Universidade Estadual de Londrina

MATHEUS RIKI NAKANISHI

DETECÇÃO DE ANOMALIAS EM REDES DEFINIDAS POR SOFTWARE SOB INFLUÊNCIA DE CONCEPT DRIFT

LONDRINA
2025

MATHEUS RIKI NAKANISHI

DETECÇÃO DE ANOMALIAS EM REDES DEFINIDAS POR SOFTWARE SOB INFLUÊNCIA DE CONCEPT DRIFT

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Mario Lemes Proença Jr.

LONDRINA

2025

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Sobrenome, Nome.
Título do Trabalho : Subtitulo do Trabalho / Nome Sobrenome. - Londrina, 2017.
100 f. : il.

Orientador: Nome do Orientador Sobrenome do Orientador.
Coorientador: Nome Coorientador Sobrenome Coorientador.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2017.
Inclui bibliografia.

1. Assunto 1 - Tese. 2. Assunto 2 - Tese. 3. Assunto 3 - Tese. 4. Assunto 4 - Tese. I. Sobrenome do Orientador, Nome do Orientador. II. Sobrenome Coorientador, Nome Coorientador. III. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

MATHEUS RIKI NAKANISHI

DETECÇÃO DE ANOMALIAS EM REDES DEFINIDAS POR SOFTWARE SOB INFLUÊNCIA DE CONCEPT DRIFT

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof. Dr. Mario Lemes Proença Jr.
Universidade Estadual de Londrina

Prof. Dr. Elieser Botelho Manhas Jr.
Universidade Estadual de Londrina

Vinícius Ferreira Schiavon
Universidade Estadual de Londrina

Londrina, 25 de novembro de 2025.

Dedico este trabalho aos meus familiares,
cujo apoio foi essencial para que eu
alcançasse esta etapa da minha trajetória.

AGRADECIMENTOS

Aos meus pais, por todo o ensinamento, amor e sacrifício realizados para que eu pudesse me desenvolver como pessoa. E aos demais familiares, por todo o carinho fornecido, em especial à Tamie, pelo cuidado durante um período tão difícil.

Ao Prof. Dr. Mario Lemes Proença Jr., por todo ensinamento e orientação, que me possibilitaram evoluir e amadurecer como pessoa e profissional.

Aos membros do grupo de pesquisa ORION da Universidade Estadual de Londrina, pelo suporte e pela companhia ao longo desta jornada.

Aos meus colegas de curso, pela amizade que tornou esse período mais leve e divertido.

Aos professores do curso de Ciência da Computação da Universidade Estadual de Londrina, pelos ensinamentos fornecidos durante a graduação.

Aos servidores da UEL, pelo trabalho fundamental que empenham, garantindo o funcionamento da universidade.

"Não há fatos eternos, como não há
verdades absolutas.
Friedrich Nietzsche

NAKANISHI, M. R.. **Detecção de Anomalias em Redes Definidas por Software sob Influência de Concept Drift**. 2025. 75f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2025.

RESUMO

As redes de computadores tornaram-se fundamentais para a sociedade, uma vez que diversos setores dependem de aplicações baseadas nessa infraestrutura. Como exemplo, tem-se o setor industrial, que utiliza as redes para automação, o setor educacional, com acesso à educação a distância, a economia, com transações financeiras, e a comunicação e mídia, com redes sociais, aplicativos de mensagens e plataformas de *streaming*. Motivados pelo valor agregado às redes, agentes maliciosos buscam comprometer seu funcionamento. Para garantir maior segurança, são desenvolvidos Sistemas de Detecção de Intrusão em Redes baseados em Aprendizado de Máquina, devido à capacidade que esses métodos apresentam de lidar com cenários complexos. No entanto, tais modelos podem ser impactados pelo Desvio de Conceito, caracterizado por mudanças na distribuição dos dados ao longo do tempo. Esse fenômeno é comum em ambientes dinâmicos, como redes de computadores, e pode resultar na degradação do desempenho dos sistemas de segurança. Este trabalho apresenta um estudo de técnicas de adaptação ao Desvio de Conceito para Sistemas de Detecção de Intrusão em Redes baseados em Aprendizado de Máquina, comparando suas principais características de operação. Um modelo baseado no algoritmo *DenStream* tem sua capacidade de detecção de anomalias, reconhecimento de ataques, adaptação ao desvio de conceito e velocidade de identificação analisadas. Foi utilizado um conjunto de dados sintético de uma Rede Definida por Software, contendo tráfego benigno e atividades maliciosas. O desempenho do modelo foi avaliado por meio de métricas como acurácia, precisão, revocação, F1-*score* e Coeficiente de Correlação de Matthews.

Palavras-chave: Aprendizado de máquina. Desvio de conceito. Detecção de anomalias. Segurança de redes. Detecção de intrusão.

NAKANISHI, M. R.. **Anomaly Detection in Software Defined Networks under the Influence of Concept Drift**. 2025. 75p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2025.

ABSTRACT

Computer networks have become fundamental to society, as various sectors rely on applications built upon this infrastructure. For example, the industrial sector uses networks for automation and the Internet of Things, the education, with access to distance learning, the economy, with financial transactions, and communication and media, with social networks, messaging applications and streaming platforms. Motivated by the value associated with networks, malicious actors seek to compromise their operation. To enhance security, Network Intrusion Detection Systems based on Machine Learning have been developed, due to the ability of these methods to handle complex scenarios. However, such models can be affected by Concept Drift, characterized by changes in the data distribution over time. This phenomenon is common in dynamic environments, such as computer networks, and may result in performance degradation of security systems. This work presents a study of Concept Drift adaptation techniques for Network Intrusion Detection Systems based on Machine Learning, comparing their main operational characteristics. A model based on the DenStream algorithm has its anomaly detection capability, attack recognition, concept drift adaptation, and identification speed analyzed. A synthetic dataset of a Software-Defined Network containing benign traffic and malicious activities was used. The model's performance was evaluated using metrics such as accuracy, precision, recall, F1-score, and Matthews Correlation Coefficient.

Keywords: Machine Learning. Concept Drift. Anomaly Detection. Network Security. Intrusion Detection.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADWIN	*Adaptive Windowing*
ADWIN2	*Adaptive Windowing 2*
ARF	*Adaptive Random Forest*
ARF-ADWIN	*Adaptive Random Forest with Adaptive Windowing*
ARF-EDDM	*Adaptive Random Forest with Early Drift Detection Method*
AUE2	*Accuracy Updated Ensemble 2*
AWE	*Accuracy-Weighted Ensembles*
C-MC	*Core Micro-Cluster*
CDDIA	*Concept Drift Detection, Interpretation and Adaptation*
CNN	*Convolutional Neural Network*
CVFDT	*Concept-adapting Very Fast Decision Tree learner*
DBSCAN	*Density-Based Spatial Clustering of Application with Noise*
DDE	*Drift Detection Ensemble*
DDM	*Drift Detection Method*
DDoS	*Distributed Denial of Service*
DWM	*Dynamic Weighted Majority*
ECDD	*Exponentially Weighted Moving Average for Concept Drift Detection*
EDDM	*Early Drift Detection Method*
EDE	*Equal Density Estimation*
EWMA	*Exponentially Weighted Moving Average*
FHDDM	*Fast Hoeffding Drift Detection Method*
FN	Falso Negativo
FP	Falso Positivo
GAN	*Generative Adversarial Network*

GB *Gigabyte*

GRU *Gated Recurrent Unit*

HAT *Hoeffding Adaptive Tree*

HDDM *Hoeffding Drift Detection Method*

HDDM_A *Hoeffding Drift Detection Method with Averages Test*

HDDM_W *Hoeffding Drift Detection Method with Weighted Averages Test*

HLFR *Hierarchical Linear Four Rates*

IB1 *Instance-Based 1-Nearest Neighbor*

IoT *Internet of Things*

IP *Internet Protocol*

KL *Kullback-Leibler*

KNN *K-Nearest Neighbors*

KNN-ADWIN *K-Nearest Neighbors with Adaptive Windowing*

Learn++.NSE *Learn++ with Non-Stationary Ensemble*

Learn++.SMOTE *Learn++ with Synthetic Minority Oversampling Technique*

LFR *Linear Four Rates*

LSDD-CDT *Least Squares Density Difference-based Change Detection Test*

LSDD-INC *Incremental LSDD-Based Change Detection Test*

LSTM *Long Short-Term Memory*

LTS *Long-Term Support*

MC *Micro-Cluster*

MCC *Matthews Correlation Coefficient*

NB *Naive Bayes*

NIDS *Network Intrusion Detection System*

NN *Neural Network*

O-MC *Outlier Micro-Cluster*

OPA	*Online Passive-Aggressive*
P	*Distribuição de Probabilidade*
P-MC	*Micro-Cluster*
PAC	*Probably Approximately Correct*
PL	*Paired Learners*
PNN	*Probabilistic Neural Network*
RAM	*Random Access Memory*
RDDM	*Reactive Drift Detection Method*
SALAD	*Split Active Learning Anomaly Detector*
SDN	*Software Defined Network*
SEED	*Stream Ensemble for Early Drift Detection*
SHAP	*SHapley Additive exPlanations*
SMOTE	*Synthetic Minority Over-sampling Technique*
SRP	*Streaming Random Patches*
SVM	*Support Vector Machine*
SVMLearn++	*Support Vector Machine with Learn++*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
VFDT	*Very Fast Decision Tree learner*
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo

SUMÁRIO

1 INTRODUÇÃO

As redes de computadores, impulsionadas pela popularização da Internet, tornaram-se ferramentas fundamentais para a sociedade, contribuindo diretamente para o avanço científico, o compartilhamento de informações e o acesso a serviços digitais [2]. Sistemas de diferentes setores, como saúde, educação, comércio e financeiro, necessitam das redes de computadores para funcionar. Essa dependência, somada ao aumento exponencial do tráfego de dados, demanda uma infraestrutura de rede mais complexa para suprir as necessidades dos usuários. Esse aumento na complexidade da infraestrutura leva à integração de dispositivos de diferentes fabricantes, dificultando o gerenciamento da rede devido às interfaces de programação proprietárias, que exigem configurações individualizadas [3].

A SDN (do inglês, *Software Defined Networks*) surge como uma alternativa flexível às redes de computadores tradicionais. Essa abordagem é caracterizada pela separação entre o plano de dados, responsável pelo encaminhamento de pacotes, e o plano de controle, responsável pelas regras de roteamento. A centralização das decisões no controlador, que se comunica com os demais dispositivos por meio de um protocolo padronizado, simplifica o gerenciamento da rede e aumenta sua escalabilidade. Essa característica torna o controlador um ponto crítico de falha que pode ser explorado por agentes maliciosos por meio de ataques cibernéticos, comprometendo toda a rede [4].

O ataque DDoS (do inglês, *Distributed Denial of Service*) é um exemplo de ataque de negação de serviço, caracterizado pelo envio coordenado de um volume excessivo de requisições a um alvo. Essa atividade maliciosa tem como objetivo esgotar os recursos computacionais e interromper o funcionamento de um sistema, rede ou serviço. Em março de 2025, a Marinha do Brasil foi alvo de um ataque DDoS destinado aos órgãos públicos brasileiros, resultando na indisponibilidade temporária de seu portal [5].

Para garantir a confidencialidade, integridade e disponibilidade dos serviços e dados presentes nas redes de computadores, são desenvolvidos os NIDS (do inglês, *Network Intrusion Detection System*). Esses sistemas monitoram continuamente o tráfego da rede, buscando identificar comportamentos anômalos, como ataques cibernéticos. Há duas abordagens principais para a detecção de intrusões em redes: a baseada em assinaturas, que identifica ataques com base em padrões previamente conhecidos, e a baseada em anomalias, que detecta ataques por meio de desvios em relação ao comportamento normal esperado [6].

Ao longo do tempo, diferentes técnicas baseadas em anomalia foram exploradas para a implementação de NIDS. Tradicionalmente, métodos estatísticos, como a análise do parâmetro de Hurst [7], foram utilizados na detecção de anomalias. Técnicas basea-

das em aprendizado de máquina ganharam relevância devido à sua maior adaptabilidade, consolidando-se como alternativas eficazes para esse tipo de sistema [2]. Abordagens de aprendizado profundo têm se destacado, uma vez que esses modelos são capazes de aprender representações complexas e processar grandes volumes de dados [8], [9], [10], [11]. Alguns exemplos de técnicas de aprendizado profundo aplicadas à detecção de anomalias incluem a CNN (do inglês, *Convolutional Neural Networks*) [12], a LSTM (do inglês, *Long Short-Term Memory*) [13], a GRU (do inglês, *Gated Recurrent Units*) [14] e a GAN (do inglês, *Generative Adversarial Networks*) [15].

Os NIDS baseados em aprendizado de máquina frequentemente assumem que o ambiente de rede apresenta um comportamento estacionário, o que não condiz com a natureza dinâmica de cenários reais [16]. A distribuição dos dados de tráfego pode variar por diversos fatores, como mudanças no comportamento dos usuários, modificações na infraestrutura da rede e adaptações nas estratégias de ataque [17]. Esse fenômeno é conhecido como desvio de conceito (do inglês, *Concept Drift*) e descreve alterações imprevisíveis nas propriedades estatísticas dos dados ao longo do tempo, o que pode levar à degradação do desempenho dos modelos [18]. A adaptação a esse fenômeno é fundamental para preservar a confiabilidade dos modelos e assegurar sua aplicabilidade em cenários reais [19].

Este trabalho tem como objetivo abordar técnicas de detecção e adaptação ao desvio de conceito em sistemas de detecção de intrusão em SDN baseados em aprendizado de máquina. A estrutura deste documento é organizada da seguinte forma: o Capítulo 2 apresenta os fundamentos teóricos necessários para o desenvolvimento do estudo proposto. No Capítulo 3, são discutidos em maior profundidade os conceitos relacionados ao desvio de conceito. O Capítulo 4 descreve trabalhos relacionados à detecção de intrusão em redes sob o contexto de desvio de conceito. Por fim, o estudo de caso e as conclusões são apresentados nos Capítulos 5 e 6, respectivamente.

2 FUNDAMENTAÇÃO TEÓRICO-METODOLÓGICA

2.1 Redes Definidas por Software

A arquitetura das redes de computadores pode ser organizada em camadas operacionais, conhecidas como planos, de acordo com as funções que desempenham. Nas redes tradicionais, destacam-se o plano de dados, responsável pela transmissão de pacotes entre os dispositivos, e o plano de controle, que determina como o tráfego deve ser encaminhado com base em políticas pré-estabelecidas [20]. Esses planos estão integrados em cada dispositivo de rede, cuja configuração é realizada manualmente e de forma local pelos administradores [21]. À medida que a infraestrutura de rede cresce e a heterogeneidade dos equipamentos aumenta, o gerenciamento de rede se torna mais complexo e custoso operacionalmente. Isso se deve às interfaces de programação proprietárias, que dificultam a implementação uniforme de políticas de controle [22].

Para superar as limitações de escalabilidade, flexibilidade e gerenciamento das redes tradicionais, surge a arquitetura SDN, caracterizada pela separação dos planos de dados, controle e aplicação, conforme ilustrado na Figura 1. Nessa abordagem, as políticas de rede, definidas pelos administradores no plano de aplicação, são transmitidas ao plano de controle por meio da interface *northbound*. O plano de controle é centralizado em um ou mais controladores, responsáveis por intermediar a comunicação entre os planos de dados e de aplicação, propagando as regras de roteamento e controle [23]. Além disso, o controlador é responsável por coletar estatísticas de fluxo, determinar as melhores políticas de encaminhamento e detectar falhas na rede [24]. O plano de dados, composto por *switches* e roteadores, tem como função encaminhar os pacotes de dados na rede, seguindo as regras e instruções propagadas pelo controlador através da interface *southbound* [25].

A centralização das configurações e a comunicação do controlador com os dispositivos de encaminhamento por meio de um protocolo padronizado conferem maior escalabilidade e flexibilidade às SDN. Essa arquitetura simplifica o gerenciamento da rede, facilita a aplicação de políticas e garante que alterações sejam propagadas de forma consistente e uniforme entre todos os dispositivos. Entretanto, a centralização torna o controlador um ponto único de falha que, caso seja explorado por agentes maliciosos, pode afetar o funcionamento de toda a rede [9].

2.2 Anomalias em Redes de Computadores

Uma anomalia é qualquer ocorrência, dado ou característica que se desvia do usual dentro de um determinado contexto [26]. Em redes de computadores, o comportamento

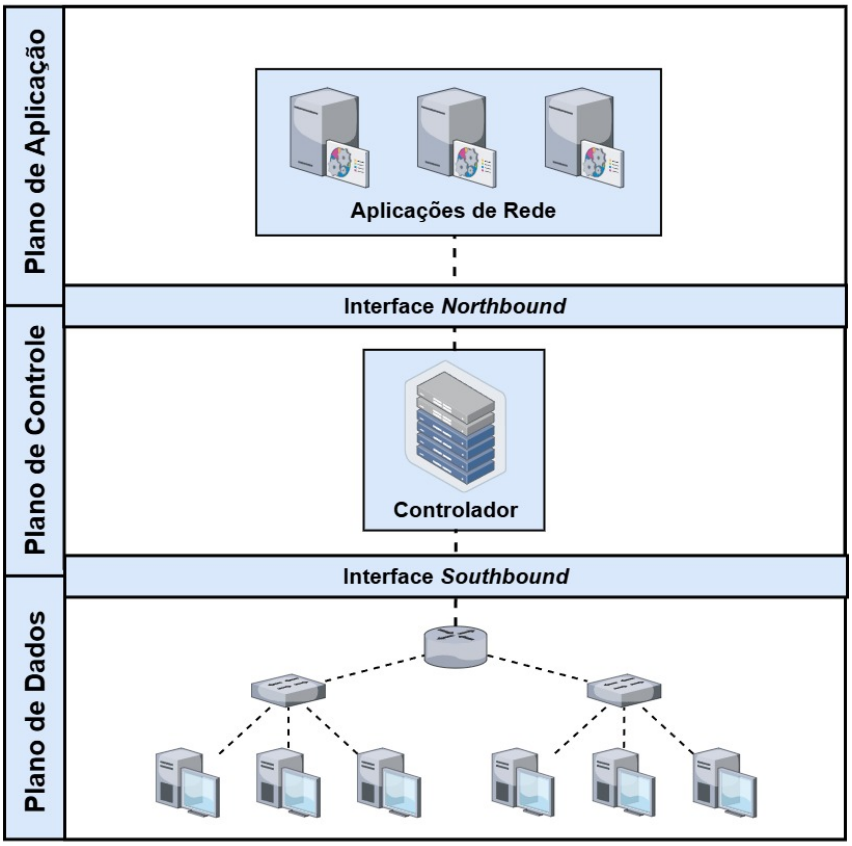

Figura 1 – Arquitetura SDN. Fonte: elaboração própria.

esperado é definido por um padrão (*baseline*) obtido a partir da análise do tráfego. Um desvio em relação a esse padrão estabelecido é classificado como uma anomalia de rede. As anomalias de rede podem causar prejuízos críticos, uma vez que diversos serviços sensíveis dependem da disponibilidade e estabilidade das redes de computadores. Suas causas podem variar desde problemas de *hardware*, *software* e falhas humanas, até atividades maliciosas, como ataques.

Entre as diferentes causas de anomalias em redes, destacam-se os ataques cibernéticos, que buscam comprometer a qualidade de um serviço ou beneficiar o usuário malicioso. Um exemplo é a negação de serviço, como no ataque DDoS, caracterizado pelo envio coordenado de um volume excessivo de requisições a um alvo, com a intenção de esgotar seus recursos computacionais e interromper seu funcionamento [27]. Para ampliar o impacto, é comum a utilização de redes de dispositivos infectados por *malwares*, conhecidas como *botnets* [28]. A Figura 2 ilustra esse cenário, destacando a atuação do agente malicioso no controle da *botnet* para a execução do ataque DDoS contra o alvo.

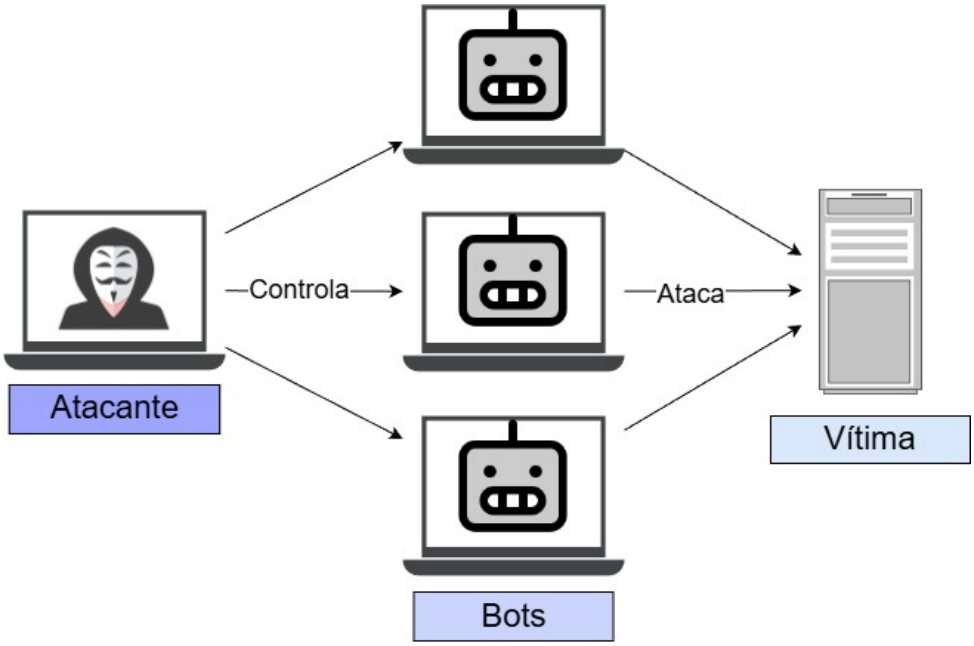

Figura 2 – Ataque DDoS utilizando *botnets*. Fonte: elaboração própria.

2.3 Sistema de Detecção de Intrusão de Redes

Os NIDS são mecanismos de defesa projetados para garantir a confidencialidade, a integridade e a disponibilidade dos serviços e dados em redes de computadores [9]. Esses sistemas monitoram continuamente o tráfego de rede e são capazes de identificar comportamentos anômalos, como tentativas de intrusão [29]. Quando uma anomalia é detectada, um alerta é emitido ao administrador da rede, permitindo que medidas corretivas sejam adotadas. O funcionamento desse processo é ilustrado na Figura 3.

