para cenários *batch* e um deles para fluxos de dados. A fim de avaliar quais tipos de ataque as técnicas analisadas conseguem identificar, foram realizados experimentos em duas sub-bases distintas da base CICIDS2017. As principais características dos experimentos realizados foram:

- Um aspecto notável é a variação no desempenho dos métodos entre os dois dias. Na sexta-feira, os modelos tendem a performar melhor na detecção de ataques do que na detecção de normais, ao contrário do que foi observado na terça-feira. Isso se deve ao fato de que as bases são distintas entre si, e a de sexta-feira apresenta um maior balanceamento entre as classes.
- Não houve um algoritmo que obteve melhor desempenho considerando as duas bases de dados.
- *SVM - One-Class* teve um desempenho diferente entre os dias, com uma queda significativa na revocação de terça para sexta-feira. Isso mostra que o tipo de ataque de terça e sexta tem impactos diferentes neste algoritmo.
- Nenhum modelo conseguiu uma revocação alta para ataques, o que sugere um desafio em capturar todos os ataques verdadeiros sem gerar muitos falsos positivos.

A detecção eficaz de ataques em sistemas de informação é crucial e desafiadora, dada a constante evolução das ameaças. A análise dos resultados demonstra a importância de selecionar e ajustar cuidadosamente os métodos de detecção de anomalias às características específicas dos dados e às variações ao longo do tempo.

Nenhum método se mostrou superior em todos os aspectos, mostrando a necessidade de estudos mais aprofundados sobre o assunto. Uma proposta possível seria uma abordagem híbrida para a segurança cibernética, combinando os pontos fortes de vários métodos para alcançar uma detecção de ataques mais confiável.

5.1 Trabalhos Futuros

A partir dos experimentos realizados e dos resultados obtidos nesta monografia, diversas oportunidades para pesquisas futuras foram identificadas:

- ❏ Estudar a utilização de uma abordagem híbrida que combine os pontos positivos de diferentes modelos de detecção de anomalia;
- ❏ Realizar testes extensivos sob diferentes condições de tráfego de rede e diferentes tipos de ataques para avaliar a robustez e a consistência dos modelos;
- ❏ Ampliação dos experimentos em outros conjuntos de dados;
- ❏ Experimentar variações nos parâmetros dos algoritmos bem como na escolha dos exemplos usados no treinamento;
- ❏ Analisar se com o atributo *Destination Port* os resultados melhoram;
- ❏ Realizar experimentos direcionados a um único tipo de ataque e verificar os resultados;
- ❏ Reduzir o número de atributos;
- ❏ Investigar o uso de técnicas que permitam reduzir o número de atributos da base a fim de melhorar o desempenho
- ❏ Estudo de novos algoritmos de fluxos contínuos de dados bem como o uso de medidas e métodos apropriados para testá-los.

Referências

AGGARWAL, C. C. Outlier analysis. **Outlier Analysis**, Springer International Publishing, 2017. Citado 3 vezes nas páginas 7, 23 e 24.

AGGARWAL, C. C.; YU, P. S.; HAN, J.; WANG, J. A framework for clustering evolving data streams. **Proceedings 2003 VLDB Conference: 29th International Conference on Very Large Databases (VLDB)**, Morgan Kaufmann, p. 81–92, 1 2003. Citado na página 25.

ANGIULLI, F.; FASSETTI, F. Distance-based outlier queries in data streams: The novel task and algorithms. **Data Mining and Knowledge Discovery**, Springer, v. 20, p. 290–324, 3 2010. ISSN 13845810. Disponível em: <https://link.springer.com/article/10.1007/s10618-009-0159-9>. Citado na página 40.

BACE, R.; MELL, P. Nist special publication on intrusion detection systems intrusion detection systems. 2001. Citado na página 17.

