UNIVERSIDADE FEDERAL FLUMINENSE

VINÍCIUS FIGUEIREDO DOS SANTOS

Detecção de Intrusão em Sistemas Ciber-Físicos com Uso de Técnicas de Aprendizado de Máquina

NITERÓI

2022

VINÍCIUS FIGUEIREDO DOS SANTOS

Detecção de Intrusão em Sistemas Ciber-Físicos com Uso de Técnicas de Aprendizado de Máquina

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação.

Orientador:
Célio Vinicius Neves de Albuquerque

Coorientador:
Diego Gimenez Passos

NITERÓI

2022

VINÍCIUS FIGUEIREDO DOS SANTOS

Detecção de Intrusão em Sistemas Ciber-Físicos com Uso de Técnicas de Aprendizado de Máquina.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação.

Aprovada em julho de 2022.

BANCA EXAMINADORA

Prof. Célio Vinicius Neves de Albuquerque - Orientador, UFF

Prof. Diego Gimenez Passos - Coorientador, UFF

Prof. Prof. Daniel Mossé, University of Pittsburgh

Prof. Débora Christina Muchaluat Saade, UFF

Prof. Silvio Ereno Quincozes, UFU

Niterói

2022

Aos meus pais: Alberto Jesus dos Santos e Maria de Fátima Figueiredo dos Santos.

A persistência é o caminho do êxito. (Charles Chaplin)

Agradecimentos

Agradeço, primeiramente, a Deus por ser meu principal guia em toda essa jornada, por me dar força e coragem para iniciar e terminar essa caminhada.

Agradeço em seguida aos meu pais que estiveram presentes, prestando seu apoio de todas as formas a mim ao longo da jornada de toda a minha vida.

Agradeço a minha esposa Rossana por permanecer ao meu lado em todos os momentos dessa caminhada. Agradeço pela compreensão durante os momentos em que me ausentei de atividades familiares para dar andamento nos trabalhos aqui apresentados e pelas palavras de apoio e incentivo nos momentos mais difíceis. Agradeço também aos meus filhos Isaac e Letícia por tornarem o mestrado um desafio muito mais prazeroso.

Gostaria de agradecer ao meu orientador Célio, que me mostrou os caminhos a serem seguidos e pela confiança depositada. Agradeço ao professor Diego Passos pela paciência, dedicação e pelos muitos ensinamentos proporcionados.

Por fim, agradeço à Marinha do Brasil por conceder a liberação integral para cursar o mestrado.

Resumo

Os Sistemas Ciber-físicos, do inglês *Cyber-Physical Systems* (CPS), estão cada vez mais interligados aos sistemas críticos de infraestrutura que têm a capacidade de impactar a vida humana no nosso dia-a-dia. A segurança das informações digitais é de vital importância para estes sistemas, como, por exemplo, nas plantas industriais de tratamento de água ou nas malhas de energia elétrica *Smart Grid* (SG), onde um invasor pode prejudicar a distribuição dos recursos vitais ou de equipamentos elétricos, se as medidas de segurança necessárias não forem adotadas. Neste contexto, os Sistemas de Detecção de Intrusão, do inglês *Intrusion Detection Systems* (IDS), se fazem necessários como uma linha de defesa para esses sistemas tão importantes do nosso cotidiano.

O uso de técnicas de Aprendizado de Máquina (AM) em IDS é importantes para a detecção de intrusões com o mínimo de intervenção humana. Neste contexto, as técnicas clássicas empregadas em um conjunto de dados (*datasets*) limitados e estáticos durante o aprendizado são conhecidas como AM *Off-line*. Porém, com a necessidade cada vez maior de utilizar os IDS em sistemas críticos em tempo real, surge a necessidade do uso de técnicas de aprendizado de máquina em fluxo de dados, conhecidas como AM *On-line*. Nesta última técnica, cada amostra deverá passar uma única vez nas etapas de aprendizado e de teste, sem serem armazenadas permanentemente na memória, devido ao grande fluxo de dados.

Muitos estudos na área de IDS em CPS não fazem uso adequado de *datasets* específicos da área, o que acaba por não representar uma relação direta com os equipamentos reais existentes nesses sistemas. Esta motivação levou a um levantamento dos principais *datasets* existentes na literatura referente a CPS, que serão posteriormente utilizados nos experimentos dessa dissertação.

Logo, este trabalho tem o objetivo de analisar a detecção de invasões em sistemas ciber-físicos com estes *datasets*, com a utilização de métodos IDS com técnicas de Aprendizado de Máquina *Off-line* e Aprendizado de Máquina *On-line*. No final, é constatado que o AM *Off-line* apresenta as melhores métricas de classificação para detecção de ataques, enquanto o AM *On-line* tem melhor capacidade de adaptação na presença de novos

ataques, com menor demanda de recursos de memória e processamento.

Palavras-chave: Sistemas Ciber-físicos; Sistemas de Detecção de Intrusão; Aprendizado de Máquina *On-line*; Aprendizado de Máquina *Off-line*.

Abstract

Cyber-Physical Systems (CPS) are increasingly interconnected to the most diverse critical infrastructure systems that can impact human life in our daily lives. The security of digital information is of vital importance for these systems, such as industrial water treatment plants or Smart Grid electrical grids, where an attack can harm a part of the distribution of vital resources or electrical equipment if the necessary security measures are not adopted. In this context, Intrusion Detection Systems (IDS) are needed as a second line of defense for these systems that are so important in our daily lives.

The use of Machine Learning (ML) techniques in IDS is important as an automatic way to detect intrusions with minimal human intervention. In this context, machine learning techniques on standard datasets are known as Offline ML. However, with the increasing need to use IDS in Big Data with time constraints and limited memory, there is a need to use data flow machine learning techniques, which are known as Online ML. In this last technique, each sample must pass the learning and testing stages only once, without being stored in memory, due to the large flow of data. Many studies in the area of IDS in CPS do not make adequate use of specific datasets of the area representing the actual existing equipment of these systems. This motivation led to a survey of the main datasets existing in the literature referring to CPS that will be later used in the experiments of this dissertation.

Therefore, this work aims to compare the intrusion detection in cyber-physical systems with these suitable datasets using IDS methods with Offline Machine Learning techniques and Online Machine Learning.

Keywords: Cyber-physical systems; Intrusion Detection Systems; Offline Machine Learning; Online Machine Learning.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AMI Infraestrutura de Medição Avançada, do inglês, *Advanced Metering Insfrastructure*

BW Janela Básica, do inglês, *Basic Window*

CAN Rede de Área Controladora, do inglês, *Controller Area Network*

CIP Protocolo Industrial Comum, do inglês, *Common Industrial Protocol*

CSV Valores separados por vírgula, do inglês, *Comma Separeted Values*

CV Validação Cruzada, do inglês, *Cross-Validation*

CVE Exposições e Vulnerabilidades Comuns, do inglês, *Common Vulnerability and Exposures*

DDoS Ataque DoS distribuído, do inglês, *Distributed DoS*

DNP3 Protocolo de Rede Distribuído versão 3, do inglês, *Distributed Network Protocol version 3*

DoS Negação de Serviço, do inglês, *Denial of Service*

DS Conjunto de dados, *Dataset*

DSA Algoritmo de Assinatura Digital, do inglês, *Digital Signature Algorithm*

ECDSA Algoritmo de Assinatura Digital de Curva Elíptica, do inglês, *Elliptic Curve Digital Signature ALgorithm*

FF Fator de Desvanecimento, do inglês, *Fading Factor*

FNR Taxa de Falso Negativo, do inglês, *False Negative Rate*

FPR Taxa de Falso Positivo, do inglês, *False Positive Rate*

FTP Protocolo de Transferência de Arquivos, do inglês, *File Transfer Protocol*

GOOSE Eventos de subestação orientados a objetos genéricos, do inglês, *Generic Object Oriented Substation Events*

HAN Rede de Área Doméstica, do inglês, *Home Area Network*

HAT Árvore Adaptativa de Hoeffding, do inglês, *Hoeffding Adaptative Tree*

HIDS *Host-based* IDS

HIL *Hardware-In-the-Loop*

HMI Human Machine Interface

HT Árvore de Hoeffding, do inglês, *Hoeffding Tree*

HTTP Protocolo de Transferência de Hipertexto, do inglês, *Hypertext Transfer Protocol*

HTTPS Protocolo de Transferência de Hipertexto Seguro, do inglês, *Hypertext Transfer Protocol Secure*

HWT Árvore de Janela de Hoeffding, do inglês, *Hoeffding Window Tree*

ICT Tecnologia de Comunicação e Informação, do inglês, *Information and Communication Technology*

IDPS Sistema de Detecção e Prevenção de Intrusões, do inglês, *Intrusion Detection and Prevention System*

IEC Comissão Eletrotécnica Internacional, do inglês, *International Electrotechnical Commission*

IED Dispositivos Eletrônicos Inteligentes, do inglês, *Intelligent Electronic Device*

IEEE Instituto de Engenheiros Eletricistas e Eletrônicos, do inglês, *Institute of Electrical and Electronics Engineers*

IoT Internet das Coisas, do inglês, *Internet of Things*

IP Protocolo de Internet, do inglês, *Internet Protocol*

JSON Notação Objeto JavaScript, do inglês, *JavaScript Object Notation*

KNN K Vizinhos Próximos, do inglês, *K-Nearest Neighbors*

LB *Leveraging Bag*

LEACH Hierarquia de Núcleos Consciente de Baixa Energia *Low Energy Aware Cluster Hierarchy*

LIN *Local Interconnect Network*

LLDP Protocolo de Descoberta da Camada de Enlace, do inglês, *Link Layer Discovery Protocol*

LOF *Local Outlier Factor*

ML Aprendizado de Máquina, do inglês, *Machine Learning*

MMS *Manufacturing Message Specification*

MOA Análise Online Massiva, do inglês, *Massive Online Analysis*

MOST *Media Oriented System Transport*

MU Unidades de Fusão, do inglês, *Merging Units*

NB Naive Bayes

NIDS *Network-based* IDS

OBA *Oza Bag Adwin*

OPC-UA *Open Platform Communications - Unified Architecture*

OSI Sistemas Abertos de Interconexão, do inglês, *Open System Interconnection*

PCAP Captura de Protocolo, do inglês, *Protocol Capture*

PLC Controlador Lógico Programável, do inglês, *Programmable Logic Controller*

PTP Protocolo de Tempo de Precisão, do inglês, *Precision Time Protocol*

R2L Ataque remoto para local, do inglês, *Remote to Local Attack*

RESTful Transferência de Estado Representacional, do inglês, *Representational State Transfer*

RSA Rivest-Shamir-Adleman

RSL Revisão Sistemática da Literatura

RTU Unidade de Terminal Remoto, do inglês, Remote Terminal Unit

SAS Sistemas de Automação de Subestação, do inglês, *Substation Automation Systems*

SCADA Sistema de Supervisão e Aquisição de Dados, do inglês, *Supervisory Control And Data Acquisition*

SDN Redes Definidas por Software, do inglês, *Software Defined Network*

SEP Perfil de Energia Inteligente, do inglês, *Smart Energy Profile*

SO Sistema Operacional

SSH *Secure Shell*

SV Valores Amostrados, do inglês, Sampled Value

SW Janela Deslizante, do inglês, *Sliding Window*

TCP Protocolo de Controle de Transmissão, do inglês, *Transmission Control Protocol*

TES Sistema de Energia Transativa, do inglês, *Transactive Energy Systems*

TI Tecnologia da Informação

TLS Segurança de Camada de Transporte, do inglês, *Transport Layer Security*

U2R Ataque de usuário para root, do inglês, *User to Root Attack*

UC Caso de Uso, do inglês *User Case*

VOIP Voz sobre IP, do inglês, *Voice Over IP*

WEKA *Waikato Environment for Knowledge Analysis*

WSN Rede de Sensores Sem Fio, do inglês, *Wireless Sensor Network*

XML Linguagem de Marcação Extensível, do inglês, *eXtensible Markup Language*

Sumário

1 Introdução

1.1 Motivações

Os sistemas ciber físicos, do inglês *Cyber-Physical Systems* (CPS), são integrações que envolvem computação, comunicação e controle através de redes e processos físicos (QUINCOZES; MOSSÉ et al., 2021). Estes sistemas estão sujeitos aos mais diversos ataques que podem afetar diversas vidas humanas.

Desde a década de 1960, muitos processos industriais de controle têm utilizado Sistema de Controle Industrial, do inglês *Industrial Control System* (ICS) (ANTON et al., 2017), e Sistema de Supervisão e Aquisição de Dados, do inglês *Supervisory Control And Data Acquisition* (SCADA). Eles são exemplos de CPS que são largamente empregados em várias áreas de infraestruturas críticas, como plantas de energia elétrica, tratamento de água e distribuição de gás. Esta ampla utilização facilita as operações, porém se torna cada vez mais sujeita a ataques cibernéticos.

Temos como exemplo a mudança no sistema de distribuição de água das infraestruturas físicas tradicionais para um sistema ciber físico nas últimas décadas. Esta nova forma de operação, aumentou a vulnerabilidade deste sistema de infraestrutura, sendo identificado em 2016 como o terceiro alvo mais atacado no mundo (TAORMINA et al., 2018).

Em 2010 ocorreu o ciberataque conhecido como Stuxnet, que prejudicou o Programa Nuclear do Irã. Este *worm* foi projetado especificamente para atacar o sistema SCADA desenvolvido pela Siemens e usado para controlar as centrífugas de enriquecimento de urânio iranianas. Ele foi descoberto em junho de 2010 por uma empresa bielorrussa desenvolvedora de antivírus (LANGNER, 2011).

Outro exemplo, foi o ciberataque na Ucrânia em 2015 conhecidos como *BlackEnergy* que foi responsável por interromper o serviço de energia elétrica para aproximadamente 225 mil usuários (KHAN et al., 2016; CASE, 2016). Os alvos foram principalmente servi-

ços do setor de energia, do governo e da mídia. O ataque foi feito por um cavalo de Troia aos sistemas de ICS e aos protocolos de equipamentos de empresas de energia elétrica. O ocorrido foi alertado em fevereiro de 2016 pela Equipe de Resposta a Emergências Cibernéticas de Sistemas de Controle Industrial dos Estados Unidos, do inglês *United States Industrial Control Systems Cyber Emergency Response Team* (US ICS-CERT), do Departamento de Segurança Interna (DHS)[1].

Todos estes ciberataques foram possíveis devido à convergência de protocolos de comunicação usados pelos CPS. Temos como exemplo a tecnologia *Smart Grid* (SG), ou redes elétricas inteligentes, onde novas funções como comunicação em duas vias, atualização constante do comportamento do consumo, controle de casas inteligentes, componentes remotos inteligentes e monitoramento da distribuição elétrica, levam os equipamentos elétricos inteligentes, do inglês *Intelliget Electronic Device* (IED), a serem responsáveis por controlar atividades importantes do cotidiano. A incorporação de novos protocolos de comunicação aos IEDs leva a uma preocupação quanto a segurança desses dispositivos, por expô-los a ameaças de segurança como quebra de privacidade, ganhos financeiros, roubo de energia e diversas atividades maliciosas (ASSIS, 2021).

1.1.1 Sistemas de Detecção de Intrusão

A segurança dos CPS deve contar a princípio com o uso de criptografia, *firewalls* e antivírus, como uma segurança primária. Porém, outra forma, tão importante quanto, são os Sistemas de Detecção de Intrusão, do inglês *Intrusion Detection Systems* (IDS), que são componentes fundamentais de defesa de redes e sistemas de computadores. Os IDS específicos de interesse dessa dissertação são os baseados em modelos de classificação avaliados a partir de *datasets* específicos de CPS.

A importância do IDS pode ser vista, por exemplo, nas tecnologias de Rede de Sensores Sem Fio, do inglês *Wireless Sensor Network* (WSN) e da Internet das Coisas, do inglês *Internet of Things* (IoT). Estas tecnologias são empregadas em aplicações em tempo real, como aplicações militares de vigilância, monitoramento de incêndios em florestas, cuidados de saúde, monitoramento de segurança de edifícios, ou seja, em partes fundamentais do nosso cotidiano. Para proteger estes dispositivos de ataques externos, a criptografia é um bom mecanismo de defesa, porém não garante proteção contra ataques internos onde as chaves de segurança estão expostas ao atacante. Por este motivo, para prover a segurança nessas redes é necessário utilizar um IDS, que faça uso de técnicas para detecção de

[1] Site: https://www.cisa.gov/uscert/ics/alerts/IR-ALERT-H-16-056-01, fevereiro de 2022.

anomalias (ALMOMANI; AL-KASASBEH; AL-AKHRAS, 2016).

Desta forma, os IDS têm um importante papel no projeto e desenvolvimento de uma infraestrutura de rede robusta para defender de ataques os recursos, através da detecção ou bloqueio. O IDS é capaz de detectar ataques em diferentes camadas, logo ele tem um papel fundamental na segurança dos CPS. *Datasets* confiáveis são importantes dados para testar e avaliar o desempenho destes sistemas de detecção.

1.1.2 Desafios

Claramente, CPS está empregado em uma vasta área de aplicações da Indústria 4.0 e a sua segurança, através de diversas técnicas, é cada vez mais importante. Estes sistemas podem ser divididos em três camadas de arquitetura: Percepção ou Física, Transmissão e Aplicação (QUINCOZES; MOSSÉ et al., 2021). A camada de Percepção consiste em sensores e atuadores que fazem uso de sistemas de controle baseados em computadores. A camada de Transmissão consiste na estrutura hierárquica de comunicação entre as pontas da rede e as centrais através de protocolos específicos de enlace, roteamento e rede. A camada de Aplicação consiste no uso prático da estrutura responsável pelo monitoramento e controle de decisão, onde as suas funções dependem da área em uso, como por exemplo, *Healthcare*, *Smart Grid* (SG), Automação Industrial, Sistemas Veiculares, etc.

Atualmente há uma tendência de incluir as aplicações CPS próximos aos componentes físicos presentes nas pontas das redes, devido à busca na redução do atraso entre os controladores (componentes físicos) e os nós centrais de decisão (componentes da aplicação), o que leva a diversos desafios na área, devido aos recursos limitados dos dispositivos físicos quanto a energia das baterias, memória e capacidades de processamento. É consenso que o componentes da camada física ou de percepção são os mais vulneráveis devido a esses limites, por não permitirem a implantação computacional de pesados recursos de segurança (QUINCOZES; PASSOS et al., 2020). Existem estudos na literatura que mostram vários *datasets* coletados dos mais diversos protocolos, porém eles não são adequados para sistemas de detecção de intrusão em sistemas ciber físicos, por apresentarem pouca relação com os reais equipamentos existentes (GOH et al., 2016). Logo, muitos trabalhos não fazem estudos adequados quanto a um CPS, o que leva a necessidade de uma levantamento aprofundado de *datasets*, o que será abordado nessa dissertação na Seção 2.6.1.

Outro tema importante é a existência cada vez maior de um grande volume de dados que exigem processamentos em tempo de aplicação em empresas e instituições científicas, o que torna críticos os recursos relacionados com espaço de memória e tempo de pro-

cessamento. Desta forma, um IDS para estes grandes volumes de dados necessita usar algoritmos de Aprendizado de Máquina, do inglês, *Machine Learning* (ML), que operem em tempo real. Se os dados a serem analisados forem tão grandes ao ponto de não poderem ser armazenados por completo, os métodos de aprendizado em fluxo de dados deverão ser usados para processar uma amostra por vez.

A efetividade deste aprendizado pode ser medida em razão do tempo e das métricas de classificação. Como foi possível observar apenas poucos estudos na literatura que façam uma comparação direta entre o aprendizado em um conjunto de dados limitados e finitos com *datasets* (ML *Off-line*) e o aprendizado em fluxo de dados em tempo de aplicação (ML *On-line*), este trabalho sobre IDS se faz necessário. Desta forma, o estudo dessa dissertação será especificamente para CPS com grande fluxo de dados, onde os gargalos de recursos são mais desafiadores, com uso de técnicas de Aprendizado de Máquina *On-line*.

1.2 Objetivo

O aprendizado proveniente de um fluxo de dados é muito diferente de um aprendizado tradicional de um conjunto de dados estáticos e finitos durante a fase de treinamento. O *software* WEKA (*Waikato Environment for Knowledge Analysis*) tem se apresentado como uma das ferramentas de avaliação dos classificadores de ataques em IDS que não levam em consideração os limites de recursos de memória e tempo de execução durante o treinamento. Porém, atualmente, com o surgimento cada vez maior de sistemas com grandes volumes de dados e com a necessidade de levar o processamento dos dados para os nós da rede, há a necessidade de fazer a classificação de ataques em tempo de execução da aplicação de um grande fluxo de dados. Nesta dissertação, este processamento será feito com o *software* MOA (*Massive Online Analysis*), levando em consideração o limite de memória do *hardware*. Cabe enfatizar que há poucos trabalhos referentes a aprendizado de máquina de IDS que utilizem *datasets* específicos de CPS, o que será o material principal dos experimentos aqui realizados.

Logo, este trabalho irá analisar e comparar estes dois paradigmas, de forma a descrever as suas funcionalidades básicas e as principais implementações no meio acadêmico. Deverá ser observado aqui qual desses paradigmas será melhor executado em dispositivos nas extremidades da rede CPS, como nos sensores ou atuadores. Em seguida, serão feitos experimentos com objetivo de comparar os seus desempenhos quanto ao consumo de

recursos e quanto às métricas de classificação dos ataques. Para isso, serão comparados o tempo de execução, assim como a acurácia, precisão, *recall* e F1*Score*. Desta forma, as seguintes perguntas deverão ser respondidas: Qual a diferença em desempenho e métrica de classificação entre o ML *Off-line* e o ML *On-line*? Qual será a melhor abordagem para IDS a ser desenvolvido para um dispositivo de *hardware* limitado? Qual técnica tem a melhor capacidade em identificar novos ataques e qual apresenta as menores ocorrências de Falso Positivo e Falso Negativo?

1.3 Organização

O restante do texto está organizado da seguinte forma. O Capítulo 2 irá descrever conceitos e definições quanto aos Sistemas de Detecção de Intrusão, seguido dos principais *datasets* em CPS encontrados na literatura. Iremos em seguida, no Capítulo 3, descrever as principais pesquisas na área referente a IDS relacionados com Aprendizado de Máquina *On-line* em CPS relacionados com essa dissertação. No Capítulo 4, são apresentados os experimentos com o objetivo de comparar as duas abordagens ML *On-line* e ML *Off-line*. Finalmente, no Capítulo 5, são apresentadas as conclusões e considerações finais.

2 Conceitos Básicos de Sistema de Detecção de Intrusão

Este capítulo será dividido em uma seção que apresentará os conceitos básicos de um IDS, assim como as características principais para classificação dos diversos sistemas existentes. Descreveremos em seguida a utilização de IDS em tecnologias CPS, como sistemas de controle, IoT e SG. Em sequência, são apresentadas diversas técnicas atuais de IDS para detecção de ataques com Aprendizado de Máquina *Off-line*. No final, são descritas as técnicas atuais de IDS que fazem uso de Aprendizado de Máquina *On-line*, seguidas da apresentação dos principais *frameworks* para os experimentos e as métricas de avaliação das classificações. Na seção seguinte, descrevemos os principais *datasets* de CPS encontrados na literatura para serem utilizados nos experimentos dessa dissertação. Por último, iremos descrever os trabalhos relacionados com esta dissertação, ou seja, estudos com ML *On-line* com *datasets* reais.

A primeira linha de defesa de qualquer sistema digital deverá fazer uso de técnicas de criptografia, autenticação e autorização, a fim de prover a confidencialidade, integridade, disponibilidade e não repúdio das informações digitais. Porém, em alguns casos, a utilização de técnicas de criptografia inviabiliza as comunicações em tempo real. Como exemplo temos o padrão IEC-61850 (QUINCOZES, S. E., 2019) para SG, onde o IDS será o principal agente de defesa.