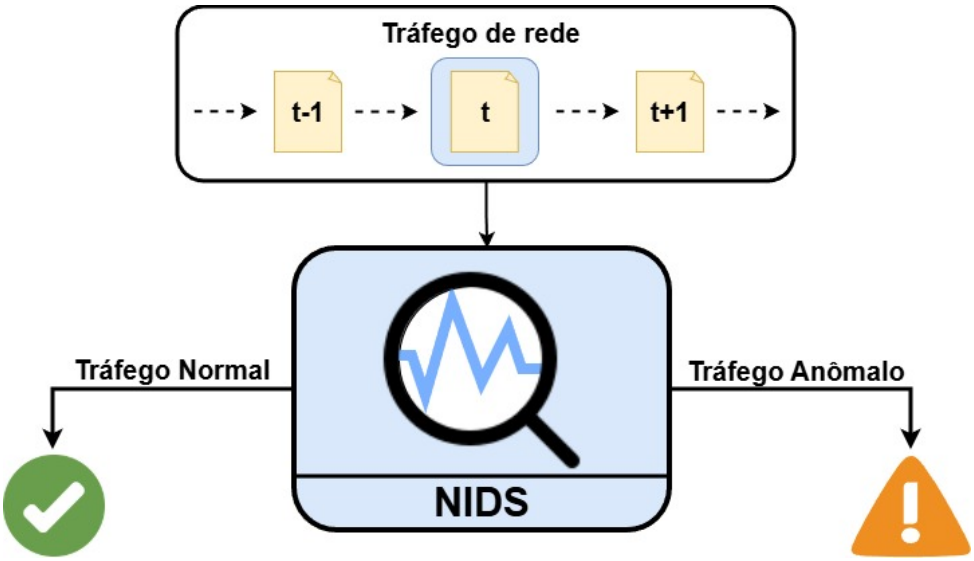

Figura 3 – Funcionamento de um NIDS. Fonte: elaboração própria.

Existem duas abordagens principais para a detecção de ataques em NIDS: baseada

em assinaturas e baseada em anomalias. A estratégia baseada em assinaturas consiste em armazenar padrões maliciosos extraídos de dados históricos e compará-los com o tráfego atual. Quando o comportamento observado corresponde a uma assinatura conhecida, o evento é classificado como ataque. A principal limitação dessa estratégia é a incapacidade de identificar ameaças desconhecidas, exigindo atualizações frequentes do conjunto de assinaturas [30]. A abordagem baseada em anomalias compara o tráfego observado com o comportamento normal da rede, possibilitando a detecção de novos ataques sem depender do conhecimento prévio de padrões maliciosos. Como desvantagem, há a possibilidade de aumento nas classificações incorretas, em que variações legítimas de comportamento podem ser interpretadas como ataques, enquanto atividades maliciosas com pequeno desvio em relação ao padrão normal podem não ser detectadas [31].

A criação constante de novos ataques, projetados para contornar os sistemas de defesa existentes, representa um desafio significativo na segurança de redes de computadores. Esse cenário torna-se ainda mais crítico devido à alta velocidade de transmissão dos dados, capaz de gerar danos severos em poucos segundos [32]. Nesse contexto, a detecção baseada em anomalias, aliada a uma resposta rápida a comportamentos suspeitos, constitui um aspecto fundamental dos NIDS [33], [34].

O grupo de pesquisa ORION, do Departamento de Computação da Universidade Estadual de Londrina, tem se dedicado ao estudo da segurança em redes de computadores, com ênfase no desenvolvimento de NIDS. Proença *et al.* (2004) [7] e Proença *et al.* (2005) [35] exploraram a aplicação de técnicas estatísticas para a geração de assinaturas digitais de segmento de rede. O Algoritmo do Vaga-lume, uma técnica heurística, foi empregado por Adaniya *et al.* [36], enquanto a otimização por colônias de formigas, uma técnica metaheurística, foi utilizada por Assis *et al.* [37] e Carvalho *et al.* [38]. Trabalhos recentes têm se concentrado no desenvolvimento de NIDS baseados em aprendizado profundo, capazes de identificar padrões complexos e manipular grandes volumes de dados, utilizando modelos como GRU [39] e GAN [40], [41].

2.4 Aprendizado de Máquina

A inteligência artificial é uma área da ciência da computação que busca desenvolver sistemas capazes de realizar atividades que tradicionalmente exigiriam inteligência humana, como resolução de problemas complexos e tomadas de decisão [42]. O aprendizado de máquina é uma subárea da inteligência artificial aplicada em diversos setores, incluindo a segurança de redes de computadores, especialmente no desenvolvimento de NIDS. O aumento na geração de dados, impulsionado pela popularização das redes sociais, dos serviços de *streaming*, da Internet das Coisas e da própria Internet, tem favorecido a criação de modelos computacionais baseados nessa técnica. Esses modelos são capazes de executar tarefas consideradas complexas para algoritmos tradicionais, identificando

automaticamente padrões nos dados e modelando problemas de forma autônoma [43].

O treinamento de algoritmos de aprendizado de máquina pode ser classificado em três abordagens principais: aprendizado supervisionado, aprendizado não supervisionado e aprendizado semi-supervisionado. O aprendizado supervisionado compreende o treinamento de modelos a partir de amostras rotuladas. Cada instância rotulada define uma correspondência entre um vetor de características e sua respectiva classe ou valor-alvo, orientando o processo de aprendizado do modelo [44]. O aprendizado não supervisionado utiliza amostras não rotuladas no processo de treinamento. O modelo é guiado apenas pelas relações entre os dados, buscando identificar afinidades e discrepâncias por meio da análise de métricas como similaridade, distância e densidade [45]. Essa técnica é fundamental em cenários que carecem de conjuntos de dados rotulados ou em que o processo de rotulação apresenta alto custo. O aprendizado semi-supervisionado é uma abordagem híbrida que combina elementos das técnicas supervisionada e não supervisionada. Durante o treinamento, as instâncias rotuladas fornecem uma base inicial de conhecimento ao modelo, que posteriormente é utilizada para identificar padrões semelhantes nos dados não rotulados [46].

2.4.1 Rede Neural Artificial

As NN (do inglês, Neural Networks), são modelos computacionais derivados do aprendizado de máquina e inspirados na estrutura e no funcionamento do cérebro humano. Sua arquitetura é composta por uma rede de nós interconectados, denominados neurônios artificiais. Na forma mais básica, como no *Perceptron*, cada neurônio é uma unidade de processamento associada a pesos e a um viés. Em arquiteturas mais avançadas, essas unidades assumem estruturas de maior complexidade, como nas células de GRU e LSTM, que incorporam mecanismos internos para controlar o fluxo de informação [14], [47].

As NN são organizadas em camadas, nas quais cada unidade processa um conjunto de entradas e propaga o resultado para a camada subsequente, conforme ilustrado na Figura 4. A camada de entrada recebe os dados e os disponibiliza para o processamento. Em seguida, uma ou mais camadas ocultas realizam o processamento das informações por meio de operações matemáticas e funções de ativação. A camada de saída converte as informações processadas pelas camadas anteriores em um valor final, aplicando uma função de ativação que adapta o formato da saída ao tipo de tarefa a ser executada [48].

2.4.2 Aprendizado Profundo

O aprendizado profundo é uma subárea do aprendizado de máquina, como ilustrado na Figura 5, que utiliza redes neurais de múltiplas camadas para aprender representações dos dados em diferentes níveis de abstração. Seus algoritmos são capazes de modelar problemas de forma autônoma, identificando padrões sem a necessidade de funções ex-

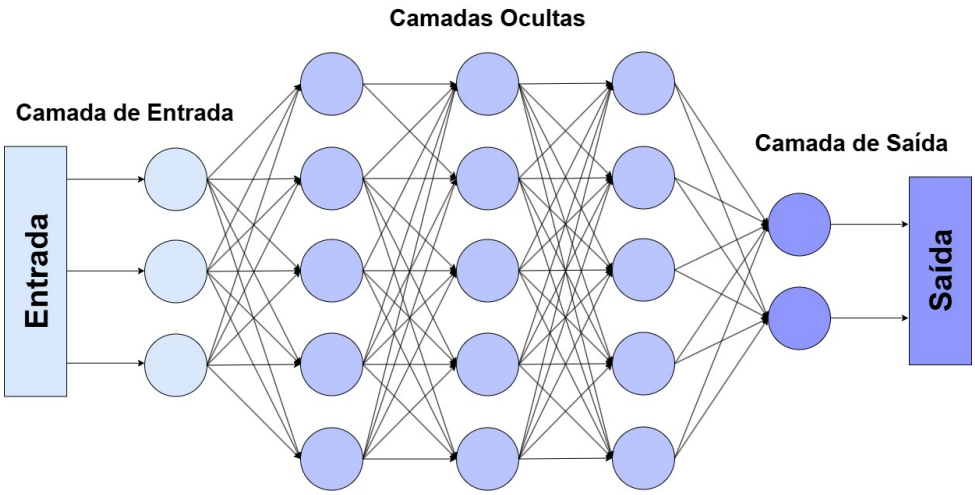

Figura 4 – Arquitetura de uma Rede Neural. Fonte: elaboração própria.

plicitamente definidas [49]. Essa abordagem tem sido aplicada em sistemas de detecção de intrusão, devido à sua capacidade de aprendizagem, generalização e processamento de grandes volumes de dados [47]. Nesse processo, camadas de representações simples são construídas internamente, sendo cada camada responsável por transformar a saída anterior em uma representação mais abstrata. A combinação dessas estruturas possibilita a modelagem de conceitos complexos [50].

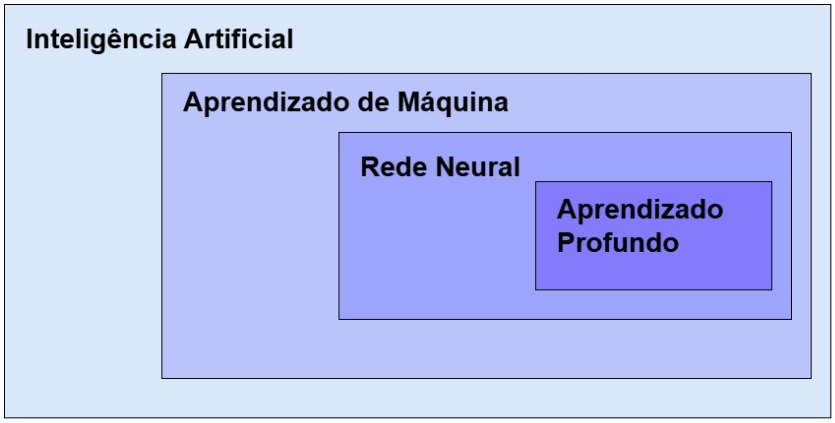

Figura 5 – Relação entre Inteligência Artificial, Aprendizado de Máquina, Redes Neurais e Aprendizado Profundo. Fonte: elaboração própria.

3 DESVIO DE CONCEITO

Fatores como investimentos em pesquisa, avanços tecnológicos, a crescente disponibilidade de dados e o maior acesso a recursos computacionais têm contribuído para a popularização do aprendizado de máquina. Essa tecnologia passou a integrar o cotidiano da sociedade, sendo aplicada em diversas áreas [51]. Os modelos tradicionais de aprendizado de máquina são treinados a partir de conjuntos históricos de dados e assumem uma distribuição fixa, pressupondo um comportamento estacionário. Quando aplicados em cenários reais, esses modelos tendem a apresentar uma queda de desempenho com o passar do tempo, devido ao fenômeno conhecido como desvio de conceito [52].

O desvio de conceito refere-se a alterações na relação entre as variáveis de entrada e a saída do modelo ao longo do tempo. Fatores como mudanças no comportamento humano, eventos externos ou até mesmo padrões sazonais recorrentes podem modificar essa relação e provocar um desvio [53]. Como consequência, o desempenho do modelo tende a se degradar, uma vez que os padrões previamente aprendidos deixam de representar adequadamente o estado atual dos dados [19].

É possível definir o conceito como uma distribuição de probabilidade conjunta [54]. Seja X o vetor de atributos e y a variável alvo, o conceito pode ser representado pela Equação 3.1:

$$\text{Conceito} = P(X, y). \tag{3.1}$$

Essa distribuição descreve a probabilidade da ocorrência simultânea de dois ou mais eventos aleatórios [55]. Por exemplo, em um sistema de detecção de fraudes em compras com cartão de crédito, X pode representar um vetor de atributos que inclua o valor, o local e o horário da compra. A variável alvo y classifica a compra como fraudulenta ou legítima. A distribuição conjunta $P(X, y)$ descreve a probabilidade de uma compra ser fraudulenta ou não, considerando todos os pares possíveis de X e y.

O desvio de conceito ocorre quando a distribuição conjunta de probabilidade entre os atributos e a variável alvo muda ao longo do tempo. Essa alteração pode ser formalmente expressa pela desigualdade apresentada na Equação 3.2, a qual indica que, em diferentes instantes, a relação entre as variáveis deixa de ser a mesma [56].

$$P_t(X, y) \neq P_{t+1}(X, y). \tag{3.2}$$

Aplicando a regra do produto, a Equação 3.2 pode ser decomposta conforme apresentado na Equação 3.3 [57]:

$$P_t(X, y) = P_t(X) \times P_t(y \mid X) \tag{3.3}$$

A probabilidade condicional, denotada por $P_t(y \mid X)$, representa a chance de ocorrência de um evento y dado que outro evento X já ocorreu [58]. Ela expressa a relação entre as variáveis de entrada e a saída no instante de tempo t. O termo $P_t(X)$ corresponde à distribuição de probabilidade das variáveis de entrada, ou seja, à probabilidade de ocorrência de X em t [59]. Essa distribuição descreve como as características estão distribuídas ao longo do tempo. A partir da decomposição apresentada na Equação (3.3), o desvio de conceito pode ser classificado em categorias que representam as diferentes formas de manifestação desse fenômeno, conforme ilustrado na Figura 6.

O *Virtual Drift* ocorre quando há alteração na distribuição marginal dos dados de entrada ($P(X)$), representada pelo deslocamento dos pontos azuis. A fronteira de decisão, ilustrada pela linha tracejada, permanece inalterada, uma vez que a relação entre os atributos e a saída ($P(y \mid X)$) não sofre modificações. Em termos de lógica preditiva, o modelo se mantém estável. Contudo, caso seja sensível à distribuição dos dados, essa mudança pode resultar na degradação do desempenho [60].

No *Real Drift*, a fronteira de decisão original sofre alteração, caracterizando uma modificação na distribuição condicional ($P(y \mid X)$). Tal desvio não está necessariamente relacionado a mudanças na distribuição marginal dos dados de entrada ($P(X)$). Esse tipo de variação implica uma mudança no conceito aprendido pelo modelo, o que resulta na redução de sua acurácia [61].

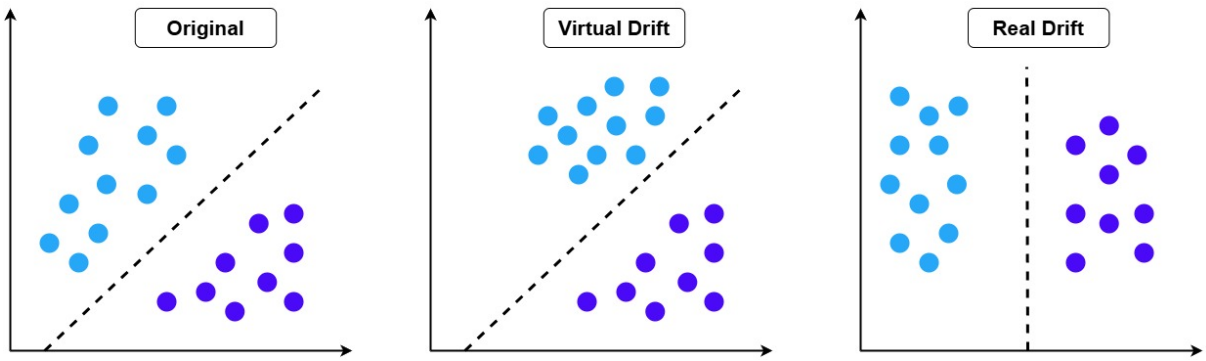

Figura 6 – *Virtual Drift* e *Real Drift*. Fonte: elaboração própria.

3.1 Transição de Conceito

O desvio de conceito pode ser categorizado conforme a forma pela qual a mudança ocorre ao longo do tempo [62], como ilustrado na Figura 7. Suas respectivas descrições são:

- **Abrupto:** ocorre de forma súbita, com uma transição rápida entre conceitos devido a um evento externo.
- **Incremental:** trata-se de uma transição lenta e contínua de conceito, à medida que os padrões de dados evoluem com o tempo.
- **Gradual:** caracteriza-se pela existência de dois conceitos distintos, que coexistem durante o período de transição. Nesse intervalo, ocorre a alternância de domínio, até que o novo conceito se estabeleça.
- **Recorrente:** conceitos previamente observados reaparecem após um intervalo de tempo, podendo apresentar comportamento cíclico ou acíclico.

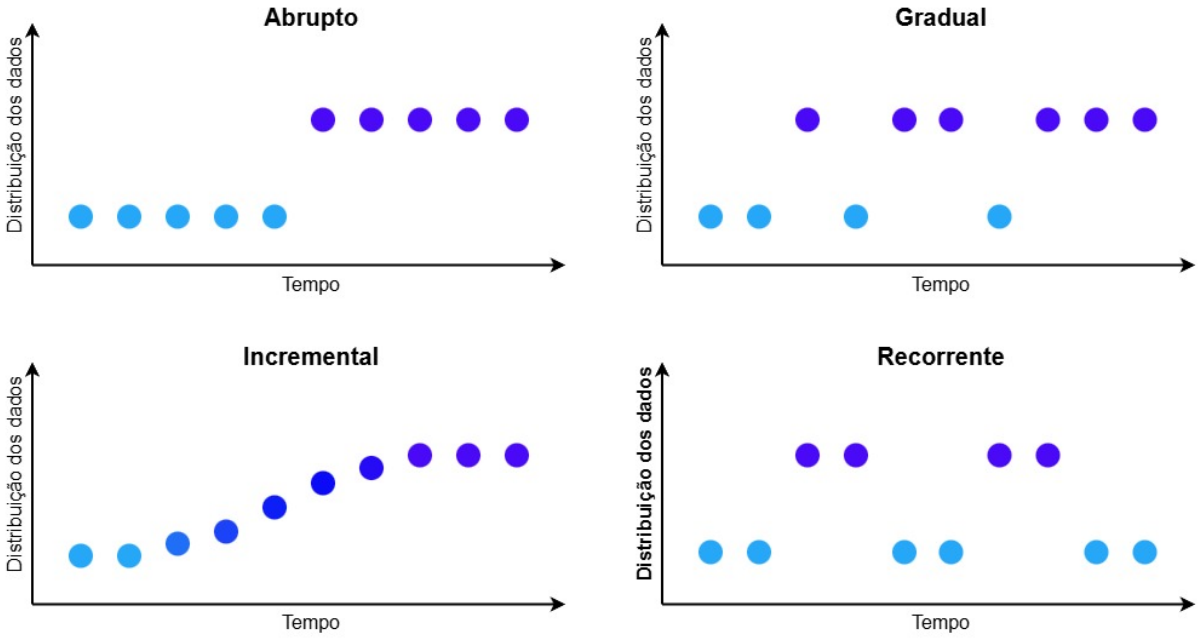

Figura 7 – Tipos de transição de conceito. Fonte: elaboração própria.

3.2 Detecção de Desvio de Conceito

A detecção constitui um processo fundamental nas estratégias de adaptação reativa ao desvio de conceito, sendo responsável por acionar os mecanismos de ajuste do modelo. Diversos algoritmos foram desenvolvidos com essa finalidade, os quais podem ser classificados em três categorias, de acordo com as estatísticas de teste aplicadas: métodos baseados na taxa de erro, métodos baseados na distribuição de dados e métodos baseados em múltiplas hipóteses [18].

3.2.1 Métodos Baseados na Taxa de Erro

Os algoritmos de detecção de desvio baseados na taxa de erro monitoram continuamente o desempenho dos classificadores. Uma variação significativa nessa taxa pode indicar a ocorrência de um desvio, acionando um alarme para que medidas adaptativas sejam aplicadas. Alguns dos algoritmos propostos com essa abordagem são:

- **DDM (do inglês, *Drift Detection Method*) [63]:** algoritmo projetado para detectar desvios de conceito com base na taxa de erro de um modelo. Desenvolvido para o contexto de aprendizado *online*, ele assume que os exemplos são processados sequencialmente, em pares (x_i, y_i), onde x_i representa o conjunto de características e y_i seu respectivo rótulo. Para cada instância, o modelo realiza uma predição $(\hat{y}_i)$, que pode estar correta $(\hat{y}_i = y_i)$ ou incorreta $(\hat{y}_i \neq y_i)$. A taxa de erro é representada, em cada instante i, pela probabilidade de ocorrer uma predição incorreta, denotada por p_i, cujo desvio padrão é dado por $s_i = \sqrt{\frac{p_i(1-p_i)}{i}}$. O DDM baseia-se na ideia de que um aumento na taxa de erro pode indicar um desvio de conceito. Para isso, a menor taxa de erro observada (p_{min}) e o seu desvio padrão correspondente (s_{min}) são armazenados. A cada novo exemplo, verifica-se a condição $p_i + s_i < p_{min} + s_{min}$ e, se satisfeita, os valores mínimos são atualizados. Com base nesses parâmetros, são definidos os níveis de aviso e de desvio. O nível de aviso é atingido quando $p_i + s_i \geq p_{min} + 2 * s_{min}$ e o nível de desvio é alcançado quando $p_i + s_i \geq p_{min} + 3 * s_{min}$, considerando a ocorrência de uma mudança de conceito. Os dados coletados entre o nível de aviso e o nível de desvio são utilizados na atualização do modelo.

- **EDDM (do inglês, *Early Drift Detection Method*) [64]:** proposto como uma melhoria do DDM para a detecção de mudanças graduais, o EDDM é um método que opera em fluxos de dados, baseado na distribuição estimada das distâncias entre erros de classificação. Parte-se do pressuposto de que a distância entre dois erros tende a aumentar conforme o modelo se adapta ao conceito vigente. O EDDM calcula, para cada erro, a distância média entre dois erros (p'_i) e do seu desvio padrão (s'_i). Esses valores são armazenados quando o modelo está mais próximo do conceito atual, isto é, quando $p'_i + 2 * s'_i$ alcança seu valor máximo. O EDDM possui dois níveis definidos pelos hiperparâmetros α e β: o nível de aviso, atingido quando $\frac{p'_i + 2 * s'_i}{p'_{max} + 2 * s'_{max}} < \alpha$, e o nível de desvio, alcançado quando $\frac{p'_i + 2 * s'_i}{p'_{max} + 2 * s'_{max}} < \beta$. As amostras armazenadas desde o nível de aviso até o nível de desvio são utilizadas para atualizar o modelo.