BELLAACHIA, A.; BHATT, R. An enhanced stream mining approach for network anomaly detection. **https://doi.org/10.1117/12.611168**, SPIE, v. 5812, p. 342–351, 3 2005. ISSN 0277786X. Disponível em: <https://www.spiedigitallibrary.org/conference-proceedings-of-spie/5812/0000/An-enhanced-stream-mining-approach-for-network-anomaly-detection/10.1117/12.611168.fullhttps://www.spiedigitallibrary.org/conference-proceedings-of-spie/5812/0000/An-enhanced-stream-mining-approach-for-network-anomaly-detection/10.1117/12.611168.short>. Citado 2 vezes nas páginas 30 e 32.

BHUYAN, M. H.; BHATTACHARYYA, D. K.; KALITA, J. K. Network traffic anomaly detection and prevention. **Network Traffic Anomaly Detection and Prevention: Concepts, Techniques, and Tools**, p. 115–169, 2017. Disponível em: <http://link.springer.com/10.1007/978-3-319-65188-0>. Citado na página 13.

BIFET, A.; HOLMES, G.; PFAHRINGER, B.; KRANEN, P.; KREMER, H.; JANSEN, T.; SEIDL, T. Moa: Massive online analysis, a framework for stream classification and clustering. v. 11, p. 44, 2010. Citado na página 27.

BREUNIQ, M. M.; KRIEGEL, H. P.; NG, R. T.; SANDER, J. Lof. **ACM SIGMOD Record**, ACMPUB27New York, NY, USA, v. 29, p. 93–104, 5

2000. ISSN 01635808. Disponível em: <https://dl.acm.org/doi/10.1145/335191.335388>. Citado na página 40.

CISCO. White paper cisco public. 2018. Citado na página 16.

D'ANDRADA, L. F. P. Um sistema de detecção de intrusão de tempo real e baseado em anomalias para redes can automotivas. Universidade Federal de Pernambuco, 2 2020. Disponível em: <https://repositorio.ufpe.br/handle/123456789/37906>. Citado na página 13.

EFFENDY, D. A.; KUSRINI, K.; SUDARMAWAN, S. Classification of intrusion detection system (ids) based on computer network. **Proceedings - 2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering, ICITISEE 2017**, Institute of Electrical and Electronics Engineers Inc., v. 2018-January, p. 90–94, 7 2017. Citado na página 17.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. d. L. F. d. **Inteligência artificial: uma abordagem de aprendizado de máquina**. [S.l.]: LTC, 2011. Citado na página 26.

FILHO, C. H. P. Técnicas de aprendizado não supervisionado baseadas no algoritmo da caminhada do turista. Biblioteca Digital de Teses e Dissertações da Universidade de São Paulo, 11 2018. Disponível em: <http://www.teses.usp.br/teses/disponiveis/82/82131/tde-20082018-122603/>. Citado na página 20.

FORTINET. **Fortinet relata que a América Latina foi alvo de mais de 360 bilhões de tentativas de ataques cibernéticos em 2022**. 2023. Disponível em: <https://www.fortinet.com/br/corporate/about-us/newsroom/press-releases/2023/fortiguard-labs-reports-destructive-wiper-malware-increases-over-50-percent>. Citado na página 12.

GALHARDI, V. V. Detecção adaptativa de anomalias em redes de computadores utilizando técnicas não supervisionadas. Universidade Estadual Paulista (Unesp), 1 2017. Disponível em: <https://repositorio.unesp.br/handle/11449/148746>. Citado 3 vezes nas páginas 14, 29 e 32.

GAMA, J. Knowledge discovery from data streams. **Knowledge Discovery from Data Streams**, CRC Press, p. 1–235, 1 2010. Disponível em: <https://www.taylorfrancis.com/books/mono/10.1201/EBK1439826119/knowledge-discovery-data-streams-joao-gama>. Citado na página 14.

IBM. **Cost of a data breach 2022 | IBM**. 2022. Disponível em: <https://www.ibm.com/reports/data-breach>. Citado na página 13.

KEMP, S. **Digital 2023 April Global Statshot Report — DataReportal – Global Digital Insights**. 2023. Disponível em: <https://datareportal.com/reports/digital-2023-april-global-statshot>. Citado na página 12.