A detecção de intrusão é o monitoramento de eventos em um sistema ou rede de comunicação em busca de sinais de possíveis incidentes, tais como violações às políticas de segurança ou às práticas comuns de segurança em sistemas de computadores. Há várias causas para tais incidentes, que vão desde acesso não autorizado por parte de terceiros ou uso mal intencionado de privilégios, até ataques de Negação de Serviço, do inglês, *Denial of Service* (DoS). Desta forma, um IDS deverá ser capaz de detectar ataques através da análise de comportamentos que saiam da operação normal do sistema.

Os IDS podem ser categorizados em três grupos de detecção: baseados em assinatura, baseados em anomalias e baseados em especificações (NIXON; SEDKY; HASSAN, 2019;

QUINCOZES; ALBUQUERQUE et al., 2021). Os baseados em assinatura caracterizam-se por ter um banco de dados com amostras que representam perfis de ataques conhecidos ou simulados. Desta forma, consiste na procura por padrões já pré-estabelecidos de atividades de cunho malicioso. Os baseados em anomalia caracterizam-se por ter no banco de dados o perfil de tráfego da rede ou o comportamento dos sistemas que representa as atividades legítimas e normais, o que os torna ideais para os casos de invasões desconhecidas, pois possuem a capacidade de detectar atividades e dados diferentes do perfil tradicional do sistema. Já os baseados em especificação são combinações dos anteriores, porém assumem também uma configuração parametrizada manualmente pelo usuário, o que pode gerar atrasos na detecção de novos ataques.

Eles podem ser classificados em três categorias dependendo da localização dos agentes: *Host-based* IDS (HIDS); *Network-based* IDS (NIDS); e *Distributed IDS* (RADOGLOU-GRAMMATIKIS; SARIGIANNIDIS, 2019). Os *Network-Based* são normalmente empregados nas fronteiras entre duas redes, próximas a *firewalls* ou roteadores, onde monitoram o tráfego e os protocolos para identificar atividades suspeitas. Os *Host-Based* são normalmente empregados em alguns dispositivos cujo funcionamento é essencial para o sistema, como servidores de acesso público, ou que contenham informações sigilosas. Eles monitoram as características de um único dispositivo e os eventos que acontecem com ele em busca de atividades suspeitas, como: tráfego da rede, *logs* do sistema, processos em execução, acesso e alteração em arquivos. Os *Distributed IDS* apresentam uma estrutura híbrida, ou distribuída dentro da rede, onde cada componente se mostra mais especializado em uma forma de detecção. Neste tipo de IDS podem ser utilizados tanto o HIDS e NIDS para apresentarem uma solução integrada para identificação de ataques.

A maioria dos IDS não apresenta uma resposta ativa às invasões e ataques, o que não o configura como um Sistema de Detecção e Prevenção de Intrusões, do inglês, *Intrusion Detection and Prevention System* (IDPS) [1]. A maioria, após detectada uma intrusão, normalmente apresenta apenas um comportamento passivo de resposta, isto é, ele apenas alerta o usuário do ocorrido. Dependendo do tipo de ataque, a detecção de intrusão por um IDS pode gerar a emissão de diversos falsos positivos (FP) e falsos negativos (FN). O primeiro é mais comum em situações onde, após o IDS detectar uma intrusão, ações com comportamentos semelhantes de uma operação normal serão também detectadas como ataques. O falso negativo por sua vez é a intrusão que não é detectada pelo IDS, desta forma é a ocorrência de um ataque bem sucedido.

[1] Site: https://www.gta.ufrj.br/grad/12_1/ids/FuncionamentobsicodeIDS.html, julho de 2022.

Existem três métodos principais para avaliar um IDS, que são: ensaios realistas que utilizam equipamentos físicos; geração sintética de *dataset* com captura de amostras simuladas ou reais de ataques; e a adoção de *datasets* sintéticos conhecidos que contenham amostras normais e de ataques. Deste modo, a maioria dos componentes de um IDS, tão fundamentais na defesa de redes e sistemas de computadores, são baseados em modelos de detecção construídos a partir de *datasets*.

Portanto, os aspectos de projeto de um IDS podem ser divididos em quatro partes: primeira, o grupo de detecção baseados em assinatura, anomalia ou especificação; segunda, a localização dos agentes, já descritas anteriormente; terceira, em ações de detecção ou prevenção, também já detalhadas; e quarta, no tipo de análise *On-line* ou *Off-line*. Este último se caracteriza como a realização das operações de consulta das assinaturas, que podem ser em tempo real, ou seja, de forma *On-line*, ou em lotes sem restrições de tempo, de forma *Off-line*.

As técnicas de ML representam uma grande evolução para a detecção de intrusão dos IDS devido à automação na detecção de ameaças. Os dados normais de um sistema monitorado em operação são geralmente rotulados, o que leva a técnicas de detecção supervisionadas. Enquanto os dados ligados a intrusão geralmente desconhecidos são não rotulados, o que leva ao uso de técnicas de detecção semi-supervisionados ou não supervisionados.

Dentro da área de ML podemos dizer então que os métodos supervisionados executam o levantamento de dados estatísticos do ataque na fase de treinamento por utilizarem rótulos de classe (método baseado em assinatura) para maior precisão nas classificações. Já os métodos não supervisionados permitem a detecção de ataques mesmo que nenhum dado específico seja utilizado para a fase de treinamento (método baseado em anomalia) (CHANDOLA; BANERJEE; KUMAR, 2009). Como exemplo temos os algoritmos de agrupamento (clusterização) *k-means* ou distanciamento, que não utilizam rótulos de classe. Os métodos semi-supervisionados utilizam técnicas híbridas que fazem uso dos rótulos apenas em determinados momentos. Desta forma, a detecção de intrusão baseada em assinaturas permite uma maior flexibilidade em relação às regras estáticas de IDS baseados em especificação e ela possui uma maior precisão em relação aos IDS baseados em detecção de anomalias.

Para mais detalhes na área de ML não supervisionado podemos destacar o trabalho de Chandola, Banerjee e Kumar (2009), onde são apresentadas as mais diversas técnicas nos mais diversos domínios de aplicação. O foco desse artigo é a descrição geral das técnicas

de detecção de anomalias, onde pouco é descrito sobre os IDS, porém os conceitos sobre detecção e classificação são passados de uma maneira didática pelos autores.

2.1 Uso de Sistema de Detecção de intrusão para *Cyber-Physical Systems*

2.1.1 Necessidade de Sistema de Detecção de Intrusão para Sistemas de Controle

Um IDS voltado para CPS deverá ser capaz de detectar maus comportamentos nas diferentes camadas, onde os componentes da camada de percepção são os mais vulneráveis devido aos limitados recursos de *hardware* dos dispositivos, conforme já mencionado. Por exemplo, podemos descrever o Controlador Lógico Programável, do inglês, *Programmable Logic Controller* (PLC) como um dos principais recursos para controle físico do CPS nas indústrias. Eles requerem Sistemas Operacionais (SO) adaptados para a necessidade das aplicações industriais, o que leva a estes sistemas a herdarem as falhas de um SO como Windows [2] (ANTON et al., 2017).

Os PLC antigamente eram restritos a redes internas industriais, desta forma a segurança era garantida pelo controle de acessos aos dispositivos físicos (ANTON et al., 2017). Porém, ao longo da década de 1970, barramentos de comunicação, como CAN (*Controller Area Network*) e Modbus, foram aos poucos sendo substituídos por soluções baseadas em TCP/IP, como ModbusTCP, ProfiNET, OPC-UA, *Local Interconnect Network* (LIN), *Media Oriented System Transport* (MOST) e FlexRay (ANTON et al., 2017). Esta convergência no uso dos barramentos de comunicação traz problemas, como expor os equipamentos de controle de produção industrial a ataques vastamente empregados na internet.

Muitos desses barramentos de comunicação também apresentam falhas de segurança, como falta de autenticação e criptografia, o que permite a execução de ataques de extração e de injeção de mensagens. Porém existem exceções, como o trabalho de Anton et al. (2017), que menciona o OPC-UA[3] (*Open Platform Communications - Unified Architecture*) como um protocolo que se destaca por conter funcionalidades de segurança, como criptografia assimétrica com autenticação. Infelizmente, nesse trabalho nenhum teste ou experimento foi feito ao longo do trabalho sobre este protocolo específico.

[2] Site: `https://bit.ly/3Pikl0D`, julho de 2022.
[3] Site: `https://opcfoundation.org/about/opc-technologies/opc-ua/`, setembro de 2021.

2.1.2 Necessidade de Sistema de Detecção de Intrusão para Internet das Coisas

Assim como o PLC, os equipamentos de Internet das Coisas, do inglês *Internet of Things* (IoT), atingiram uma vasta área de aplicações da indústria 4.0 (ZANELLA et al., 2014). A tecnologia IoT embarca componentes computacionais que permitem aos objetos físicos ver e ouvir, de forma a executar tarefas e decisões através do compartilhamento de informações ou de outras decisões pela Internet (AL-FUQAHA et al., 2015). Com uma arquitetura distribuída, esta tecnologia é vulnerável a diversos ciberataques que podem prejudicar a privacidade e a disponibilidade dos serviços fornecidos, logo deverá ser protegida através das diversas técnicas de segurança, em que o IDS é uma delas.

Porém, a partir da leitura de trabalhos relacionados com o tema, pode ser observado que as técnicas tradicionais de IDS não são adequadas para as redes IoT (BENKHELIFA; WELSH; HAMOUDA, 2018). Isso se deve à dificuldade de projetar um sistema de detecção para dispositivos de localização imprevisível na rede e com uma vasta gama de tecnologias empregadas que geram uma natureza não definido do tráfego, como o da internet. Logo, as técnicas de detecção dos IDS requerem uma atualização constante para manter preciso o banco de dados e os modelos de classificação, o que torna os algoritmos pesados para serem executados em limitados *hardwares* de IoT.

2.1.3 Necessidade de Sistema de Detecção de Intrusão para *Smart Grid*

As redes elétricas inteligentes são a evolução das malhas elétricas tradicionais com a introdução da Tecnologia de Comunicação e Informação, do inglês, *Information and Communication Technology* (ICT) (RADOGLOU-GRAMMATIKIS; SARIGIANNIDIS, 2019). A tecnologia proporciona uma comunicação bidirecional entre os consumidores e as empresas fornecedoras em tempo real, o que pode proporcionar múltiplos benefícios como medições e manutenções automáticas, gerenciamento eficiente de energia, segurança e confiabilidade.

O surgimento dessa nova forma de distribuição de energia requer a instalação de Sistemas de Automação de Subestação, do inglês, *Substation Automation Systems* (SAS), com funções automatizadas para unificação de diferentes protocolos de comunicação em um único protocolo (USTUN; FAROOQ; HUSSAIN, 2019), o que pressupõe um aumento da conectividade entre os IED de diferentes protocolos. Um dos benefícios de se adotar as SG é o aumento da eficiência no consumo de energia de forma a fornecer apenas a potência necessária aos equipamentos inteligentes que solicitarem. Outro benefício é a possibilidade

de proteção do sistema elétrico de forma remota e automática (QUINCOZES, S. E., 2019). Porém, essa concentração de diferentes procolos pode expor estes sistemas a diversos ataques.

As SG são exemplos de CPS que fazem uso de componentes com recursos limitados na camada de percepção, o que dificulta a implementação de técnicas de segurança, devido à baixa capacidade de processamento, memória e energia deste equipamentos (JACOBSEN; MIKKELSEN, 2014). As SG fazem uso também de sistemas e equipamentos IoT, que além de apresentarem os problemas já descritos, possuem uma vasta variedade de protocolos e de *hardwares* que levam a soluções pobres de integração, o que têm prejudicado a escalabilidade de dispositivos e a especificação de tempo das SG (MELONI; ATZORI, 2016).

Logo, como em qualquer camada de percepção de um CPS, o IDS é importante como uma linha de defesa para os sistemas SG onde as técnicas de criptografia, autenticação e autorização não são suficientes para garantir a segurança. Podemos citar como exemplo os ataques que têm como objetivo interferir no Sistema de Energia Transativa, do inglês, *Transactive Energy Systems* (TES) (ZHANG et al., 2019), que é um registro de transações cujo objetivo é realizar o equilíbrio entre o fornecimento e o consumo de energia elétrica entre produtores e consumidores. Os TES são executados e controlados por equipamentos IoT, o que os torna vulneráveis a ciberataques cujo objetivo é causar danos financeiros. Logo, para estes casos, o IDS é uma importante ferramenta de defesa. No Apêndice B, apresentamos a importância de se utilizar os IDS nos novos padrões SG atualmente em desenvolvimento.

Embora as técnicas de detecção de intrusão sejam comumente estudadas em redes e sistemas convencionais de comunicação, apenas alguns estudos abordam as SG. As técnicas tradicionais de detecção de IDS apresentam limitações quanto à escalabilidade para operar nestes sistemas (RADOGLOU-GRAMMATIKIS; SARIGIANNIDIS, 2019), devido em parte ao limite em monitorar e interpretar dados vindos de múltiplas fontes. Outro ponto observado é que poucos estudos se referem ao atraso ou consumo dos recursos computacionais em sistemas SG.

Como exemplo, nos trabalhos de Sílvio Ereno Quincozes (2019) e Radoglou-Grammatikis e Sarigiannidis (2019) as técnicas de inteligência artificial, a operação em tempo real e a implementação de Redes Definidas por Software, do inglês, *Software Defined Network* (SDN), são apontadas como promissoras para o desenvolvimento de um IDS para SG por possibilitarem um aumento de acurácia na detecção e a redução de métricas de Taxa de

Falso Positivo, do inglês, *False Positive Rate* (FPR) e Taxa de Falso Negativo, do inglês, *False Negative Rate* (FNR).

2.2 Abordagem por Aprendizado de Máquina *Off-line*

A adoção de um IDS isolado, que opere em modo *Host-Based*, terá um baixo desempenho em bloquear ataques devido à visão limitada dos serviços da rede. IDS de baixo desempenho geram muitos falsos positivos quando qualquer atividade foge do comportamento normal. Logo, uma forma de reduzir o FPR é adotar técnicas com regras pré definidas. Porém, a atualização manual constante destas regras se mostra inviável (PAN; MORRIS; ADHIKARI, 2015). A adoção de *Network-Based* IDS é uma solução que tem causado muitos benefícios nos últimos anos com o surgimento de métodos de cooperação em nuvem em sistemas com grande quantidade de dados (FARHAT et al., 2020). Porém, a utilização desse grande fluxo de dados em tempo real gera problemas quanto ao armazenamento e processamento (BIFET; HOLMES et al., 2010). Logo, os principais métodos para otimização no desempenho dos NIDS são agrupados em dois grandes grupos: a detecção temporal e a seleção de atributos.

Os estudos relacionados à detecção temporal vêm ocorrendo em pesquisas relacionadas com geração de grande fluxo de dados em tempo real que fazem uso do *framework* como MOA[4] (*Massive Online Analysis*), que serão mais bem detalhadas na Seção 2.3. Já estudos relacionados com precisão podem ser feitos com seleção de atributos, ou *Feature Selection* (FS), utilizadas em IDS de forma a permitir descartar dados irrelevantes ou redundantes, o que gera uma menor carga de processamento e memória, além de possibilitar um aumento na acurácia.

A seleção de atributos é muito importante para identificar os ataques desejados, porém em alguns casos deve-se saber qual ataque queremos defender para selecionar quais características serão mais importantes de serem avaliadas. Como exemplo, para as técnicas de classificação baseadas em assinaturas, em que as características estatísticas dos ataques são conhecidas, a FS pode representar uma melhora na acurácia com melhoria no desempenho, como é citada no trabalho Quincozes, Passos et al. (2020). Porém, nestas técnicas, os ataques desconhecidos não são facilmente detectados. Para estes casos, podem ser empregadas técnicas baseadas em anomalias, em que as atividades que fogem do padrão normal conhecido são identificadas como possíveis ataques.

[4] Site: `https://moa.cms.waikato.ac.nz/`, setembro de 2021.

As técnicas de seleção de atributos são vistas como operações frequentemente utilizadas no aprendizado de máquina para redução de dimensão, que consiste na seleção de um subgrupo da dimensão original ou construção de uma nova dimensão (CHANDOLA; BANERJEE; KUMAR, 2009; ESSEGHIR, 2010; YUSTA, 2009). O objetivo da FS é encontrar atributos específicos retirados de um grupo grande de atributos que maximizem as habilidades de um classificador. A busca de uma combinação de seleção de atributos não é muito usual em aplicações em tempo real (QUINCOZES; PASSOS et al., 2020), devido a restrições de memória e tempo. Muitos casos de FS podem levar um tempo excessivo para encontrar um subgrupo de *features* dentro de um intervalo de tempo viável.

As FS podem ser divididas em três modalidades (YUSTA, 2009): *Wrappers*, que testam subgrupos de *features* em relação aos resultados das métricas dos algoritmos de aprendizagem de máquina; Filtros, ou do inglês *Filters*, que não levam em consideração o processo de aprendizado de máquina, mas sim as relações de entropia, distância, informação ou consistência; Embarcado, ou do inglês *Embedded*, que seleciona atributos durante o processo interno de geração de um classificador. Podemos notar na literatura que a modalidade *Wrapper* provê resultado melhor que a modalidade Filtro, porém é computacionalmente mais lenta e custosa. Os trabalhos de Yusta (2009), Esseghir (2010) e Quincozes, Mossé et al. (2021) são referências da área de FS.

Trabalhos mostram que os IDS apresentam um bom desempenho para os ataques que são conhecidos durante a fase de treinamento, porém apresentam um desempenho ruim para os novos ataques que estão presentes apenas na fase de teste. Como exemplo, temos o trabalho de Lippmann et al. (2000). Para os ataques conhecidos, as métricas de acurácia foram entre 63% a 93%, com a taxa de 10 falsos positivos por dia. Já para os ataques presentes apenas na fase de teste, as métricas de acurácia foram abaixo de 25%. Desta forma, os resultados mostram que os IDS podem detectar muitos ataques existentes com baixo falso positivo, contanto que os ataques estejam presentes na fase de treinamento. Logo, as pesquisas deverão então ser voltadas para a detecção de novos ataques ou variantes de antigos ataques.

Uma solução, a princípio, é a utilização de técnicas de clusterização, o que permite agrupar o fluxo normal dos dados em grupos, onde qualquer atividade que fuja destes agrupamentos conhecidos seja identificada, o que provê uma grande capacidade em detectar novos ataques.

Um dos trabalhos que busca fazer uso desta forma para identificar novos ataques é o Portnoy (2000). Ele propõe detectar intrusão sem a necessidade de classificação dos

dados na fase de treinamento. Para isso, é utilizada a técnica de detecção de anomalias conhecida como clusterização *k-means*.

O algoritmo referenciado passa diversas vezes pelas amostras durante a fase de treinamento. Em cada passagem, o centro do *cluster* é deslocado para a média dos pontos de todas as amostras e continua a interação até que nenhuma mudança significativa no centro do *cluster* ocorra mais. Para determinar quais *clusters* são ataques e quais são normais, os autores partiram do suposto que as instâncias normais são maiores do que as de ataque. Logo, cada instância será rotulada com o *label* de classificação do *cluster* de menor distância.

Quanto mais intrusões forem apresentadas e treinadas pelo classificador, o conjunto de *clusters* e a taxa de detecção ficam melhores (PORTNOY, 2000). Porém, pode ser concluído que os métodos de classificação não supervisionados de clusterização possuem uma baixa acurácia e um aumento no número de falsos positivos, por classificarem qualquer atividade que foge da operação normal como ataque. Por esta razão, há a necessidade de se usar técnicas híbridas que contenham tanto os métodos não supervisionados quanto os métodos supervisionados. Novos métodos híbridos de IDS apresentam desempenhos melhores do que os métodos usuais, o que pode ser visto nos artigos Al-Nashif et al. (2008), Baek et al. (2017), Aljamal et al. (2019), Farhat et al. (2020), Ha et al. (2016), Quincozes, Raniery et al. (2021) e Benkhelifa, Welsh e Hamouda (2018).

2.2.1 Classificadores Supervisionados *Off-Line*

A partir da leitura dos diversos artigos descritos na seção anterior, os principais classificadores de Aprendizado de Máquina supervisionados que serão utilizados nos experimentos desta dissertação são detalhados (QUINCOZES; SANTOS et al., 2019; WITTEN; FRANK, 2002):

- *Random Tree*, o algoritmo constrói uma árvore que consiste em K atributos escolhidos aleatoriamente em cada nó;
- *J48*, semelhante ao algoritmo árvore de decisão, onde regras de classificação são extraídas do conceito de entropia;
- *REPTree*, consiste em um algoritmo de aprendizado de árvore de decisão rápida que, de forma semelhante ao J48, é baseado também em entropia;
- *Naive Bayes*, este classificador utiliza medidas estatísticas do teorema de Bayesian;

- *Random forest*, este método combina preditores de árvores baseados em amostras de vetores aleatórios com a mesma distribuição para todas as árvores;

Quatro classificadores em árvore (J48, *REPTree*, *Random Tree* e *Random Forest*) foram selecionados por serem os mais citados nas referências estudas durante o mestrado. O classificador *Naive Bayes* foi selecionado por ser um do mais básico para servir de parâmetro de comparação.

2.3 Abordagem por Aprendizado de Máquina *On-line*

A proliferação de grandes volumes de dados de empresas e instituições científicas torna necessário que os algoritmos de aprendizado de máquina operem com grandes *datasets*, o que torna crítico os recursos relacionados como espaço de memória e tempo de processamento (WITTEN; FRANK, 2002). Se os dados a serem analisados forem tão grandes ao ponto de não poderem ser armazenados por completo na memória principal, os métodos de aprendizado incremental deverão ser usados de forma a processar uma amostra por vez. Esta solução requer mecanismos sofisticados de armazenamento de memória utilizados de modo a prover rápido acesso às partes mais usadas das amostras do *dataset*. Outro aspecto é o tempo de processamento do aprendizado de máquina que deverá crescer de maneira linear à medida que o número de amostras do *dataset* torna-se grande.

Existem alguns procedimentos que tornam o processo de aprendizado de grandes *datasets* viáveis, como a utilização de:

- **Subconjunto de *dataset* para treinamento**, o que pode resultar em sub amostragens e perda de informação;
- **Paralelização com uso de processadores separados**, o que exige a utilização de aprendizado que possam ocorrer de maneira paralela, como os métodos de Árvores de Decisão que podem usar subárvores;
- **Fatias do *dataset* em pedaços de tamanhos fixos** onde cada parte será submetida ao processo de aprendizado separadamente, e os pedaços serão combinados no final através de média ou votação;
- **Seleção de atributos**, conforme mencionado anteriormente, são importantes também pois desconsideram dados irrelevantes para a classificação.

Porém, os métodos tradicionais, mesmo que empreguem algum desses procedimentos, não são suficientes pois sofrem um elevado custo para perceberem que o método de classificação não está com uma métrica adequada ao fluxo recente de dados e deverá logo descartar o modelo antigo, para então aprender um novo a partir das informações mais recentes. Desta forma, as técnicas tradicionais não apresentam um processamento adequado e gerenciamento de memória para este tipo de operação, por consequência são pouco escaláveis para as aplicações práticas no mundo real.

Uma forma promissora para lidar com grandes *datasets* é desenvolver algoritmos que consideram os dados de entrada como um fluxo contínuo (WITTEN; FRANK, 2002). Desta forma, o algoritmo deverá analisar apenas uma amostra por vez para ceder espaço para a próxima amostra no fluxo. Como o algoritmo não tem controle na ordem das amostras, ele deverá atualizar o modelo de aprendizagem de maneira incremental a cada chegada de uma nova amostra. Assim sendo, o modelo deverá ser capaz de testar uma amostra em qualquer momento durante o treinamento, o que o torna ideal para operar com fluxos de dados em tempo real (WITTEN; FRANK, 2002).