- **RDDM (do inglês, *Reactive Drift Detection Method*) [65]:** método derivado do DDM, desenvolvido para superar limitações relacionadas à perda de desempenho. Uma das causas desse problema é a redução da sensibilidade do DDM à medida que um conceito se prolonga, exigindo um número maior de erros de predição para

alterar significativamente a taxa de erro média a ponto de indicar a ocorrência de desvios. Outra limitação é a permanência prolongada no nível de alerta, o que reduz a velocidade do sistema e compromete a capacidade de detecção de desvios graduais. Para contornar essas deficiências, o RDDM introduz algumas modificações. Uma delas é o acionamento de um desvio de conceito suave (desvio de RDDM) quando um conceito estável atinge um número máximo de instâncias, definido pelo parâmetro max. Nesse caso, as estatísticas são recalculadas utilizando apenas um número mínimo de instâncias recentes, definido pelo parâmetro min. Outra modificação consiste em impedir o acionamento do desvio de conceito suave durante o nível de alerta, evitando um desvio prematuro com um classificador insuficientemente treinado. Para lidar com períodos de alerta excessivamente longos, o método força um desvio de DDM quando o número de instâncias em alerta ultrapassa o limite predefinido $warnLimit$, assumindo uma alta probabilidade de que o desvio já tenha ocorrido. Quando um desvio de DDM é confirmado após um período de alerta, as estatísticas são recalculadas a partir da primeira instância que gerou o alerta, uma vez que o novo classificador base já vinha sendo treinado desde então. Essas alterações resultaram em maior precisão, menores intervalos de confiança e maior velocidade em comparação ao DDM.

- **HDDM (do inglês, *Hoeffding's Inequality-based Drift Detection Method*) [66]:** algoritmo baseado no DDM que busca identificar desvios de conceito aplicando a Desigualdade de Hoeffding à sequência de dados analisada. Essa abordagem permite comparar estatisticamente dados anteriores e posteriores a um possível ponto de corte. Existem duas variantes: o $HDDM_A$, que utiliza médias móveis limitadas, e o $HDDM_W$, que aplica médias móveis ponderadas delimitadoras. O algoritmo define três estados: o *STABLE*, quando não há evidências de desvio, o *WARNING*, quando há indícios iniciais de desvio e novas instâncias passam a ser armazenadas, e o *DRIFT*, quando o desvio é identificado e o modelo é substituido por outro treinado com dados mais recentes.

- **FHDDM (do inglês, *Fast Hoeffding Drift Detection Method*) [67]:** método desenvolvido para detectar desvio através do monitoramento de desempenho do classificador, utilizando uma janela deslizante e a Desigualdade de Hoeffding. O algoritmo é baseado no modelo PAC (do inglês, *Probably Approximately Correct*), segundo o qual a acurácia da classificação deve se manter ou aumentar à medida que mais dados são processados. Caso contrário, a probabilidade de ocorrência de um desvio aumenta. A janela deslizante, de tamanho fixo n, armazena os resultados das classificações mais recentes, registrando o valor 1 para acerto e 0 para erro. À medida que os dados são processados, a probabilidade de acerto, p_1^t, é calculada no instante de tempo t e o maior valor observado, p_{max}^t, é atualizado. Um desvio é detectado

quando a diferença entre p^t_{max} e p^t_1 torna-se estatisticamente significativa, de acordo com o limiar ϵ_d, definido pela Desigualdade de Hoeffding. Experimentalmente, o FHDDM apresentou menor atraso de detecção e menos falsos positivos e negativos, em comparação a algoritmos como HDDM, DDM, EDDM e *Adaptive Windowing*.

- **ECDD (do inglês, *Exponentially Weighted Moving Average for Concept Drift Detection*) [68]:** algoritmo de detecção baseado no desempenho do classificador em fluxos de dados. Ele gera um fluxo binário de erros, atribuindo o valor 0 a classificações corretas e 1 a incorretas. Este fluxo é analisado pelo estimador *Exponentially Weighted Moving Average* (EWMA), sensível a mudanças recentes, em comparação a uma média simples de longo prazo, que estima a taxa de erro estável. Um desvio de conceito é identificado quando a diferença entre a estimativa do EWMA e a média de longo prazo excede um limiar predefinido.

- **SEED (do inglês, *Stream Ensemble for Early Drift Detection*) [69]:** algoritmo que utiliza blocos de instâncias de dados como unidade básica, ao invés de instâncias individuais. Cada bloco armazena uma sequência binária, onde 0 indica uma classificação correta e 1 representa uma classificação incorreta. O algoritmo mantém uma janela deslizante W com o bloco mais recente, que é dividida em uma subjanela esquerda (W_l), com instâncias mais antigas, e uma subjanela direita (W_r), com instâncias mais recentes. Se a diferença entre as médias das subjanelas for superior a um limiar calculado pela Desigualdade de Hoeffding com correção de Bonferroni, um desvio é detectado e a W_l é descartada. O SEED também realiza compressão de blocos, fundindo blocos adjacentes que possuem distribuições semelhantes. Isso reduz pontos de corte desnecessários, aumentando a eficiência do algoritmo.

- **PL (do inglês, *Paired Learners*) [70]:** combina um aprendiz estável S, que efetua as predições com base em toda a experiência obtida, e um aprendiz reativo R_w, com conhecimento limitado a uma janela de dados recentes de tamanho w. Um desvio é detectado através da comparação entre as classificações do S e do R_w. Para cada exemplo em que S erra e R_w acerta, um bit é marcado em uma lista circular C. Se a proporção de bits marcados ultrapassar um limiar θ, um desvio é identificado. Nesse caso, S é substituído e seu conhecimento é redefinido com base no conhecimento do R_w.

3.2.2 Métodos Baseados na Distribuição de Dados

Os métodos de detecção baseados na distribuição de dados identificam desvios de conceito por meio da comparação entre dados históricos e atuais. Nessa abordagem, é comum o uso de técnicas baseadas em janelas, que permitem calcular as mudanças na dis-

tribuição de dados em diferentes instantes temporais. Dentre os algoritmos desenvolvidos com base nessa abordagem, podem ser citados:

- **ADWIN (do inglês, *Adaptive Windowing*) [71]:** algoritmo que utiliza uma janela deslizante adaptativa W para armazenar e analisar dados. A janela é particionada em duas subjanelas, W_0 e W_1. Se a distribuição dos dados em W for estacionária, as médias das subjanelas permanecem próximas. À medida que novos dados são recebidos, o tamanho de W aumenta. Um desvio é detectado quando a diferença absoluta entre as médias de W_0 e W_1 excede um limiar predefinido. Nesse caso, a subjanela mais antiga, W_0, é descartada, reduzindo o tamanho de W. Todas as possíveis partições da janela são verificadas até que a condição de estabilidade seja restabelecida. O ADWIN2 foi desenvolvido para reduzir o custo computacional do ADWIN. Esse algoritmo utiliza uma variação do histograma exponencial como estrutura de dados, permitindo armazenar e gerenciar as amostras de forma comprimida, reduzindo o uso de memória e o tempo de processamento.

- **EDE (do inglês, *Equal Density Estimation*) [72]:** no EDE, a medida *DensityScale* é calculada para quantificar as diferenças locais de densidade entre duas janelas consecutivas. Essa medida é então integrada para verificar se há uma tendência geral de aumento da *DensityScale* nas sub-regiões do espaço de características. O resultado desse processo é um valor global que representa a diferença entre as distribuições. Um teste estatístico não paramétrico baseado em permutação é aplicado para avaliar se esse valor excede um limiar predefinido, indicando a ocorrência de um desvio de conceito.

- **LSDD-CDT (do inglês, *Least Squares Density Difference-based Change Detection Test*) [73]:** estima a diferença de densidade por mínimos quadrados entre uma janela de referência Z_p, que representa a condição estável do sistema, e uma janela de teste Z_q, composta pelos dados mais recentes. A cada nova instância de dado, a janela Z_q é atualizada. A comparação entre as duas janelas gera um valor de dissimilaridade $\hat{D}^2$, que quantifica a diferença entre as distribuições. A detecção de desvio é realizada por meio de um mecanismo de limiar de três níveis: alerta, mudança e seguro. No nível de alerta, a atualização da janela de referência é temporariamente suspensa. No nível de mudança, registram-se os pontos correspondentes ao alerta e à mudança. No nível seguro, o estado de alerta é desativado, e a janela de referência é atualizada utilizando amostragem por reservatório, combinando dados antigos e novos.

- **LSDD-INC (do inglês, *Incremental LSDD-Based Change Detection Test*) [74]:** assim como o LSDD-CDT, o LSDD-INC estima a diferença de densidade por mínimos quadrados ($\hat{D}^2$) entre uma janela de referência Z_p (amostras estáveis) e

uma janela de teste Z_q (amostras recentes). A cada nova instância processada, a janela Z_q é atualizada, e o valor de $\hat{D}^2$ é recalculado de forma incremental, reduzindo a carga computacional do processo. O valor estimado $\hat{D}^2$ é então comparado a um limiar adaptativo $T_{u'}$. Quando $\hat{D}^2 > T_{u'}$, considera-se que a diferença entre as distribuições de probabilidade das janelas é estatisticamente significativa, caracterizando a ocorrência de um desvio de conceito.

- **OnePassSampler [75]:** identifica desvios de conceito por meio da comparação entre dois blocos de dados: o bloco esquerdo, que armazena os dados referentes ao conceito estável, e o bloco direito, que contém os dados mais recentes. Um teste estatístico é aplicado para verificar se há uma diferença significativa entre as médias das amostras dos dois blocos. Para acelerar o processo e reduzir a dependência entre os dados, as médias são calculadas a partir de amostras aleatórias de tamanho igual em cada bloco. A significância da diferença é avaliada utilizando o Limite de Bernstein. Quando a diferença entre as médias excede o limiar predefinido, um desvio de conceito é detectado. Nesse caso, os dados do bloco esquerdo são descartados, e os dados do bloco direito são transferidos para o esquerdo, representando o novo conceito. Caso não seja detectado um desvio de conceito, os dados do bloco direito são adicionados ao bloco esquerdo, estendendo o período de estabilidade.

3.2.3 Métodos Baseados em Múltiplas Hipóteses

A abordagem baseada em múltiplas hipóteses emprega diversos testes estatísticos com o objetivo de obter uma detecção de desvio de conceito mais abrangente e confiável, podendo adotar uma metodologia paralela ou hierárquica. Nos testes paralelos de hipóteses múltiplas, diferentes testes estatísticos são executados simultaneamente, cada um monitorando um aspecto distinto do modelo ou dos dados. A detecção do desvio ocorre a partir da combinação de seus resultados. Nos testes hierárquicos de hipóteses múltiplas, os testes são organizados em camadas, de modo que o resultado de um teste mais abrangente, em um nível superior, pode acionar análises mais específicas nos níveis seguintes. Com base nessa estratégia, podem ser mencionados os seguintes algoritmos:

- **DDE (do inglês, *Drift Detection Ensemble*) [76]:** algoritmo que combina três detectores de desvio de conceito distintos, com o objetivo de aprimorar a precisão da detecção, aproveitando as vantagens individuais de cada método em diferentes cenários. Para que um alerta de desvio de conceito seja acionado, é necessário que um número mínimo de detectores, definido por um parâmetro de sensibilidade, identifique a ocorrência do fenômeno. Essa abordagem permite criar diferentes combinações de métodos de detecção, ajustando-se também o valor do parâmetro de sensibilidade. Os detectores empregados nas diferentes configurações incluem o DDM, o ADWIN, HDDM e o ECDD.

- **LFR (do inglês, *Linear Four Rates*) [77]:** método de detecção que monitora quatro taxas derivadas da matriz de confusão de um classificador: a taxa de verdadeiros positivos, a taxa de verdadeiros negativos, o valor preditivo positivo e o valor preditivo negativo. Um desvio é detectado quando a hipótese nula, que assume a estabilidade do conceito, é rejeitada. Isso ocorre quando é observada uma variação significativa em qualquer uma das quatro taxas monitoradas, de acordo com os limites definidos para os níveis de aviso e de detecção. Esses limites são determinados por meio de simulações de Monte Carlo. O nível de aviso inicia o armazenamento das instâncias subsequentes, enquanto o nível de detecção confirma o desvio e aciona o retreinamento do classificador com os dados acumulados após o nível de aviso.

- **HLFR (do inglês, *Hierarchical Linear Four Rates*) [78]:** o HLFR apresenta uma arquitetura hierárquica composta por duas camadas de testes de hipóteses. A primeira camada utiliza o algoritmo LFR para monitorar continuamente o fluxo de dados e identificar potenciais desvios. A premissa é que, se o conceito dos dados permanece estável, as quatro métricas derivadas da matriz de confusão de um classificador (taxa de verdadeiros positivos, taxa de verdadeiros negativos, valor preditivo positivo e valor preditivo negativo) também permanecem estáveis. Uma mudança em qualquer uma das taxas que exceda o limiar predefinido indica um desvio potencial e aciona a segunda camada. A função da segunda camada é validar as detecções realizadas pela camada anterior, reduzindo a ocorrência de falsos positivos. Para isso, aplica-se o teste de permutação de perda zero-um, utilizando segmentos de dados anteriores e posteriores ao ponto de desvio potencial. O classificador é avaliado com uma divisão ordenada dos dados e com divisões embaralhadas. Caso nenhum desvio de conceito tenha ocorrido, a taxa de erro da divisão ordenada não deve se desviar significativamente da distribuição de erros das divisões embaralhadas. A significância dessa diferença é avaliada por um teste de permutação. O p-valor é calculado como a fração de simulações embaralhadas em que o erro foi igual ou pior que o erro da divisão ordenada. Se o p-valor, que representa a probabilidade do erro da divisão ordenada ter sido obtido por acaso, for menor que o limiar predefinido η, o desvio é confirmado.

As diferentes estratégias de detecção de desvio de conceito apresentadas, incluindo abordagens baseadas na taxa de erro, na distribuição dos dados e em múltiplas hipóteses, estão sintetizadas na Tabela 1. Os métodos baseados na taxa de erro tendem a ser menos complexos, detectando o desvio de conceito de forma mais objetiva. Os algoritmos fundamentados na distribuição dos dados permitem identificar desvios mesmo em cenários sem rótulos imediatos. As abordagens baseadas em múltiplas hipóteses oferecem uma análise mais robusta e abrangente, por combinar diferentes técnicas de detecção. A escolha da técnica mais adequada está diretamente relacionada com o contexto de aplica-

ção e com as características do fluxo de dados. Uma vez detectado o desvio de conceito, torna-se necessário adotar mecanismos de adaptação capazes de atualizar o modelo de forma eficiente.

Tabela 1 – Algoritmos de detecção de desvio de conceito. Fonte: elaboração própria.

Algoritmo	**Categoria**	**Ano**
DDM	Taxa de erro	2004
EDDM	Taxa de erro	2006
RDDM	Taxa de erro	2017
HDDM	Taxa de erro	2014
FHDDM	Taxa de erro	2016
ECDD	Taxa de erro	2012
SEED	Taxa de erro	2012
PL	Taxa de erro	2012
ADWIN	Distribuição dos dados	2007
ADWIN2	Distribuição dos dados	2007
EDE	Distribuição dos dados	2016
LSDD-CDT	Distribuição dos dados	2018
LSDD-INC	Distribuição dos dados	2017
OnePassSampler	Distribuição dos dados	2013
DDE	Múltiplas hipóteses	2015
LFR	Múltiplas hipóteses	2015
HLFR	Múltiplas hipóteses	2017

3.3 Adaptação ao Concept Drift

Devido à natureza não estacionária dos dados, frequentemente observada em ambientes reais, modelos de aprendizado de máquina precisam incorporar mecanismos adaptativos para preservar seu desempenho e confiabilidade ao longo do tempo. Essas estratégias de adaptação podem atuar de maneira ativa, quando a adaptação é acionada a partir da detecção explícita de um desvio, ou passiva, quando ocorre de forma contínua, antecipando possíveis mudanças sem recorrer a detectores de desvio [61].

3.3.1 Estratégias Gerais de Adaptação

As estratégias adaptativas podem ser classificadas em três grupos principais, de acordo com a forma de atualização dos modelos: retreinamento do modelo, métodos baseados em conjuntos e ajuste do modelo existente [18]. Essas técnicas não são exclusivas de abordagens ativas ou passivas, podendo ser aplicadas em ambos os contextos.

3.3.1.1 Retreinamento do Modelo

O retreinamento é a forma mais direta de adaptação ao desvio de conceito. Nessa estratégia, dados recentes são incorporados ao processo de treinamento de um novo modelo ajustado ao conceito atual, que substitui o modelo obsoleto. O retreinamento pode ser

acionado por um detector de desvio, que indica o momento adequado para a atualização, ou realizado de forma periódica, independentemente da detecção explícita de mudanças.

3.3.1.2 Métodos Baseados em Conjunto

Os métodos baseados em conjunto utilizam a combinação da saída de múltiplos classificadores para definir a classificação final. Cada modelo pode possuir diferentes tipos, parâmetros ou especializações. O gerenciamento desse conjunto permite operações como adicionar, remover, retreinar ou atualizar os modelos base, tornando a abordagem flexível e robusta diante de diferentes padrões de mudança.

3.3.1.3 Ajuste do Modelo Existente

O ajuste dos modelos existentes utiliza algoritmos de aprendizado adaptativo, como modelos baseados em árvores de decisão capazes de analisar e modificar sub-regiões específicas do modelo. Essa abordagem é caracterizada pela capacidade de atualização dinâmica à medida que a distribuição dos dados se altera, sem a necessidade de um retreinamento completo, buscando equilibrar a eficiência computacional e a capacidade de adaptação.

3.3.2 Adaptação Ativa

A abordagem ativa representa um dos tipos de adaptação ao desvio de conceito, seguindo a lógica de detectar e reagir, como representado pelo fluxograma da Figura 8. Essa estratégia caracteriza-se pela presença de um módulo específico de detecção de desvios, que monitora continuamente o modelo ou o fluxo de dados, acionando o mecanismo de atualização quando uma mudança é identificada [79]. Uma vez confirmado o desvio, aplicam-se as estratégias gerais de adaptação discutidas anteriormente, com o objetivo de adequar o modelo às características atuais dos dados. Alguns exemplos de algoritmos com abordagens ativas são:

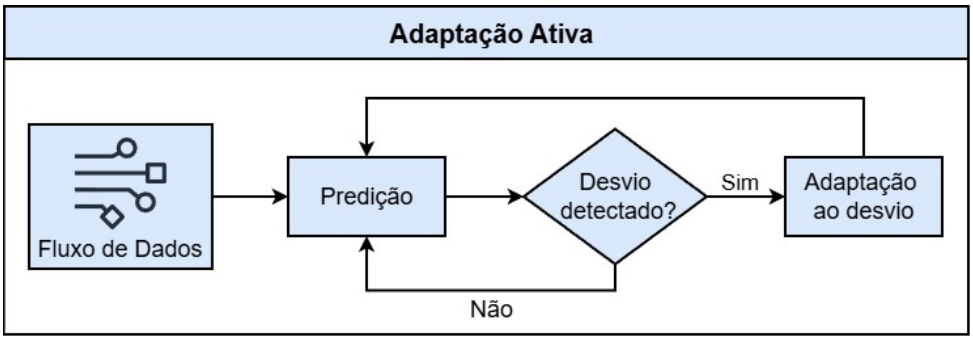

Figura 8 – Funcionamento geral da estratégia ativa de adaptação ao desvio de conceito. Fonte: elaboração própria.

- **HAT (do inglês, *Hoeffding Adaptive Tree*) [80]:** árvore de decisão desenvolvida para a classificação de fluxos de dados contínuos em cenários dinâmicos. Esse algoritmo aplica o limite de Hoeffding para determinar quantos exemplos são necessários para escolher o melhor atributo na divisão de um nó. Esse limite garante, com alta probabilidade, que o atributo escolhido com a menor amostra possível seria o mesmo selecionado utilizando-se infinitos exemplos. Quando um desvio é identificado por um detector explícito em um nó, uma árvore alternativa inicia seu crescimento naquele ponto, utilizando os dados mais recentes. Caso o desempenho da árvore alternativa seja superior ao da subárvore atual, ela a substitui.

- **ARF (do inglês, *Adaptive Random Forest*) [81]:** adaptação da *Random Forest*, projetada para a classificação de fluxos de dados em evolução. Esse algoritmo utiliza as Árvores de Hoeffding como classificadores base e associa a cada uma delas um detector de desvio. Quando um alerta de desvio é sinalizado, uma árvore de segundo plano é criada e treinada com novas instâncias. Caso o desvio seja confirmado, a árvore original é substituída por sua respectiva árvore de segundo plano.

- **SRP (do inglês, *Streaming Random Patches*) [82]:** método de conjunto projetado para classificação em fluxos de dados em evolução. Cada modelo base é treinado com um subconjunto aleatório de instâncias e características, a fim de aumentar a diversidade entre os modelos. Para se adaptar às mudanças na distribuição dos dados, um detector de desvio de conceito, como o ADWIN, é empregado para monitorar o desempenho de cada modelo base. Quando um alerta é sinalizado pelo detector, um novo modelo base inicia o treinamento em segundo plano. Caso o desvio de conceito seja confirmado, o modelo em questão é substituído pelo modelo de segundo plano.