KONTAKI, M.; GOUNARIS, A.; PAPADOPOULOS, A. N.; TSICHLAS, K.; MANOLOPOULOS, Y. Continuous monitoring of distance-based outliers over data streams. **Proceedings - International Conference on Data Engineering**, p. 135–146, 2011. ISSN 10844627. Citado na página 40.

KUROSE, J. F.; ROSS, K. W. Redes de computadores e a internet : uma abordagem top-down. Pearson, 2013. Citado 2 vezes nas páginas 12 e 16.

LI, K. L.; HUANG, H. K.; TIAN, S. F.; XU, W. Improving one-class svm for anomaly detection. **International Conference on Machine Learning and Cybernetics**, v. 5, p. 3077–3081, 2003. Citado na página 40.

LIAO, H.-J.; LIN, C.-H. R.; LIN, Y.-C.; TUNG, K.-Y. Intrusion detection system: A comprehensive review. 2012. Disponível em: <http://dx.doi.org/10.1016/j.jnca.2012.09.004>. Citado 2 vezes nas páginas 7 e 18.

LIMA, L. da S. Sistema de detecção de intrusão baseado no algoritmo local outlier factor (lof). Universidade Tecnológica Federal do Paraná, 8 2021. Disponível em: <http://repositorio.utfpr.edu.br:8080/jspui/handle/1/29982>. Citado 3 vezes nas páginas 14, 29 e 32.

LIU, F. T.; TING, K. M.; ZHOU, Z. H. Isolation forest. **Proceedings - IEEE International Conference on Data Mining, ICDM**, p. 413–422, 2008. ISSN 15504786. Citado na página 40.

LUDERMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. **Estudos Avancados**, Instituto de Estudos Avancados da Universidade de Sao Paulo, v. 35, p. 85–94, 1 2021. ISSN 18069592. Citado na página 18.

MAFRA, P. M.; FRAGA, J. da S.; MOLL, V.; SANTIN, A. O. Polvo-iids: Um sistema de detecção de intrusão inteligente baseado em anomalias. In: SBC. **Anais do VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais**. [S.l.], 2008. p. 61–72. Citado na página 13.

MILLIGAN, G. W. Clustering validation: Results and implications for applied analyses. **Clustering and Classification**, WORLD SCIENTIFIC, p. 341–375, 1 1996. Citado na página 21.

MITCHELL, T. M. et al. **Machine learning**. [S.l.]: McGraw-hill New York, 2007. v. 1. Citado 2 vezes nas páginas 18 e 20.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas inteligentes-Fundamentos e aplicações**, v. 1, n. 1, p. 32, 2003. Citado na página 18.

MONTEIRO, P. M. Propostas de métodos baseados em co-op training para aprendizado semi-supervisionado em fluxos contínuos de dados. Universidade Federal de Pernambuco, 10 2022. Disponível em: <https://repositorio.ufpe.br/handle/123456789/49433>. Citado 2 vezes nas páginas 25 e 28.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011. Citado na página 41.

PEREIRA, M. A. Arcabouço para detecção online de outliers para algoritmos de agrupamento em fluxos contínuos de dados. Universidade Federal de Viçosa, 7 2017. Disponível em: <https://locus.ufv.br//handle/123456789/17942>. Citado na página 40.

PRASAD, N. R.; ALMANZA-GARCIA, S.; LU, T. T. Anomaly detection. **Computers, Materials and Continua**, v. 14, p. 1–22, 2009. ISSN 15462218. Disponível em: <http://doi.acm.org/10.1145/1541880.1541882>. Citado 2 vezes nas páginas 6 e 24.

RESEARCH, C. P. **Check Point Research Reports a 38Global Cyberattacks - Check Point Blog**. 2023. Disponível em: <https://blog.checkpoint.com/2023/01/05/38-increase-in-2022-global-cyberattacks/>. Citado na página 14.

RIBEIRO, A. d. R. L.; SANTOS, R. Y. C.; NASCIMENTO, A. C. A. Anomaly detection technique for intrusion detection in sdn environment using continuous data stream machine learning algorithms. In: IEEE. **2021 IEEE international systems conference (SysCon)**. [S.l.], 2021. p. 1–7. Citado 2 vezes nas páginas 30 e 32.