Estas características tornam possível a operação desses algoritmos por tempo indefinido utilizando uma quantidade limitada de memória. Outra funcionalidade, que estes algoritmos deverão apresentar é capacidade de processamento por amostra. O processamento deverá ocorrer em uma velocidade superior ao fluxo de dados de entrada, de preferência dentro de um limite de tempo de curta duração. Desta forma, estudos relacionados à detecção de intrusão em tempo real vêm ocorrendo em pesquisas relacionadas com sistemas com grande fluxo de dados, o que leva a adoção de técnicas de mineração de dados para a detecção de ciberataques (WISESA et al., 2016; BIFET; FRANCISCI MORALES et al., 2015). Logo, estes modelos de predição em tempo real são descritos na literatura também como métodos de Aprendizagem de Máquina *On-line* em fluxos de dados. São uma família de métodos que tentam prever a classe de uma instância a partir do aprendizado de uma sequência de dados (HOI et al., 2021).

O objetivo desse aprendizado é maximizar as métricas para uma sequência de predições com restrição de tempo. De maneira geral, podemos agrupar os classificadores em três categorias principais (NIXON; SEDKY; HASSAN, 2019):

- **Aprendizado Supervisionado *On-line***, atualiza seu modelo de predição com uso de rótulos de classificação através de alguma estratégia para melhorar o seu desempenho utilizando função de perda como parâmetro (conforme será descrito, Figura 3) ;

- **Aprendizado Semi-Supervisionado *On-line* (com retorno parcial)**, busca fazer a predição da classe dos fluxos de forma a receber como retorno partes dos rótulos das classes reais para avaliação das predições;
- **Aprendizado Não Supervisionado *On-line***, aprende a partir de fluxos de dados sem rótulos e sem alimentação de retorno (*feedback*) para realizar predições.

A classificação ou predição dos fluxos de dados traz novas demandas e tarefas desafiadoras na área de aprendizado de máquina, como utilizar um tempo constante por dados de amostra com uma quantidade fixa de memória principal. Nestes cenários, é necessário atualizar o classificador em tempo real, que consiste em incorporar novas informações na velocidade que os dados chegam, identificar as mudanças e adaptar os modelos de decisão com os dados mais recentes (HIDALGO; MACIEL; BARROS, 2019). O processo de classificação para aprendizagem de fluxo de dados é dividido em duas etapas, conforme mostrado na Figura 1:

- **Construção do modelo**: cada amostra pertence a uma classe predefinida, onde o modelo será construído para determinar qual atributo e valor determina a classe rotulada. Dessa forma, esse modelo é representado por regras de classificação, árvores de decisão, ou fórmulas matemáticas.
- **Uso do modelo**: é a classificação ou estimação das amostras futuras desconhecidas utilizando o modelo construído. Nessa etapa, também são calculadas as métricas do modelo, feito a partir do rótulo conhecido da amostra por *datasets* sintéticos ou devidamente rotulados previamente, comparado com o resultado classificado do modelo para um processo de classificação supervisionado. Como exemplo, uma das métricas calculadas na etapa de uso do modelo pode ser a taxa de acurácia.

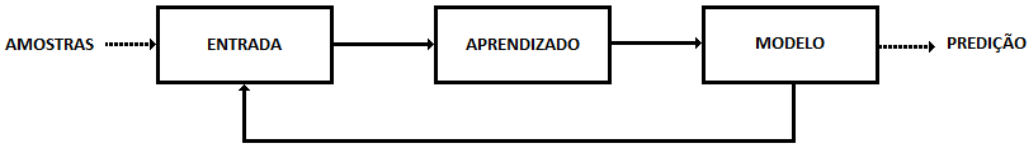

Figura 1: Ciclo de classificação do fluxo de dados. Adaptado de: (BIFET; HOLMES et al., 2010)

2.3.1 Mudança de Conceito

Geralmente as aplicações em tempo real são dinâmicas, desta forma o fluxo de dados muda ao longo do tempo, o que diminui a relevância em construir um modelo fixo para fazer futuras predições (DAHAL et al., 2015). O fluxo de dados pode ser tanto para processos do tipo estacionários, cuja distribuição de probabilidade não muda ao longo do tempo, quanto para processos do tipo não estacionários, cuja distribuição de probabilidade muda ao longo do tempo. Atualmente as fontes de dados são distribuídas e geradas por um número crescente de dispositivos, onde os dados são transientes e não devem ser armazenados permanentemente (GAMA; SEBASTIAO; RODRIGUES, 2009). Desta forma, os fluxos de dados serão do tipo não estacionários, o que diminui a relevância em construir um modelo fixo para fazer futuras predições. Estas mudanças temporais não estacionárias são chamadas de Mudança de Conceito (*Concept Drift*) e desempenham um papel fundamental em ambientes dinâmicos. Elas podem ser classificadas como abruptas ou graduais, conforme Figura 2.

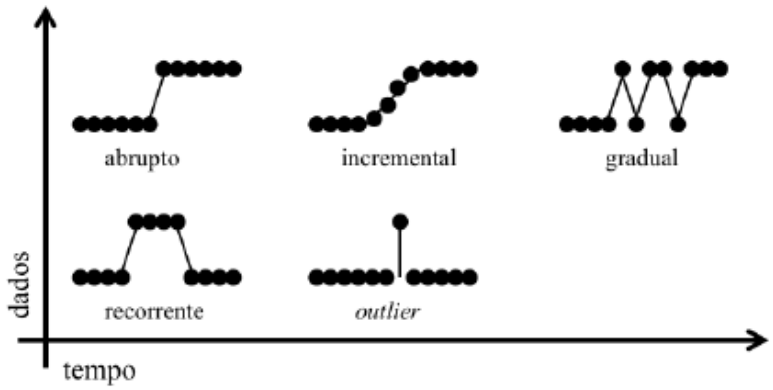

Figura 2: Tipos de Mudança de Conceito. Extraído de: (CORRÊA, 2017)

2.3.2 Método de Validação

Os problemas relacionados com o fluxo de dados passam pela falta de métodos de validação (BIFET; FRANCISCI MORALES et al., 2015). As formas de avaliação dos métodos clássicos de aprendizado utilizam conjunto finitos de dados com técnicas como Treino/Teste ou *cross-validation* (CV), com as suas variantes *leave-one-out* e *bootstrap*. Elas não são apropriadas para fluxo de dados devido ao tamanho restrito do *dataset* e a distribuição estacionária de amostras independentes, o que não é aplicável para fluxo de dados (GAMA; SEBASTIAO; RODRIGUES, 2009). Duas alternativas são mais usuais para o caso de fluxo de dados, conforme Gama, Sebastiao e Rodrigues (2009):

- ***Holdout***, aplica o modelo de classificação em testes com intervalos de tempo regulares (configurado pelo usuário);
- ***Prequential***, todas as amostras são testadas (predição) e depois aprendidas.

A metodologia para avaliação de modelos de decisão evolutivos, que lidam com Mudança de Conceito, mais apropriada é o prequencial (junção das palavras preditivo e sequencial) (BIFET; FRANCISCI MORALES et al., 2015; GAMA; SEBASTIAO; RODRIGUES, 2009), ou do inglês *prequential* (junção de duas palavras *predictive* e *sequential*), com mecanismos de janela W e fator de desvanecimento α. A abordagem é baseada na premissa de que o objetivo da inferência estatística é fazer previsões de probabilidade sequencial para observações futuras, em vez de expressar informações sobre observações passadas. Desta forma, suponha que no tempo t o modelo de previsão preveja $\hat{y}_{t+k}$, e que após k instantes, no tempo t + k, o sensor meça a quantidade $\bar{y}_{t+k}$. Então, poderá ser estimada a perda da predição $L(\hat{y}_{t+k}, \bar{y}_{t+k})$, conforme ilustrado na Figura 3. O cálculo deste método é obtido a partir da soma cumulativa dos erros sequenciais a cada janela ao longo do tempo, com a função de perda, entre as previsões e os valores observados.

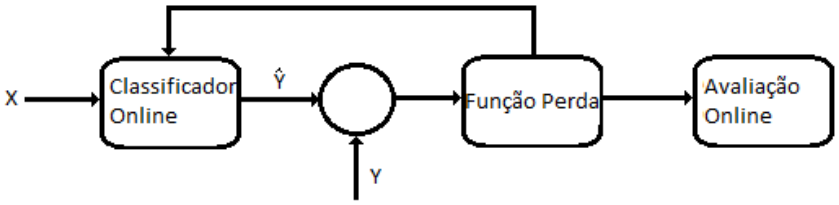

Figura 3: Cálculo da função perda. Adaptado de: (HIDALGO, 2017)

O método prequencial pode ser um aprendizado semi-supervisionado, ou seja, uma técnica com *feedback* limitado, onde não é necessário saber a classe real de todas as amostras, apenas onde precisa ser calculada a função de perda. Porém, nada impede que o método seja também um aprendizado supervisionado, onde todas as amostras são utilizadas com *feedback* para comparar a classe real com as classes previstas. Cabe ao desenvolvedor verificar o parâmetro do projeto do IDS ou a qualidade do *dataset* para determinar qual método de aprendizado será adotado, assim como o momento adequado para fazer a configuração do *feedback*.

A avaliação dos modelos de classificação é muitas vezes realizada de acordo com as métricas, o tempo de execução e o consumo de memória. Em geral, o método prequencial

pode ser usado para avaliar qualquer algoritmo de aprendizagem em cenários de fluxos de dados. Desta forma, cada amostra individual é testada pelo modelo antes de ser usada para treinamento e, portanto, a métrica é atualizada incrementalmente com cálculo em tempo real, que poderá ser feito pela acurácia média, por exemplo. Logo, este cálculos podem ser usados para determinar a ocorrência de Mudança de Conceito também ou para determinar a construção de um novo modelo de classificação.

Como pode ser visto, os modelos são dinâmicos, logo não existe um conjunto de treinamento finito específico, isto é, cada amostra rotulada pode ser utilizada para treinamento. Deve-se observar que o conjunto de teste deverá ser independente do conjunto de treinamento, caso contrário ocorrerá um excesso de treinamento, conhecido como *overfitting* (HIDALGO; MACIEL; BARROS, 2019).

Apesar de haver poucos estudos de metodologias que façam comparação e avaliação de modelos, o prequencial tem vantagens em relação ao procedimento *holdout* para avaliar algoritmos de aprendizagem. Isso se deve ao fato de o modelo ser sempre testado em exemplos não vistos e, portanto, não é necessário um conjunto de treinamento fixo, de modo que os dados disponíveis podem ser totalmente utilizados (HIDALGO; MACIEL; BARROS, 2019). Outra vantagem é que o método prequencial pode utilizar um *feedback* limitado dos rótulos das classes, onde não é necessário saber a classe real de todas as amostras.

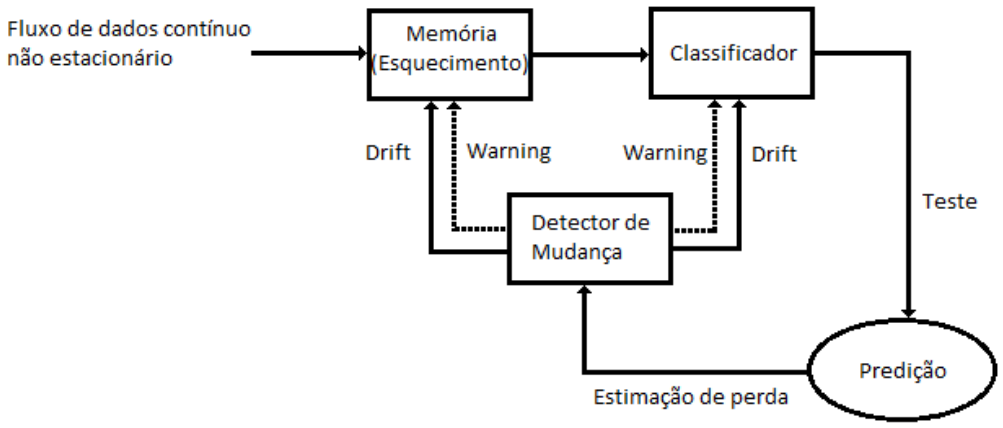

Figura 4: Detector de Mudança de Conceito. Adaptado de: (NIXON; SEDKY; HASSAN, 2019)

A metodologia prequencial, independente da abordagem adotada para o cálculo estatístico, tem a função de avaliar a tarefa de classificação e, para isso, cada amostra é

repassada ao classificador, esperando-se uma resposta. As respostas das predições do classificador são disponibilizadas aos algoritmos de detecção de mudanças de conceitos, conforme ilustra a Figura 4. A função de um detector de mudanças de conceito é notificar ao classificador que ocorreu uma mudança na distribuição de probabilidades no fluxo dos dados e determinar a substituição do modelo de predição por um novo (HIDALGO; MACIEL; BARROS, 2019).

Em geral, os detectores analisam as predições do classificador base e adotam um modelo de decisão para detectar as mudanças na distribuição dos dados. Em específico, estes métodos trabalham com dois níveis de alarmes (HIDALGO, 2017): *warning* e *drift*. Quando um *warning* é sinalizado, uma nova instância do classificador base é criada e mantida em paralelo com o classificador antigo. Quando um *drift* é sinalizado, representa um nível maior de mudança na distribuição analisada e simboliza que, de fato, ocorreu uma modificação de conceito. Desta forma, caso o nível de *drift* seja alcançado, o detector exclui o antigo classificador e mantém apenas o novo. Por outro lado, caso o sinal de *warning* seja considerado um alarme falso positivo (FP) da Mudança de Conceito, a nova instância do classificador é excluída.

2.3.3 Janelas de Amostras Prequencial

Os modelos de decisão incrementais são dinâmicos e evoluem ao longo do tempo, com tendência de melhora no seu desempenho, logo o erro prequencial estimado num fluxo de dados pode ser fortemente influenciado pela primeira parte da sequência do erro, quando poucos exemplos foram usados para treinar o classificador. Esta observação leva à necessidade de calcular o erro usando um mecanismo de esquecimento. Isto pode ser conseguido usando uma janela dinâmica, que detecta automaticamente a taxa de mudança em um fluxo de dados, ou usando fatores de desvanecimento. A janela dinâmica pode ser divida em três tipos (HIDALGO, 2017): *Basic Window* (BW), *Sliding Window* (SW) e *Fading Factors* (FF).

- **Abordagem clássica BW**, também conhecida como *Interleaved Test-Then-Train* (HIDALGO; MACIEL; BARROS, 2019), todas as instâncias processadas do fluxo são usadas no cálculo que constrói o modelo de decisão.
- **Abordagem SW** é criado um mecanismo de esquecimento com o objetivo de minimizar as questões referentes aos cálculos que usam um número muito grande de amostras. A maior dificuldade neste sentido é delimitar o tamanho adequado da

janela. As janelas pequenas garantem uma rápida adaptabilidade em cenários onde acontecem as Mudanças de Conceito e as janelas maiores produzem mais baixos estimadores de variância em fases estáveis, mas não podem reagir rapidamente à mudança.

- **Abordagem FF** é outra variação que adota o fator de esquecimento. Como foi explicado anteriormente, o cálculo para atualizar o modelo do método prequencial dessa janela é através da estimativa do erro baseado em um fator de desvanecimento α. Teoricamente eles são multiplicativos, correspondendo ao esquecimento exponencial dos dados dentro da janela.

Cada uma das variações de janelas prequencial tem algumas dificuldades durante o processo de avaliação dos algoritmos de aprendizagem (HIDALGO, 2017):

- **BW** é difícil fazer a classificação real em um dado momento devido ao uso de todas as amostras processadas no fluxo de dados para atualizar o modelo de classificação;
- **SW** é difícil delimitar o tamanho adequado da janela para determinar as diferentes fases nas mudanças de conceito;
- **FF** é difícil determinar o valor adequado do fator de desvanecimento usado na atualização do modelo de decisão.

Porém, no trabalho de Hidalgo, Maciel e Barros (2019) foram avaliadas experimentalmente as três variações prequencial (BW, SW, FF) em diferentes ambientes onde acontecem mudanças de conceito, fazendo uso dos classificadores *Naive Bayes* (NB) e *Hoeffding Tree* (HT) para cada conjunto de dados artificiais utilizados no trabalho. Os resultados identificam a SW como a abordagem mais adequada a ser usada nos experimentos com a metodologia prequencial pois adquiriu as melhores métricas com os fluxos de dados. Desta forma, a principal contribuição deste trabalho foi identificar a estratégia mais apropriada para ser utilizada nas avaliações experimentais da acurácia na classificação em testes relacionados com Mudanças de Conceito.

Já no trabalho de Gama, Sebastiao e Rodrigues (2009) é apontado que os métodos que fazem uso do FF utilizam menos memória, o que é uma propriedade importante em cenários de fluxo de dados. Logo, este método é mais vantajoso que o método SW e BW que requerem maior memória para armazenar os elementos dentro da janela.

2.3.3.1 Janela Adaptativa (ADWIN)

Durante os testes, a escolha do tamanho da janela e do fator de esquecimento são preponderantes nos resultados alcançados, o que pode levar a resultados equivocados. As janelas SW de tamanho pequeno apresentam flutuações nos resultados, enquanto as janelas SW de tamanho grande apresentam atrasos na detecção de alterações. Como o fluxo de dados pode oscilar, uma janela adaptativa é a melhor solução para resolver estes problemas de parâmetros de configuração das janelas. Logo, a técnica *Adaptive Window* (ADWIN) (BIFET; GAVALDA, 2007) pode ser utilizada.

O algoritmo do ADWIN mantém uma janela de tamanho variável. Ele cresce automaticamente a janela quando não ocorrem mudanças de conceito, e diminui quando as mudanças são detectadas. O algoritmo ajusta o tamanho da janela para o ponto ótimo de equilíbrio entre tempo de reação e pequenas variações, de forma que pode trabalhar junto com um algoritmo de aprendizado de máquina para monitorar a taxa de erro do modelo atual. O ADWIN foi desenvolvido para trabalhar com fluxo de dados de forma a usar pouca memória e em tempo baixo, o que possibilita criar diversas instâncias da janela para acompanhar as diversas estatísticas para construção do modelo de aprendizado. A ideia do algoritmo é simples. Quando duas subjanelas grandes o suficiente são caracterizadas como distintas nas médias, pode ser concluído que os valores esperados são diferentes e a subjanela mais antiga é descartada. Desta forma, ele mantém os valores estimados da média e variância do fluxo de dados para detectar mudanças de conceito e ajustar o tamanho da janela de forma a manter apenas os dados recentes mais relevantes.

Após o exposto, podemos concluir que o Aprendizado de Máquina *On-line* resolve muitos problemas relacionados aos IDS que operam com fluxo de dados, conforme elencados abaixo (ADHIKARI; MORRIS, T. H.; PAN, 2017):

- **Primeiro**: tem a capacidade de processamento de dados em quantidades infinitas.
- **Segundo**: supera a dificuldade dos métodos clássicos de aprendizado de máquina (AM *Off-line*) em processar em tempo real grande quantidade de dados.
- **Terceiro**: detecta Mudança de Conceito nos fluxos de dados, pois possui a capacidade de atualizar os modelos através de um constante treinamento.

2.3.4 Classificadores Supervisionados *On-line*

Os principais métodos de Aprendizado de Máquina *On-line* encontrados na literatura sobre fluxo de dados, conforme será demonstrado na Seção 3.1, são o *Naive Bayes* (NB), *Hoeffding Tree* (HT) e o *Hoeffding Adaptive Tree* (HAT) que são brevemente descritos a seguir.

- **NB** é um classificador probabilístico, também chamado de classificador simples de Bayes ou Independente de Bayes, baseado no teorema de Bayes, com capacidade de prever e diagnosticar problemas através de hipóteses de probabilidade robustas a ruídos. Ele utiliza uma variação da regra de Bayes para prever a classe para uma instância de teste, assumindo que as características são condicionalmente independentes umas das outras, dada a classe.

- **HT** é um algoritmo de classificação incremental popular que combina os dados em uma árvore enquanto o modelo é construído (aprendizado) incrementalmente. A classificação pode ocorrer em qualquer momento. Suporta a indução de árvores de decisão e fornece soluções para incerteza no tempo de aprendizagem.

- **HAT** é um algoritmo de árvores de decisão, construída a partir do HT com a inclusão de um detector de mudança e um estimador. Desta forma, ele é capaz de lidar com fluxos de dados com mudança de conceito e mudança de distribuição sem precisar saber o tamanho ótimo da janela.

As Árvores de Decisão são métodos tradicionais de aprendizado de máquina em grade uso (conforme será visto na Seção 3), muito utilizados para a geração de respostas compreensíveis para análise humana e integração com recursos legados como firewalls (CORRÊA, 2017), por exemplo. Na árvore cada nó contém um teste de atributo, cada ramo corresponde a um possível resultado e cada folha contém uma predição de classe. Ela aprende recursivamente substituindo as folhas por novos testes em nós. O atributo a ser testado no nó é escolhido por comparação entre todos os atributos disponíveis conforme medidas heurísticas. Desta forma, elas são projetadas para trabalhar com limitada quantidade de amostras, que são analisadas múltiplas vezes para construir um modelo que será depois utilizado para fazer as predições.

Já o algoritmo *Hoeffding Tree* (HT) é uma solução que pode utilizar dados estacionários e não estacionários. Ele constrói as árvores de decisão através da leitura dos dados de entrada apenas uma vez, sem a necessidade de armazenamento destes dados para outras

análises (DAHAL et al., 2015). A árvore construída HT, uma árvore de decisão incremental para fluxo de dados, contém as estatísticas suficientes para crescer e fazer previsões das classificações dos dados recebidos. Os seus nós são atualizados a partir dos atributos testados geralmente com uma heurística conhecida, como Ganho de Informação, do inglês *Information Gain* (IG).

As árvores Hoeffding fazem uso do fato de que pequenas amostras podem ser suficientes para escolher uma divisão de atributos ótima, o que é sustentado pela propriedade matemática do limite de Hoeffding (CORRÊA, 2017). Ele quantifica o número de observações necessárias para estimar estatisticamente a divisão de um nó da árvore dentro de uma precisão determinada.

O limite de Hoeffding é usado para garantir que o resultado esteja próximo de um aprendizado convencional (DOMINGOS; HULTEN, 2000). Desta forma, as árvores de HT proporcionam um método de aprendizado de árvore de decisão que aprende com um tempo constante por amostra, vendo as amostras uma única vez, enquanto produzem resultados próximos das árvores convencionais. Logo, o algoritmo HT irá requerer memória da ordem de $O(ldvc)$, onde l é o número de folhas, d é o número de atributos, v é o número máximo de valores por atributo e c é o número de classes.

Muitos algoritmos possuem como requisito inserir parâmetros referentes a velocidade ou frequência da mudança de conceito, como a definição do tamanho da janela deslizante, o valor de decaimento ou esquecimento, limites explícitos de mudança máxima, entre outros, o que requer uma pré-concepção sobre o comportamento dos dados, que pode ser completamente equivocada. Por esta razão, os autores do artigo Bifet e Gavalda (2009) propõem algoritmos que podem aprender de forma adaptativa a partir de um fluxo de dados que muda ao longo do tempo, com uso do algoritmo CVFDT (*Concept-adapting Very Fast Decision Tree*), um evolução do algoritmo VFDT (*Very Fast Decision Tree*) (DOMINGOS; HULTEN, 2000) usado no classificador HT.