3.3.3 Adaptação Passiva

Os algoritmos passivos não apresentam módulos explícitos de detecção de desvios, como ilustrado no fluxograma inferior da Figura 9. Nessa abordagem, a atualização do modelo é realizada de forma contínua, independentemente da ocorrência de mudanças no conceito [79]. Diversas técnicas foram desenvolvidas com o propósito de manter o sistema atualizado em cenários dinâmicos. São exemplos dessa categoria:

- **CVFDT (do inglês, *Concept-adapting Very Fast Decision Tree learner*) [83]:** algoritmo baseado em árvores de decisão, desenvolvido como uma extensão do VFDT (do inglês, *Very Fast Decision Tree learner*) para lidar com o desvio de conceito. Ele mantém uma janela deslizante de tamanho fixo contendo os exemplos mais recentes. Cada nova instância processada é adicionada à janela e suas informações são utilizadas para atualizar as estatísticas dos nós da árvore, enquanto o exemplo mais antigo é removido e suas estatísticas são decrementadas dos nós

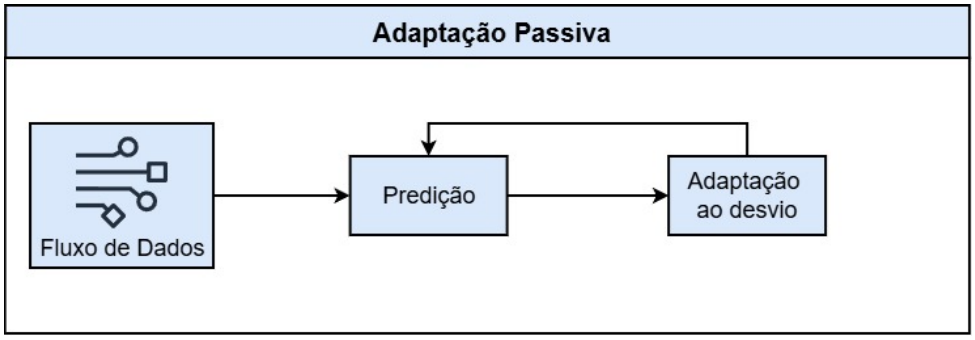

Figura 9 – Funcionamento geral da estratégia passiva de adaptação ao desvio de conceito. Fonte: elaboração própria.

pelos quais passou. O algoritmo analisa periodicamente os nós internos da árvore para verificar se o atributo de divisão permanece o mais adequado de acordo com a janela atual. Caso outro atributo se torne significativamente melhor, segundo os limites de Hoeffding, o CVFDT inicia o crescimento de uma subárvore alternativa, que substitui a original quando apresenta desempenho superior.

- **AWE (do inglês, *Accuracy-Weighted Ensembles*) [84]:** utiliza um conjunto de classificadores ponderados, em que, a cada lote de dados processado, um novo classificador base é treinado e seu peso é definido de acordo com o desempenho obtido no lote mais recente, refletindo a distribuição atual dos dados. Para limitar o crescimento do conjunto, o AWE mantém apenas os K classificadores com melhor acurácia. A classificação final de uma instância é obtida por meio da média ponderada das previsões de todos os membros do conjunto.

- **DWM (do inglês, *Dynamic Weighted Majority*) [85]:** mantém um conjunto de modelos de aprendizado, cada um associado a um peso que reflete seu desempenho histórico. Esses classificadores são atualizados incrementalmente a cada novo exemplo processado. Quando um modelo comete um erro de classificação, seu peso é reduzido, diminuindo sua influência sobre o conjunto. Modelos com peso inferior a um limiar predefinido são removidos. Caso a predição global do conjunto, resultante da votação ponderada dos classificadores, esteja incorreta, um novo modelo é criado e incorporado, permitindo ao DWM aprender novos conceitos.

- **SVMLearn++ (do inglês, *Support Vector Machine with Learn++*) [86]:** abordagem de aprendizado incremental baseada em um conjunto de classificadores SVM (do inglês, *Support Vector Machine*), projetada para ambientes dinâmicos. O fluxo de dados é processado em lotes, onde cada lote é utilizado para gerar subconjuntos de treino e teste. Um classificador base é treinado e avaliado, e os pesos das instâncias são atualizados conforme os erros cometidos. Instâncias classificadas incorretamente recebem pesos maiores, forçando os classificadores subsequentes a

focarem nos conceitos ainda não aprendidos. Um mecanismo de esquecimento é aplicado para remover modelos obsoletos do conjunto à medida que novos membros são adicionados. Entre as estratégias de manutenção estão manter apenas os N classificadores com melhor desempenho ou descartar aqueles cujo erro ultrapasse um limiar predefinido. A classificação final de uma instância é obtida pela votação majoritária ponderada de todos os classificadores do conjunto.

- **Learn++.NSE (do inglês, *Learn++ with Non-Stationary Ensemble*) [87]:** algoritmo baseado em conjunto, projetado para aprendizado incremental em ambientes não estacionários. Cada lote processado tem seus dados classificados pelos membros do conjunto e, com base no erro geral do conjunto, são atribuídos pesos às instâncias. As amostras classificadas incorretamente recebem pesos maiores, gerando uma distribuição de penalidade que destaca exemplos de difícil classificação. Um novo classificador é então treinado, e todos os modelos do conjunto são reavaliados, recebendo pesos de acordo com o desempenho recente. O voto final de cada classificador é definido a partir de sua média ponderada de desempenho, privilegiando aqueles mais precisos nos últimos lotes. A classificação final é obtida por meio da votação majoritária ponderada do conjunto.

- **Learn++.SMOTE (do inglês, *Learn++ with Synthetic Minority Over-sampling Technique*) [88]:** integra os algoritmos Learn++.NSE e SMOTE (do inglês, *Synthetic Minority Over-sampling Technique*) para lidar simultaneamente com dados não estacionários e desbalanceados. Cada bloco do fluxo de dados é utilizado para avaliar o conjunto de classificadores, atualizando os pesos das instâncias. As amostras classificadas incorretamente recebem pesos maiores, criando uma distribuição que atribui mais importância aos exemplos de difícil classificação para o conjunto. O SMOTE é aplicado para gerar amostras sintéticas da classe minoritária, equilibrando a distribuição de classes. Em seguida, um novo classificador base é treinado com o bloco real de dados e os exemplos sintéticos. Se o novo modelo classificador apresentar um erro superior a 0,5, ele será descartado e outro será criado em seu lugar. Classificadores antigos cujo erro ultrapasse 0,5 têm seu voto anulado, fixando o erro em 0,5, o que corresponde a um desempenho equivalente a um palpite aleatório. A preservação desses classificadores no conjunto permite sua reutilização em cenários cíclicos. O peso final de cada classificador é calculado pela soma ponderada de seus erros normalizados ao longo do tempo, suavizada por uma função logística sigmoide. A classificação final é obtida pela votação majoritária ponderada, em que os classificadores com melhor desempenho recente exercem maior influência.

- **AUE2 (do inglês, *Accuracy Updated Ensemble 2*) [89]:** algoritmo que utiliza um conjunto de classificadores para fluxos de dados, projetado para reagir de forma eficiente ao desvio de conceito. Ele combina a natureza incremental das Árvores

de Hoeffding com um mecanismo de ponderação baseado em acurácia. O fluxo de dados é processado em blocos de tamanho fixo e, a cada novo bloco, os classificadores do conjunto são avaliados e seus pesos recalculados a partir das estimativas de erro de predição. Um novo classificador é treinado e substitui o membro de pior desempenho. Todos os classificadores são atualizados incrementalmente com os dados mais recentes, permitindo que modelos mais antigos também se ajustem ao conceito corrente.

As estratégias de adaptação ao desvio de conceito oferecem diferentes formas de preservar o desempenho dos modelos em ambientes dinâmicos. As abordagens ativas utilizam detectores explícitos para identificar o momento em que a adaptação é necessária, enquanto as passivas realizam atualizações contínuas, sem a necessidade de um módulo de detecção. A Tabela 2 sintetiza as principais características dos algoritmos passivos discutidos.

Tabela 2 – Algoritmos de adaptação ao desvio de conceito. Fonte: elaboração própria.

Algoritmo	**Estratégia de Adaptação**	**Abordagem**
CVFDT	Ajuste de modelo existente	Passiva
AWE	Conjunto	Passiva
DWM	Conjunto	Passiva
SVMLearn++	Conjunto	Passiva
Learn++.SMOTE	Conjunto	Passiva
Learn++.NSE	Conjunto	Passiva
AUE2	Conjunto	Passiva
HAT	Ajuste de modelo existente	Ativa
ARF	Conjunto	Ativa
SRP	Conjunto	Ativa

3.3.4 *DenStream*

O *DenStream* [90] é um algoritmo de clusterização baseado em densidade, projetado para operar em fluxos de dados contínuos e em evolução. Seu principal objetivo é superar os desafios de memória limitada, processamento em uma única passagem e a natureza dinâmica dos dados. Para lidar com a restrição de memória, em vez de armazenar todos os pontos de dados, o *DenStream* utiliza *Micro-Clusters* (MC) para criar, excluir e atualizar *clusters* dinamicamente.

Os MC são estruturas de resumo, definidas como $mc_i = \{\overline{CF_i^1}, \overline{CF_i^2}, w_i\}$, onde $\overline{CF_i^1}$ é a soma linear ponderada dos pontos, $\overline{CF_i^2}$ é a soma quadrática ponderada dos pontos e w_i é a soma dos pesos das instâncias. O centroide do MC é definido como $c_{mc_i} = \frac{\overline{CF_i^1}}{w_i}$ e o raio como $r_{mc_i} = \sqrt{\frac{|\overline{CF_i^2}|}{w_i} - \left(\frac{|\overline{CF_i^1}|}{w_i}\right)^2}$. Os MC são classificados em três tipos, de acordo com seu peso:

- **C-MC (do inglês, *Core Micro-Cluster*):** representa um grupo denso e estável de pontos próximos, cujo peso é superior ou igual ao limiar μ.
- **P-MC (do inglês, *Potential Micro-Cluster*):** corresponde a um estágio inicial de formação de um C-MC, apresentando um peso maior ou igual ao limiar $\beta\mu$.
- **O-MC (do inglês, *Outlier Micro-Cluster*):** MC com peso inferior ao limiar $\beta\mu$, representando ruídos ou o início potencial de novos agrupamentos.

Durante a fase *online* de manutenção dos MC, P-MC e O-MC são atualizados de forma incremental, garantindo que o modelo se adapte continuamente à chegada de novos dados. Quando um novo ponto é processado, o algoritmo tenta incorporá-lo ao P-MC mais próximo e, caso isso não seja possível, ao O-MC mais próximo. A fusão somente ocorre se o novo raio do MC não exceder o limite ϵ. Caso a fusão não seja possível, um novo O-MC é criado. Os O-MC cujo peso ultrapassa o limiar $\beta\mu$ são promovidos a P-MC. Já os P-MC que não recebem novos pontos têm seus pesos reduzidos gradualmente por meio da aplicação de um fator de decaimento exponencial. Isso permite eliminar os agrupamentos que deixam de representar o comportamento atual dos dados. Os O-MC com peso inferior ao limiar ξ são descartados por não apresentarem potencial de evolução. Esse limiar é definido com base no tempo de existência do O-MC, exigindo que ele atraia mais pontos para sobreviver à medida que envelhece.

A criação de *clusters* finais do *DenStream* é um processo executado sob demanda, que utiliza os P-MC ativos para realizar a clusterização. Cada P-MC é tratado como um ponto virtual, posicionado em seu centro e com seu respectivo peso. O algoritmo aplica uma variação do algoritmo DBSCAN (do inglês, *Density-Based Spatial Clustering of Application with Noise*) para agrupar esses pontos virtuais com base no conceito de conectividade por densidade. Dois MC são considerados diretamente conectados se o peso de um deles for maior que o limiar μ (agindo como um C-MC) e se a distância entre seus centroides for menor ou igual a duas vezes o limite ϵ. Dessa forma, os *clusters* finais são produzidos pelo agrupamento dos P-MC que são densamente conectados através de uma cadeia de conexões.

4 TRABALHOS RELACIONADOS

Técnicas de aprendizado de máquina e aprendizado profundo têm sido amplamente empregadas em sistemas de detecção de anomalias e intrusões em redes de computadores. Entretanto, a ocorrência de desvio de conceito representa um desafio para a aplicabilidade desses modelos em cenários reais, uma vez que a natureza dinâmica do tráfego de rede e a constante evolução dos ataques podem comprometer seu desempenho ao longo do tempo. Para mitigar esse fenômeno, diversas estratégias adaptativas de detecção de anomalias em redes de computadores têm sido propostas na literatura, abrangendo tanto abordagens ativas quanto passivas.

Xu Wang *et al.* [91] propuseram um sistema de detecção de tráfego anômalo projetado para tolerar o desvio de conceito em ambientes IoT (do inglês, *Internet of Things*). O sistema emprega um módulo de detecção baseado em conjunto, combinando três modelos de aprendizado profundo para analisar o tráfego de rede sob diferentes perspectivas: uma CNN para extrair características espaciais, uma GRU para capturar dependências temporais e uma PNN (do inglês, *Probabilistic Neural Network*) para extrair características estatísticas. Os resultados dos três modelos são processados por uma camada de ponderação de saída, composta por duas camadas totalmente conectadas, responsáveis por atribuir pesos distintos às saídas de cada submodelo. Quando todos os submodelos concordam no mesmo resultado de classificação, o dado analisado é considerado confiável e é rotulado. A partir de um conjunto de dados rotulados, realiza-se um treinamento incremental, permitindo que os modelos aprendam a distribuição mais recente sem esquecer o conhecimento previamente adquirido.

Lijuan Xu *et al.* [92] desenvolveram uma estrutura chamada CDDIA (do inglês, *Concept Drift Detection, Interpretation and Adaptation*), que aborda o problema de desvio de conceito na detecção de anomalias em IoT. Inicialmente, um *Autoencoder* é treinado com um conjunto de dados históricos. O erro de reconstrução, que representa a saída do modelo, é utilizado na aplicação de uma técnica de calibração para obter estimativas de probabilidade mais precisas. As distribuições de probabilidade calibradas das amostras antigas e recentes são comparadas utilizando histogramas de frequência, gerando distribuições discretas. A divergência de Kullback-Leibler (KL) é aplicada para medir a diferença entre as duas distribuições. Em seguida, um teste de permutação é realizado para calcular o p-valor, que indica a significância estatística dessa diferença. Se o p-valor ultrapassar um limiar predefinido, o sistema sinaliza a ocorrência de um desvio de conceito. A interpretação do desvio é feita em nível de amostra e em nível de característica. No nível de amostra, identificam-se quais instâncias específicas são as principais causadoras do desvio. Por meio de um algoritmo de busca, as amostras são classificadas em três

categorias: amostras antigas desatualizadas, amostras antigas não desatualizadas e novas amostras com desvio. No nível de características, para as novas amostras com desvio, a técnica SHAP (do inglês, *SHapley Additive exPlanations*) é empregada para identificar quais características mais contribuíram para a mudança. Com base nos resultados da interpretação em nível de amostra, um novo conjunto de treinamento é formado combinando as amostras antigas não desatualizadas com as novas amostras que apresentam desvio. O modelo de detecção de anomalias é então retreinado, aprendendo a nova distribuição normal dos dados sem esquecer totalmente os padrões antigos ainda relevantes.

Hanli Qiao *et al.* [93] apresentaram um método para aprimorar a detecção de ciberataques de *botnets* em cenários de IoT, adaptando-se ao desvio de conceito. O sistema proposto baseia-se na premissa de que a ocorrência de um desvio de conceito resulta em um aumento súbito na quantidade de anomalias. Para isso, a Projeção de Subespaço Residual é empregada para calcular o número de instâncias anômalas no conjunto de amostras analisado. Uma janela dinâmica é utilizada para armazenar as amostras atuais conforme são processadas, em conjunto com uma janela deslizante de tamanho fixo, que coleta os dados mais recentes do fluxo. O desvio de conceito é detectado pela comparação da quantidade de instâncias anômalas entre as duas janelas, e, quando identificado, o modelo é retreinado para adaptar-se ao novo conceito.

Omar Abdel Wahab *et al.* [94] propuseram um sistema de detecção de intrusão em ambientes IoT com módulos de detecção e adaptação ao desvio de conceito. A Análise de Componentes Principais é aplicada para medir e comparar a variância das características dos dados antigos e recentes, identificando um desvio caso a diferença entre os dois conjuntos ultrapasse um limiar predefinido. Para a adaptação, é utilizada uma Rede Neural Profunda Online que emprega o mecanismo de ponderação Hedge para atribuir um peso e um classificador de saída a cada camada oculta. Após uma previsão, o peso de cada classificador é ajustado com base no erro observado, e a predição final é obtida através de uma agregação ponderada, atribuindo maior importância às camadas com melhor desempenho.

Meenal Jain *et al.* [95] apresentaram uma estrutura híbrida para detecção de anomalias em tráfego de rede, projetada para lidar com o desvio de conceito. O sistema combina a clusterização *K-Means* com um classificador SVM. O tráfego de rede é armazenado em janelas deslizantes, sobre as quais a clusterização é aplicada para modelar os dados. A detecção de desvio é realizada por meio de duas técnicas: uma baseada na taxa de erro e outra na distribuição dos dados. A primeira monitora a taxa de erro do classificador SVM, onde um aumento pode indicar um possível desvio. A segunda técnica mede a dissimilaridade entre janelas consecutivas utilizando a divergência de KL como teste estatístico para identificar uma mudança na distribuição. Quando um desvio é detectado, a adaptação do sistema é realizada por meio do retreinamento do classificador SVM com

um conjunto de dados atualizado.

Zinuo Yin *et al.* [96] propuseram uma estrutura incremental e adaptativa de detecção de intrusão em redes IoT. A estrutura utiliza um *Autoencoder* Contrativo para aprender representações latentes de baixa dimensão do tráfego e a teoria do valor extremo para analisar características estatísticas dos dados, servindo como uma análise auxiliar. Ambas as técnicas classificam as amostras de teste, e aquelas que recebem rótulos consistentes de ambos os modelos são adicionadas ao conjunto de treinamento com pseudo-rótulos. O *Autoencoder* é retreinado periodicamente com os dados históricos e os dados recentes pseudo-rotulados, permitindo que se mantenha atualizado enquanto preserva o conhecimento prévio.

Christopher Nixon *et al.* [97] apresentaram o SALAD (do inglês, *Split Active Learning Anomaly Detector*), um método de detecção de anomalias semi-supervisionado projetado para identificar ataques cibernéticos. O sistema utiliza um *Autoencoder* para aprender a representação do comportamento normal da rede. A detecção de anomalias é realizada pela comparação entre o erro de reconstrução de uma amostra e um limiar predefinido. O SALAD introduz duas abordagens para ajustar esse limiar em tempo real: o *Stochastic Anomaly Threshold with Fading Factor* e o *Adaptive Anomaly Threshold*. Esses métodos permitem ao sistema adaptar-se a mudanças graduais e abruptas no fluxo de dados. Para reduzir a necessidade de rotulagem manual de dados e os custos associados, é empregada uma estrutura de aprendizado ativo. O sistema adota uma estratégia de consulta dividida que combina uma seleção aleatória e uma seleção baseada em incerteza, que seleciona amostras com base na distância entre seu erro de reconstrução e o limiar de anomalia. O DDM é utilizado para detectar desvios de conceito. Quando um sinal de alerta é emitido, um novo *Autoencoder* é treinado com os dados mais recentes. Se um desvio de conceito for confirmado, o modelo atual é substituído pelo novo.

Muhammad Amin *et al.* [98] desenvolveram um sistema de detecção de *malware* em fluxos de dados de IoT, projetado para lidar com o desvio de conceito. Os métodos estatísticos aplicados para identificar mudanças nos padrões de dados incluem o DDM, EDDM, *Adaptive Classifiers-Ensemble* e *Detection Method Using Statistical Testing*. Após serem processados pelo módulo de detecção de desvio, os dados são encaminhados aos algoritmos de aprendizado de máquina para classificação. Os classificadores utilizados no estudo foram IB1 (do inglês, *Instance-Based 1-Nearest Neighbor*), NB (do inglês, *Naive Bayes*), SVM e LSTM. Se o *F1-score* diminuir além de um limiar predefinido, ou se o detector de desvio indicar a necessidade, é acionado um processo de rotulagem dos dados e retreinamento do modelo. Dessa forma, o sistema se adapta às mudanças na distribuição dos dados, preservando seu desempenho.

Yafeng Wu *et al.* [99] propuseram um sistema de aprendizado de conjunto adaptável ao desvio de conceito. Quatro métodos de aprendizado *online* são empregados para

realizar previsões: OPA (do inglês, *Online Passive-Aggressive*), KNN-ADWIN (do inglês, *K-Nearest Neighbors with Adaptive Windowing*), ARF-ADWIN (do inglês, *Adaptive Random Forest with Adaptive Windowing*) e ARF-EDDM (do inglês, *Adaptive Random Forest with Early Drift Detection Method*). O OPA ajusta seus pesos em tempo real conforme os dados são processados. O KNN-ADWIN combina o algoritmo KNN (do inglês, *K-Nearest Neighbors*) com o detector de desvio de conceito ADWIN, respondendo dinamicamente às mudanças nos padrões dos dados. O ARF-ADWIN e o ARF-EDDM são variantes do ARF que incorporam, respectivamente, os detectores ADWIN e EDDM para detectar desvios e adaptar as árvores de decisão do conjunto. As previsões desses modelos são combinadas atribuindo pesos a cada classificador de acordo com seu desempenho recente. Esses pesos são calculados por meio da Média Móvel Exponencialmente Ponderada da taxa de erro. A Otimização por Enxame de Partículas é utilizada para ajustar o fator de decaimento dos pesos, definindo a importância relativa dos erros históricos em relação aos recentes. Essas características permitem que o sistema se adapte continuamente às mudanças no fluxo de dados.

Firoz Khan *et al.* [100] desenvolveram um sistema de detecção de ataques em ambientes IoT, projetado para superar as limitações associadas ao desvio de conceito. Uma abordagem de aprendizado de máquina baseada em conjunto é empregada, utilizando árvores de decisão como classificadores base. O desempenho individual das árvores é avaliado continuamente, de modo que classificações incorretas resultam na redução de seus pesos de influência. Árvores com peso inferior a um limiar predefinido são removidas do conjunto. Caso a classificação final, obtida pela soma ponderada das previsões de todas as árvores, esteja incorreta, o número total de classificadores base é incrementado, e uma nova árvore de decisão treinada com os dados mais recentes é adicionada ao conjunto.

Observa-se uma tendência crescente na literatura em tratar explicitamente o problema do desvio de conceito. Os trabalhos analisados evidenciam avanços significativos na detecção de anomalias em ambientes dinâmicos, especialmente em redes de computadores. Diversas estratégias de adaptação têm sido exploradas, tanto ativas quanto passivas, abrangendo métodos baseados em conjunto, aprendizado *online*, mecanismos de ponderação e técnicas híbridas. A maioria dos trabalhos abordados emprega classificadores baseados em aprendizado profundo e adota o retreinamento do modelo como estratégia de adaptação. A Tabela 3 apresenta uma síntese dos principais trabalhos discutidos neste capítulo.

Tabela 3 – Trabalhos recentes de adaptação ao desvio de conceito. Fonte: elaboração própria.