ROBINSON, D. **The Incredible Growth of Python - Stack Overflow**. 2017. Disponível em: <https://stackoverflow.blog/2017/09/06/incredible-growth-python/>. Citado na página 40.

ROCHA, M. S. Identificação de ataques não-conhecidos em sistemas de detecção de intrusão baseados em anomalia. Universidade Federal de Uberlândia, 6 2023. Disponível em: <https://repositorio.ufu.br/handle/123456789/38147>. Citado na página 13.

SAH, S. Machine learning: A review of learning types. Preprints, 7 2020. Disponível em: <https://www.preprints.org/manuscript/202007.0230/v1>. Citado 3 vezes nas páginas 6, 19 e 20.

SALEHI, M.; LECKIE, C.; BEZDEK, J. C.; VAITHIANATHAN, T. Local outlier detection for data streams in sensor networks: Revisiting the utility problem invited paper. **2015 IEEE 10th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2015**, Institute of Electrical and Electronics Engineers Inc., 5 2015. Citado na página 40.

SANTOS, S. G. T. C.; BARROS, R. S. M.; GONçALVES, P. M. Optimizing the parameters of drift detection methods using a genetic algorithm. **Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI**, IEEE Computer Society, v. 2016-January, p. 1077–1084, 1 2016. ISSN 10823409. Disponível em: <https://www.researchgate.net/publication/281901238_Optimizing_the_Parameters_of_Drift_Detection_Methods_Using_a_Genetic_Algorithm>. Citado na página 25.

VIEIRA, L. H. B. Aprendizado não supervisionado para detecção de ataques em internet das coisas. 2022. Citado 2 vezes nas páginas 29 e 32.

XU, D.; TIAN, Y. A comprehensive survey of clustering algorithms. **Annals of Data Science**, Springer Science and Business Media LLC, v. 2, p. 165–193, 6 2015. ISSN 2198-5804. Citado 4 vezes nas páginas 7, 21, 22 e 23.

YANG, D.; RUNDENSTEINER, E. A.; WARD, M. O. Neighbor-based pattern detection for windows over streaming data. **Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT'09**, p. 529–540, 2009. Disponível em: <https://dl.acm.org/doi/10.1145/1516360.1516422>. Citado na página 40.

YANG, Z.; LIU, X.; LI, T.; WU, D.; WANG, J.; ZHAO, Y.; HAN, H. A systematic literature review of methods and datasets for anomaly-based network intrusion detection. **Computers Security**, Elsevier Advanced Technology, v. 116, p. 102675, 5 2022. ISSN 0167-4048. Citado na página 13.

YILMAZ, S. F.; KOZAT, S. S. Pysad: A streaming anomaly detection framework in python. 9 2020. Disponível em: <https://arxiv.org/abs/2009.02572v1>. Citado na página 29.

_____. Pysad: A streaming anomaly detection framework in python. **arXiv preprint arXiv:2009.02572**, 2020. Citado na página 41.

YIN, C.; ZHANG, S.; YIN, Z.; WANG, J. Anomaly detection model based on data stream clustering. **Cluster Computing**, Springer New York LLC, v. 22, p. 1729–1738, 1 2019. ISSN 15737543. Disponível em: <https://link.springer.com/article/10.1007/s10586-017-1066-2>. Citado 2 vezes nas páginas 31 e 32.

ZHANG, J.; LI, H.; GAO, Q.; WANG, H.; LUO, Y. Detecting anomalies from big network traffic data using an adaptive detection approach. **Information Sciences**, Elsevier, v. 318, p. 91–110, 10 2015. ISSN 0020-0255. Citado 2 vezes nas páginas 30 e 32.

ZHAO, Y.; NASRULLAH, Z.; LI, Z. Pyod: A python toolbox for scalable outlier detection. **Journal of Machine Learning Research**, v. 20, p. 1–7, 2019. Disponível em: <https://pyod.readthedocs.io>. Citado na página 28.