São adicionados dois elementos ao algoritmo, o detector de mudança e o estimador, o que gera um novo método capaz de lidar com mudança de conceito, o *Hoeffding Adaptive Tree* (HAT). A vantagem principal desse método é que ele não precisa ter o conhecimento da velocidade da mudança no fluxo de dados, o que em outros métodos requer diversos parâmetros, a serem definidos pelos usuários, para medir este efeito.

Um exemplo da sua implementação é o algoritmo *Hoeffding Window Tree* (HWT) da Figura 5. Ele é chamado como qualquer árvore de decisão que faz uso do limite de Hoeffding com uso de uma janela deslizante com o código geral da referida figura. Desta

forma, o algoritmo proposto possibilita que: os detectores de mudança sejam inseridos em cada nó; a capacidade de criar, gerenciar, trocar e deletar árvores alternativas; e manter estimadores com estatísticas relevantes nos nós da janela deslizante atual.

HOEFFDING WINDOW TREE($Stream, \delta$)

▷ Let HT be a tree with a single leaf(root)
▷ Init sufficient node statistics at root
for each example (x, y) in Stream
    do HWTREEGROW($(x, y), HT, \delta$)

HWTREEGROW($(x, y), HT, \delta$)

▷ Sort (x, y) to leaf l using HT
▷ Update sufficient node statistics
    at leaf l and nodes traversed in the sort
if this node has an alternate tree T_{alt}
    HWTREEGROW($(x, y), T_{alt}, \delta$)
▷ Compute information gain G for each attribute
if G(Best Attr.)$-G$(2nd best)$> \epsilon = \sqrt{R^2 \ln(1/\delta)/(2n)^a}$
    then
▷ Split leaf on best attribute
for each branch
        do ▷ Start new leaf
            and initialize sufficient node statistics
if one accuracy change detector has detected change
    then
▷ Create an alternate tree with the new best attribute at its root, if there is none
if existing alternate tree is more accurate
    then
▷ replace current node with alternate tree

Figura 5: Algoritmo genérico *Hoeffding Window Tree*. Extraído de: (BIFET; GAVALDA, 2009).

Conforme mencionado na Seção 2.3.3, uma das melhores janelas deslizantes proposta é o algoritmo detector de mudança ADWIN (*ADaptive WINdow*) que mantém um tamanho variável da janela com os itens vistos recentemente com uso de um fator de desvanecimento (BIFET; GAVALDA, 2009). O algoritmo tem a propriedade de que a janela mantenha um tamanho máximo com consistência estatística, de forma que não haja mudanças no valor médio das amostras dentro da janela. Desta forma, a janela deslizando dentro do HWT irá funcionar como um estimador e o algoritmo ADWIN irá funcionar como um detector de mudança em cada nó.

Logo, a combinação HWT-ADWIN possibilitará: a detecção de árvores alternativas tão logo uma mudança seja detectada, sem necessidade de esperar uma determinada quantidade de amostras; e substituição de uma árvore antiga por uma árvore alternativa de maior precisão. Pode ser observado que o algoritmo HWT-ADWIN será um CVFDT porém sem a necessidade de configuração dos parâmetros e o tamanho da janela definido

pelo usuário.

A *Hoeffding Adaptive Tree* (HAT) é, então, uma evolução do HWT-ADWIN que aprende de forma adaptativa as mudanças no tempo do fluxo de dados sem precisar de valores fixos para o tamanho das janelas. Desta forma, podemos definir o HAT como um algoritmo de árvores de decisão construídas de maneira incremental capaz de lidar com fluxos de dados com mudança de conceito e mudança de distribuição (ADHIKARI; MORRIS, T. H.; PAN, 2017). Ele cria uma árvore de decisão do fluxo de dados e a atualiza após inspecionar cada instância. Desta forma, as amostras não precisam ser armazenadas na memória pois a sua árvore já armazena os dados necessários em cada nó ao fazer a classificação. Esta característica faz com que a HAT apresente as seguintes vantagens sobre o HWT-ADWIN: não há necessidade de determinar o tamanho ótimo das janelas; cada nó decide qual instância é mais relevante; não há necessidade de janelas adicionais para armazenar instâncias; redução da quantidade de memória necessária.

Para se apurar a qualidade do HAT com uso de ADWIN, no artigo Bifet e Gavalda (2009), são propostas três variantes, conforme o detector de mudança adotado: HAT-INC (*linear INCremental estimator*); HAT-EWMA (*Exponencial Weight Moving Average*); HAT-ADWIN (detector de mudança ADWIN). Eles são testados com dois *datasets* do repositório UCI (*Adult*[5] e *Poker-Hand*[6]). O HAT-INC e HAT-EWMA são os métodos que utilizam menos memória. Porém, o artigo conclui no final que HAT-ADWIN é o melhor método e que o HAT-EWMA é o método mais rápido com resultados próximos.

Desta forma, podemos concluir que os principais métodos de aprendizado encontrados na literatura sobre fluxo de dados são o *Naive Bayes* (NB), *Hoeffding Tree* (HT) e a sua evolução o *Hoeffding Adaptive Tree* (HAT) com ADWIN, que por esta razão serão utilizados nos experimentos desta dissertação junto com o método Prequencial, todos disponíveis no *framework* MOA.

2.4 *Frameworks*

Optamos neste trabalho de dissertação pela utilização de dois *frameworks*, WEKA e MOA, para detecção de intrusão por Aprendizado de Máquina *Off-line* e *On-line*, respectivamente, detalhados a seguir.

[5]Site: https://archive.ics.uci.edu/ml/datasets/Adult, outubro de 2021.
[6]Site: https://archive.ics.uci.edu/ml/datasets/Poker+Hand, outubro de 2021.

2.4.1 *Waikado Environment for Knowledge Analysis* (WEKA)

Uma ferramenta muito usual na área de Aprendizado de Máquina é o *Waikato Environment for Knowledge Analysis* (WEKA). Ele contém muitos algoritmos de classificação de dados, assim como outras técnicas como agrupamento, regras de associação, regressão, visualização e diversas métricas. Ela tem se apresentado como a principal ferramenta de desenvolvimento de classificadores de ataques em IDS na sua forma mais clássica de operação, sem levar em consideração os limites de recursos de memória e tempo de execução. Esta ferramenta de mineração é construída em código aberto em linguagem de programação Java desenvolvida pela Universidade de Waikato, Nova Zelândia.

2.4.2 *Massive Online Analysis* (MOA)

Os métodos incrementais de aprendizado de máquina em tempo real devem processar os exemplos rapidamente e no momento em que eles chegam, o que requer novas funcionalidades de estrutura de *software* quanto à velocidade de processamento e uso de memória ao longo do tempo, bem como considerações quanto ao uso dessas funcionalidades sobre os resultados do modelo. Há *softwares* disponíveis publicamente que realizam o aprendizado de máquinas de um fluxo de dados, que são capazes de prever as classificações, são eles: VFML[7] (*Very Fast Machine Learning*), MOA[8] (*Massive Online Analisys*), *Rapid-Miner*[9] e *Vowpal Wabbit*[10].

Esses programas têm como características principais (GAMA; SEBASTIAO; RODRIGUES, 2009): baixo tempo de processamento por dados; quantidade fixa de memória independente da quantidade total de dados; construção de modelos de classificação que fazem uma varredura apenas nos dados de treinamento; capacidade de classificação dos dados em qualquer momento independente da quantidade já processada; e capacidade de lidar com mudança de conceito.

Nessa dissertação, o foco será na ferramenta de desenvolvimento MOA (BIFET; HOLMES et al., 2010), que é um sistema de código aberto que contém uma coleção de algoritmos de aprendizado *On-line*, assim como ferramentas de avaliação, capaz de prover geração de dados, algoritmos e medidas de avaliação de fluxos de dados. Ele suporta interação com o WEKA, o que possibilita ao MOA realizar a classificação, regressão, detecção

[7]Site: https://www.cs.washington.edu/dm/vfml/, outubro de 2021.
[8]Site: https://github.com/Waikato/moa, outubro de 2021.
[9]Site: https://rapidminer.com/, outubro de 2021.
[10]Site: https://github.com/VowpalWabbit/vowpal_wabbit/wiki, outubro de 2021.

de mudança de conceito, clusterização, entre outros, todos relacionados a fluxo de dados.

2.5 Métricas

Neste trabalho, são utilizadas 6 métricas principais no WEKA para avaliação de desempenho para a realização da classificação, são elas: FPR (*False Positive Rate*), taxa de falso positivo; FNR (*False Negative Rate*), taxa de falso negativo; *Overall Accuracy* (A), acurácia geral; *Precision* (P), precisão; *Recall* (R), retorno; e F1*Score*, pontuação F1.

$$FPR = \frac{FP}{FP + TN} \tag{2.1}$$

$$FNR = \frac{FN}{FN + TP} \tag{2.2}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.5}$$

$$F1Score = 2 \times \frac{P \times R}{P + R} \tag{2.6}$$

As fórmulas de (2.1) a (2.6) apresentam como estas métricas são calculadas. O FPR (2.1) representa a taxa de não ataques identificados de forma equivocada (FP) entre o total de classificações normais (TN) do sistema. O FNR (2.2) representa a taxa de casos de ataques identificados como normais entre o total de ataques do sistema. O *Accuracy* (2.3) é a taxa total de decisões corretas para identificar um ataque corretamente ou identificar que não há ataque do total das identificações. A *Precision* (2.4) representa os casos positivos previstos que foram corretamente classificados. O *Recall* (2.5) representa os ataques identificados como verdadeiros em relação ao total de ataques. A pontuação F1 (2.6) representa uma métrica melhor da precisão balanceada pelo *recall*.

Para os métodos de aprendizado *On-line*, são utilizadas as mesmas métricas dos méto-

dos tradicionais do WEKA, descritas acima. Porém, testes executados, como no trabalho Bifet, Francisci Morales et al. (2015), mostram que a acurácia de um teste prequencial é apropriada somente quando todas as classes estão aproximadamente balanceadas. Porém, para esta dissertação preferimos dar destaque às métricas de Acurácia, Precisão, *Recall* e F1*Score*, já consolidadas e largamente utilizadas para fazer comparações entre o ML *Off-line* e ML *On-line*.

Descrevemos, assim, nessa Seção 2, os principais conceitos relacionados a IDS com ML *On-line* e ML *Off-line*, as suas principais abordagens, os *frameworks* para o experimento e as métricas mais empregadas.

2.6 *Datasets* para IDS

Podemos ver até aqui que os IDS e IDPS têm um importante papel no projeto e desenvolvimento de uma infraestrutura de rede robusta para defender recursos de ataques maliciosos através da detecção ou bloqueio. Desta forma, *datasets* confiáveis são importantes para testar e avaliar o desempenho destes sistemas de detecção.

Os *datasets* podem ser gerados de cinco formas principais (GHARIB et al., 2016): (F1) repetição de tráfego normal e de ataques reais; (F2) geração artificial de tráfego com geradores de *hardware* e *software*; (F3) simulação de uma instalação para experimento com um *software* de geração de tráfego; (F4) emulação com uso de máquinas reais de atacantes e usuários com uso de simulação em *software* da rede para experimento; (F5) representação física direta com uma topologia de rede desejada e com máquinas reais de atacantes e usuários. A Tabela 1 extraída do artigo Gharib et al. (2016) descreve as principais características de cada tipo de geração de *dataset*.

Tabela 1: Principais características na geração de *dataset*. Extraído de: (GHARIB et al., 2016).

	F1	F2	F3	F4	F5
Reconfigurabilidade	✓	✓	✓	x	x
Baixo Custo	✓	✓	✓	✓	x
Monitoramento	x	x	x	x	✓
Diversidade de Ataques	x	x	✓	✓	✓
Escalabilidade	x	✓	✓	x	x
Tráfego Interativo	x	x	✓	✓	✓

A seleção ou desenvolvimento de um *dataset* adequado é um desafio significativo, devido à privacidade, a falta de disponibilidade e uma forte necessidade de anonimato dos

dados. Outro importante fato é a necessidade de atualização constante desses *datasets* devido à evolução contínua das estratégias de ataque. Podemos destacar, como exemplo, no trabalho de Sharafaldin, Lashkari e Ghorbani (2018) a análise de 11 *datasets*, que estão disponíveis na comunidade acadêmica desde 1998, que se encontram desatualizados e com pouca diversidade e volume de tráfegos, são eles:

- **DARPA**, dados retirados de uma estrutura de rede desatualizados com antigos ataques;
- **KDD CUP 99**, atualização do dataset DARPA com muitas redundâncias e dados tendenciosos, porém ele é muito referenciado na literatura e será detalhado mais abaixo;
- **DEFCON**, criado para a competição CTF (*Capture the Flag*) com dados irreais devido ao grande número de ataques;
- **CAIDA**, com ataques e eventos muito específicos com alto anonimato na carga útil, protocolos, informação e destino;
- **LBNL**, sem carga útil e com dados omitidos por questões de anonimato;
- **CDX**, sofre de falta de diversidade de tráfego e volume;
- **KYOTO**, tráfego gerado apenas com dados de aplicações DNS e email, o que não reflete um fluxo de rede real;
- **TWENTE**, contém alguns alertas de tráfego desconhecidos e descorrelacionados, além de sofrer da falta de diversidade de tráfego e de volume;
- **UMASS**, não é útil para testes em técnicas de IDS e IPS devido à ausência de variedade de tráfego e de ataques;
- **ISCX2012**, não é baseado nas estatísticas atuais da internet por ter um baixo tráfego de protocolo HTTPS;
- **ADFA**, tem pouca diversidade e variedade de ataques.

Logo, para mitigar problemas como os descritos com os *datasets* acima, é importante a utilização de conceitos de avaliação para a elaboração de *datasets* que sirvam para comparação entre diferentes trabalhos na área, conforme os 11 critérios de construção apresentados por Gharib et al. (2016): Configuração de rede completa, Tráfego completo,

Datasets rotulados, Interação Completa, Captura completa, Protocolos disponíveis, Diversidade de ataque, Anonimato, Heterogeneidade, Conjunto de atributos e Metadados bem documentados.

Os métodos conhecidos e consolidados de detecção de anomalias em redes clássicas de Tecnologia da Informação (TI) não podem ser utilizados diretamente no tráfego de redes de instalações industriais por fazerem uso apenas de protocolos comunicação específicos. Tal abordagem não se mostra adequada para um IDS que opere com CPS, devido principalmente aos seguintes aspectos, conforme Schneider e Böttinger (2018): primeiro, a diferença na taxa de transferência de pacotes, que nos processos industriais são mais elevados porém com carga útil menor quando comparados com os sistemas clássicos de TI; segundo, para se ter êxito nos sistemas industriais não se pode negligenciar a carga útil dos pacotes, pois eles contém dados alterados que manipulam a operação dos PLC; outro desafio é o uso de protocolos de comunicação proprietários nos sistemas industriais, o que não permite a inspeção correta do fluxo de pacote de dados na rede.

Os *datasets* KDD Cup e CIC-IDS (descritos no Apêndice C), apesar de serem referenciados por alguns trabalhos como camadas de aplicação, transmissão e percepção, respectivamente (QUINCOZES; MOSSÉ et al., 2021), se enquadram como casos de sistemas clássicos de TI. Os *datasets* WDN-DS e o SUTD-IoT se referem apenas aos parâmetros da camada física de protocolo de comunicação sem fio e não a atividades de CPS, o que leva a necessidade de encontrar *dataset* mais específicos.

2.6.1 *Dataset* para *Cyber-Physical Systems*

Com objetivo de levantar *datasets* específicos de CPS, foi feita uma busca mais aprofundada sobre o tema. A Tabela 2 resume os principais *datasets* analisados, onde para os da cor vermelha não foi possível encontrar metadados que descrevessem como montar os dados de uma forma a serem prontamente utilizados nos *frameworks* empregados nos experimentos dessa dissertação. Os primeiros trabalhos encontrados são referentes aos *datasets* SWaT e BATADAL.

O *dataset* SWaT 2016 (*Secure Water Treatment*) (GOH et al., 2016) foi criado para futuros estudos específicos nesta área. O *dataset* SWaT foi elaborado de um projeto de CPS constituído de um sistema de tratamento de água moderno de seis estágios que produz 5 galões de água duplamente filtrada por minuto. O artigo de apresentação do *dataset* é bastante citado — com 193 referências, em setembro de 2021, conforme site Google Scholar

Tabela 2: *Datasets* para *Cyber-Physical Systems*.

Dataset	Classes	Tamanho	Features	Descrição
SWaT	Normal e Ataque	946719	51	Tratamento de água
Quadruple Tank	Multiclasse	N/A	N/A	Dados não disponíveis
Aqua-Tank	Multiclasse	100000	N/A	Descontinuado
BATADAL	Normal e Ataque	13938	43	Distribuição de água
Morris-1	41 classes	76035	128	Rede Elétrica
Morris-2	Binário	29310	26	Rede de Gasoduto
Morris-3 gas	7 classe	97019	26	Rede de Gasoduto
Morris-3 water	7 classe	236179	23	Tratamento de Água
Morris-4	Normal e Ataque	274628	16	Rede de Gasoduto
Morris-5	Multiclasse	1048575	8	Gerenciamento de Energia Elétrica
Lemay	N/A	N/A	N/A	Dois MTU e Três controladores
Rodofile	Multiclasse	N/A	N/A	Controle de uma refinaria de mineração
4SICS	N/A	N/A	N/A	Equipamentos PLC em laboratório
S4x15CTF	N/A	N/A	N/A	N/A
DEFCON23	N/A	N/A	N/A	N/A
ERENO	7 classes	7 Casos de Uso de 1114152	69	Rede Elétrica

—, e encontra-se também disponível no site *iTrust Lab* [11]. O sistema foi desenvolvido dentro do laboratório da universidade SUTD (*Singapore University of Technology and Desing*) e representa um sistema de tratamento de água de grandes cidades e plantas industriais.

O *dataset* foi constituído com ataques que ocorreram no enlace de comunicação entre SCADA e o PLC. A coleta dos dados ocorreu durante 24 horas por dia durante 11 dias, onde nos primeiros 7 dias não ocorreram ataques e nos 4 dias remanescentes houve a ocorrência de ataques. O protocolo utilizado foi o *Common Industrial Protocol* (CIP) sobre EtherNet/IP (EN/IP).

Os ataques foram executados em vários instantes e duraram entre alguns minutos e algumas horas. Os dados coletados são provenientes da camada de percepção, vindos da planta de tratamento de água através dos sensores e atuadores, e da camada de transmissão, vindos dos barramentos de comunicação entre SCADA e PLC. Os ataques na camada de percepção foram feitos alterando os valores dos sensores, com alteração da informação no nível dos tanques, e nos atuadores, com alteração no estado ligado e desligado das bombas. Os ataques na camada de transmissão foram feitos com o sequestro dos pacotes

[11] Site: https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/, setembro de 2021.

de comunicação falsificando os valores dos sensores e atuadores.

Na camada de percepção, há as propriedades físicas do sistema de tratamento de água em operação, onde no total foram coletadas 946.722 amostras contendo 53 atributos coletados dos sensores e atuadores nos 11 dias. Nesta parte do *dataset*, todos os ataques foram rotulados com a sua classificação. Desta forma, o *dataset* de características ciber físicas pode ser classificado por métodos de Aprendizado de Máquina supervisionado.

Na camada de transmissão, os *logs* fizeram registros em intervalos de 1 segundo com o limite de 500.000 pacotes, logo muitos pacotes foram transmitidos no mesmo instante de tempo representando diferentes atividades, o que torna a rotulação dos ataques inviável. Desta forma, o *dataset* do barramento de comunicações só pode ser classificado por um método de Aprendizado de Máquina não supervisionado.

Um outro *dataset*, conhecido pela competição acadêmica em inglês de *BATtle of the Attack Detection ALgorithms* (BATADAL), foi criado para testar algoritmos de detecção de ataques em CPS de sistemas de distribuição de água. O BATADAL é apresentado no artigo Taormina et al. (2018), O *dataset* foi constituído por uma rede de 429 tubulações, 388 junções, 7 tanques, 11 bombas hidráulicas, 5 válvulas e 1 reservatório. No total, há uma rede de 9 PLC localizados próximos de bombas hidráulicas, tanques de armazenamento e válvulas, onde são controlados os estados das válvulas (abertas ou fechadas), pressão de entrada e pressão de saída das bombas, assim como o fluxo de água. Os dados referentes à distribuição de água foram incorporados ao simulador hidráulico EPANET2 [12] e os ataques foram gerados a partir da ferramenta epanetCPA do MATLAB. Ele contém 43 atributos e encontra-se disponível também para o meio acadêmico pelo site da iTrust Lab.

O BATADAL foi validado no trabalho de Taormina et al. (2018) com a distribuição aos participantes de três *datasets*, dois para treinamento e um para detecção de ataques. As métricas serviram para avaliar o tempo de detecção dos ataques e a acurácia dos algoritmos utilizados pelas equipes participantes da competição. O trabalho documentou principalmente o tempo de detecção dos ataques dos algoritmos de detecção e pouco foi descrito quanto à acurácia e ocorrência de FNR e FPR. Conforme já descrito, o *dataset* BATADAL surgiu com o propósito de fazer um competição acadêmica com sete participantes de diferentes universidades organizado para testar algoritmos de detecção de ataques em sistemas de distribuição de água. Desta forma, este *dataset* é o primeiro de uma competição que trata de segurança em sistemas ciber físicos.

[12] Site: https://www.epa.gov/water-research/epanet, setembro de 2021.

Dentre os *datasets* sobre CPS pesquisados, o BATADAL e o SWaT foram os de mais fácil acesso, porém o estudo sobre o SWaT foi aprofundado por ser o mais referenciado, o que levou a leitura de mais artigos referentes à utilização deste *dataset* e a contribuição no artigo Quincozes, Mossé et al. (2021).

Continuando nos trabalhos descritos da Tabela 2, o trabalho de Hink et al. (2014) apresenta o *dataset* Morris-1, que é constituído de um conjunto de controles supervisórios em interação com vários IED, junto com dispositivos de monitoramento de rede como o sistema SNORT [13] e Syslog [14]. A rede é constituída por quatro disjuntores controlados por relés inteligentes, que são ligados a uma subestação de *switch*, através de um roteador até o controle supervisório e o sistema de aquisição da rede. Os cenários de ataque foram construídos a partir do suposto que um atacante já tenha acesso a rede de subestação e tenha a capacidade de injetar comandos a partir da subestação *switch*.

Os *datasets* Morris-3 foram descritos por Morris e Gao (2014). Eles foram capturados com a utilização de registros de dados de rede que monitoram e armazenam tráfego MODBUS de uma conexão RS-232. Foram utilizados dois sistemas SCADA em escala laboratorial de uma rede de gasoduto (Morris-3 gas) e de um tanque de armazenamento de água (Morris-3 water). O programa *Wireshark* foi utilizado para capturar e armazenar o tráfego da rede, durante as operações normais e sob ataque sem afetar as operações de controle do sistema e os ataques simulados. Para a simulação dos ataques foi utilizado o equipamento MU-400 Analyser, um testador de rede capaz de executar ataques DoS, congestionamento de rede, teste de mutilação de protocolos e testes contra vulnerabilidades comuns de rede.

O *dataset* Morris-4 foi descrito por Morris, Thornton e Turnipseed (2015) e se refere também a uma rede de gasoduto. Este *dataset* tem a mesma origem do Morris-3, porém ele apresenta melhorias como 35 ciber ataques rotulados, simulados em uma rede de gás virtual desenvolvida em Python, que inclui Interface de Homem-Máquina, do inglês, *Human Machine Interface* (HMI), um processo físico virtual, um PLC virtual e uma rede virtual. O ambiente virtual foi escolhido por possibilitar outros experimentos sem a necessidade de ter acesso a dispositivos físicos e possibilidade de expansão. A natureza aleatória dos ataques garante uma representação mais próxima da realidade a este *dataset*.