Referência	**Classificador**	**Tipo de Adaptação**	**Estratégia de Adaptação**
[91]	CNN + GRU + PNN	Passiva	Conjunto + Ajuste de modelo existente
[92]	AutoEncoder	Ativa	Retreinamento
[93]	CNN / LSTM	Ativa	Retreinamento
[94]	Rede Neural Profunda Online	Ativa	Ajuste de modelo existente
[95]	SVM	Ativa	Retreinamento
[96]	AutoEncoder Contrastivo + Improved Extreme Value Theory	Passiva	Retreinamento
[97]	AutoEncoder	Ativa	Ajuste de modelo existente + Retreinamento
[98]	IB1 / NB / SVM / LSTM	Ativa	Retreinamento
[100]	Árvore de Decisão	Passiva	Conjunto
[99]	OPA + KNN-ADWIN + ARF-ADWIN + ARF-EDDM	Ativa	Conjunto + Ajuste de modelo existente

5 ESTUDO DE CASO

A seguir, apresenta-se a implementação de um modelo de detecção de anomalias baseado no algoritmo *DenStream*, elucidado anteriormente. Avalia-se sua aplicabilidade como Sistema de Detecção de Intrusão em ambientes SDN, bem como sua capacidade de adaptação a desvios de conceito.

5.1 Conjunto de Dados

Conjuntos de dados são fundamentais para o desenvolvimento de NIDS, sendo utilizados no treinamento, na avaliação e na comparação de modelos. A qualidade dos dados influencia diretamente o desempenho e a robustez desses sistemas em cenários reais [101].

Na implementação do sistema proposto, foram utilizados conjuntos de dados criados pelo Grupo de Pesquisa ORION da Universidade Estadual de Londrina. A geração dos dados foi realizada utilizando o emulador Mininet para criar um ambiente SDN virtual composto por um controlador, *switches* e *hosts*. Cada conjunto de dados representa um dia de tráfego, contendo 86.400 instâncias, uma para cada segundo do dia, compostas por tráfego normal e anômalo. Os fluxos de rede apresentam seis atributos: endereço IP de origem, endereço IP de destino, porta de origem, porta de destino, quantidade de bits e quantidade de pacotes.

O tráfego normal de rede apresenta aleatoriedade nas características dos pacotes e variações no volume de dados ao longo do dia, simulando mudanças legítimas no uso da rede. A biblioteca *Scapy* foi utilizada para a criação e o envio dos pacotes de rede. O tráfego anômalo dos conjuntos de dados é composto por ataques DDoS e *Portscan*. Os ataques DDoS foram gerados por meio da ferramenta hping3, direcionando pacotes UDP a uma porta específica do alvo. A intensidade desses ataques é regulada pela quantidade de *hosts* atacantes. O ataque *Portscan* foi realizado com o uso da biblioteca *Scapy*, na qual um *host* envia sequencialmente pacotes TCP para diferentes portas do alvo. Sua intensidade é controlada pelo intervalo de tempo entre o envio dos pacotes.

Apesar da aleatoriedade aplicada na criação dos pacotes, o tráfego benigno gerado em diferentes dias apresenta um comportamento semelhante, simulando um padrão diário de uso da rede. Foram gerados quatro dias de tráfego contendo ataques com diferentes combinações de duração, de modo a avaliar a robustez do modelo frente a cenários variados. As possíveis combinações em relação à duração dos ataques e seus respectivos conjuntos de dados são apresentadas na Tabela 4.

Tabela 4 – Duração dos ataques nos conjuntos de dados (em segundos). Fonte: elaboração própria.

Conjunto de dados	**DDoS**	**Portscan**
Dia 1	679 (curta duração)	606 (curta duração)
Dia 2	3.691 (longa duração)	4.245 (longa duração)
Dia 3	656 (curta duração)	3.661 (longa duração)
Dia 4	3.687 (longa duração)	627 (curta duração)

5.2 Sistema de Detecção de Intrusão em Redes

O NIDS desenvolvido neste estudo tem como objetivo identificar comportamentos anômalos em ambientes SDN. O sistema inspira-se na implementação realizada por Scaranti *et al.* [1], sendo capaz de adaptar-se a desvios incrementais de conceito. Implantado no plano de aplicação, o sistema comunica-se com o controlador por meio da interface *northbound*, permitindo o acesso às estatísticas de fluxo utilizadas na detecção de intrusões.

Conforme ilustrado na Figura 10, o NIDS é composto por módulos que atuam de forma integrada para garantir maior segurança da rede. O coletor de tráfego e pré-processador de dados preparam as informações do tráfego para a análise posterior. O núcleo de detecção, baseado no algoritmo *DenStream*, realiza a caracterização inicial do tráfego, detecção de anomalias e adaptação a desvios de conceito. As instâncias classificadas como anômalas são interpretadas de acordo com seu comportamento.

5.2.1 Pré-processamento de Dados

Os fluxos de rede coletados passam por uma etapa de pré-processamento, responsável por preparar os dados antes de serem processados pelo NIDS, buscando extrair o melhor desempenho do modelo. São utilizados na análise de tráfego os atributos de endereço IP de origem, endereço IP de destino, porta de origem e porta de destino. Características volumétricas do fluxo, como a quantidade de bits e de pacotes transmitidos, não são consideradas, uma vez que podem gerar falsos positivos em situações de aumento legítimo no uso da rede.

Os atributos dos fluxos de rede selecionados para a análise do tráfego possuem valores qualitativos, o que os torna inadequados para o processamento em modelos de aprendizado de máquina [14]. Para contornar essa limitação, aplica-se a Entropia de Shannon [102], que transforma valores de endereços IP e portas em informações quantitativas. Essa métrica fornece uma medida do grau de concentração ou dispersão dos dados no intervalo de tempo analisado. Seu cálculo é apresentado na Equação 5.1, em que x_t^A representa o histograma da quantidade de ocorrências dos diferentes valores possíveis do atributo A no período t. Tem-se $x_t^A = \{x_1, x_2, x_3, ..., x_N\}$, onde x_i indica a quantidade de ocorrências do valor i, N é o número de possíveis eventos, e S representa o total de valores observados

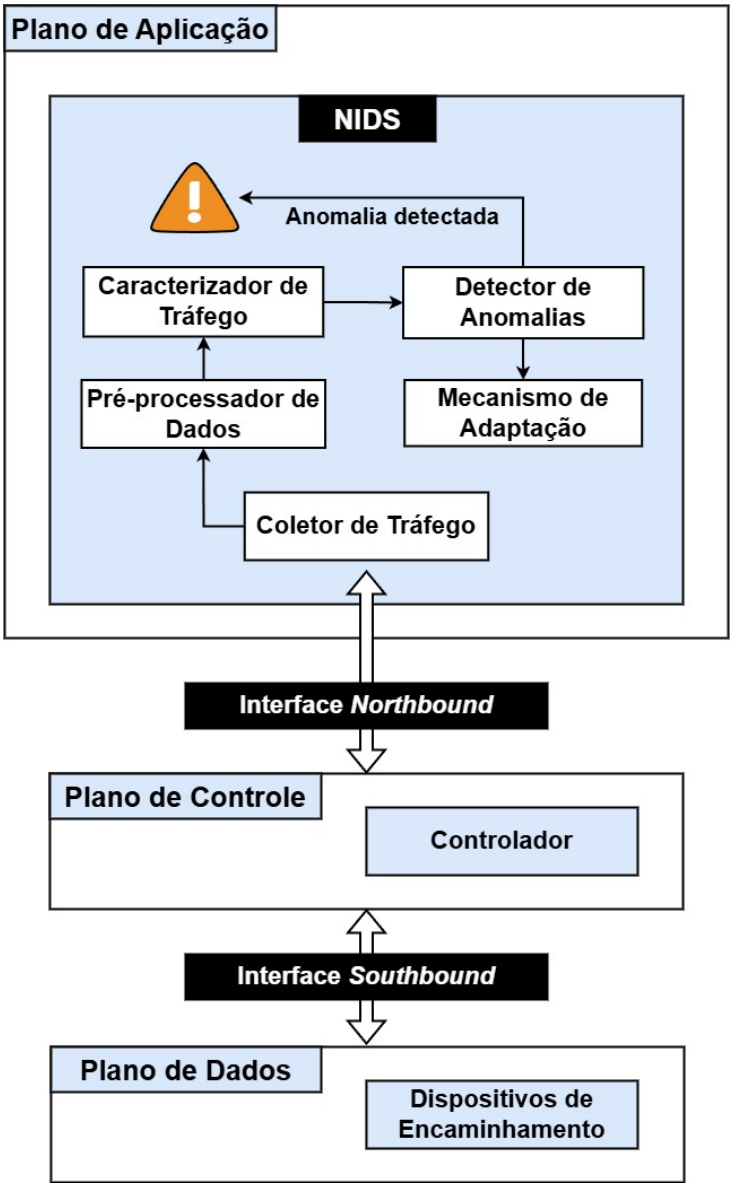

Figura 10 – Ambiente de rede considerado para aplicação do sistema de segurança desenvolvido. Fonte: elaboração própria.

para o atributo A no período de análise.

$$H(x_t^A) = -\sum_{i=1}^{N} \left(\frac{x_i}{S}\right) \log_2 \left(\frac{x_i}{S}\right) \tag{5.1}$$

Para que os atributos resultantes contribuam de forma equilibrada na classificação do tráfego, faz-se necessária a aplicação da normalização dos dados. Esse processo reduz a influência negativa de atributos dominantes e de *outliers* [103]. Nesta implementação, foi utilizada a técnica *MinMaxScaler*, da biblioteca *scikit-learn*, a qual escala os dados para um intervalo comum, entre 0 e 1, neste caso.

5.2.2 NIDS-*DenStream*

O NIDS-*DenStream* constitui o núcleo do sistema de detecção de intrusões e baseia-se no funcionamento do algoritmo *DenStream*, que opera de forma incremental para processar fluxos contínuos de dados com comportamento dinâmico. O processo envolve duas fases principais: a inicialização, em que são formados os agrupamentos iniciais, e a fase online, responsável pela atualização dinâmica dos MC, pela detecção de anomalias e pela adaptação a mudanças no comportamento da rede. A Figura 11 apresenta uma visão geral da arquitetura do sistema proposto, destacando seus principais módulos e o fluxo de processamento do tráfego de rede.

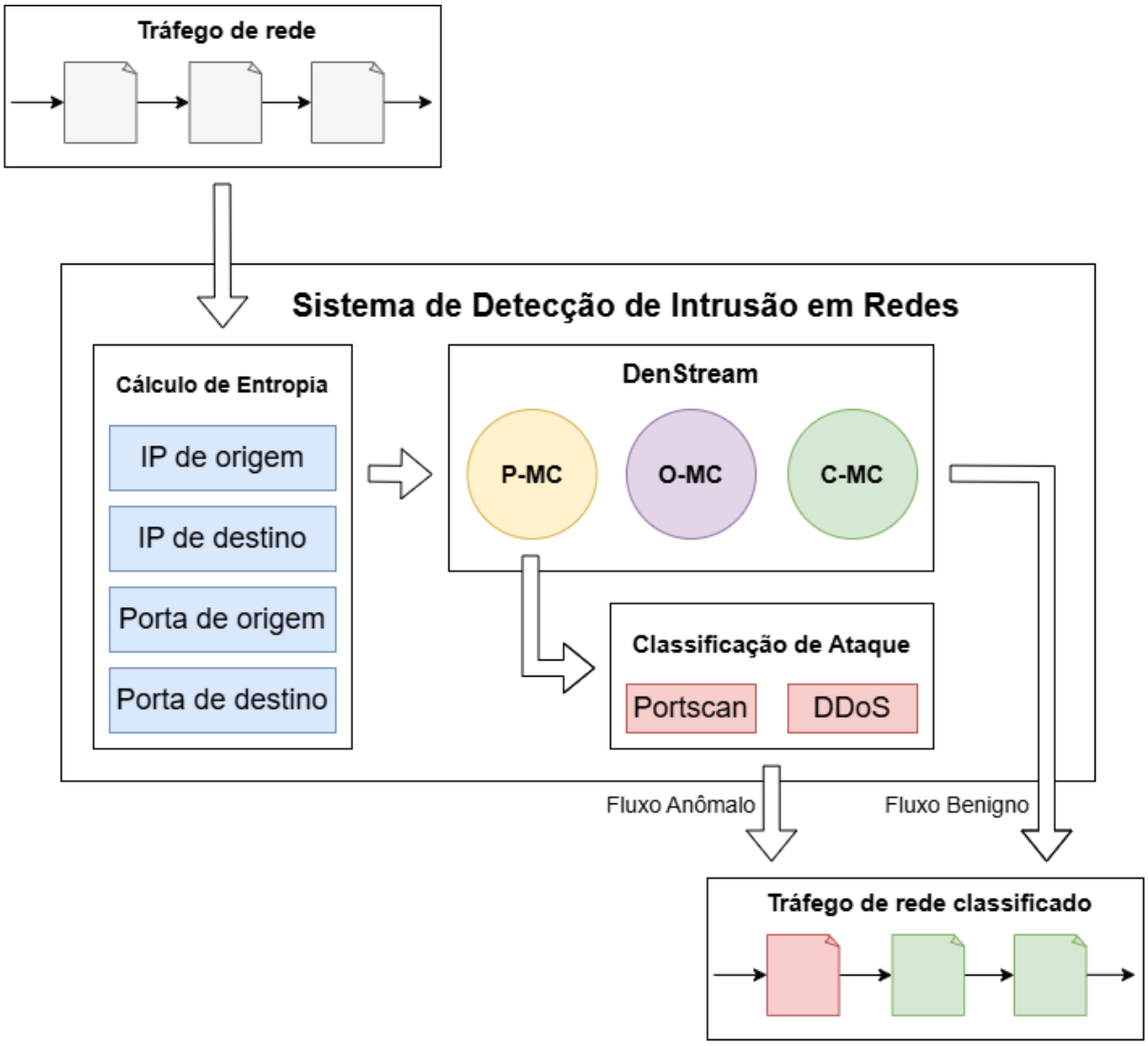

Figura 11 – NIDS-*DenStream*. Fonte: adaptado de Scaranti *et al.* [1]

Na fase de inicialização, o sistema é alimentado com amostras benignas coletadas da rede monitorada. Essas amostras são utilizadas para definir o padrão comportamental do tráfego. Para isso, o algoritmo DBSCAN é aplicado sobre ρ (hiperparâmetro) instâncias para criar os MC iniciais. Cada MC armazena a soma linear ponderada dos atributos das

instâncias ($\overline{CF^1}$), a soma quadrática ponderada ($\overline{CF^2}$) e o peso (w), que representa a quantidade de amostras incorporadas. Esses agrupamentos são classificados com base no peso e nos hiperparâmetros β e μ. Quando $w < \beta\mu$, o MC é considerado um O-MC, e quando $w \geq \beta\mu$, é considerado um P-MC.

Na fase online, o sistema processa continuamente o fluxo de tráfego de rede. Para cada nova instância analisada, o algoritmo tenta incorporá-la ao P-MC mais próximo, desde que, após a incorporação, seu raio não exceda o hiperparâmetro ϵ. Caso isso não seja possível, tenta-se incorporá-la ao O-MC mais próximo, aplicando o mesmo critério. Se nenhuma dessas opções for válida, um novo O-MC é criado para representar a instância. Em seguida, os O-MC são avaliados, e aqueles com peso suficiente ($w \geq \beta\mu$) são promovidos a P-MC. Aplica-se um decaimento exponencial aos P-MC e O-MC não atualizados há υ instâncias processadas, definido como $2^{-\lambda}$, sendo υ e λ hiperparâmetros do modelo. Dessa forma, comportamentos obsoletos são gradualmente esquecidos.

A cada υ instâncias processadas, realiza-se o processo de criação dos *clusters* finais, que representam um comportamento estável da rede. O DBSCAN é aplicado sobre os P-MC existentes, podendo resultar na criação de novos C-MC. Nesse processo, um agrupamento é criado a partir de um P-MC aleatório (pmc_a), ao qual são adicionados os P-MC (pmc) cuja distância euclidiana dos centroides seja menor ou igual à soma de seus respectivos raios, definida por $distance(pmc_a, pmc) \leq r_{pmc_a} + r_{pmc}$. Esse procedimento é repetido para todos os P-MC incorporados ao agrupamento, até que não seja possível adicionar mais nenhum. Se a soma dos pesos dos P-MC de um agrupamento exceder o limiar μ, ele é promovido a C-MC. O processo é repetido até que todos os P-MC tenham sido incorporados a algum *cluster* final ou não possam mais ser agrupados. Durante essa etapa, os P-MC localizados na área potencial são desconsiderados garantindo que os C-MC, que representam o comportamento normal da rede, não sejam contaminados por atividades anômalas.

A área potencial corresponde a regiões com maior probabilidade de conter tráfego malicioso, utilizadas para distinguir P-MC legítimos de ataques. Um P-MC é considerado pertencente a essa área se a distância euclidiana da entropia da porta de destino em relação ao C-MC mais próximo for maior que o hiperparâmetro κ. As instâncias incorporadas em P-MC localizados dentro da área potencial são consideradas ataques. As amostras identificadas como maliciosas são classificadas entre DDoS e *Portscan*, de acordo com o valor da entropia da porta de destino. Esse atributo é utilizado na distinção dos ataques devido às características inversas entre eles. Enquanto o DDoS tende a direcionar pacotes a uma única porta do alvo, com o objetivo de sobrecarregá-la, o *Portscan* envia pacotes a múltiplas portas, visando mapeá-las. Se a entropia de porta de destino for superior à do C-MC mais próximo, o ataque é definido como um *Portscan*. Caso a entropia seja inferior à do C-MC mais próximo, a classificação será de DDoS. Quando os valores de entropia

são iguais, o ataque é definido como desconhecido. Aplica-se um decaimento exponencial adicional aos P-MC dentro da área potencial, definido como $1.1^{-\lambda}$, para que o fim do ataque seja identificado de forma mais precisa e *outliers* não sejam incorretamente classificados como atividades maliciosas. Os O-MC não são considerados maliciosos para evitar falsos positivos decorrentes de *outliers*. O pseudo-código da fase *online* é apresentado no Algoritmo 1.

Algoritmo 1: Atualização dos Micro-Clusters

```
Input: Tráfego de rede, λ, β, μ, ϵ, υ, κ
Obtêm a amostra p do tráfego de rede no instante atual t;
//Tenta inserir p no P-MC mais próximo;
if r_p (raio do P-MC) ≤ ϵ then
    Insere p no P-MC ;
    //Verifica se o P-MC está na área potencial com base no C-MC mais
      próximo ;
    if distância_euclidiana(centroide_PMC, centroide_CMC) > κ then
        Classifica a amostra p como ataque ;
else
    //Tenta inserir p no O-MC mais próximo;
    if r_p(raio do O-MC) ≤ ϵ then
        Insere p no O-MC ;
        if w_OMC (peso do O-MC) ≥ βμ then
            Cria um novo P-MC a partir do O-MC e exclui o O-MC ;
    else
        Cria um novo O-MC a partir de p ;
//Aplicação do decaimento exponencial ;
if (υ mod t) = 0 then
    Aplica o decaimento exponencial nos P-MC e O-MC não atualizados nas
      últimas υ instâncias processadas ;
    Aplica o decaimento exponencial extra aos P-MC dentro da área potencial
      ;
    for cada P-MC do
        if w_PMC (peso do P-MC) ≤ βμ then
            Exclui o P-MC ;
    for cada O-MC do
        if w_OMC (peso do O-MC) < 1 then
            Exclui o O-MC ;
```

A Tabela 5 sumariza os hiperparâmetros utilizados no NIDS-*DenStream* e suas respectivas descrições. Abordagens tradicionais de ajuste de hiperparâmetros, como o *Grid Search* e o *Random Search* desperdiçam tempo e recursos testando configurações não promissoras. Já a Otimização Bayesiana, constrói um modelo estatístico substituto que funciona como um mapa probabilístico da função objetivo. Este mapa estima a média e a variância para todas as combinações de hiperparâmetros nos intervalos definidos. A cada iteração, uma função de aquisição é utilizada para analisar o mapa e selecionar o próximo ponto a testar. A seleção busca equilibrar a explotação, testando configurações com alta probabilidade de alto desempenho, e a exploração, avaliando regiões de alta incerteza, de modo a identificar picos de performance ainda desconhecidos. Após a avaliação do ponto, o resultado é utilizado para atualizar o mapa, ajustando a direção da busca para explorar regiões mais promissoras do espaço de parâmetros. Dessa forma, a Otimização Bayesiana é capaz de encontrar um ótimo resultado com consumo de recursos computacionais e tempo gasto menor em comparação às técnicas tradicionais [104].Esse procedimento foi realizado utilizando o conjunto de dados correspondente ao Dia 1, adotando como pontuação de cada iteração a média das métricas de acurácia, precisão, revocação e *F1-score*. A Tabela 6 apresenta os hiperparâmetros avaliados e os melhores valores encontrados dentro do intervalo definido.

Tabela 5 – Hiperparâmetros utilizados no NIDS-*DenStream*. Fonte: elaboração própria.

Hiperparâmetros	**Descrição**
λ	Fator de decaimento que limita a influência de comportamentos obsoletos
β	Fator de tolerância de *outlier*
μ	Limiar principal de peso
ϵ	Raio máximo de um *micro-cluster*
υ	Intervalo de instâncias processadas para aplicar o decaimento exponencial e a criação de *clusters* finais
ρ	Número de instâncias utilizadas no processo de inicialização
κ	Limiar de distância para um P-MC estar dentro da área potencial

Tabela 6 – Valores da Otimização Bayesiana. Fonte: elaboração própria.

Hiperparâmetros	**Intervalo definido**	**Melhor valor**
λ	[0,05; 0,1]	0,095
β	[0,001; 0,005]	0,004
μ	[500; 1000]	637
ϵ	[0,01; 0,07]	0,048
υ	[500; 1500]	1266
ρ	[500; 1000]	637
κ	[0,05; 0,1]	0,099

5.2.3 Adaptação ao Desvio de Conceito

A forma como o algoritmo *DenStream* opera favorece a adaptação a mudanças graduais no comportamento dos dados. Por meio da atualização incremental dos MC a cada instância processada, os agrupamentos conseguem acompanhar alterações sutis

nas estatísticas do tráfego de rede. Além disso, o decaimento exponencial aplicado aos MC não atualizados permite ao sistema esquecer comportamentos obsoletos, mantendo a representação do modelo coerente com o estado atual da rede.

5.3 Resultados e Discussão

O Sistema de Detecção de Intrusão foi avaliado utilizando três dias distintos de tráfego de rede, cada um representando cenários contrastantes quanto à duração dos ataques. Essa abordagem permite analisar o desempenho do modelo diante de diferentes padrões de atividade maliciosa, tanto de curta quanto de longa duração. Os experimentos foram executados em uma máquina cujas configurações de *hardware* e *software* estão descritas na Tabela 7.

Tabela 7 – Configuração de desenvolvimento. Fonte: elaboração própria.