_____. Pyod: A python toolbox for scalable outlier detection. **Journal of Machine Learning Research**, v. 20, n. 96, p. 1–7, 2019. Disponível em: <http://jmlr.org/papers/v20/19-011.html>. Citado na página 41.

ZHU, C.; WANG, X.; ZHU, L. Application research of a data stream clustering algorithm in network security defense. **Journal of Physics: Conference Series**, IOP Publishing, v. 1423, n. 1, p. 012027, dec 2019. Disponível em: <https://dx.doi.org/10.1088/1742-6596/1423/1/012027>. Citado 2 vezes nas páginas 31 e 32.

APÊNDICE A – Resultados *Isolation Forest*

A.1 Terça-Feira

Tabela 14 – Isolation Forest Terça | Classe: Normal

Isolation Forest Terça-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 71575	fn = 57969
	Negativo	fp = 2930	tn = 10902

Tabela 15 – Isolation Forest Terça | Classe: Ataque

Isolation Forest Terça-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 10902	fn = 2930
	Negativo	fp = 57969	tn = 71575

A.2 Sexta-Feira

Tabela 16 – Isolation Forest Sexta | Classe: Normal

Isolation Forest Sexta-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 68928	fn = 55252
	Negativo	fp = 174883	tn = 113902

Tabela 17 – Isolation Forest Sexta | Classe: Ataque

Isolation Forest Sexta-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 113902	fn = 174883
	Negativo	fp = 55252	tn = 68928

APÊNDICE B – Resultados *SVM - One-Class*

B.1 Terça-Feira

Tabela 18 – SVM Terça | Classe: Normal

SVM Terça-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 116905	fn = 12639
	Negativo	fp = 6672	tn = 7160

Tabela 19 – SVM Terça | Classe: Ataque

SVM Terça-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 7160	fn = 6672
	Negativo	fp = 12639	tn = 116905

B.2 Sexta-Feira

Tabela 20 – SVM Sexta | Classe: Normal

SVM Sexta-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 111586	fn = 12594
	Negativo	fp = 231853	tn = 56932

Tabela 21 – SVM Sexta | Classe: Ataque

SVM Sexta-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 56932	fn = 231853
	Negativo	fp = 12594	tn = 111586

APÊNDICE C – Resultados *LOF Novelty*

C.1 Terça-Feira

Tabela 22 – LOF-Novelty Terça | Classe: Normal

LOF-Novelty Terça-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 108081	fn = 21463
	Negativo	fp = 9562	tn = 4270

Tabela 23 – LOF-Novelty Terça | Classe: Ataque

LOF-Novelty Terça-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 4270	fn = 9562
	Negativo	fp = 21463	tn = 108081

C.2 Sexta-Feira

Tabela 24 – LOF-Novelty Sexta | Classe: Normal

LOF-Novelty Sexta-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 103134	fn = 21046
	Negativo	fp = 169806	tn = 118979

Tabela 25 – LOF-Novelty Sexta | Classe: Ataque

LOF-Novelty Sexta-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 118979	fn = 169806
	Negativo	fp = 21046	tn = 103134

APÊNDICE D – Resultados *LOF - Stream*

D.1 Terça-Feira

Tabela 26 – LOF-Stream Terça | Classe: Normal

LOF-Stream Terça-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 98065	fn = 31479
	Negativo	fp = 9583	tn = 4249

Tabela 27 – LOF-Stream Terça | Classe: Ataque

LOF-Stream Terça-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 4249	fn = 9583
	Negativo	fp = 31479	tn = 98065

D.2 Sexta-Feira

Tabela 28 – LOF-Stream Sexta | Classe: Normal

LOF-Stream Sexta-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 93562	fn = 30618
	Negativo	fp = 194885	tn = 93900

Tabela 29 – LOF-Stream Sexta | Classe: Ataque

LOF-Stream Sexta-feira			
		Predito	
		Positivo	**Negativo**
Real	**Positivo**	tp = 93900	fn = 194885
	Negativo	fp = 30618	tn = 93562