O *dataset* ERENO pode ser verificado no artigo Silvio Quincozes (2022). Ele foi desenvolvido como uma estrutura de geração de tráfego sintético baseado no padrão IEC-

[13] Site: `https://www.snort.org/`, janeiro de 2022.
[14] Site:`https://pt.wikipedia.org/wiki/Syslog`, janeiro de 2022.

61850 para SG. O seu desenvolvimento veio da ausência de dados específicos sobre o padrão de estudos na área de segurança relacionados com IDS. Logo, o objetivo é fornecer um referência para as pesquisas na área de detecção de intrusão. Os dados são originados da ferramenta PSCAD (*Power Systems Computer-Aided Design*) [15] para reprodução real de linha de transmissão elétrica. Desta forma, são gerados dados reais dos protocolos GOOSE e SV. No final, o *dataset* é montado com 7 Casos de Uso (UC) de cenários com 7 ataques comuns.

Foi descrito até aqui, ao longo do Capítulo 2, os conceitos básicos de IDS, assim como os principais classificadores utilizados, as métricas de classificação e os principais desafios da área. Descrevemos também os *datasets* mais importantes (SWaT, BATADAL, ERENO, Morris-1, Morris-3 gas, Morris-3 water e Morris-4) de CPS, Tabela 2, que serão usados nos experimentos desta dissertação mais a frente.

[15] Site: https://www.pscad.com/software/pscad/overview, janeiro de 2022

3 Trabalhos Relacionados

Nessa capítulo, serão descritos os principais trabalhos relacionados com o tema de pesquisa desta dissertação a partir da Revisão Sistemática da Literatura (RSL), descrita no Apêndice D. Desta forma, os tópicos foram definidos em áreas de modo a possibilitar a identificação das principais tendências. Na subseção seguinte serão detalhados os principais trabalhos relacionados a partir da RSL descrita.

Tabela 3: Análise detalhada dos artigos selecionados da Revisão Sistemática da Literatura.

Referência	**Ano**	**Dataset**	**Experimento**
Securing advanced metering infrastructure using intrusion detection system with data stream mining (FAISAL et al., 2012)	2012	KDD Cup 99	Comparar técnicas ML *On-line* para SG
Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study (FAISAL et al., 2014)	2014	KDD Cup 99 e NSL-KDD	Comparar técnicas ML *On-line* para SG
Efficient intrusion detection system using stream data mining classification technique (DESALE; KUMATHEKAR; CHAVAN, 2015)	2015	NSL-KDD	Testar técnicas ML *On-line* para IDS
Event Stream Processing for Improved Situational Awareness in the Smart Grid (DAHAL et al., 2015)	2015	Real Time Digital Simulator (RTDS®)	Descobrir a melhor técnica ML *On-line* ou ML *Off-line* para SG
An investigation of the hoeffding adaptive tree for the problem of network intrusion detection (CORRÊA; ENEMBRECK; SILLA, 2017)	2017	Kyoto2006	Descobrir a melhor técnica ML *On-line* para Rede de Computadores
Applying hoeffding adaptive trees for real-time cyber-power event and intrusion classification (ADHIKARI; MORRIS, T. H.; PAN, 2017)	2017	SG simulador HIL	Verificar a viabilidade das técnicas ML *On-line* para SG
Practical application of machine learning based online intrusion detection to internet of things networks (NIXON; SEDKY; HASSAN, 2019)	2019	KDD Cup 99 e UNSW-NB15	Comparar técnicas ML *On-line* para IoT
Comprehensive analysis for class imbalance data with concept drift using ensemble based classification (PRIYA; UTHRA, 2021)	2021	SEA, KDD e ANSWEM	Compara técnicas ML *On-line* para Rede de Computadores
Adversarial Attack by Inducing Drift in Streaming Data (SERAPHIM; POOVAMMAL, 2021)	2021	RBF, UNSW-NB15, NSL-KDD e ISCX	Comparar técnicas ML *On-line* para Redes de Computadores
Drift-based approach for evolving data stream classification in Intrusion detection system (SETHA; SINGHA; CHAHALA, 2021)	2021	CICIDS 2018	Comparar técnicas ML *On-line* para Redes de Computadores
Detecção de Intrusão em Sistemas Ciber-Físicos com Uso de Técnicas de Aprendizado de Máquina (DISSERTAÇÃO)	2022	SWaT, ERENO, BATADAL, Morris-1, Morris-3 gas, Morris-3 water e Morris-4	Comparar técnicas de ML *On-line* e ML *Off-line* em CPS

3.1 Trabalhos Relacionados com Aprendizado de Máquina *On-line*

A partir da Tabela 3 gerada da RSL, podemos observar a ausência de trabalhos que falam sobre ML *On-line* com uso do *framework* MOA e com *datasets* reais de CPS, o que possibilitaria uma comparação entre técnicas de *On-line* e Off-line. Um trabalho que busca demonstrar o benefício do ML *On-line* é o Desale, Kumathekar e Chavan (2015), onde são apresentados mecanismos para melhorar a eficiência de um IDS. São aplicados quatro algoritmos de classificação para fluxo de dados com o *datasets* NSL-KDD: *Naive Bayes* (NB), algoritmo baseado em probabilidade; *Hoeffding Tree* (HT), algoritmo baseado em árvore de decisão; *Accuracy Updated Ensembles* (AUE) e *Accuracy Weighted Ensemble* (AWE), algoritmos baseados em conjunto que utilizam o *Minimum Square Error* (MSE) como entrada de atualização dos classificadores selecionados. Todos os algoritmos classificadores foram testados na ferramenta MOA.

Os resultados mostram que o classificador AWE possui melhores métricas de acurácia do que os outros classificadores, porém o classificador HT é o que apresenta menor tempo de processamento com métricas de classificação um pouco piores. Infelizmente, o artigo não compara as técnicas ML *On-line* e ML *Off-line*. Outro problema que podemos apontar é a falta de testes com outros *datasets* mais atuais e específicos de CPS.

O artigo Corrêa, Enembreck e Silla (2017) apresenta um estudo de ML *On-line* em fluxo de dados com uso de diferentes métodos para predição em sistemas de detecção de intrusão. Os autores adotam o algoritmo principal do classificador HAT como modelo de predição, por ser um dos mais usuais no meio acadêmico. O *dataset* utilizado no *framework* MOA é o Kyoto 2006 (SONG et al., 2011), onde é comparado o HAT com os classificadores kNN, *Naive Bayes* e *Perceptron*. Os resultados mostram que tanto o HAT e kNN apresentam os melhores resultados, porém o tempo de processamento do kNN é 30 vezes maior que o HAT e o NB é 3 vezes menor que o HAT. Na conclusão o HAT é apontado como o melhor algoritmo em termos acurácia (95%), com um tempo de processamento ainda aceitável, o que possibilita a sua implementação em dispositivos de arquitetura de *hardware* simples e baratos, como Arduino e Raspberry PI.

O artigo Nixon, Sedky e Hassan (2019) busca mostrar técnicas de ML para detecção de intrusão mais adequadas em uma arquitetura com limitação de recursos como a camada de percepção IoT. A principal contribuição foi a execução prática de técnicas ML *On-line* com os métodos semi-supervisionado (*Informed*) e não supervisionados (*Unsupervised*)

com o uso do MOA e dos *datasets* KDD Cup 99 e UNSW-NB15. Os classificadores para os métodos semi-supervisionados *Naive Bayes* (NB) e *Hoeffding Tree* (HT) foram utilizados com os seguintes detectores de mudança de conceito: *Adaptive Window* (ADWIN), *Drift Detection Method* (DDM) e *Hoeffding Drift Detection Method* (HDDM). No final do artigo, é possível verificar os seguintes benefícios do ML *On-line*: Gerenciamento de Memória, o tamanho de memória utilizada pelos algoritmos é sempre o mesmo, por causa do método de esquecimento; Aprendizado eficiente, as técnicas semi-supervisionadas são mais adequadas do que as supervisionadas, pois a rotulação dos dados traz custo de processamento, logo as supervisionadas não são viáveis para uma rede IoT; Detecção de Anomalia não Supervisionada, o que permite a detecção de ataques desconhecidos mantendo os benefícios anteriormente descritos.

No trabalho de Priya e Uthra (2021), são feitos experimentos comparativos entre diferentes técnicas de mudança de conceito e de classificação conjunta (*Ensemble Classification*) com uso de *dataset* sintético, SEA (do inglês *Streaming Ensemble Algorithm*), e *datasets* reais, o KDD Cup e ELEC2 [1] (*Australian New South Wales Electricity Market* - ANSWEM). A ferramenta MOA é utilizada com o método de avaliação prequencial. Como a classificação baseada em conjunto é uma função das predições de diferentes classificadores, ela possui uma maior capacidade de lidar com a mudança de conceito e melhorar a acurácia quando comparada com um único classificador. No final, os resultados mostram que a classificação conjunta apresenta um desempenho superior em relação a um classificador isolado, como o NB.

O trabalho de Seraphim e Poovammal (2021) segue o mesmo objetivo. Ele faz uso de técnicas de aprendizado de máquina supervisionado do *framework* MOA, com fluxo de dados, com e sem mudança de conceito e com os classificadores Naive Bayes, *Hoeffding Tree*, *Leveraging Bag* (LB) e *Oza Bag Adwin* (OBA). São usados nos experimentos os algoritmos geradores de dados sintéticos, o SEA, *Random RBF* e *Hyperplane* para inclusão de mudança de conceito, e os *datasets* reais, o UNSW-NB15, KDD Cup, NSL-KDD e ISCX, com extração de métricas de classificação. Os experimentos com os *datasets* reais foram feitos com o método de avaliação *Holdout*, divididos em razões de treinamento e teste de 80:20, 70:30 e 60:40. O algoritmo HT tem um bom desempenho quando comparado com os outros algoritmos devido ao menor consumo de tempo e memória. O *dataset* ISCX[2] se mostrou tendencioso, pois contém grande número de dados normais e uma pequena quan-

[1] Site: https://www.aemc.gov.au/energy-system/electricity/electricity-system/NEM, novembro de 2021.

[2] https://www.unb.ca/cic/datasets/index.html

tidade de dados anormais, o que resulta em resultados tendenciosos. Para os experimentos com mudança de conceito com os dados sintéticos, o algoritmo LB apresenta o melhor desempenho com acurácia de 94,55%, porém, devido ao método utilizar classificadores em conjunto, há um maior tempo de processamento e consumo de memória.

O artigo Setha, Singha e Chahala (2021) faz uso do método de ML *On-line* para IDS com a utilização do classificador *Adaptive Random Forest* (ARF), o detector de mudança ADWIN e a utilização do dataset CIC-IDS 2018. Os autores propõem desta forma um IDS de grande desempenho capaz de detectar mudança de conceito com um modelo de classificação incremental e atualizável, o que permite uma rápida adaptação. No final são comparados os métodos ARF, kNN e *Naive Bayes*, onde o ARF se apresenta melhor com 99,5% de acurácia, 99,9% de precisão e 99,8% de *recall*. Infelizmente, as comparações foram apenas entre estes três classificadores de ML *On-line* sem nenhuma comparação com ML *Off-line*.

Desta forma, podemos observar que os artigos até agora descritos nesta seção fazem uso de *datasets* reais, porém não relacionados com CPS. Na seção seguinte serão detalhados os artigos que fazem uso de ML *On-line* e que utilizam *datasets* de CPS.

3.1.1 Sistemas de Detecção de Intrusão com Aprendizado de Máquina *On-line* em *Chyber-Physical System*

Um IDS tradicional, que utilize regras criadas manualmente por especialistas, não é ideal no contexto de um grande fluxo de dados, como é o caso de uma rede SG distribuída. Muitas dessas tecnologias ainda estão em desenvolvimento, logo novos tipos de ataques podem surgir, o que faz com que as soluções de hoje sejam ainda custosas e não muito efetivas. Logo, conforme os artigos descritos até aqui na Tabela 3, a adoção de um IDS com ML *On-line* é a melhor solução para aprender sobre o fluxo de dados vindos de diversas fontes. Desta forma, este tipo de classificador pode ser empregado para aprender sobre o comportamento normal e a assinatura dos ataques, de forma a criar de maneira automática regras de detecção.

Por este motivo, os autores do artigo Faisal et al. (2012) escolheram um método de IDS baseado em anomalias com uso de ML *On-line*, por se assemelhar mais à realidade das redes SG. Isso se deve à existência de várias fontes de dados distribuídos e ao alto fluxo de dados, com distribuição estatística não estacionária, além da limitação de recursos que os equipamentos na malha SG apresentam, como baixa capacidade de processamento e limitada capacidade de memória.

Com objetivo de encontrar os algoritmos de maior acurácia e também os melhores parâmetros para o teste "EvaluatePrequencial", foram selecionados no artigo basicamente 4 classificadores de ML *On-line*, descritos a seguir: *Leveraging Bagging* (LeveragingBag), *Limited Attribute Classifier* (LimAttClassifier), *Bagging using ADWIN* (OzagBagAdwin) e *Single Classifier Drift* (SingeClassifierDrift). Foram coletadas as métricas de classificação e resultados de tempo de processamento e consumo de memória. O *dataset* utilizado para os testes foi o KDD *Cup* 99 por ser uma referência na área.

Como conclusão do artigo, é apontado que as técnicas ML *On-line* em fluxo de dados são mais adequadas na prática para serem utilizadas na detecção de intrusão em redes SG, pois a velocidade e o tempo de resposta são primordiais. O algoritmo *LimitAttClassifier* é o que mostrou melhor desempenho entre os quatro classificadores, porém os seus requisitos de tempo e memória tornam o seu emprego inviável em equipamentos com restrições de recursos conforme os autores. O algoritmo *SingleClassifierDrift* é descrito como o mais promissor devido à sua baixa demanda de recursos, logo é um tema a ser analisado em estudos futuros para melhorar as suas métricas de classificação.

Desta forma, o artigo Faisal et al. (2012) se preocupou em descrever uma arquitetura de IDS para um sistema integrado de medidores inteligentes conhecidos como uma Infraestrutura de Medição Avançada, proveniente do termo em inglês *Advanced Metering Infrastructure* (AMI), para atender aos critérios de segurança provendo suporte a protocolos legados, escalabilidade, suporte a *hardware* legado, cumprimento de padrões, adaptabilidade, determinismo e confiabilidade. Apesar de os autores apontarem também como estudo futuro a localização dos sensores de IDS na rede SG como relevante para o desempenho do sistemas, no trabalho mais atual Faisal et al. (2014), o objetivo foi descrever em detalhes os experimentos, com diversos classificadores, a fim de encontrar aqueles que possuem o melhor desempenho com *dataset* KDD *Cup* 99 e o mais atual do NSL-KDD. Neste artigo é descrito também um IDS baseado em anomalias para SG que utiliza métodos de ML *On-line* com fluxo de dados. Os autores fazem a identificação e avaliação de diferentes algoritmos com uso de ML *On-line*. Desta forma, o objetivo do estudo é avaliar os algoritmos dinâmicos que possuem a capacidade de evoluírem e detectarem mudanças de conceito. São selecionados 7 classificadores dinâmicos, três a mais que o estudo anterior, que apresentam as maiores acurácias avaliadas pelo método "*EvaluatePrequential*", detalhados a seguir: *Accuracy Updated Ensemble* (AUE), *Active Classifier (ActiveClassifier)*, *Leveraging Bagging (LeveragingBag)*, *Limited Attribute Classifier (LimAttClassifier)*, *Bagging using ADWIN (OzagBagAdwin)*, *Bagging using Adaptive Size Hoeffding Tree (OzagBagASHT)* e *Single Classifier Drift (SingleClassifierDrift)*.

Nos testes de acurácia com o KDD 99, o melhor algoritmo foi o *LeveragingBag* seguido pelo *LimAttClassifier*, porém eles não apresentaram o menor consumo de recursos. O *ActiveClassifier* é que apresenta a menor exigência quanto aos recursos, porém ele apresenta resultados baixos das métricas de classificação. O *SingleClassifierDrift* é o que apresenta bons resultados mantendo o baixo consumo de recursos de tempo e memória, porém apresenta número elevado de FNR. Para os testes de acurácia com NSL-KDD, o melhor algoritmo é o *LimAttClassifier*, porém com os maiores custos de memória e tempo. O mesmo desempenho foi verificado do *ActiveClassifier* que apresenta a menor exigência quanto aos recursos, porém ele apresenta resultados pobres. O *SingleClassifierDrift* e o *LeveragingBag* apresentam bons resultados mantendo o baixo consumo de recursos de tempo e memória, porém apresentam número elevado de FNR.

Os classificadores mantiveram as mesmas ordens entre os dois *datasets* utilizados, porém uma observação importante é que todos os classificadores consumiram mais memória com o NSL-KDD atualizado, da ordem de 10 vezes mais, do que o KDD CUP 99 completo. Os autores argumentam que esse efeito é devido ao *dataset* NSL-KDD apresentar maior ocorrência de mudanças de conceito. No final, são apontados o ActiveClassifier e o SingleClassifierDrift como os classificadores mais promissores, por possuírem limitadas exigências de tempo e memória, apesar do primeiro apresentar resultados com métricas pobres de classificação e o segundo apresentar resultados com métricas moderadas de classificação. Um ponto favorável ao ActiveClassifier é que ele não faz uso de todos os rótulos em todas as amostras, o que se aproxima ao ambiente real de uma SG quando exposto a novos ataques.

Desta forma, nos dois trabalhos Faisal et al. (2012) e Faisal et al. (2014) os testes foram executados apenas sobre o *dataset* KDD, que não é o mais adequado para um rede SG. Logo, as conclusões apontadas pelos autores precisam ser verificadas em outros *datasets*, o que pode ser visto no trabalho de Dahal et al. (2015). Nele, também são utilizados os algoritmos de ML *On-line* para fluxo de dados, com o objetivo de prover a predição de um grande fluxo de dados em SG. O *dataset* foi retirado de um simulador *Hardware-In-the-Loop* (HIL) para manter os experimentos próximos ao real. Infelizmente, os autores não disponibilizaram publicamente o *dataset*. Algumas métricas foram escolhidas pelos autores para medir o desempenho dos algoritmos, como a acurácia, o tempo de processamento e o uso de memória. No total, são realizados quatro experimentos.

No experimento I, foi feito o teste *interleaved-test-then-train* com os algoritmos HT adaptativo e não-adaptativo. A métrica do algoritmo não adaptativo apresenta um de-

sempenho ruim, que vai piorando à medida que o fluxo prossegue. Já com o algoritmo adaptativo ocorreu o contrário: a métrica do algoritmo melhora à medida que o fluxo prossegue, apesar deste algoritmo apresentar maior consumo de recursos computacionais.

No experimento II, foram adotados os mesmos algoritmos de predição com fluxos de dados que não apresentaram mudança de conceito. Desta forma, o HT não-adaptativo apresentou um desempenho satisfatório quando comparado com o experimento I, porém o algoritmo HAT ainda apresenta o melhor resultado apesar do maior consumo de recursos computacionais.

No experimento III, apenas o algoritmo HT não adaptativo foi usado com o tamanho fixo da árvore de decisão em 25 Kbytes, 50 Kbytes e 75 Kbytes. O desempenho foi o mesmo até o número de amostras chegar a 140 mil para as árvores fixadas em 25 e 50 kbytes, o que comprova o argumento que os algoritmos de ML *On-line* com fluxo de dados podem se adaptar a níveis baixos de memória sem perder muita precisão.

No último experimento IV, é feita a comparação de desempenho do algoritmo HT do ML *On-line* com os algoritmos tradicionais do ML *Off-line*, o J48 e o REPTree. O HT é o que apresenta o menor consumo de recursos, mesmo considerando o tempo de aprendizado e teste juntos. Com relação à precisão, o HT apresentou um desempenho um pouco inferior aos outros dois algoritmos, da ordem de 15%, apesar de a acurácia apresentar uma tendência de alta à medida que a quantidade de amostras aumenta. O grande limitador deste trabalho foi utilizar apenas um *dataset*, que não foi publicamente disponibilizado, o que não permite recriar os experimentos apresentados.

O artigo Adhikari, Thomas H Morris e Pan (2017) utiliza a combinação de classificadores *Hoeffding Adaptive Trees* (HAT) adicionados com DDM (*Drift Detection Method*) e ADWIN (*Adaptive Windowing*) para classificar ataques e falhas em tempo real em *Smart Grids*. O *dataset* utilizado foi extraído do sistema IEEE *9-bus 3-generator* simulados em um HIL com dados de fluxo normal, falhas no sistema elétrico, manutenção do sistemas, flutuação de carga e quatro cenários de ciberataques, todo o processo foi detalhado no artigo anterior dos autores (ADHIKARI; MORRIS, T.; PAN, 2016). Os dados brutos de sensores elétricos foram fundidos em um único fluxo de dados, quantizados e comprimidos antes de serem enviados ao MOA para classificação. Todas as 426.009 amostras foram armazenadas com as informações de tempo e os rótulos de classificação de forma tabular em um arquivo CSV.

Os testes foram executados com o método prequencial com o ML *On-line*, onde os resultados foram plotados em gráficos com as métricas de acurácia e mudança de conceito.

Pode ser observado que a acurácia média foi de 98% para a classificação binária e 94% para a classificação multiclasse. Os autores ainda quantificaram o tempo médio de avaliação das amostras: foram 0,028 milissegundos para a classificação binária e 0,089 milissegundos para a classificação multiclasse.

Quando comparado com os métodos tradicionais de classificação, vemos que os resultados apontam que o ML *On-line* de fluxo de dados é aplicável para a classificação de cenários complexos e amplos sem sacrificar a acurácia e recursos computacionais. Quando comparado com vários outros métodos, *Random Forest*, JRip, Adaboost+JRip (HINK et al., 2014), *Common Path Mining*(PAN; MORRIS; ADHIKARI, 2015) e NNGE+STEM (ADHIKARI, 2015), o método proposto apresenta desempenho superior nas métricas de acurácia e estatística Kappa em um cenário de fluxo de dados alto e em constante mudança de conceito. Podemos concluir que os algoritmos HAT+DDM+ADWIN podem ser usados para IDS que operem com fluxo de dados em tempo real por demandarem baixo recursos de memória. Infelizmente, o *dataset* não está disponível abertamente na internet, o que dificulta a reprodução dos testes.

Foi descrito até aqui, ao longo do Capítulo 3, as principais pesquisas na área, a fim de definir a principal contribuição quanto à comparação entre métodos de classificação de ML *On-line* e ML *Off-line* com *datasets* reais de CPS. Desta forma, este trabalho será uma continuidade do Dahal et al. (2015), porém o trabalho será ampliado com os diversos *datasets* de CPS disponíveis na literatura e descritos no Capítulo 2. Faremos também uso de outros classificadores, a fim de identificar quais são os melhores e piores quanto às métricas de classificação para cada *dataset* em uso.

4 Experimentos

Neste capítulo serão feitos experimentos com classificação de detecção de intrusão com *datasets* de CPS. O objetivo é verificar uma forma eficiente e eficaz de classificação com grande capacidade de processamento de um grande fluxo de dados. Para isso, serão utilizados o *framework* WEKA, que é um dos mais conhecidos na literatura especializado em métodos de classificação de ML *Off-line* em conjunto de dados limitados, e o *framework* MOA, especializado em técnicas de classificação de ML *On-line* em grande fluxo de dados, que pode ser utilizado em dispositivos de recursos limitados.

Os *datasets* empregados foram SWaT, BATADAL, ERENO, Morris-1, Morris-3 (*gas* e *Water*) e Morris-4, conforme descrito na Seção 2.6.1, com as principais características resumidas na Tabela 4. Todos os experimentos com os *datasets* foram executados com a classificação binária, ou seja, só se verificou a ocorrência e detecção de ataques, e não o tipo de ataque específico. Todos os *datasets* foram tratados para serem executados nos devidos *frameworks*, WEKA e MOA. Particularmente, o *dataset* ERENO foi aglutinado em um único arquivo contendo todos os seus 7 casos de uso.