Sistema Operacional	Ubuntu 22.04.5 LTS
Processador	AMD Ryzen 5 5600G
RAM	16 GB DDR4
Python	3.10.12
Sklearn	1.6.1

A avaliação do modelo foi estruturada em três aspectos principais. O primeiro refere-se à capacidade de detecção de anomalias, medida por meio de métricas de desempenho derivadas da matriz de confusão. O segundo aspecto está relacionado ao reconhecimento de diferentes tipos de ataque, especificamente DDoS e *Portscan*, com base na distribuição e posição espacial dos P-MC. O terceiro aspecto avalia o atraso na identificação das atividades maliciosas, mensurado pelo número de instâncias processadas entre o início do ataque e a criação do P-MC correspondente. Dessa forma, é possível analisar a efetividade do sistema na distinção entre tráfego benigno e malicioso, a capacidade de identificar corretamente o tipo de ameaça e o tempo de resposta do sistema.

5.3.1 Cenário de Avaliação

Os Dias 2, 3 e 4 foram utilizados para avaliar o desempenho do modelo. As estatísticas referentes ao segundo e terceiro dia estão ilustradas nas Figuras 12 e 13, respectivamente. No Dia 2, ocorre um ataque DDoS de curta duração e um ataque *Portscan* de longa duração. O Dia 3 apresenta a situação inversa, com um ataque DDoS de longa duração e um ataque *Portscan* de curta duração. Esses dois cenários permitem analisar o desempenho do modelo sob diferentes padrões temporais de atividade maliciosa.

Para avaliar a capacidade de adaptação do modelo ao desvio de conceito, o conjunto referente ao Dia 4 foi modificado. Um valor incremental de desvio foi adicionado às entropias durante o período compreendido entre as instâncias 5.000 e 60.000. Após esse intervalo, o valor adicional foi mantido constante, de modo que o conjunto preservasse

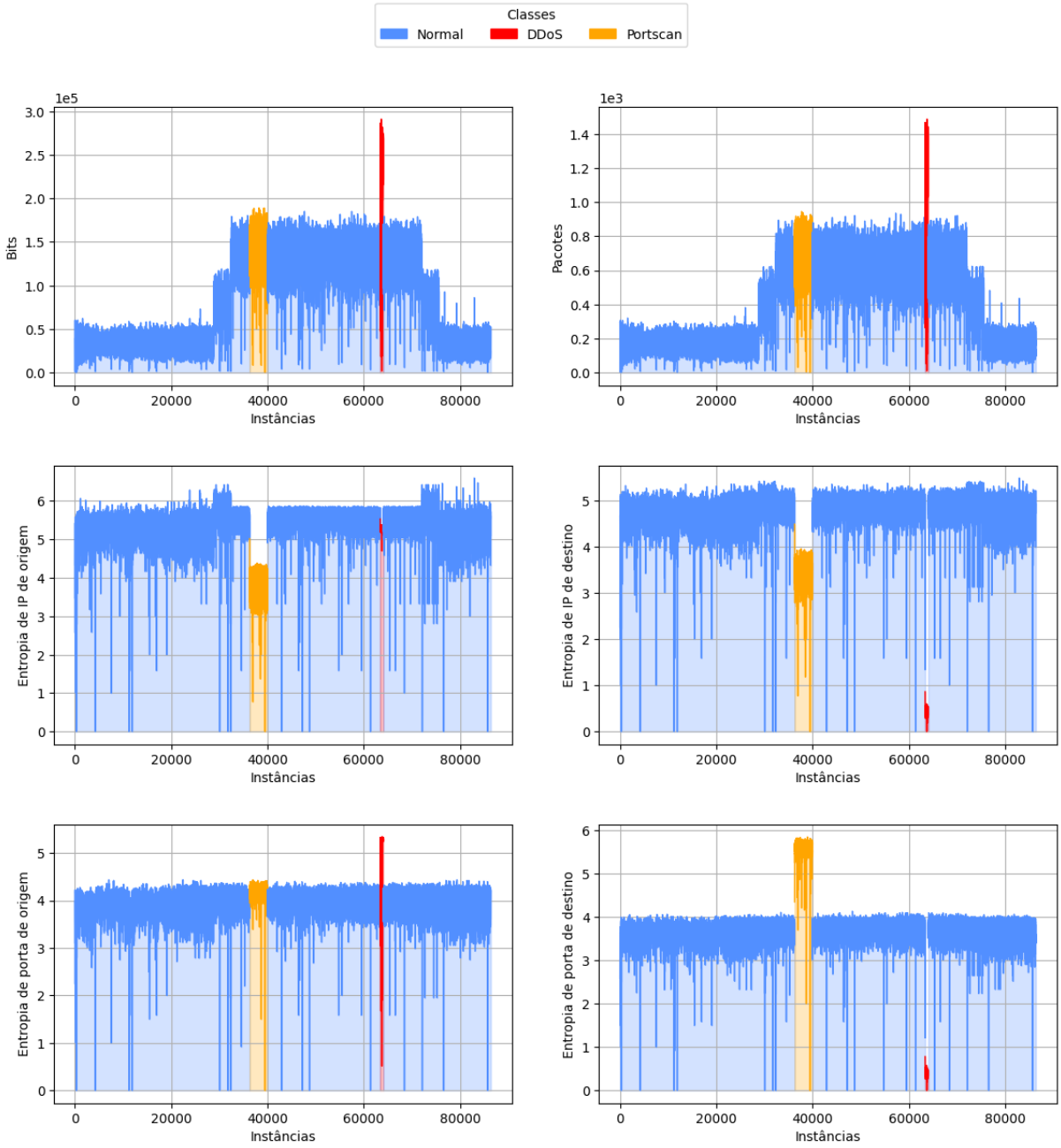

Figura 12 – Estatísticas do Dia 2. Fonte: elaboração própria.

a nova distribuição. Esse cenário simula uma mudança no comportamento do tráfego, por meio do aumento incremental nos valores de entropia. Como ilustrado na Figura 14, o valor médio da entropia da porta de destino, que inicialmente caracterizava o tráfego associado ao ataque *Portscan*, passa a corresponder ao padrão do tráfego normal.

5.3.2 Métricas de Avaliação

As métricas de avaliação são fundamentais no desenvolvimento de sistemas de detecção de anomalias, pois permitem analisar e comparar o desempenho do modelo. As métricas utilizadas neste trabalho baseiam-se na matriz de confusão, ilustrada na Figura 15.

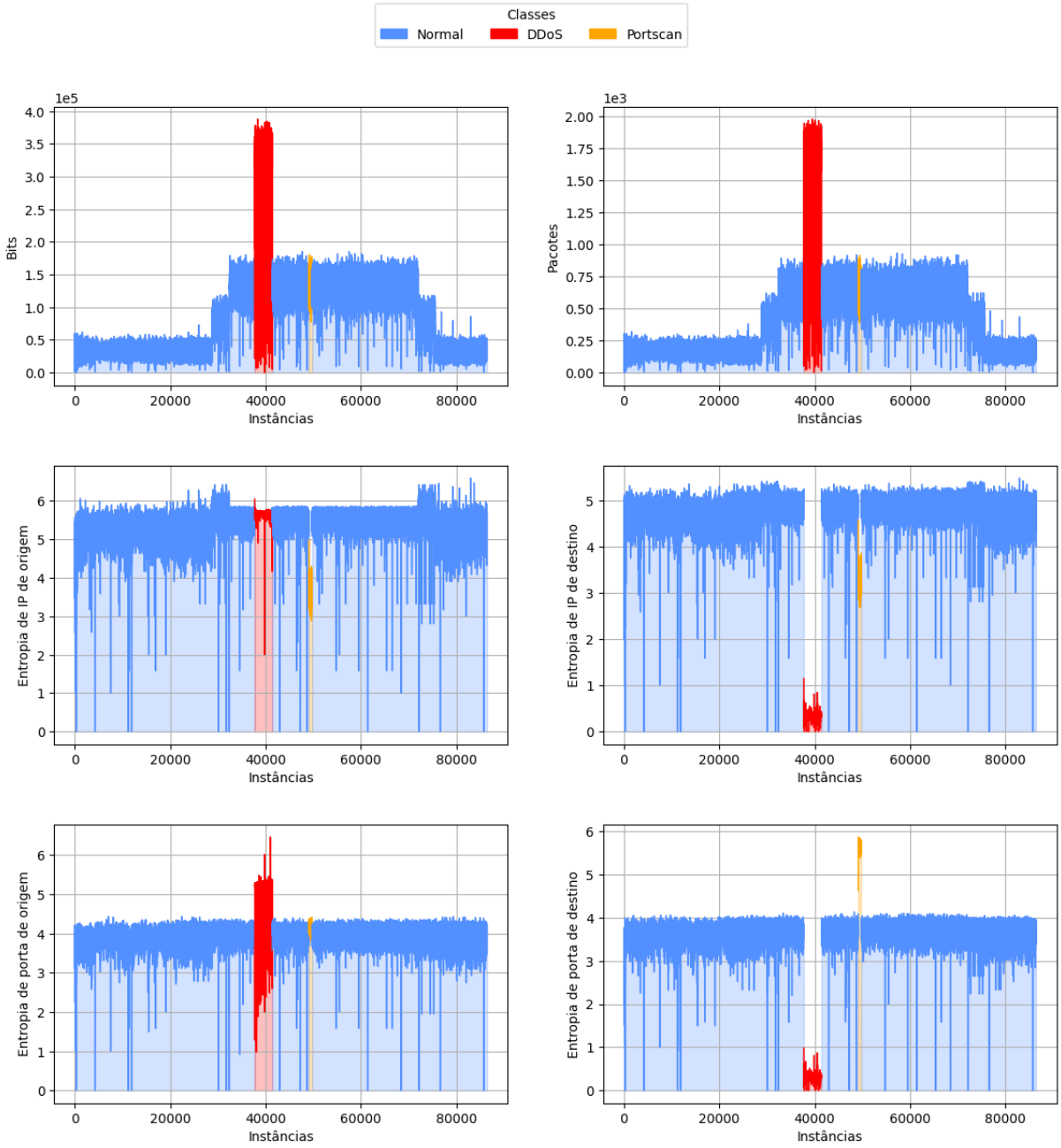

Figura 13 – Estatísticas do Dia 3. Fonte: elaboração própria.

Essa estrutura organiza as classificações realizadas em quatro categorias: verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN). A partir dessa matriz, são calculadas métricas como *Accuracy*, *Precision*, *Recall*, *F1-score* e *Matthews Correlation Coefficient* (MCC).

A *Accuracy*, representada pela Equação 5.2, indica a proporção de classificações corretas realizadas pelo modelo.

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN} \tag{5.2}$$

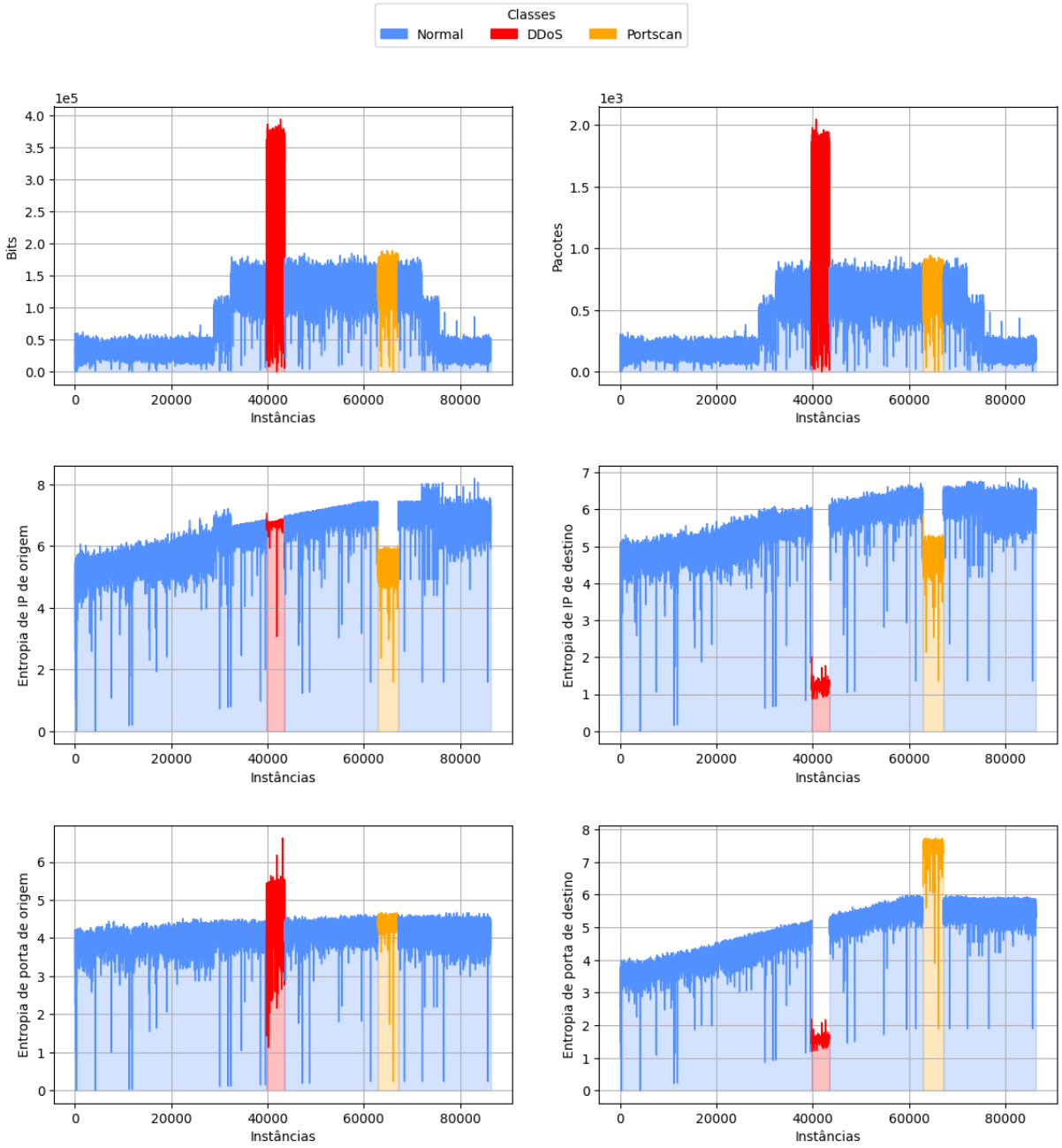

Figura 14 – Estatísticas do Dia 4 (com desvio). Fonte: elaboração própria.

A *Precision* mensura a proporção de classificações positivas corretas, ou seja, a fração das instâncias classificadas como anômalas que realmente correspondem a anomalias, conforme a Equação 5.3.

$$Precision = \frac{VP}{VP + FP} \tag{5.3}$$

A *Recall*, calculada pela Equação 5.4, avalia a capacidade do modelo de identificar corretamente as instâncias anômalas existentes.

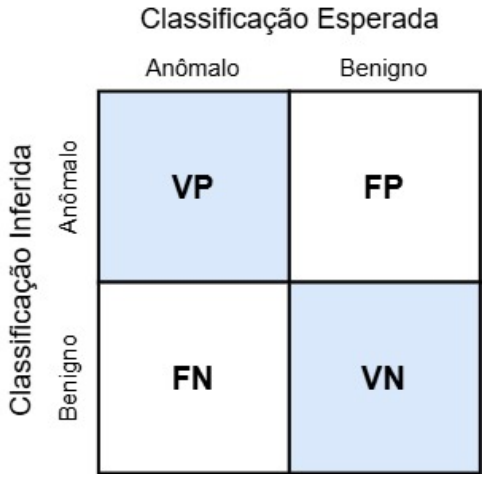

Figura 15 – Matriz de Confusão. Fonte: elaboração própria.

$$Recall = \frac{VP}{VP + FN} \tag{5.4}$$

O *F1-score* mede o equilíbrio entre a *Precision* e a *Recall* por meio da média harmônica entre ambas, conforme a Equação 5.5.

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.5}$$

O MCC considera todas as categorias da matriz de confusão, fornecendo uma medida mais robusta, especialmente em cenários com desbalanceamento entre classes. Sua definição está apresentada na Equação 5.6.

$$MCC = \frac{VP * VN - FP * FN}{\sqrt{(VP + FP)(VP + FN)(VN + FP)(VN + FN)}} \tag{5.6}$$

No contexto de classificação multiclasse, as métricas são adaptadas para considerar três classes distintas: tráfego normal, ataque DDoS e ataque *Portscan*. Conforme ilustrado na Figura 16, a matriz de confusão assume a dimensão 3×3. As métricas *Precision*, *Recall*, *F1-score* e MCC são calculadas individualmente para cada classe, e o desempenho global é obtido por meio da média macro.

5.3.3 Detecção de Anomalias

A função do sistema proposto é classificar os fluxos IP pertencentes ao tráfego de rede analisado como normais ou anômalos. Essa distinção é realizada com base nas características dos MC. Considera-se que um C-MC representa um comportamento normal da rede, enquanto um P-MC localizado na área potencial corresponde a um agrupamento de amostras maliciosas. Um O-MC pode representar tanto um agrupamento inicial quanto

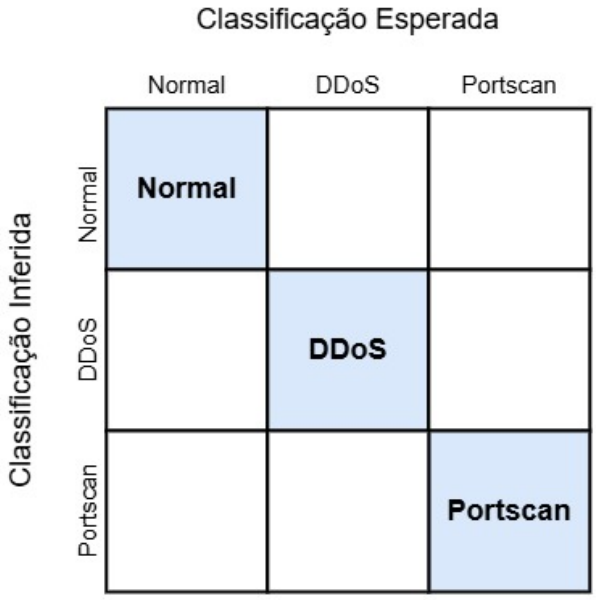

Figura 16 – Matriz de confusão multiclasse. Fonte: elaboração própria.

um ruído. Dessa forma, o desempenho do modelo é avaliado em três cenários distintos, apresentados na Seção 5.3.1, utilizando as métricas abordadas na Seção 5.3.2.

As matrizes de confusão obtidas nesses cenários são apresentadas nas Figuras 17, 18 e 19. As métricas correspondentes encontram-se na Tabela 8.

Tabela 8 – Desempenho do NIDS-*DenStream* na detecção de anomalias. Fonte: elaboração própria.

	Accuracy	***Precision***	***Recall***	***F1-score***	**MCC**
Dia 2	0,9994	0,9994	0,9994	0,9994	0,9938
Dia 3	0,9996	0,9996	0,9996	0,9996	0,9955
Dia 4 (com desvio)	0,9994	0,9994	0,9994	0,9994	0,9963

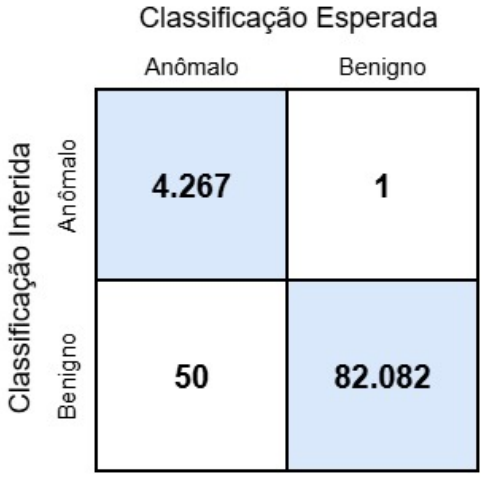

Figura 17 – Matriz de confusão para detecção de anomalias no Dia 2. Fonte: elaboração própria.

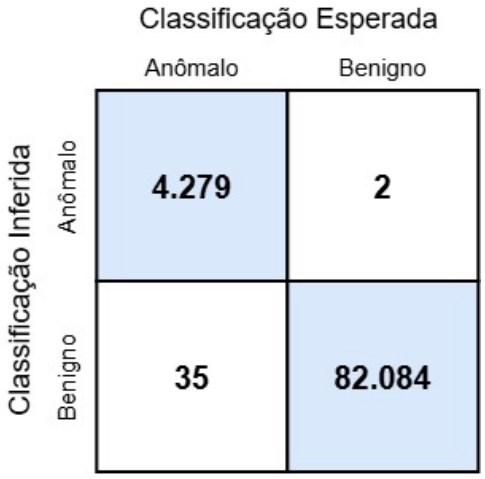

Figura 18 – Matriz de confusão para detecção de anomalias no Dia 3. Fonte: elaboração própria.

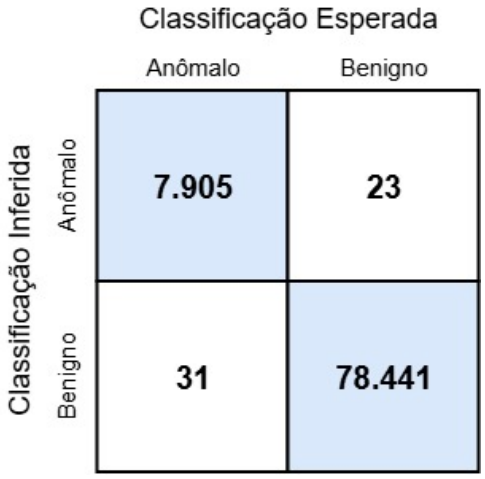

Figura 19 – Matriz de confusão para detecção de anomalias no Dia 4 (com desvio). Fonte: elaboração própria.

5.3.4 Reconhecimento de Ataques

Após detectar uma amostra anômala, o sistema procede à sua classificação como ataque DDoS ou *Portscan*. A avaliação do desempenho do sistema na identificação dos tipos de ataque é realizada com base na classificação multiclasse, considerando as classes Normal, DDoS e *Portscan*. Para essa análise, utilizam-se as métricas adaptadas para o contexto de três classes, conforme descrito na Seção 5.3.2. As Figuras 20, 21 e 22 apresentam as matrizes de confusão referentes aos três cenários avaliados. As Tabelas 9, 10 e 11 apresentam o desempenho individual de cada classe nos Dias 2, 3 e 4, respectivamente.

As Figuras 23, 24 e 25 apresentam a distribuição dos dados nos três cenários de teste, comparando as classificações realizadas com os respectivos rótulos reais. No terceiro dia, observa-se um deslocamento dos dados, causado pelo desvio de conceito, principalmente aqueles pertencentes ao tráfego normal. Conforme ilustrado na Figura 26,

Tabela 9 – Desempenho do NIDS-*DenStream* no reconhecimento de ataques para o Dia 2. Fonte: elaboração própria.

	Precision	***Recall***	***F1-score***	**MCC**
Normal	0,9994	1,000	0,9997	0,9938
DDoS	1,000	0,9787	0,9892	0,9892
Portscan	0,9997	0,9902	0,9949	0,9947

Tabela 10 – Desempenho do NIDS-*DenStream* no reconhecimento de ataques para o Dia 3. Fonte: elaboração própria.

	Precision	***Recall***	***F1-score***	**MCC**
Normal	0,9996	1,000	0,9998	0,9955
DDoS	0,9997	0,9965	0,9981	0,9980
Portscan	0,9983	0,9649	0,9813	0,9813

Tabela 11 – Desempenho do NIDS-*DenStream* no reconhecimento de ataques para o Dia 4 (com desvio). Fonte: elaboração própria.

	Precision	***Recall***	***F1-score***	**MCC**
Normal	0,9996	0,9997	0,9997	0,9962
DDoS	0,9984	0,9967	0,9976	0,9974
Portscan	0,9960	0,9955	0,9958	0,9955

o centroide do C-MC, representado pelo ponto verde, desloca-se progressivamente para acompanhar essa mudança no padrão do tráfego benigno conforme novas instâncias são processadas. Esse comportamento demonstra a capacidade do modelo em adaptar-se a variações graduais na distribuição dos dados.

5.3.5 Atraso de Detecção

A identificação rápida de ataques é essencial para minimizar o impacto causado por ataques à rede. Para avaliar essa capacidade, foi medido o atraso entre o início de cada ataque e a sua detecção, expresso em número de instâncias processadas. A Tabela 12 apresenta o atraso de detecção observado nos três cenários de teste.

Tabela 12 – Atraso de detecção de anomalias do modelo em número de instâncias processadas. Fonte: elaboração própria.

	Dia 2	**Dia 3**	**Dia 4 (com desvio)**
DDoS	9	7	6
Portscan	27	20	13

O modelo apresenta um atraso médio de 13,667 instâncias na detecção de ataques, sendo esse valor maior na identificação de *Portscan*, com média de 20 instâncias, enquanto para ataques DDoS o atraso médio é de 7,333 instâncias. Esse comportamento pode estar relacionado às características individuais dessas atividades maliciosas, onde ataques DDoS tendem a gerar alterações mais abruptas, favorecendo uma detecção mais rápida.

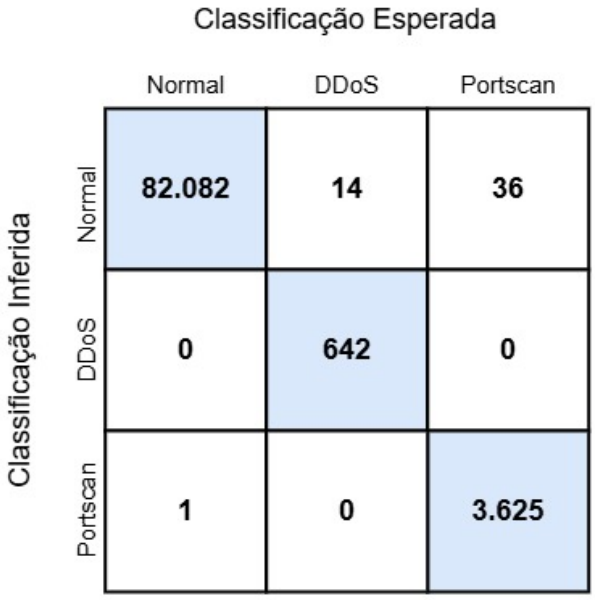

Figura 20 – Matriz de confusão para reconhecimento de ataques no Dia 2. Fonte: elaboração própria.

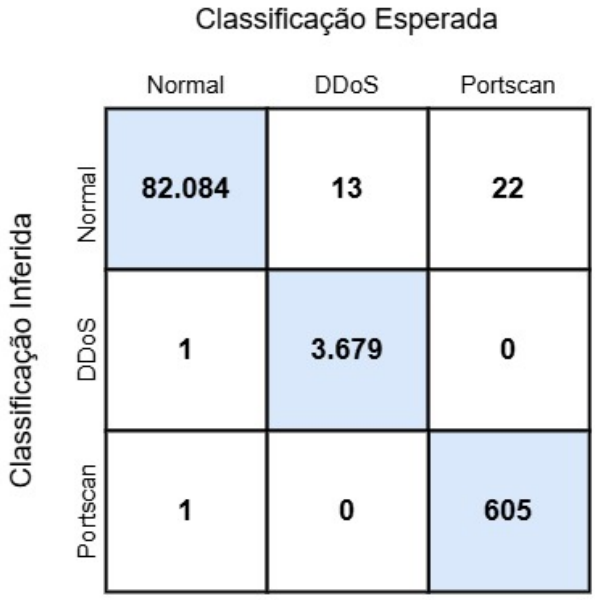

Figura 21 – Matriz de confusão para reconhecimento de ataques no Dia 3. Fonte: elaboração própria.

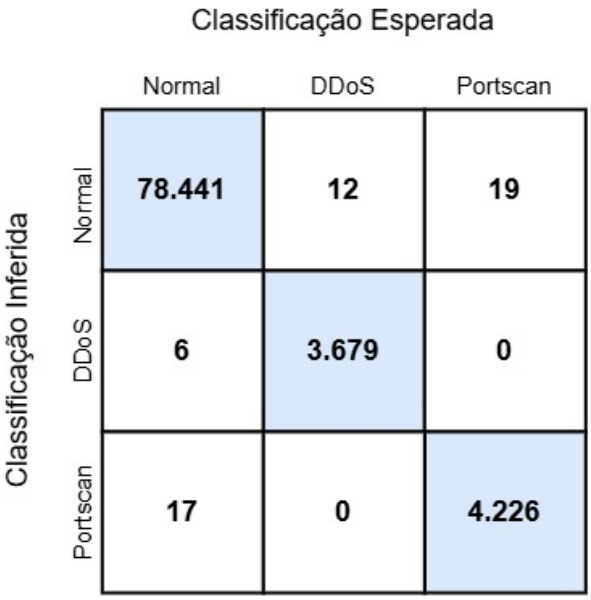

Figura 22 – Matriz de confusão para reconhecimento de ataques no Dia 4 (com desvio). Fonte: elaboração própria.

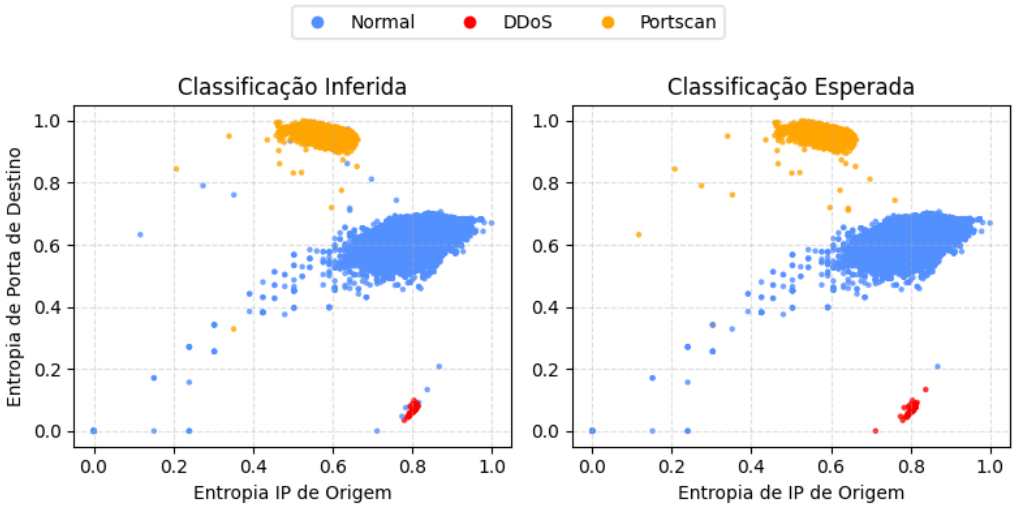

Figura 23 – Distribuição dos dados do Dia 2. Fonte: elaboração própria.

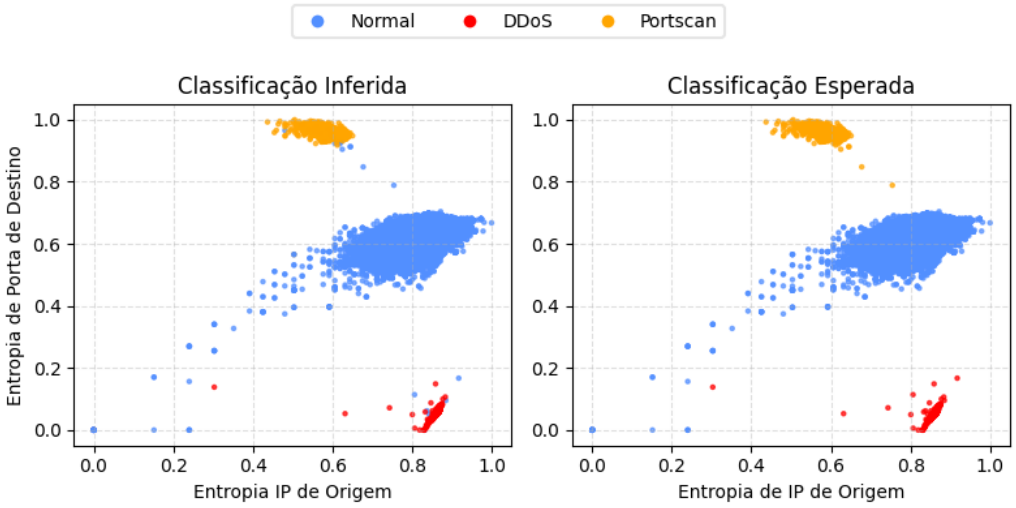

Figura 24 – Distribuição dos dados do Dia 3. Fonte: elaboração própria.

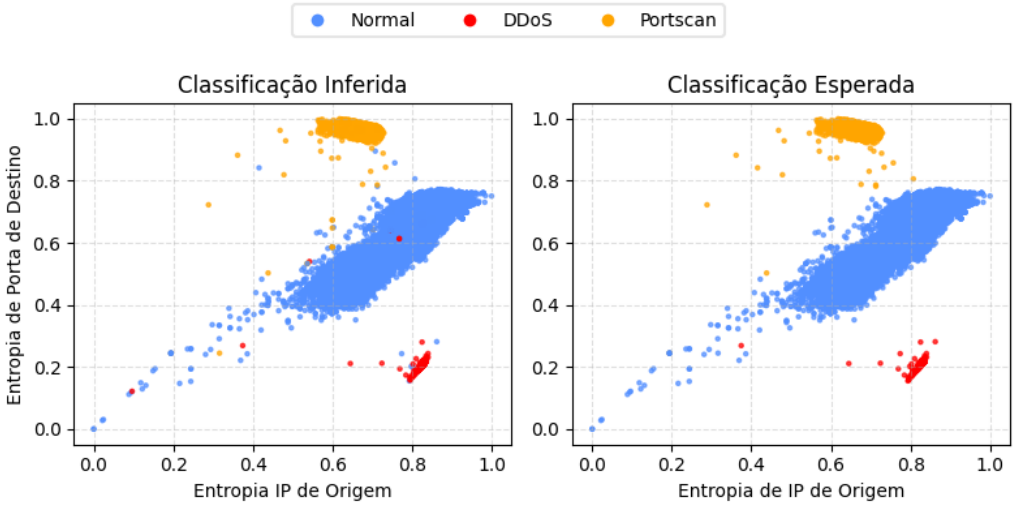

Figura 25 – Distribuição dos dados do Dia 4 (com desvio). Fonte: elaboração própria.

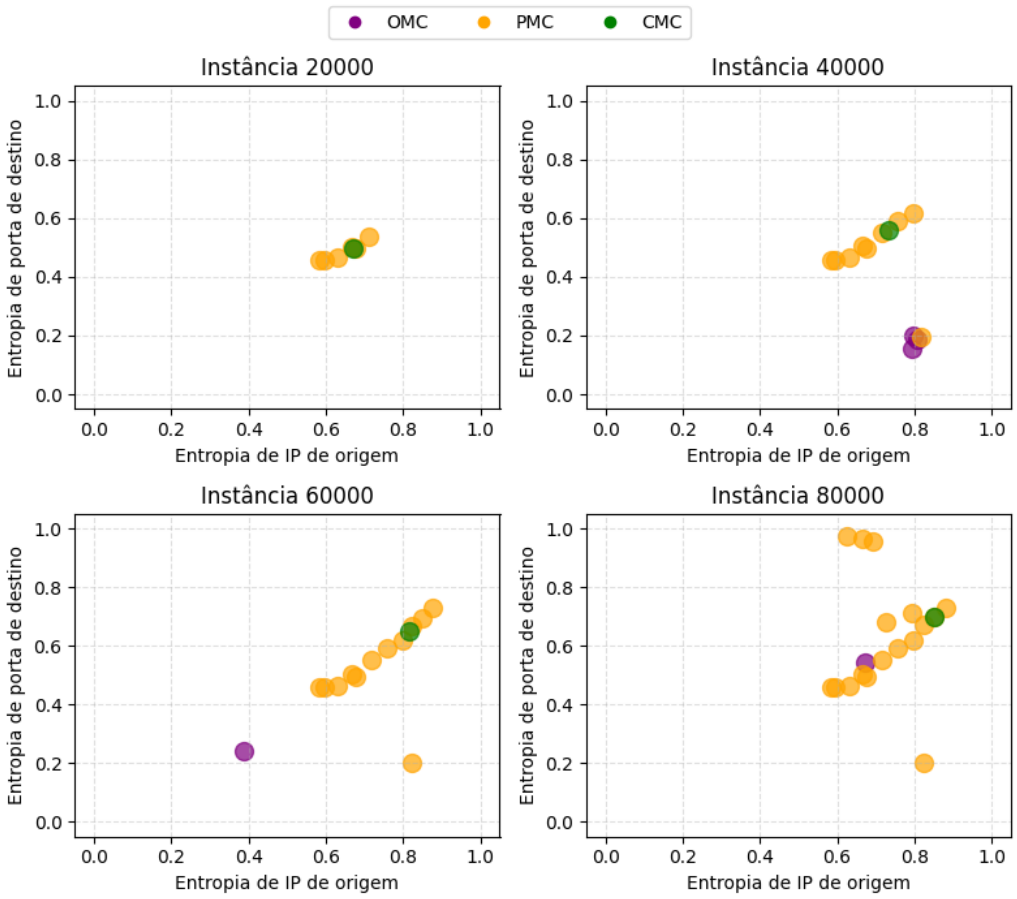

Figura 26 – Distribuição dos Micro-Cluster do Dia 4 (com desvio). Fonte: elaboração própria.

6 CONCLUSÃO

O desvio de conceito representa um desafio para modelos de aprendizado de máquina, sendo caracterizado pela mudança na distribuição dos dados ao longo do tempo. Cenários dinâmicos, como o tráfego em redes de computadores, tendem a apresentar esse fenômeno, o que pode causar degradação no desempenho de sistemas de detecção de intrusão.

Para manter a eficácia dos modelos diante do desvio de conceito, técnicas adaptativas são empregadas, podendo ser ativas ou passivas. A abordagem ativa utiliza um módulo dedicado à detecção de desvio de conceito, responsável por acionar a atualização do modelo. Já a abordagem passiva realiza atualizações contínuas, independente da ocorrência de um desvios.

Este trabalho abordou a tarefa de Detecção de Intrusão em Rede utilizando um modelo de aprendizado de máquina com adaptação passiva ao desvio de conceito. Uma variação do algoritmo *DenStream* foi empregada na implementação do NIDS. Trata-se de um método de agrupamento por densidade, desenvolvido para processar dados em fluxo e adaptar-se a comportamentos dinâmicos.

O sistema foi avaliado em diferentes cenários, utilizando três conjuntos de dados referentes a tráfegos diários com ataques DDoS e *Portscan*. Foram considerados três aspectos principais nesse processo: capacidade de detecção de anomalias, capacidade de reconhecimento de ataques (DDoS e *Portscan*) e atraso de detecção. Para analisar a capacidade adaptativa do modelo, um dos conjuntos de dados apresentava um desvio de conceito, caracterizado por um aumento incremental nos valores de entropia.

O NIDS obteve resultados consistentes na detecção de anomalias, com valores superiores a 99% para acurácia, precisão, revocação, *F1-score* e MCC. Com base na entropia da porta de destino, o modelo obteve mais de 98% de *F1-score* na classificação dos ataques, onde as classificações incorretas estão majoritariamente associadas ao *Portscan*. O atraso médio na identificação das atividades maliciosas foi de 13,667 instâncias, sendo o *Portscan* o ataque com maior atraso, com média de 20 instâncias, enquanto o DDoS registrou um atraso médio de 7,333 instâncias. O modelo mostrou-se resiliente a mudanças graduais no comportamento dos dados, apresentando valores de acurácia, precisão, revocação, *F1-score* e MCC superiores a 99% na detecção de anomalias, e mais de 98% de *F1-score* na classificação dos ataques. O atraso de detecção de anomalias foi de 6 instâncias para o ataque DDoS e 13 instâncias para o *Portscan*.

O modelo apresentou resultados semelhantes na classificação do tráfego de rede entre os diferentes cenários. As características do algoritmo *DenStream* conferiram ao

sistema de detecção de intrusão uma capacidade adaptativa para ambientes com desvios de conceito incrementais. Entretanto, desvios abruptos podem ser incorretamente classificados, fazendo com que novos padrões de comportamento benigno sejam interpretados como ataques. Assim, o desvio de conceito mostra-se um tópico de grande relevância na área de segurança de redes, influenciando diretamente a confiabilidade e a aplicabilidade de sistemas de detecção de intrusão em cenários reais.

Como trabalhos futuros, pretende-se ampliar a capacidade adaptativa do modelo, de modo a torná-lo adaptável a diferentes tipos de transições de conceito, como desvios de conceito abruptos. Outra possibilidade consiste em empregar uma abordagem totalmente não supervisionada, com o objetivo de aumentar a aplicabilidade do sistema em cenários com escassez de rótulos. Além disso, propõe-se avaliar o desempenho do modelo em diferentes bases de dados, com padrões de tráfego de rede distintos, e compará-lo a resultados apresentados em trabalhos científicos relacionados.

REFERÊNCIAS

[1] SCARANTI, G. F. et al. Unsupervised online anomaly detection in software defined network environments. *Expert Systems with Applications*, v. 191, p. 116225, 2022. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2021.116225>.

[2] SIDDIQI, M. A.; PAK, W. Tier-based optimization for synthesized network intrusion detection system. *IEEE Access*, v. 10, p. 108530–108544, 2022. Disponível em: <https://doi.org/10.1109/ACCESS.2022.3213937>.

[3] FERNANDES, G. et al. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, v. 70, n. 3, p. 447–489, 2019. ISSN 1572-9451. Disponível em: <https://doi.org/10.1007/s11235-018-0475-8>.

[4] ASSIS, M. V. O. D. et al. A game theoretical based system using holt-winters and genetic algorithm with fuzzy logic for dos/ddos mitigation on sdn networks. *IEEE Access*, v. 5, p. 9485–9496, 2017. Disponível em: <https://doi.org/10.1109/ACCESS.2017.2702341>.

[5] Security Report. *Portal da Marinha do Brasil sofre ataque DDoS e fica fora do ar*. 2024. Acesso em: 08 nov. 2025. Disponível em: <https://securityleaders.com.br/marinha-do-brasil-confirma-ataque-ddos-apos-instabilidade-no-site/>.

[6] da Silva Ruffo, V. G. et al. Generative adversarial networks to detect intrusion and anomaly in ip flow-based networks. *Future Generation Computer Systems*, v. 163, p. 107531, 2025. ISSN 0167-739X. Disponível em: <https://doi.org/10.1016/j.future.2024.107531>.

[7] PROENÇA, M. L. et al. The hurst parameter for digital signature of network segment. In: SOUZA, J. N. de; DINI, P.; LORENZ, P. (Ed.). *Telecommunications and Networking - ICT 2004*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 772–781. ISBN 978-3-540-27824-5. Disponível em: <https://doi.org/10.1007/978-3-540-27824-5_103>.

[8] RUFFO, V. G. d. S. et al. f-anogan for unsupervised attack detection in sdn environment. *IEEE Transactions on Network Science and Engineering*, p. 1–17, 2025. Disponível em: <https://doi.org/10.1109/TNSE.2025.3558936>.

[9] da Silva Ruffo, V. G. et al. Anomaly and intrusion detection using deep learning for software-defined networks: A survey. *Expert Systems with Applications*, v. 256, p. 124982, 2024. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2024.124982>.

[10] CHINNASAMY, R. et al. Deep learning-driven methods for network-based intrusion detection systems: A systematic review. *ICT Express*, v. 11, n. 1, p. 181–215, 2025. ISSN 2405-9595. Disponível em: <https://doi.org/10.1016/j.icte.2025.01.005>.

[11] KWON, D. et al. A survey of deep learning-based network anomaly detection. *Cluster Computing*, v. 22, n. 1, p. 949–961, 2019. ISSN 1573-7543. Disponível em: <https://doi.org/10.1007/s10586-017-1117-8>.

[12] de Assis, M. V. et al. Near real-time security system applied to sdn environments in iot networks using convolutional neural network. *Computers & Electrical Engineering*, v. 86, p. 106738, 2020. ISSN 0045-7906. Disponível em: <https://doi.org/10.1016/j.compeleceng.2020.106738>.

[13] NOVAES, M. P. et al. Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. *IEEE Access*, v. 8, p. 83765–83781, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2020.2992044>.

[14] LENT, D. M. B. et al. A gated recurrent unit deep learning model to detect and mitigate distributed denial of service and portscan attacks. *IEEE Access*, v. 10, p. 73229–73242, 2022. Disponível em: <https://doi.org/10.1109/ACCESS.2022.3190008>.

[15] LENT, D. M. B. et al. An unsupervised generative adversarial network system to detect ddos attacks in sdn. *IEEE Access*, v. 12, p. 70690–70706, 2024. Disponível em: <https://doi.org/10.1109/ACCESS.2024.3402069>.

[16] MAHDI, O. A. et al. Roadmap of concept drift adaptation in data stream mining, years later. *IEEE Access*, v. 12, p. 21129–21146, 2024. Disponível em: <https://doi.org/10.1109/ACCESS.2024.3358817>.