Como citado na Introdução desta dissertação, busca-se, através destes experimentos, responder as seguintes questões de pesquisa:

1. É possível identificar nos *datasets* estudados alguma relação entre as métricas dos classificadores ao longo do tempo? Resposta no final da Subseção 4.3.1.

2. Qual diferença é observada nas métricas de classificação quando são utilizados os classificadores adaptativos do ML *On-line*? Resposta no final da Subseção 4.3.1.

3. Qual é a diferença nas métricas de Acurácia, Precisão ou *Recall* entre o ML *Off-line* e o ML *On-line*? Resposta no final da Subseção 4.3.2.

4. Qual é a técnica de IDS mais adequada para dispositivos de *hardware* limitado? Resposta no final da Subseção 4.3.4.

5. Qual é a diferença entre os ML *On-line* e ML *Off-line* na detecção de novos ataques em sequência? Resposta no final da Subseção 4.3.5.

4.1 Experimentos com o *Framework* WEKA

Os experimentos com o *Framework* WEKA foram executados nos computadores do Laboratório MidiaCom em cinco máquinas, cada uma com a configuração de oito núcleos de 2,80 GHz de CPU e 32 GB de RAM. Os classificadores foram executados com o método *Holdout*, com divisão mais conhecida na literatura entre treino e teste de 50/50, mantendo o balanceamento original das classes do *dataset*. Os resultados são mostrados no Apêndice A para cada *dataset* com resultados da Matriz Confusão com o classificador de melhor acurácia em azul.

Conforme já mencionado, os quatro classificadores em árvore (J48, *REPTree*, *Random Tree* e *Random Forest*) foram selecionados por seus modelos serem de fácil visualização gráfica por parte dos usuários. O classificador *Naive Bayes* foi selecionado por ser um dos mais básico para servir de parâmetro de comparação. O objetivo destes experimentos é obter dados de consumo de memória, tempo de processamento (aprendizado e teste), acurácia, F1*Score*, precisão e *recall* entre os *datasets*.

4.2 Experimentos com o *Framework* MOA

Os experimentos com o *Framework* MOA não exigem *hardware* de grande desempenho e foram executados em um computador pessoal, com a configuração de quatro núcleos de 2,10GHz de CPU e 8GB de RAM. Os classificadores foram executados com o método *prequencial*. Observa-se, portanto, que não foi possível dividir o tempo gasto no treinamento e no teste, como foi feito com o *framework* WEKA, devido às características de como o método é executado no método prequencial. Este método foi executado na forma padrão com janela de tamanho de 1000 amostras e fator de desvanecimento de 0,01. A frequência de amostragem foi igual a unidade de forma a capturar os valores de VP, VN, FP e FN de cada instância. Os resultados são registrados no Apêndice A por cada *dataset*, com a Matriz Confusão usando o classificador de melhor acurácia (HAT) em vermelho.

A proposta dos experimentos no *framework* MOA, conforme já mencionado, contém os classificadores HT, HAT e NB executados com o teste prequencial sobre os mesmos *datasets*. O objetivo dos experimentos é obter dados de acurácia, *recall* e F1*Score*, assim como

Tabela 4: Principais características dos *datasets* com os melhores e piores classificadores do ML *Off-line*.

Dataset	**Tamanho**	**Features**	**Normal**	**Ataque**	**Melhor em Acurácia do ML *Off-line***	**Pior em Acurácia do ML *Off-line***
SWAT	946719	51	94,23%	5,77%	Random Forest	Naive Bayes
BATADAL	12938	43	98,31%	1,69%	REPTree	Random Tree
Ereno	5750000	69	93,40%	6,60%	REPTree	Naive Bayes
Morris-1	78400	128	22,15%	77,85%	Random Forest	Naive Bayes
Morris-3 gas	97020	26	63,04%	36,96%	J48	Naive Bayes
Morris-3 water	236179	23	73,00%	27,00%	REPTree	Naive Bayes
Morris-4	274725	35	78,13%	21,87%	Random Tree	Naive Bayes

as quantidades de Verdadeiro Positivo (VP), Verdadeiro Negativo (VN), Falso Positivo (FP) e Falso Negativo (FN), que serão posteriormente comparados com os experimentos do *framework* WEKA, descritos anteriormente.

4.3 Análise dos Resultados

Primeiramente, serão apresentados os resultados dos *datasets* com o melhor e o pior em acurácia classificador do ML *Off-line* e os classificadores NB, HT e HAT do ML *On-line* com as TFN e TFP do HAT ao longo do tempo. Em seguida, serão comparadas as métricas de classificação Acurácia, Precisão, *Recall* e F1*Score* do melhor classificador do ML *Off-line* e MOA em cada *dataset*. Na sequência, será feita a analise entre a acurácia e o tempo de processamento por instância com o melhor e o pior classificador em acurácia do ML *Off-line* e os classificadores NB, HT e HAT do ML *On-line* em relação ao tamanho do *dataset*. Serão apresentados também a relação entre a acurácia e o consumo de recursos de memória e processamento. Finalmente, apresentaremos uma comparação ao longo do tempo entre o ML *Off-line* e o ML *On-line* na ocorrência de novos ataques.

4.3.1 Comparação entre os ML *On-line* e ML *Off-line* pela Acurácia e Taxas de Falso Negativo e Falso Positivo ao longo do Tempo

Os resultados seguintes são referentes à evolução em gráficos da acurácia ao longo do tempo no ML *On-line* com os classificadores NB, HT e HAT. São inseridos para comparação o classificador de melhor e pior acurácia do ML *Off-line*, entre os cinco classificadores utilizados. São apresentados, nas Figuras de 6 a 12, no eixo vertical secundário os valores da Taxa de Falso Negativo (TFN) e Taxa de Falso Positivo (TFP) extraídos do classificador HAT do tamanho total dos *datasets* divididos em 100 partes, a fim de identificar os

instantes de erro.

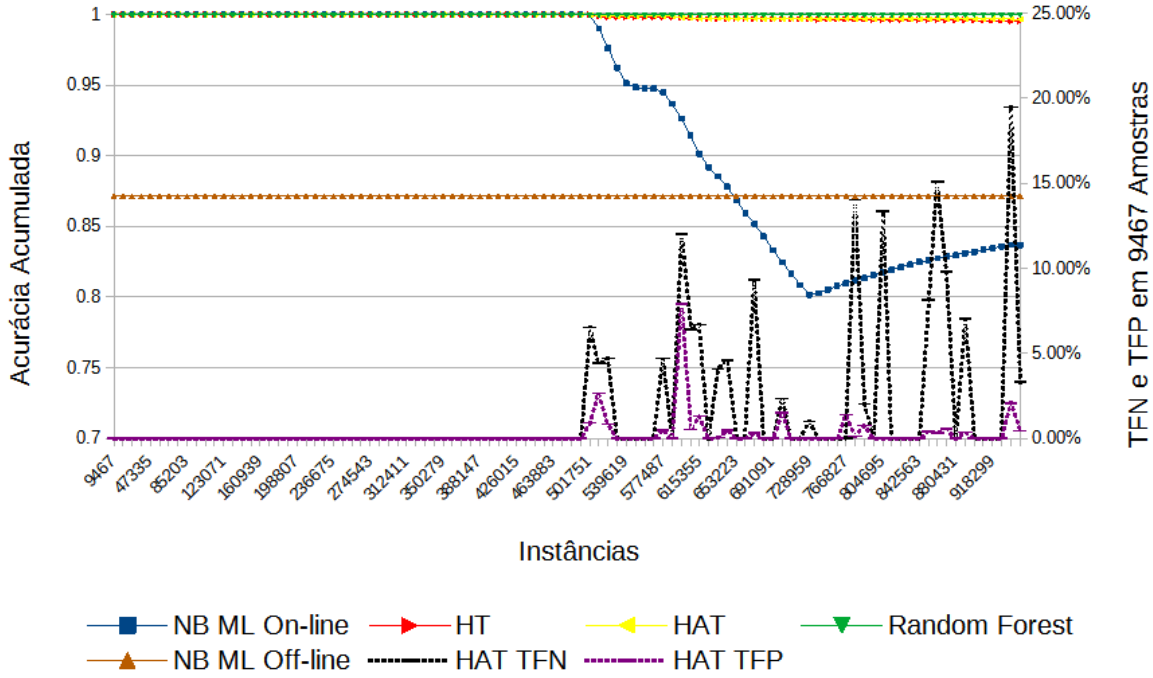

Figura 6: Acurácia dos ML *Off-line* e ML *On-line* (eixo y principal), Taxa de Falso Negativo e Taxa de Falso Positivo do classificador *Hoeffding Adaptive Tree* do ML *On-line* em 9467 amostras (eixo y secundário) do *dataset* SWaT.

Na Figura 6, temos o desempenho dos classificadores no *dataset* SWaT, onde podemos ver que os ataques não identificados começam no meio do *dataset*, conforme os pontos em preto (TFN) no gráfico. O melhor classificador em acurácia do ML *Off-line* foi o *Random Forest* com 99,97% e o prior classificador em acurácia do ML *Off-line* foi o *Naive Bayes* com 87,15%. Os classificadores HT e HAT do ML *On-line* apresentaram desempenhos muito próximos do *Random Forest* com 99,52% e 99,65% de acurácia, respectivamente, com uma diferença muito pequena de 0,45% e 0,32%, em comparação com o classificador RF do ML *Off-line*.

Na Figura 7, temos o desempenho dos classificadores no *dataset* BATADAL, onde podemos ver que os ataques ocorrem nos instantes finais do *dataset*, conforme os resultados de TFP aparecem no gráfico. O melhor classificador em acurácia do ML *Off-line* foi o *Random Forest* com 99,87% e o prior classificador em acurácia do ML *Off-line* foi o *Random Tree* com 98,27%. O classificadores NB, HT e HAT do ML *On-line* apresentaram desempenhos parecidos com o classificador NB do ML *Off-line* apresentando uma recuperação mais elevada no final com acurácia de 98,82% e os outros dois classificadores com 98,44% e 98,43%, respectivamente. A diferença entre o melhor em acurácia do ML *Off-line* (RF) e o melhor do ML *On-line* em acurácia (NB) é um valor pouco perceptível

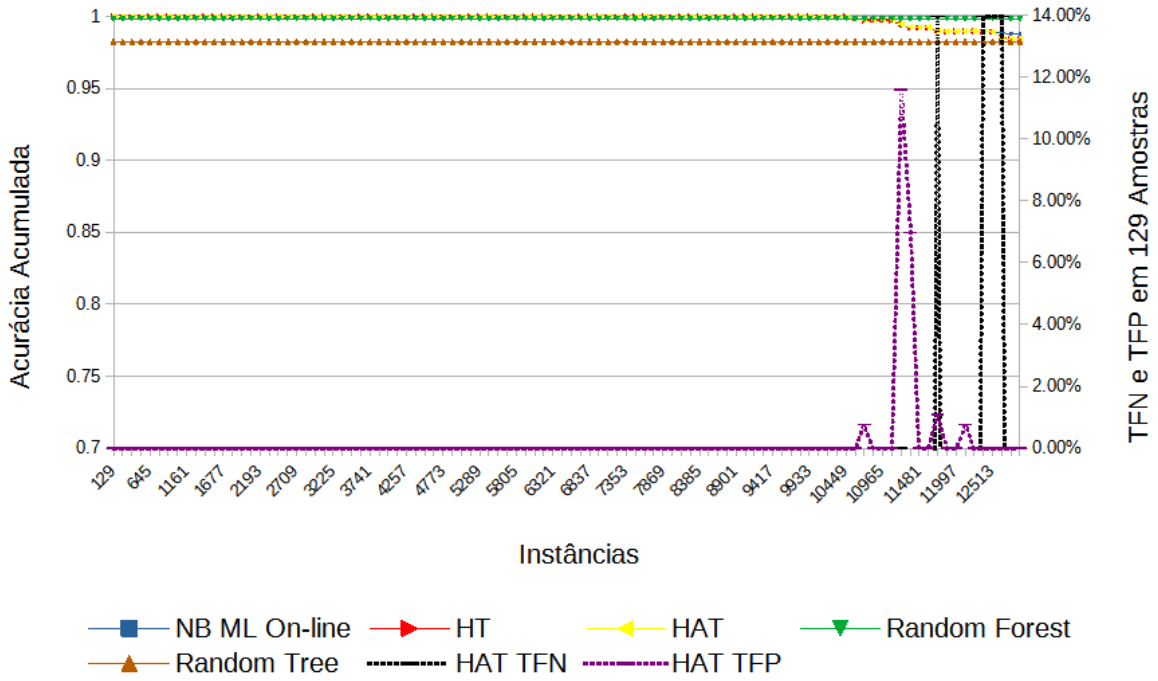

Figura 7: Acurácia do ML *Off-line* e ML *On-line* (eixo y principal), Taxa de Falso Negativo e Taxa de Falso Positivo do classificador *Hoeffding Adaptive Tree* do ML *On-line* em 129 amostras (eixo y secundário) do *dataset* BATADAL.

de 1,05%.

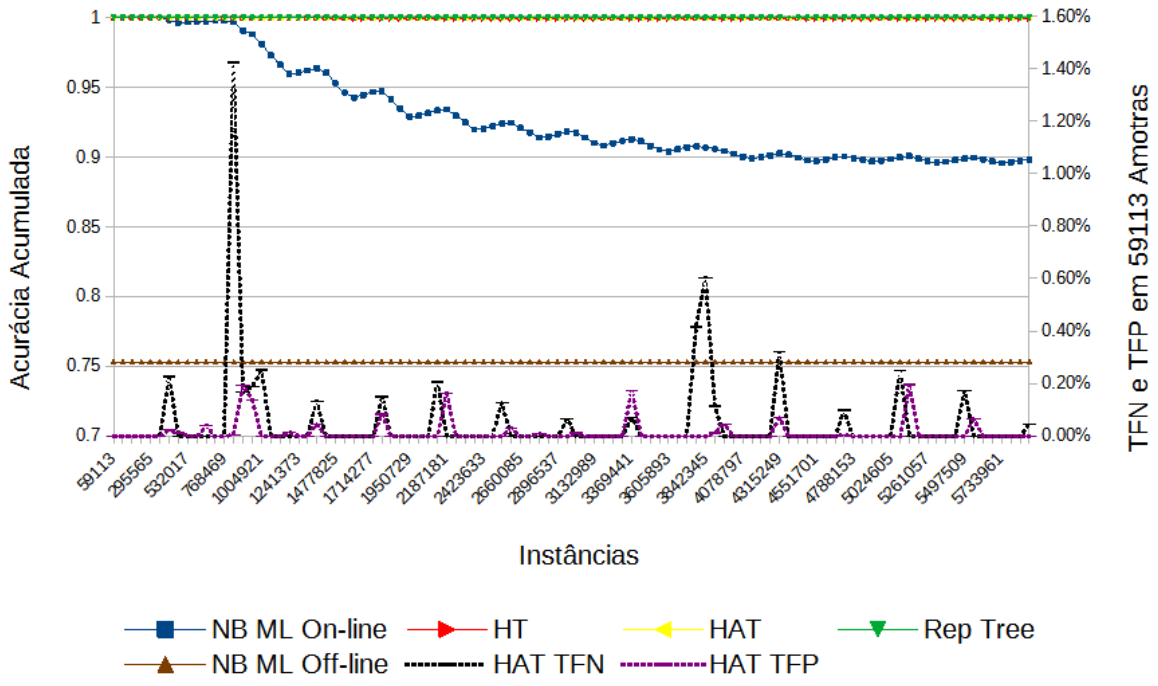

Figura 8: Acurácia do ML *On-line* e ML *Off-line* (eixo y principal), Taxa de Falso Negativo e Taxa de Falso Positivo do classificador *Hoeffding Adaptive Tree* do ML *On-line* em 59113 amostras (eixo y secundário) do *dataset* ERENO.

Na Figura 8, temos o desempenho dos classificadores no *dataset* ERENO. O melhor

classificador em acurácia do ML *Off-line* foi o *REPTree* com 99,99% e o pior classificador em acurácia do ML *Off-line* foi o *Naive Bayes* com 75,31%. O NB do ML *On-line* apresenta uma queda suave para 90% de acurácia com o passar das amostras, ou seja, ao passar pelos casos de uso de 1 a 7. Os classificadores HT e HAT apresentam um desempenho semelhante ao classificador *REPTree* do ML *Off-line*, com acurácia de 99,91% e 99,98%, respectivamente.

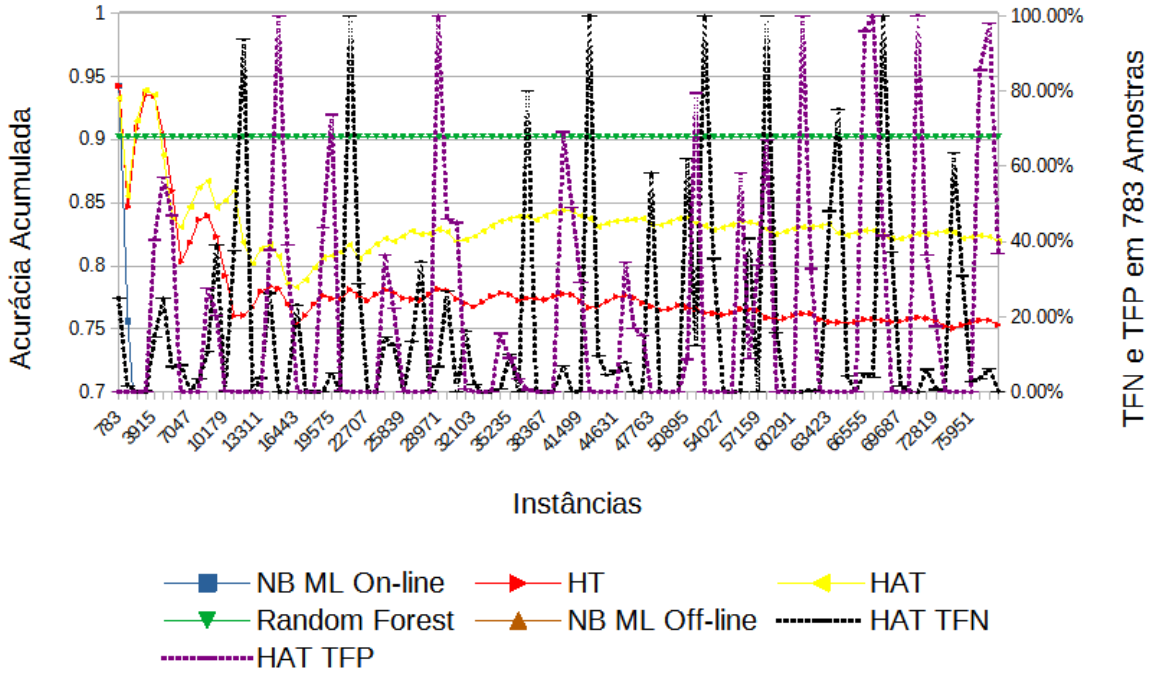

Figura 9: Acurácia do ML *Off-line* e ML *On-line* (eixo y principal), Taxa de Falso Negativo e Taxa de Falso Positivo do classificador *Hoeffding Adaptive Tree* do ML *On-line* em 783 amostras (eixo y secundário) do *dataset* Morris-1.

Na Figura 9 temos o desempenho dos classificadores no *dataset* Morris-1, onde podemos observar a grande ocorrência de ataques ao longo de todas as amostras. Cabe lembrar que este *dataset* possui 77,85% das classes em forma de ataque. Por esta razão, o classificador NB do ML *On-line* apresenta um baixo desempenho de 31,06% no final. Os classificadores HT e HAT apresentam uma maior capacidade de acompanhar a ocorrência de ataques com a acurácia de 75,34% e 81,98% respectivamente. O melhor classificador em acurácia do ML *Off-line* foi o *Random Forest* com 90,23% e o prior classificador em acurácia do ML *Off-line* foi o *Naive Bayes* com 31,42%. Podemos observar que, no *dataset* Morris-1, o classificador *Random Forest* do ML *Off-line* tem um desempenho superior aos do ML *On-line* devido à forte ocorrência de ataques ao longo do tempo.

Na Figura 10, temos o desempenho dos classificadores do *dataset* Morris-3 gas. O classificador NB do ML *On-line* apresenta um desempenho baixo ao longo do tempo

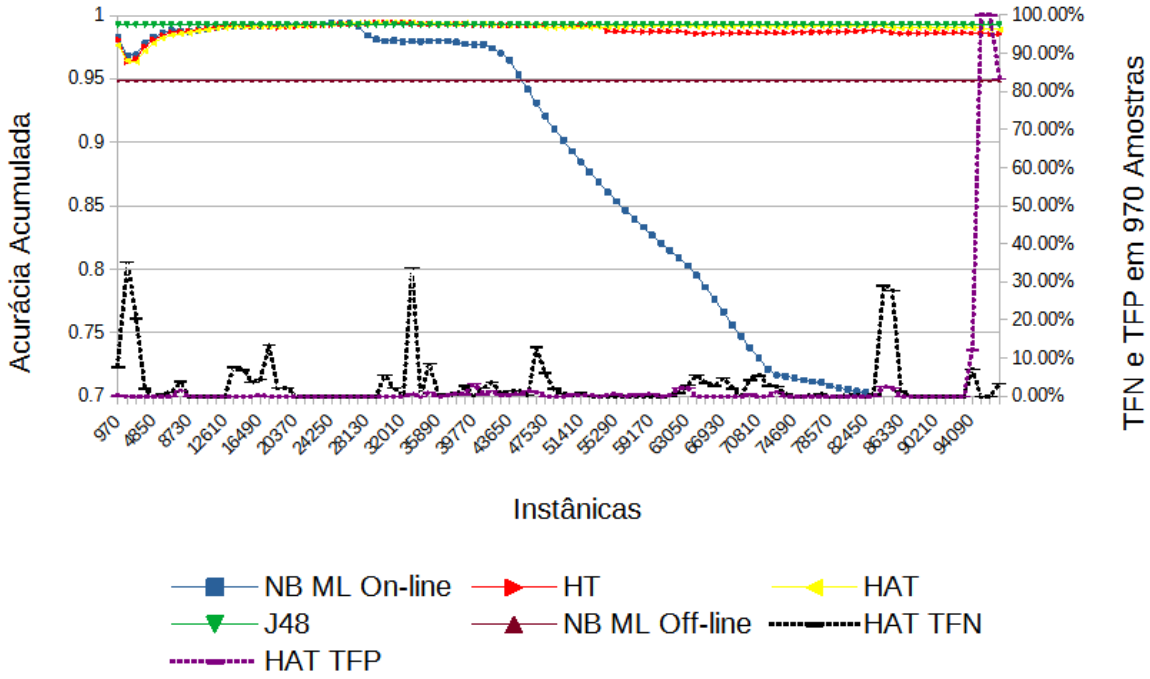

Figura 10: Acurácia do ML *Off-line* e ML *On-line* (eixo y principal), Taxa de Falso Negativo e Taxa de Falso Positivo do classificador *Hoeffding Adaptive Tree* do ML *On-line* em 970 amostras (eixo y secundário) do *dataset* Morris-3 gas.

terminando com a acurácia em 67,94%. O melhor classificador em acurácia do ML *Off-line* foi o J48 com 99,26% e o pior classificador em acurácia do ML *Off-line* foi o *Naive Bayes* com 94,86%. Logo, o desempenho do classificador NB do ML *On-line* foi muito inferior ao NB do ML *Off-line*. Os classificadores HT e HAT apresentaram resultados praticamente iguais com acurácia de 98,51% e 98,91% respectivamente, uma diferença de 0,35% a favor do ML *Off-line*.

A Figura 11 apresenta o desempenho dos classificadores do *dataset* Morris-3 water. O classificador NB do ML *On-line* apresenta um desempenho em queda a partir da ocorrência de FP e finaliza a sua acurácia com 36,83%. O melhor classificador em acurácia do ML *Off-line* foi o *REPTree* com 99,13% e o prior classificador em acurácia do ML *Off-line* foi o *Naive Bayes* com 90,78%. Logo, o desempenho do classificador NB do ML *On-line* foi muito inferior ao NB do ML *Off-line*. Os classificadores HT e HAT apresentaram resultados de acurácia de 96,03% e 98,25%, respectivamente, o que representa uma pequena diferença de 0,88% a favor do ML *Off-line*.

Na Figura 12, temos o desempenho dos classificadores do *dataset* Morris-4. Para nossa surpresa, ambos os classificadores NB e HAT do ML *On-line* apresentam desempenhos baixos (83,42% e 83,47% respectivamente), porém acima do pior classificador em acurácia do ML *Off-line*, o classificador *Naive Bayes* (82,18%). O classificador HT é o que apresenta

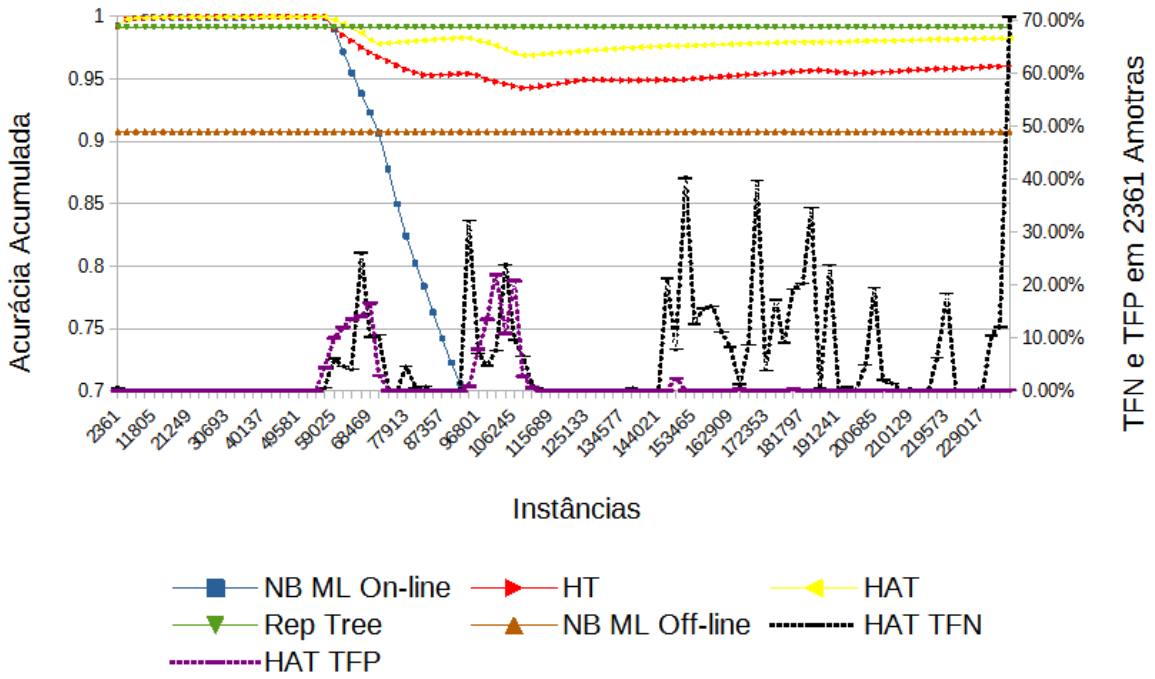

Figura 11: Acurácia do ML *Off-line* e ML *On-line* (eixo y principal), Taxa de Falso Negativo e Taxa de Falso Positivo do classificador *Hoeffding Adaptive Tree* do ML *On-line* em 2361 amostras (eixo y secundário) do *dataset* Morris-3 water.

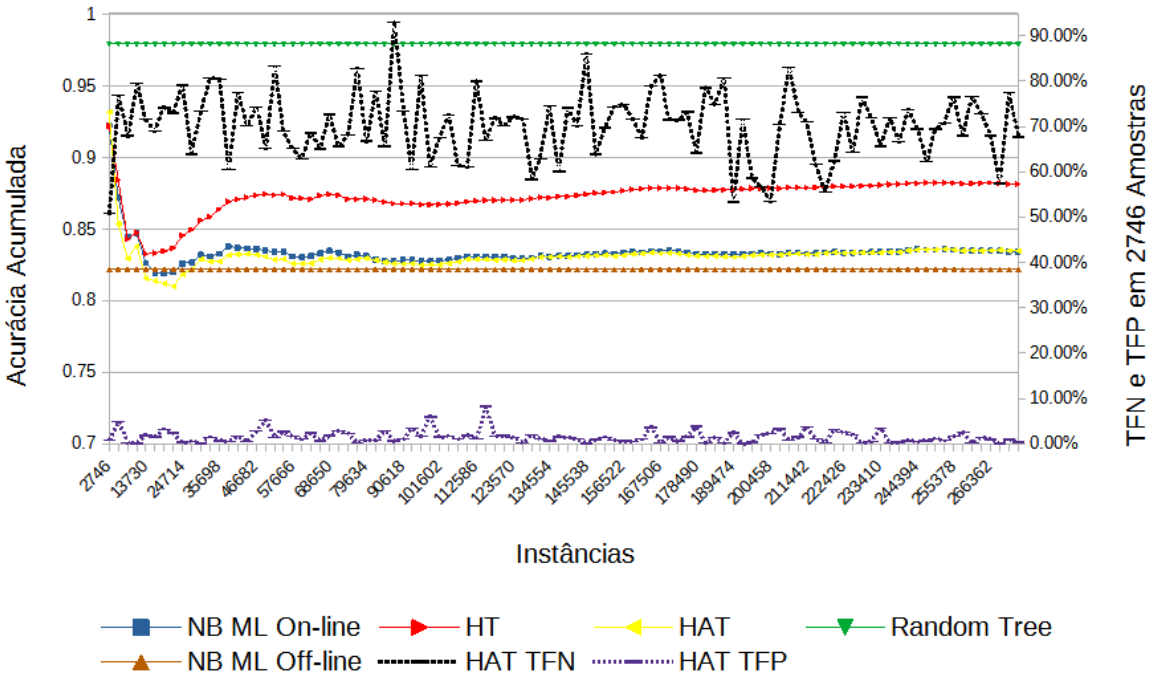

Figura 12: Acurácia do ML *Off-line* e ML *On-line* (eixo y principal), Taxa de Falso Negativo e Taxa de Falso Positivo do classificador *Hoeffding Adaptive Tree* do ML *On-line* em 2746 amostras (eixo y secundário) do *dataset* Morris-4.

o melhor resultado do ML *On-line*, com acurácia final de 88,14%, um valor muito abaixo do melhor classificador em acurácia do ML *Off-line*, o classificador *Random Tree* 97,94%,

uma diferença de 9,8% para o ML *Off-line*.

Os melhores classificadores do ML *Off-line* apresentaram resultados da acurácia entre os *datasets* em média de 97,95%, com destaque para o melhor resultado, no *dataset* ERENO com acurácia do classificador *REPTree* de 99,98%, e o pior, o *dataset* Morris-1 com acurácia no classificador *Random Forest* de 90,23%. O destaque no ML *Off-line* foi com os classificadores *Random Forest*, presente em três *datasets* (SWaT, Morris-1 e Morris-4), e o classificador *REPTree*, presente em outros três (BATADAL, ERENO e Morris-3 Water). No ML *On-line*, os melhores classificadores apresentaram em média uma acurácia de 95,35%, com destaque para o melhor em acurácia, o classificador HAT no *dataset* ERENO com 99,98%, e o pior em acurácia, o classificador HAT do *dataset* Morris-1 com 83,26%. O classificador que teve mais destaque no ML *On-line* foi o HAT, pois apresentou o melhor resultado em cinco *datasets*: SWaT, ERENO, Morris-1, Morris-3 gas e Morris-3 water. Desta forma, os *datasets* com os melhores resultados em acurácia do ML *Off-line* e ML *On-line*, do melhor para o pior, foram: ERENO, SWaT, Morris-3 gas, BATADAL, Morris-3 water, Morris-4 e Morris-1. Esta ordem é dada tanto pela observação da acurácia, quanto pelos resultados finais de TFN e TFP do classificador HAT do ML *On-line*.

Em resposta à pergunta 1: É possível identificar nos *datasets* estudados alguma relação entre as métricas dos classificadores ao longo do tempo? Com as Figuras de 6 a 12 foi possível observar a forte dependência dos classificadores do ML *On-line* quanto à ocorrência de ataques ao longo da sequência temporal. Geralmente, quando o ataque surge, há a ocorrência de FN e o classificador leva um tempo para começar a detectar o ataque. Logo depois há ocorrência de FP e novamente o classificador leva um tempo para detectar a ausência do ataque.

Em resposta à pergunta 2: Qual diferença é observada nas métricas de classificação quando são utilizados os classificadores adaptativos do ML *On-line*? O grande diferencial em usar um classificador adaptativo, o classificador HAT com ADWIN, é a sua capacidade em detectar os ataques em menor tempo e com menos ocorrência de erros FP e FN, quando comparado com os classificadores *Naive Bayes* e *Hoeffding Tree*. Desta forma, os classificadores adaptativos terão resultados mais próximos dos sistemas físicos reais que operam em tempo real (DAHAL et al., 2015).

4.3.2 Comparação entre ML *Off-line* e ML *On-line* pela Acurácia, Precisão, *Recall* e F1*Score*

O ML *Off-line* apresenta um desempenho superior ao ML *On-line*, o que é justificado pelo fato de ele fazer o seu aprendizado com o *dataset* completo e sem restrição de recursos de memória e processamento. Apesar do ML *Off-line* apresentar uma acurácia superior em cerca de 3,2% (média retirada das Figuras de 6 a 12), pouco foi dito quanto à qualidade desta acurácia, pois para termos mais detalhes é importante ter outras métricas como F1*Score*, Precisão e *Recall*, além das Matrizes Confusão. Estas métricas são importantes pois, conforme a Tabela 4, se os classificadores fizerem uso da classe de maior ocorrência (classe majoritária), a acurácia poderá apresentar valores altos, porém a capacidade do modelo de fazer a predição correta em casos de ataque pode ser reduzida.

Nas Figuras 13 e 14 o melhor classificador em acurácia do ML *Off-line* apresenta a acurácia superior ao classificador HAT do ML *On-line* e que o mesmo ocorre com os valores de F1*Score*, *Precision* e *Recall*, o que representa uma capacidade superior do ML *Off-line* em detectar ataques, mesmo que o *dataset* seja desbalanceado e com pouca presença de ataques. Os *datasets* BATADAL, SWAT e ERENO apresentam as menores quantidades de ataques, com os seguintes valores, respectivamente: 1,69%, 5,77% e 6,60%. O forte desbalanceamento nos dados pode ser um indício da baixa capacidade dos classificadores em detectar os ataques quando comparados com um classificador de maior ocorrência, o que justifica a alta acurácia com baixas métricas, principalmente no ML *On-line*.

Já os *datasets* Morris-3 gas e Morris-3 water apresentam dados mais balanceados, com ataques distribuídos, respectivamente, em: 36,96% e 27,00%. Esta característica mostra um forte indício para o bom desempenho do classificador HAT do ML *On-line* com métricas próximas ao melhor classificador em acurácia do ML *Off-line*. Os *datasets* Morris-1 e Morris-4, apesar de serem também balanceados, não apresentam boas métricas. O *dataset* Morris-4 possui 21,87% de ataques nas amostras, porém as suas métricas de F1*Score* e *Recall* do classificador HAT do ML *On-line* são muito baixas quando comparado com o melhor classificador em acurácia do ML *Off-line*, o que mostra que este *dataset* é um dos mais difíceis para o ML *On-line*. Já o *dataset* Morris-1 possui 77,85% de ataques nas amostras, e os seus resultados do classificador HAT do ML *On-line* são piores que o melhor classificador em acurácia do ML *Off-line* também, com baixos valores de F1*Score* e *Recall*.

O classificador HAT do ML *On-line* apresenta valores médios de F1*Score* 30,62% inferior ao melhor classificador em acurácia do ML *Off-line*. Isso se deve à relação direta

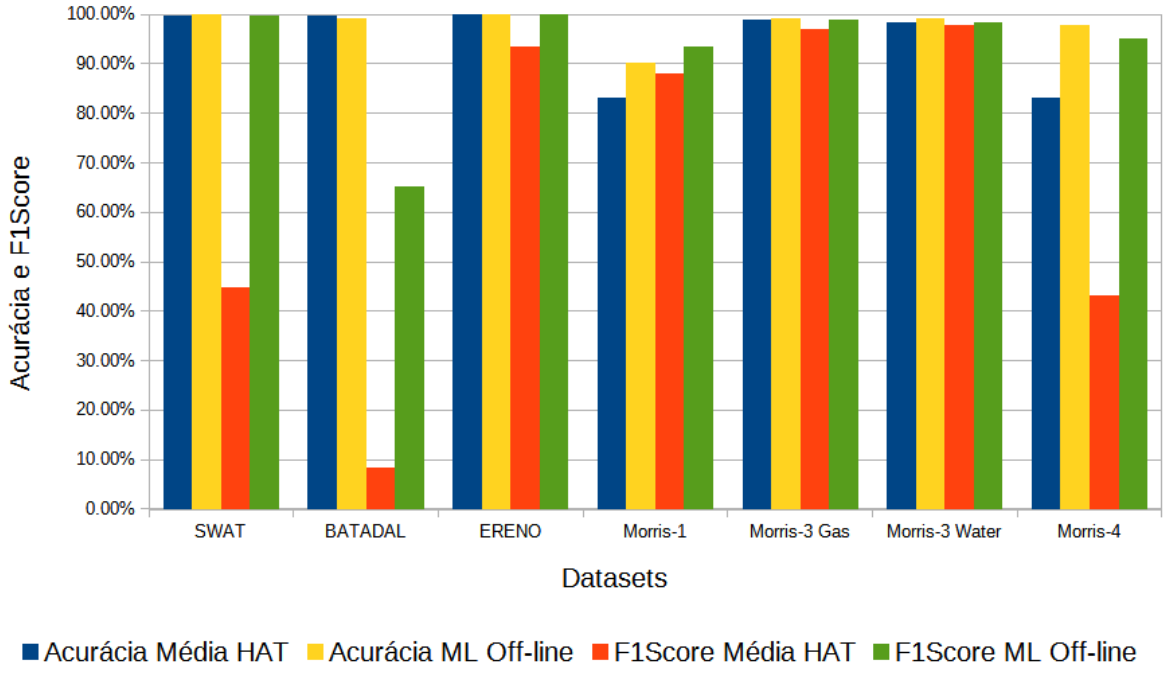

Figura 13: Acurácia e F1*Score* dos *datasets* selecionados.

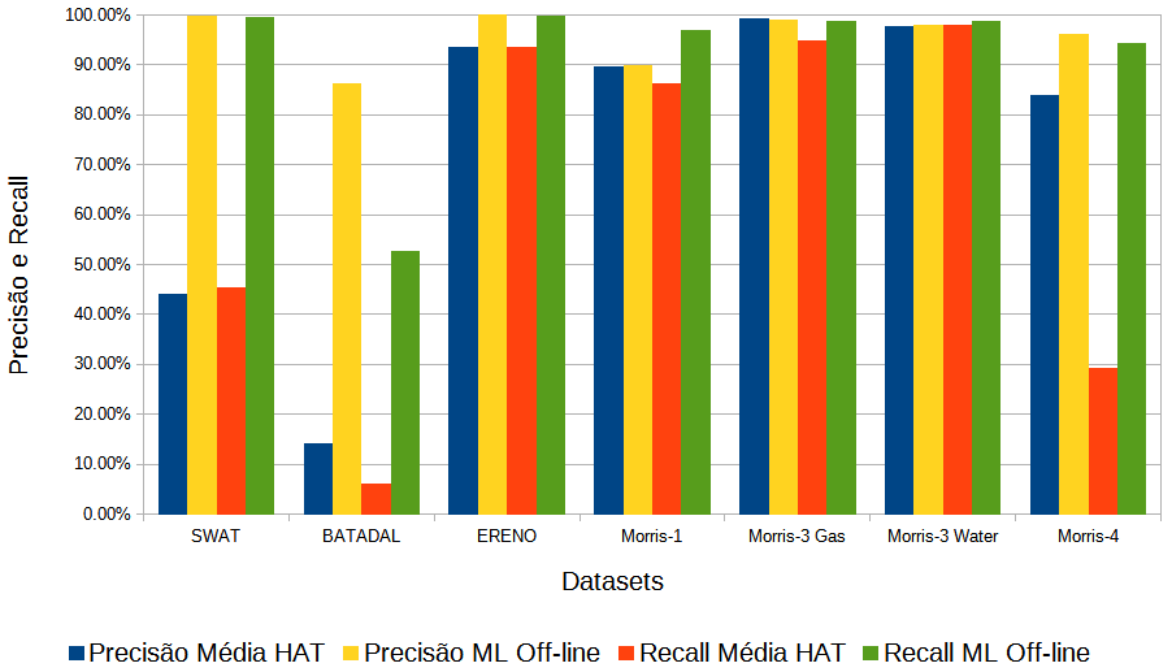

Figura 14: Precisão e *Recall* dos *datasets* selecionados.

que as fórmulas de Precisão e *Recall*, Fórmulas 2.4 e 2.5, têm quanto às variáveis FP e FN, respectivamente. Quanto maior o FP e FN, menor será a Precisão e o *Recall*. Logo, o classificador HAT do ML *On-line* apresentou mais erros na detecção dos ataques. Estas observações podem ser analisadas também no Apêndice A a partir das Matrizes Confusão. Podemos logo observar que o ML *On-line* sofre uma forte influência quanto ao

balanceamento dos *datasets* e que o ML *Off-line* tem um desempenho superior.

Em resposta à pergunta 3: Qual é a diferença nas métricas de Acurácia, Precisão ou Recall entre o ML Off-line e o ML On-line? Conforme as Figuras 13 e 14, os valores da Acurácia, F1*Score*, Precisão e *Recall* são melhores para o classificador HAT do ML *Off-line*, o que mostra a sua grande capacidade em identificar os ataques. O mesmo pode ser visto também nas Matrizes Confusão no Apêndice A: os valores de FN e FP são menores no ML *Off-line*.

Desta forma, podemos observar que o ML *Off-line* tem uma vantagem na acurácia quando comparado com o ML *On-line*, porém a avaliação das métricas não é completa, pois não leva em consideração os custo de tempo e memória. No consumo de recursos, iremos verificar a principal diferença entre ML *Off-line* e ML *On-line*, o que será visto na próxima subseção.

4.3.3 Comparação Entre ML *Off-line* e ML *On-line* pela Acurácia e Tempo de Processamento por Instância

Para a execução dos experimentos com tempo, foram utilizados os computadores do laboratório MidiaCom, já descritos na Seção 4.1, para uma comparação adequada entre o ML *On-line* e ML *Off-line*. Durante este experimento, a coleta das métricas não foi feita para não prejudicar no tempo total de execução.

Na Figura 15, temos o tempo de execução no ML *Off-line* e MOA dos sete *datasets* com os valores de acurácia nas barras laterais. Porém, nos treinamentos do ML *Off-line* elas não foram coletadas. Podemos observar que o classificador NB do ML *On-line* apresentou um valor médio de 38,5 microsegundos por instância, enquanto o classificador HT teve 44,2 microsegundos por instância e o classificador HAT gastou 58,3 microsegundos por instância. Desta forma, o classificador NB é o que apresenta o menor tempo de processamento por instância com uma acurácia média entre os *datasets* de apenas de 78,73%, enquanto os classificadores HT e HAT tiveram em média 93,29% e 94,75%, respectivamente. Podemos concluir que o tempo na média entre os *datasets* é de 14,74% para o classificador HT e 51,60% para o classificador HAT a mais quando comparados com o classificador NB, o que justifica o uso principalmente do classificador HT devido ao pequeno aumento de tempo e ao seu bom desempenho na acurácia.

O melhor do ML *Off-line* teve um tempo médio entre os *datasets* de treinamento de 530 microsegundos por instância e de teste de 40,1 microsegundos por instância, o que

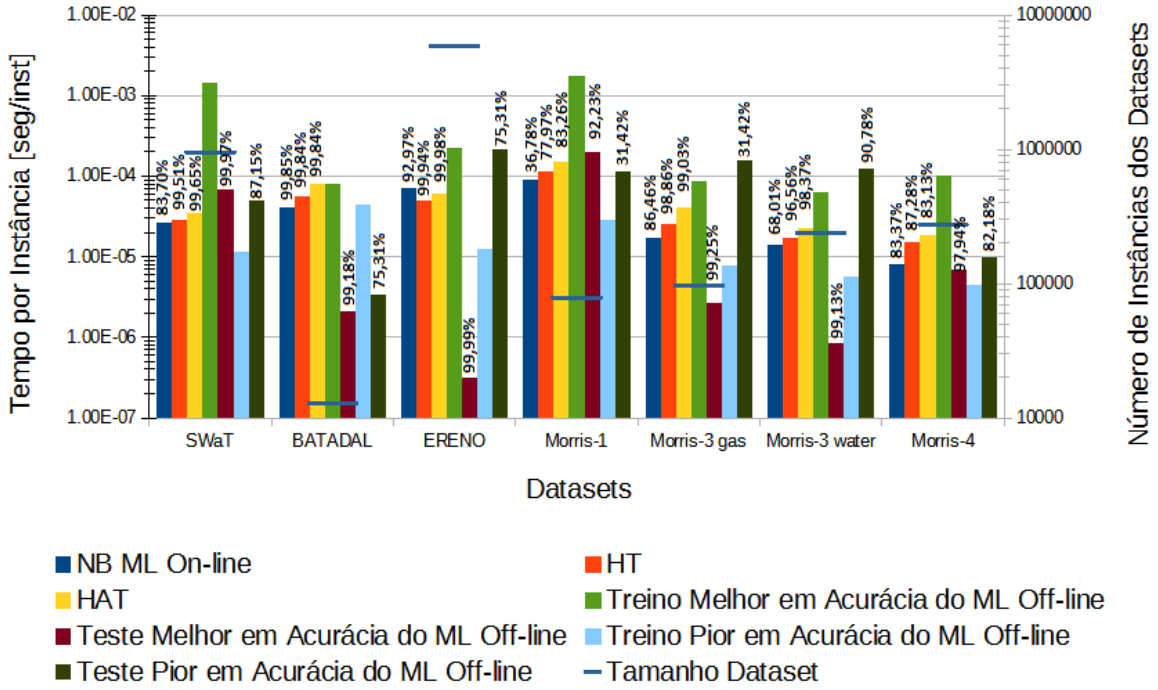

Figura 15: Tempo do ML *Off-line* e ML *On-line* em Segundos por Instância no eixo y principal por *datasets*. Tamanho dos *datasets* no eixo y secundário.

mostra um tempo 13 vezes maior do treinamento em relação ao teste. Podemos concluir, a principio, que a fase de treinamento do ML *Off-line* não deverá ocorrer em dispositivos de *hardware* simples, como os sensores e atuadores de um CPS. Essa diferença de tempo ocorre pois durante o treinamento o classificador tem que montar o modelo. Para o caso dos classificadores *Random Tree* (RaT), REPTree (ReT), J48 e *Random Forest* (RF) deverão ser montadas árvores de decisão. Já para o classificador *Naive Bayes* (NB), monta-se o modelo de decisão de condições de probabilidade. Desta forma, o melhor do ML *Off-line* tem uma acurácia média entre os *datasets* de 97,96%, o que é uma melhora em 3,21% do classificador HAT e 4,67% do classificador HT, porém o seu tempo total de treino e teste por instância é 9,15 vezes maior que o classificador HAT e 12,08 vezes maior que o classificador HT.