[17] PéREZ, J. L. M.; BARROS, R. S. M.; SANTOS, S. G. T. C. Enhancing semi-supervised learning with concept drift detection and self-training: A study on classifier diversity and performance. *IEEE Access*, v. 13, p. 24681–24697, 2025. Disponível em: <https://doi.org/10.1109/ACCESS.2025.3538710>.

[18] LU, J. et al. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, v. 31, n. 12, p. 2346–2363, 2019. Disponível em: <https://doi.org/10.1109/TKDE.2018.2876857>.

[19] LIMA, M. et al. Learning under concept drift for regression—a systematic literature review. *IEEE Access*, v. 10, p. 45410–45429, 2022. Disponível em: <https://doi.org/10.1109/ACCESS.2022.3169785>.

[20] SAROHE, S.; HARIT, S.; KUMAR, M. A systematic and comprehensive survey of load balancing techniques in software defined network based internet of things. *Computer Networks*, p. 111412, 2025. ISSN 1389-1286. Disponível em: <https://doi.org/10.1016/j.comnet.2025.111412>.

[21] RAHOUTI, M. et al. Sdn security review: Threat taxonomy, implications, and open challenges. *IEEE Access*, v. 10, p. 45820–45854, 2022. Disponível em: <https://doi.org/10.1109/ACCESS.2022.3168972>.

[22] SAHAY, R.; MENG, W.; JENSEN, C. D. The application of software defined networking on securing computer networks: A survey. *Journal of Network and Computer Applications*, v. 131, p. 89–108, 2019. ISSN 1084-8045. Disponível em: <https://doi.org/10.1016/j.jnca.2019.01.019>.

[23] ZHANG, Q.-Y. et al. Software defined networking meets information centric networking: A survey. *IEEE Access*, v. 6, p. 39547–39563, 2018. Disponível em: <https://doi.org/10.1109/ACCESS.2018.2855135>.

[24] Kengne Tchendji, V.; VELEMPINI, M.; Chassem Kamdem, P. Multi-objective game for fighting against distributed reflection dos attacks in software-defined network. *Array*, v. 26, p. 100410, 2025. ISSN 2590-0056. Disponível em: <https://doi.org/10.1016/j.array.2025.100410>.

[25] İPEK, A. D.; CICIOğLU, M.; ÇALHAN, A. Airsdn: Ai based routing in software-defined networks for multimedia traffic transmission. *Computer Communications*, p. 108222, 2025. ISSN 0140-3664. Disponível em: <https://doi.org/10.1016/j.comcom.2025.108222>.

[26] FILHO, J. E. D. A. et al. A review of neural networks for anomaly detection. *IEEE Access*, v. 10, p. 112342–112367, 2022. Disponível em: <https://doi.org/10.1109/ACCESS.2022.3216007>.

[27] ZARGAR, S. T.; JOSHI, J.; TIPPER, D. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE Communications Surveys & Tutorials*, v. 15, n. 4, p. 2046–2069, 2013. Disponível em: <https://doi.org/10.1109/SURV.2013.031413.00127>.

[28] HOQUE, N.; BHATTACHARYYA, D. K.; KALITA, J. K. Botnet in ddos attacks: Trends and challenges. *IEEE Communications Surveys & Tutorials*, v. 17, n. 4, p. 2242–2270, 2015. Disponível em: <https://doi.org/10.1109/COMST.2015.2457491>.

[29] HORE, S. et al. A sequential deep learning framework for a robust and resilient network intrusion detection system. *Computers & Security*, v. 144, p. 103928, 2024. ISSN 0167-4048. Disponível em: <https://doi.org/10.1016/j.cose.2024.103928>.

[30] BORGIOLI, N. et al. A convolutional autoencoder architecture for robust network intrusion detection in embedded systems. *Journal of Systems Architecture*, v. 156, p. 103283, 2024. ISSN 1383-7621. Disponível em: <https://doi.org/10.1016/j.sysarc.2024.103283>.

[31] RAHMAN, M. M.; SHAKIL, S. A.; MUSTAKIM, M. R. A survey on intrusion detection system in iot networks. *Cyber Security and Applications*, v. 3, p. 100082, 2025. ISSN 2772-9184. Disponível em: <https://doi.org/10.1016/j.csa.2024.100082>.

[32] KIM, T.; PAK, W. Early detection of network intrusions using a gan-based one-class classifier. *IEEE Access*, v. 10, p. 119357–119367, 2022. Disponível em: <https://doi.org/10.1109/ACCESS.2022.3221400>.

[33] FAN, G.; CHEN, S. Design of network security anomaly detection model based on slna cell structure. *IEEE Access*, v. 12, p. 172004–172017, 2024. Disponível em: <https://doi.org/10.1109/ACCESS.2024.3494242>.

[34] FATHIMA, A. H. N.; IBRAHIM, S. P. S.; KHRAISAT, A. Enhancing network traffic anomaly detection: Leveraging temporal correlation index in a hybrid framework. *IEEE Access*, v. 12, p. 136805–136824, 2024. Disponível em: <https://doi.org/10.1109/ACCESS.2024.3458903>.

[35] PROENCA, M.; ZARPELAO, B.; MENDES, L. Anomaly detection for network servers using digital signature of network segment. In: *Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications*

Workshop (AICT/SAPIR/ELETE'05). [s.n.], 2005. p. 290–295. Disponível em: <https://doi.org/10.1109/AICT.2005.26>.

[36] ADANIYA, M. H. A. C. et al. Anomaly detection using dsns and firefly harmonic clustering algorithm. In: *2012 IEEE International Conference on Communications (ICC)*. [s.n.], 2012. p. 1183–1187. Disponível em: <https://doi.org/10.1109/ICC.2012.6364088>.

[37] ASSIS, M. V. O. de et al. Holt-winters statistical forecasting and aco metaheuristic for traffic characterization. In: *2013 IEEE International Conference on Communications (ICC)*. [s.n.], 2013. p. 2524–2528. Disponível em: <https://doi.org/10.1109/ICC.2013.6654913>.

[38] CARVALHO, L. F. et al. A novel anomaly detection system to assist network management in sdn environment. In: *2017 IEEE International Conference on Communications (ICC)*. [s.n.], 2017. p. 1–6. Disponível em: <https://doi.org/10.1109/ICC.2017.7997214>.

[39] ASSIS, M. V. et al. A gru deep learning system against attacks in software defined networks. *Journal of Network and Computer Applications*, v. 177, p. 102942, 2021. ISSN 1084-8045. Disponível em: <https://doi.org/10.1016/j.jnca.2020.102942>.

[40] NOVAES, M. P. et al. Adversarial deep learning approach detection and defense against ddos attacks in sdn environments. *Future Generation Computer Systems*, v. 125, p. 156–167, 2021. ISSN 0167-739X. Disponível em: <https://doi.org/10.1016/j.future.2021.06.047>.

[41] ZACARON, A. M. et al. Generative adversarial network models for anomaly detection in software-defined networks. *Journal of Network and Systems Management*, v. 32, n. 4, p. 93, 2024. ISSN 1573-7705. Disponível em: <https://doi.org/10.1007/s10922-024-09867-z>.

[42] do Rosário, C. R. et al. Modeling of tacit knowledge in industry: Simulations on the variables of industrial processes. *Expert Systems with Applications*, v. 42, n. 3, p. 1613–1625, 2015. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2014.09.023>.

[43] WANG, S. et al. Machine learning in network anomaly detection: A survey. *IEEE Access*, v. 9, p. 152379–152396, 2021. Disponível em: <https://doi.org/10.1109/ACCESS.2021.3126834>.

[44] FOSIć, I. et al. Anomaly detection in netflow network traffic using supervised machine learning algorithms. *Journal of Industrial Information Integration*, v. 33, p. 100466, 2023. ISSN 2452-414X. Disponível em: <https://doi.org/10.1016/j.jii.2023.100466>.

[45] XIN, Y. et al. Machine learning and deep learning methods for cybersecurity. *IEEE Access*, v. 6, p. 35365–35381, 2018. Disponível em: <https://doi.org/10.1109/ACCESS.2018.2836950>.

[46] ENGELEN, J. E. van; HOOS, H. H. A survey on semi-supervised learning. *Machine Learning*, v. 109, n. 2, p. 373–440, February 2020. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/s10994-019-05855-6>.

[47] NOVAES, M. P. et al. Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. *IEEE Access*, v. 8, p. 83765–83781, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2020.2992044>.

[48] CAO, W. et al. A review on neural networks with random weights. *Neurocomputing*, v. 275, p. 278–287, 2018. ISSN 0925-2312. Disponível em: <https://doi.org/10.1016/j.neucom.2017.08.040>.

[49] YUAN, X. et al. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, v. 30, n. 9, p. 2805–2824, 2019. Disponível em: <https://doi.org/10.1109/TNNLS.2018.2886017>.

[50] MAHDAVIFAR, S.; GHORBANI, A. A. Application of deep learning to cybersecurity: A survey. *Neurocomputing*, v. 347, p. 149–176, 2019. ISSN 0925-2312. Disponível em: <https://doi.org/10.1016/j.neucom.2019.02.056>.

[51] SARKER, I. H. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, v. 2, n. 3, p. 160, 2021. ISSN 2661-8907. Disponível em: <https://doi.org/10.1007/s42979-021-00592-x>.

[52] LIU, N.; ZHAO, J. Streaming data classification based on hierarchical concept drift and online ensemble. *IEEE Access*, v. 11, p. 126040–126051, 2023. Disponível em: <https://doi.org/10.1109/ACCESS.2023.3327637>.

[53] ZHANG, H. et al. Multilayer concept drift detection method based on model explainability. *IEEE Access*, v. 12, p. 190791–190808, 2024. Disponível em: <https://doi.org/10.1109/ACCESS.2024.3517697>.

[54] WEBB, G. I. et al. Characterizing concept drift. *Data Mining and Knowledge Discovery*, v. 30, n. 4, p. 964–994, July 2016. ISSN 1573-756X. Disponível em: <https://doi.org/10.1007/s10618-015-0448-4>.

[55] BISHOP, C. *Pattern Recognition and Machine Learning*. Springer, 2006. Disponível em: <https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/>.

[56] WEBB, G. I. et al. Analyzing concept drift and shift from sample data. *Data Mining and Knowledge Discovery*, v. 32, n. 5, p. 1179–1199, 2018. ISSN 1573-756X. Disponível em: <https://doi.org/10.1007/s10618-018-0554-1>.

[57] HáJEK, A. Conditional probability. In: BANDYOPADHYAY, P. S.; FORSTER, M. R. (Ed.). *Philosophy of Statistics*. Amsterdam: North-Holland, 2011, (Handbook of the Philosophy of Science, v. 7). p. 99–135. Disponível em: <https://doi.org/10.1016/B978-0-444-51862-0.50003-4>.

[58] GARCíA-PAVIONI, A.; LóPEZ, B. Dimensionality reduction and features visual representation based on conditional probabilities applied to activity classification. *Computers in Biology and Medicine*, v. 167, p. 107595, 2023. ISSN 0010-4825. Disponível em: <https://doi.org/10.1016/j.compbiomed.2023.107595>.

[59] BAYRAM, F.; AHMED, B. S.; KASSLER, A. From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems*, v. 245, p. 108632, 2022. ISSN 0950-7051. Disponível em: <https://doi.org/10.1016/j.knosys.2022.108632>.

[60] GONçALVES, P. M. et al. A comparative study on concept drift detectors. *Expert Systems with Applications*, v. 41, n. 18, p. 8144–8156, 2014. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2014.07.019>.

[61] DITZLER, G. et al. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, v. 10, n. 4, p. 12–25, 2015. Disponível em: <https://doi.org/10.1109/MCI.2015.2471196>.

[62] PEREIRA, E. V.; SILVA, W. S. d. A comparison of approaches for handling concept drifts in data processed with machine learning. *IEEE Access*, v. 13, p. 61109–61121, 2025. Disponível em: <https://doi.org/10.1109/ACCESS.2025.3557229>.

[63] GAMA, J. et al. Learning with drift detection. In: BAZZAN, A. L. C.; LABIDI, S. (Ed.). *Advances in Artificial Intelligence – SBIA 2004*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 286–295. ISBN 978-3-540-28645-5. Disponível em: <https://doi.org/10.1007/978-3-540-28645-5_29>.

[64] BAENA-GARCIA, M. et al. Early drift detection method. In: *Fourth international workshop on knowledge discovery from data streams*. [S.l.: s.n.], 2006. v. 6, p. 77–86.

[65] BARROS, R. S. et al. Rddm: Reactive drift detection method. *Expert Systems with Applications*, v. 90, p. 344–355, 2017. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2017.08.023>.

[66] FRíAS-BLANCO, I. et al. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, v. 27, n. 3, p. 810–823, 2015. Disponível em: <https://doi.org/10.1109/TKDE.2014.2345382>.

[67] PESARANGHADER, A.; VIKTOR, H. L. Fast hoeffding drift detection method for evolving data streams. In: FRASCONI, P. et al. (Ed.). *Machine Learning and Knowledge Discovery in Databases*. Cham: Springer International Publishing, 2016. p. 96–111. ISBN 978-3-319-46227-1. Disponível em: <https://doi.org/10.1007/978-3-319-46227-1_7>.

[68] ROSS, G. J. et al. Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, v. 33, n. 2, p. 191–198, 2012. ISSN 0167-8655. Disponível em: <https://doi.org/10.1016/j.patrec.2011.08.019>.

[69] HUANG, D. T. J. et al. Detecting volatility shift in data streams. In: *2014 IEEE International Conference on Data Mining*. [s.n.], 2014. p. 863–868. Disponível em: <https://doi.org/10.1109/ICDM.2014.50>.

[70] BACH, S. H.; MALOOF, M. A. Paired learners for concept drift. In: *2008 Eighth IEEE International Conference on Data Mining*. [s.n.], 2008. p. 23–32. Disponível em: <https://doi.org/10.1109/ICDM.2008.119>.

[71] BIFET, A.; GAVALDà, R. Learning from time-changing data with adaptive windowing. In: *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM)*. [s.n.], 2007. p. 443–448. Disponível em: <https://doi.org/10.1137/1.9781611972771.42>.

[72] GU, F. et al. Concept drift detection based on equal density estimation. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. [s.n.], 2016. p. 24–30. Disponível em: <https://doi.org/10.1109/IJCNN.2016.7727176>.

[73] BU, L.; ALIPPI, C.; ZHAO, D. A pdf-free change detection test based on density difference estimation. *IEEE Transactions on Neural Networks and Learning Systems*, v. 29, n. 2, p. 324–334, 2018. Disponível em: <https://doi.org/10.1109/TNNLS.2016.2619909>.

[74] BU, L.; ZHAO, D.; ALIPPI, C. An incremental change detection test based on density difference estimation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, v. 47, n. 10, p. 2714–2726, 2017. Disponível em: <https://doi.org/10.1109/TSMC.2017.2682502>.

[75] SAKTHITHASAN, S.; PEARS, R.; KOH, Y. S. One pass concept change detection for data streams. In: PEI, J. et al. (Ed.). *Advances in Knowledge Discovery and Data Mining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 461–472. ISBN 978-3-642-37456-2. Disponível em: <https://doi.org/10.1007/978-3-642-37456-2_39>.

[76] MACIEL, B. I. F.; SANTOS, S. G. T. C.; BARROS, R. S. M. A lightweight concept drift detection ensemble. In: *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*. [s.n.], 2015. p. 1061–1068. Disponível em: <https://doi.org/10.1109/ICTAI.2015.151>.

[77] WANG, H.; ABRAHAM, Z. Concept drift detection for streaming data. In: *2015 International Joint Conference on Neural Networks (IJCNN)*. [s.n.], 2015. p. 1–9. Disponível em: <https://doi.org/10.1109/IJCNN.2015.7280398>.

[78] YU, S.; ABRAHAM, Z. Concept drift detection with hierarchical hypothesis testing. In: *Proceedings of the 2017 SIAM International Conference on Data Mining (SDM)*. [s.n.], 2017. p. 768–776. Disponível em: <https://doi.org/10.1137/1.9781611974973.86>.

[79] KRAWCZYK, B. et al. Ensemble learning for data stream analysis: A survey. *Information Fusion*, v. 37, p. 132–156, 2017. ISSN 1566-2535. Disponível em: <https://doi.org/10.1016/j.inffus.2017.02.004>.

[80] BIFET, A.; GAVALDÀ, R. Adaptive learning from evolving data streams. In: ADAMS, N. M. et al. (Ed.). *Advances in Intelligent Data Analysis VIII*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 249–260. ISBN 978-3-642-03915-7. Disponível em: <https://doi.org/10.1007/978-3-642-03915-7_22>.

[81] GOMES, H. M. et al. Adaptive random forests for evolving data stream classification. *Machine Learning*, v. 106, n. 9, p. 1469–1495, 2017. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/s10994-017-5642-8>.

[82] GOMES, H. M.; READ, J.; BIFET, A. Streaming random patches for evolving data stream classification. In: *2019 IEEE International Conference on Data Mining (ICDM)*. [s.n.], 2019. p. 240–249. Disponível em: <https://doi.org/10.1109/ICDM.2019.00034>.

[83] HULTEN, G.; SPENCER, L.; DOMINGOS, P. Mining time-changing data streams. In: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2001. (KDD '01), p. 97–106. ISBN 158113391X. Disponível em: <https://doi.org/10.1145/502512.502529>.

[84] WANG, H. et al. Mining concept-drifting data streams using ensemble classifiers. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2003. (KDD '03), p. 226–235. ISBN 1581137370. Disponível em: <https://doi.org/10.1145/956750.956778>.

[85] KOLTER, J. Z.; MALOOF, M. A. Dynamic weighted majority: An ensemble method for drifting concepts. *J. Mach. Learn. Res.*, JMLR.org, v. 8, p. 2755–2790, dez. 2007. ISSN 1532-4435. Disponível em: <https://dl.acm.org/doi/10.5555/1314498.1390333>.

[86] YALCIN, A.; ERDEM, Z.; GURGEN, F. Ensemble based incremental svm classifiers for changing environments. In: *2007 22nd international symposium on computer and information sciences*. [s.n.], 2007. p. 1–5. Disponível em: <https://doi.org/10.1109/ISCIS.2007.4456862>.

[87] DITZLER, G.; POLIKAR, R.; CHAWLA, N. An incremental learning algorithm for non-stationary environments and class imbalance. In: *2010 20th International Conference on Pattern Recognition*. [s.n.], 2010. p. 2997–3000. Disponível em: <https://doi.org/10.1109/ICPR.2010.734>.

[88] ELWELL, R.; POLIKAR, R. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, v. 22, n. 10, p. 1517–1531, 2011. Disponível em: <https://doi.org/10.1109/TNN.2011.2160459>.

[89] BRZEZINSKI, D.; STEFANOWSKI, J. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, v. 25, n. 1, p. 81–94, 2014. Disponível em: <https://doi.org/10.1109/TNNLS.2013.2251352>.

[90] CAO, F. et al. Density-based clustering over an evolving data stream with noise. In: *Proceedings of the 2006 SIAM International Conference on Data Mining (SDM)*. [s.n.], 2006. p. 328–339. Disponível em: <https://doi.org/10.1137/1.9781611972764.29>.

[91] WANG, X. et al. A concept drift tolerant abnormal detection method for network traffic. In: *2023 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. [s.n.], 2023. p. 323–330. Disponível em: <https://doi.org/10.1109/CyberC58899.2023.00057>.

[92] XU, L. et al. Addressing concept drift in iot anomaly detection: Drift detection, interpretation, and adaptation. *IEEE Transactions on Sustainable Computing*, v. 9, n. 6, p. 913–924, 2024. Disponível em: <https://doi.org/10.1109/TSUSC.2024.3386667>.

[93] QIAO, H.; NOVIKOV, B.; BLECH, J. O. Concept drift analysis by dynamic residual projection for effectively detecting botnet cyber-attacks in iot scenarios. *IEEE Transactions on Industrial Informatics*, v. 18, n. 6, p. 3692–3701, 2022. Disponível em: <https://doi.org/10.1109/TII.2021.3108464>.

[94] WAHAB, O. A. Intrusion detection in the iot under data and concept drifts: Online deep learning approach. *IEEE Internet of Things Journal*, v. 9, n. 20, p. 19706–19716, 2022. Disponível em: <https://doi.org/10.1109/JIOT.2022.3167005>.

[95] JAIN, M.; KAUR, G.; SAXENA, V. A k-means clustering and svm based hybrid concept drift detection technique for network anomaly detection. *Expert Systems with Applications*, v. 193, p. 116510, 2022. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2022.116510>.

[96] YIN, Z. et al. Caeaid: An incremental contrast learning-based intrusion detection framework for iot networks. *Computer Networks*, v. 262, p. 111161, 2025. ISSN 1389-1286. Disponível em: <https://doi.org/10.1016/j.comnet.2025.111161>.

[97] NIXON, C. et al. Salad: A split active learning based unsupervised network data stream anomaly detection method using autoencoders. *Expert Systems with Applications*, v. 248, p. 123439, 2024. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2024.123439>.

[98] AMIN, M. et al. Cyber security and beyond: Detecting malware and concept drift in ai-based sensor data streams using statistical techniques. *Computers and Electrical Engineering*, v. 108, p. 108702, 2023. ISSN 0045-7906. Disponível em: <https://doi.org/10.1016/j.compeleceng.2023.108702>.

[99] WU, Y. et al. Online ensemble learning-based anomaly detection for iot systems. *Applied Soft Computing*, v. 173, p. 112931, 2025. ISSN 1568-4946. Disponível em: <https://doi.org/10.1016/j.asoc.2025.112931>.

[100] KHAN, F.; KUMAR, B. S. S.; SANGANI, S. A feature-level ensemble machine learning approach for attack detection in iot networks. *Discover Internet of Things*, v. 5, n. 1, p. 81, jul. 2025. ISSN 2730-7239. Disponível em: <https://doi.org/10.1007/s43926-025-00185-7>.

[101] GOLDSCHMIDT, P.; CHUDá, D. Network intrusion datasets: A survey, limitations, and recommendations. *Computers & Security*, v. 156, p. 104510, 2025. ISSN 0167-4048. Disponível em: <https://doi.org/10.1016/j.cose.2025.104510>.

[102] SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, v. 27, n. 3, p. 379–423, 1948. Disponível em: <https://doi.org/10.1002/j.1538-7305.1948.tb01338.x>.

[103] SINGH, D.; SINGH, B. Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, v. 97, p. 105524, 2020. ISSN 1568-4946. Disponível em: <https://doi.org/10.1016/j.asoc.2019.105524>.

[104] SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. In: . Red Hook, NY, USA: Curran Associates Inc., 2012. (NIPS'12), p. 2951–2959. Disponível em: <https://dl.acm.org/doi/10.5555/2999325.2999464>.