O pior em acurácia do ML *Off-line* apresentou na média entre os *datasets* a acurácia de 80%, o que é superior ao classificador NB do ML *On-line* (pior em acurácia) em 1,27% apenas com um tempo médio 1,62 vezes maior. Logo, o pior do ML *Off-line* em acurácia (classificador NB presente em seis do sete *datasets*) se mostra um pouco melhor na acurácia com um tempo maior que o classificador NB do ML *On-line* (o pior do MOA). Porém, o mesmo apresenta uma acurácia média ainda inferior de 13,29% em relação ao classificador HT e 14,75% em relação ao classificador HAT. Logo, podemos concluir que o pior do ML *Off-line* em acurácia, apesar de ter um tempo 8,6 vezes menor que o melhor

em acurácia do ML *Off-line*, o seu uso ainda não é justificado quando comparado com o classificador HAT ou HT em acurácia e nem em tempo.

Podemos então concluir a princípio que o aumento na acurácia no ML *Off-line* não justifica o tempo a mais de processamento quando comparado com o classificador HAT do ML *On-line*. Porém, para dar total credibilidade a este argumento, é necessário verificar as demais métricas de F1*Score*, Precisão e *Recall*, onde podemos verificar que o ML *Off-line* apresenta um melhor desempenho nas métricas. Logo, deve ser analisado durante uma escolha entre os dois métodos a relação entre as melhores métricas do ML *Off-line* e o menor tempo do ML *On-line*.

Podemos observar no final, pela Tabela 5, onde em verde constam os melhores classificadores em acurácia no ML *Off-line*, que o tempo por instância de treinamento está relacionado com a qualidade da acurácia. Ou seja, quanto maior é a acurácia mais tempo é gasto no aprendizado. Logo, o classificador NB é o que apresenta a pior acurácia com o menor tempo e a sequência REPTree, J48, RaT e RaF são os que apresentam as melhores acurácias com o maior tempo de treinamento.

Já o tempo de teste por instância está relacionado também com a qualidade da acurácia, porém quanto melhor o classificador menor é o tempo por instância no teste. O classificador REPTree, por exemplo, é o que mais se destaca com acurácias acima de 99% (Tabela 5) e tempo médio de 0,5 microsegundos por instância no teste. A razão para isso ocorrer no teste é o tamanho do modelo aprendido (ou seja, o tamanho dos ramos e folhas do modelo) para classificar cada instância. Logo, as árvores do *REPTree* dos *datasets* ERENO, Morris-3 water e BATADAL devem ser pequenas quando comparadas com as demais. Infelizmente, nessa dissertação de mestrado não conseguimos no prazo medir o tamanho do modelo aprendido (como o tamanho das arvores de decisão) de todos os classificadores, o que pode ser um tema para trabalhos futuros.

4.3.4 Comparação entre ML *Off-line* e ML *On-line* pela Acurácia e Memória

Os cálculos de consumo de memória foram feitos de maneiras diferentes entre os *frameworks* MOA e WEKA. No *framework* MOA foi utilizado o próprio arquivo exportado pela aplicação, que após cada amostragem calcula a quantidade de memória utilizada em RAM-Hora, que depois foi convertida para bytes multiplicando o resultado pela Equação 4.1. O computador utilizado para este experimento é o computador pessoal descrito na Seção 4.2.

Tabela 5: Ordenação do tempo de treino e teste do WEKA entre classificadores e *datasets*.

Classificação em Tempo de Treinamento por Instância	**Acurácia**	**Classificação em Tempo de Teste por Instância**	**Acurácia**
Morris-4 NB	82.18%	ERENO REPTree	100.00%
Morris-3 water NB	90.78%	Morris-3 water REPTree	99.13%
Morris-3 gas NB	94.86%	BATADAL REPTree	99.18%
SWaT NB	87.15%	Morris-3 gas J48	99.26%
ERENO NB	75.31%	BATADAL RaT	98.27%
Morris-1 NB	31.42%	Morris-4 RaT	97.94%
BATADAL RaT	98.27%	Morris-4 NB	82.18%
Morris-3 water REPTree	99.13%	SWaT NB	87.15%
BATADAL REPTree	99.26%	SWaT RaF	99.97%
Morris-3 gas J48	99.18%	Morris-1 NB	31.42%
Morris-4 RaT	97.94%	Morris-3 water NB	90.78%
ERENO REPTree	100.00%	Morris-3 gas NB	94.86%
SWaT RaF	99.97%	Morris-1 RaF	90.39%
Morris-1 RaF	90.39%	ERENO NB	75.31%

$$\frac{\left(1024 \times \frac{MB}{GB} \times 1024 \times \frac{KB}{MB} \times 1024 \times \frac{Byte}{KB} \times 3600 \times \frac{segundos}{hora}\right)}{Tempo\ Total\ do\ Experimento\ em\ segundos} \tag{4.1}$$

Podemos verificar no ML *On-line*, Figura 16, que o classificador NB é o que menos consome memória, uma média de 27,86 KB. O classificador HAT vem logo após com consumo médio de 61,75 KB, que é 2,21 vezes maior que o classificador NB. O classificador HT é o que consome mais memória com uma média de 1,59 MB, que é 26,36 vezes maior que o classificador HAT. Quando levamos em consideração a acurácia, podemos notar que o classificador HAT apresenta melhor métrica com um consumo reduzido de memória em comparação com o classificador HT. O classificador NB apresenta a menor acurácia com o menor consumo de memória, o que não justifica o seu uso diante do classificador HAT. Podemos notar aqui que o classificador NB tem uma pequena capacidade de adaptação na percepção de novos ataques, o que justifica a sua baixa acurácia. O classificador HT não faz uso da janela ADWIN como o classificador HAT faz, o que justifica o seu menor consumo de memória.

No *framework* WEKA, foram utilizadas as bibliotecas do Java e foi retirada a diferença entre o método *totalMemory* e o *freeMemory* executados antes e depois do treino e antes e depois do teste, para obter o total de memória utilizada em cada processo. Porém, foi possível apenas observar uma relação direta do tamanho do *dataset* com a quantidade de memória consumida, logo não foi possível observar quanto exclusivamente foi utilizado

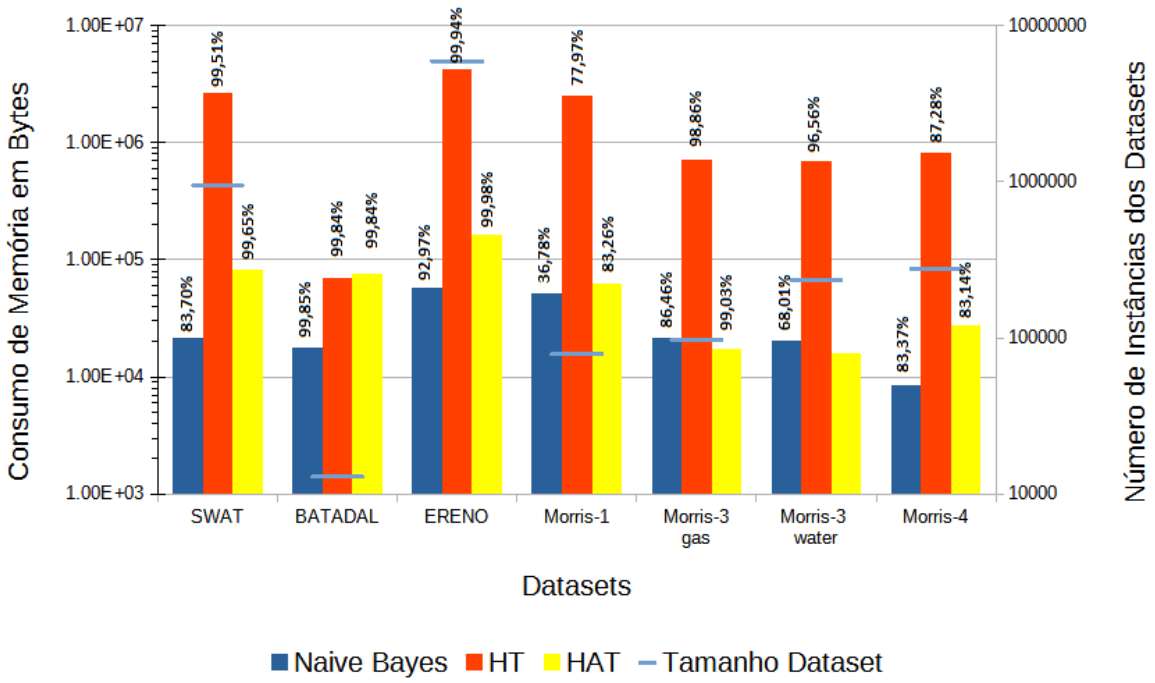

Figura 16: Consumo de memória do *framework* MOA no eixo y principal dos *datasets* selecionados. Acurácia nas barras verticais. Tamanho dos *datasets* no eixo y secundário.

pelo classificador.

Uma outra tentativa foi executar no Linux os dois *frameworks* de forma a verificar com o comando *cat* os arquivos *proc/$$/statm* e *proc/$$/status*, com o PID do processo, a fim de adquirir os valores de memória da pilha e do segmento de dados com o *dataset* ERENO. Esta abordagem também não se mostrou adequada pois a alocação da memória é fortemente dependente do tamanho do *dataset*. Seria importante verificar quanto da memória é usada exclusivamente para o armazenamento do *dataset* e quanto é usado exclusivamente pelo método de classificação.

Desta forma, um estudo mais completo se faz necessário de forma a fazer com que as comparações sejam mais justas. Assumindo que a fase de teste seja simples e rápida para os dois *frameworks* e que no *framework* MOA não há como separar pela interface o tempo de treino e teste, uma forma promissora seria comparar o tamanho das árvores de decisão, conforme o trabalho de Dahal et al. (2015). Isso poderá ser feito em trabalhos futuros, como uma continuação da pesquisa desta dissertação.

Em resposta à pergunta 4: Qual é a técnica de IDS mais adequada para dispositivos de hardware limitado? Na Figura 15 que o ML *Off-line* possui as melhores métricas de classificação, porém o ML *On-line* apresenta menor consumo de recursos de tempo. Considerando que a memória consumida no ML *On-line* é pequena, devido ao treinamento

ocorrer de maneira dinâmica, acompanhada do teste, sem a necessidade de armazenar grande quantidade de informações, ele apresenta também menor consumo de recursos de memória. Este resultado fica de acordo com trabalhos dos autores Dahal et al. (2015), Faisal et al. (2014) e Adhikari, Thomas H Morris e Pan (2017), ou seja, o ML *On-line* é o melhor para dispositivos de CPS com limitado *hardware* devido ao baixo consumo de recursos. O ML *Off-line* poderá ser até usado nas ponta de uma rede CPS em dispositivos como sensores e atuadores na fase de teste, com a fase de treinamento sendo feita em um servidor central da rede com maior disposição de recursos de processamento e memória.

4.3.5 Desempenho do ML *Off-line* e ML *On-line* com Ocorrência de Novos Ataques ao Longo do Tempo

Foram selecionados, para comparar o desempenho entre ML *Off-line* e ML *On-line*, os classificadores com as melhores métricas, o HAT e o *REPTree*, respectivamente. No *framework* WEKA, o experimento foi feito com o *dataset* ERENO com os Casos de Uso em sequência. Procuramos dessa forma fazer o aprendizado em sequência seguido do teste, conforme as Figuras 17 e 18, montadas a partir da Tabela 16 do Apêndice A. A classificação ocorreu de forma binária, ou seja, foi verificado apenas a identificação quanto a ocorrência ou não de ataque, e não quanto a identificação do tipo de ataque. Foram utilizados seis *datasets* de treino com os Casos de Uso acumulados: UC1, UC1-2, UC1-3, UC1-4, UC1-5 e UC1-6. Eles foram testados respectivamente com os Casos de Uso: UC2, UC3, UC4, UC5, UC6 e UC7, de forma a representar um desenvolvimento temporal entre eles. No ML *On-line*, foi utilizado o *dataset* ERENO contendo todos os Casos de Uso em sequência do UC1 a UC7, onde foram calculadas as métricas de acurácia, TFN e TFP, isoladamente com uma janela em cada UC. Os resultados são detalhados a partir da Tabela 17 do Apêndice A.

4.3.5.1 Análise no *Framework* WEKA

No *Framework* WEKA, conforme as Figuras 17 e 18, os casos UC1/UC2 (treina no Caso de Uso 1 e testa no 2), UC1-2/UC3 (treina nos Casos de Uso de 1 a 2 e testa no 3), UC1-3/UC4 (treina nos Casos de Uso 1 a 3 e testa no 4) e UC1-4/UC5 (treina nos Casos de Uso 1 a 4 e testa no 5) apresentaram desempenhos abaixo da Classe Majoritária, com a TFN e TFP muito reduzidos para o classificador *REPTree*, o melhor do ML *Off-line* para este *dataset*. Estes valores mostram que o classificador não foi capaz de perceber novos ataques. Logo, não tem a capacidade de se adaptar a mudanças de conceito, o que

já era esperado devido à característica de funcionamento do ML *Off-line*. Os dois últimos casos, UC1-5/UC6 (treina nos Casos de Uso de 1 a 5 e testa no 6) e UC1-6/UC7 (treina nos Casos de Uso de 1 a 6 e testa no 7), apresentaram valores consistentes após análise de 59% dos dados do *dataset*.

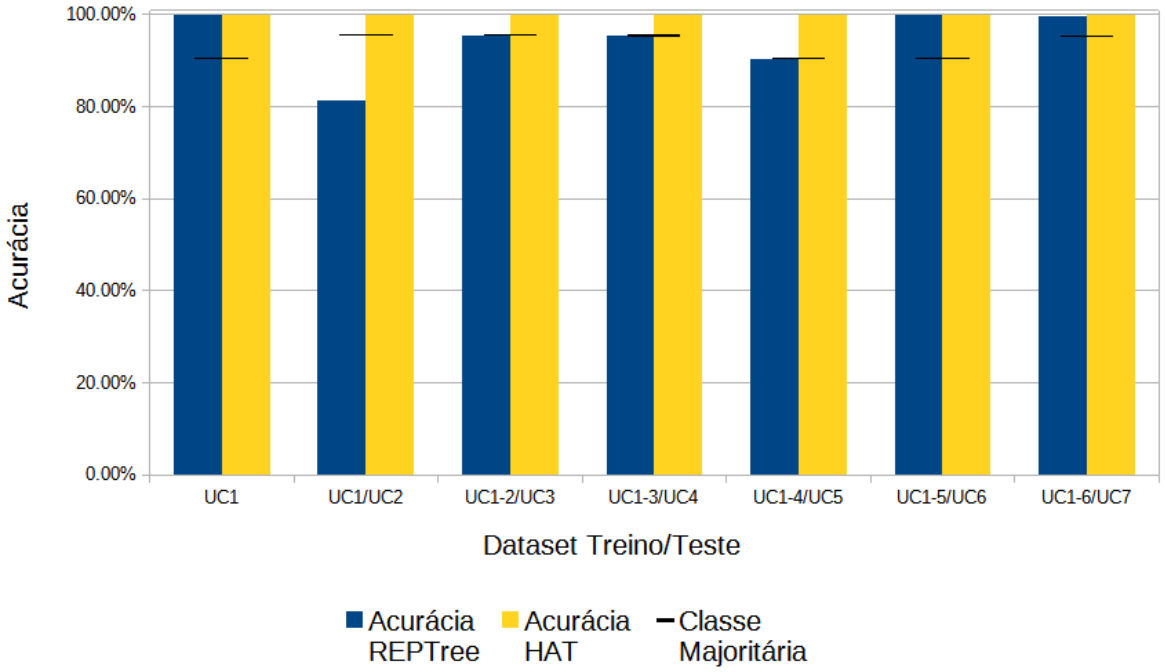

Figura 17: Acurácia entre o ML *Off-line* e ML *On-line*, com o *dataset* ERENO em diferentes casos de uso para Treino e Teste.

Uma investigação mais aprofundada deve ser feita para verificar os motivos desses dois últimos valores, que podem ser devidos à facilidade em identificar os ataques, por serem semelhantes a um dos anteriores já usados para treino. Um dos experimentos executados com este objetivo, cujos resultados são mostrados na Tabela 6, utiliza a abordagem *Leave-One-Out*, onde o treinamento é executado com todos os Casos de Uso menos o que será usado no teste. Os valores em vermelho mostram a combinação entre treino e teste dos UC onde a acurácia ficou abaixo da Classe Majoritária e com valor baixo de Precisão ou *Recall*, os valores em verde mostram a combinação entre treino e teste onde a acurácia ficou superior a Classe Majoritária e com valor alto de Precisão ou *Recall*. Os casos de uso onde os valores de FP e VP foram nulos foram representados por VP/FP(Ø), pois a precisão foi uma divisão de zero sobre zero. Os valores de F1*Score* para estes mesmos casos de uso foi representado por Indef., o que mostra a indefinição no cálculo da fórmula desta métrica, devido ao que ocorreu com o valor da Precisão já descrito.

Estes experimentos mostram a relação entre os ataques: UC1, *Random Replay Attacks*; UC2, *Inverse Replay Attacks*; UC3, *Masquerade Attacks - Outage*; UC4, *Masque-*

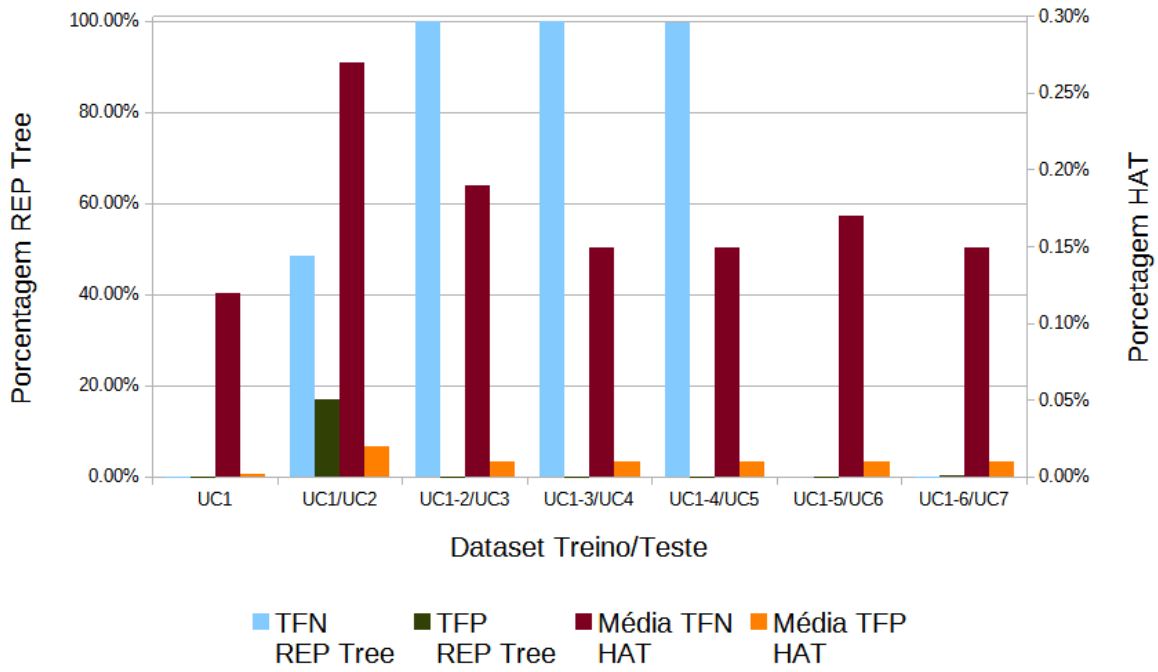

Figura 18: Taxa de Falso Positivo e Taxa de Falso Negativo entre o classificador REPTree do ML *Off-line* no eixo y principal e o classificador *Hoeffding Adaptive Tree* do ML *On-line* no eixo y secundário, com o *dataset* ERENO com diferentes casos de uso em Treino e Teste.

Tabela 6: Teste *Leave-One-Out*, Treino com todos os Casos de Uso menos o Caso de Uso em Teste, com o *dataset* ERENO.

Leave One Out	Acuracy	Precision	Recall	F1Score	VP	VN	FP	FN
UC-1	99.09%	100.00%	90.46%	94.99%	70560	742794	0	7440
UC-2	98.66%	100.00%	73.46%	84.70%	41397	1057800	0	14955
UC-3	95.58%	100.00%	0.46%	0.92%	160	742794	0	34327
UC-4	95.52%	VP/FP(Ø)	0.00%	Indef.	0	742794	0	34839
UC-5	90.50%	100.00%	0.02%	0.05%	19	742794	0	77981
UC-6	99.99%	100.00%	99.94%	99.97%	77950	742794	0	50
UC-7	99.69%	93.90%	100.00%	96.86%	37144	740383	2411	0

rade Attacks - Equipment Damage; UC5, *Random Message Injection attacks*; UC6, *High Status Number attacks*; e UC7, *High-Rate Flooding Attack*. Desta forma, foi possível verificar que os ataques *Masquerade Attacks - Outage*, *Masquerade Attacks - Equipment Damage* e *Random Message Injection attacks* (Casos de Uso 3, 4 e 5, respectivamente), não são identificados por nenhum outro UC já analisado. Estes ataques são apontados também por Silvio Quincozes (2022) como os mais difíceis de serem detectados. O ataque *Inverse Replay Attacks*, Caso de Uso 2, ficou em uma posição intermediária. Os ataques *Random Replay Attacks*, *High Status Number attacks* e *High-Rate Flooding Attack* (Casos de Uso 1, 6 e 7, respectivamente) são facilmente identificados por outros UC.

Para complementar esta análise e encontrar uma relação individual entre os Casos de

Tabela 7: Classificador *REPTree* com Treino e Teste (UC1, UC2, UC3 e UC4) entre os Casos de Uso do *dataset* ERENO.

Treino\Teste	UC1		UC2		UC3		UC4	
UC1	**Accuracy**	**Precision**	81.31%	13.84%	49.00%	0.00%	95.52%	VP/FP(Ø)
	Recall	**F1Score**	51.57%	21.82%	0.00%	0.00%	0.00%	Indef.
UC2	99.01%	100.00%	**Accuracy**	**Precision**	95.74%	100.00%	95.52%	VP/FP(Ø)
	89.58%	94.50%	**Recall**	**F1Score**	3.92%	7.54%	0.00%	Indef.
UC3	94.78%	99.77%	95.82%	98.40%	**Accuracy**	**Precision**	95.52%	VP/FP(Ø)
	45.22%	62.23%	17.73%	30.05%	**Recall**	**F1Score**	0.00%	Indef.
UC4	93.11%	100.00%	94.94%	VP/FP(Ø)	95.56%	0.00%	**Accuracy**	**Precision**
	27.47%	43.10%	0.00%	Indef.	0.00%	0.00%	**Recall**	**F1Score**
UC5	90.50%	VP/FP(Ø)	94.94%	VP/FP(Ø)	95.56%	VP/FP(Ø)	95.52%	VP/FP(Ø)
	0.00%	Indef.	0.00%	Indef.	0.00%	Indef.	0.00%	Indef.
UC6	90.50%	VP/FP(Ø)	94.94%	VP/FP(Ø)	95.56%	VP/FP(Ø)	94.94%	VP/FP(Ø)
	0.00%	Indef.	0.00%	Indef.	0.00%	Indef.	0.00%	Indef.
UC7	98.82%	100.00%	97.33%	95.11%	95.92%	100.00%	95.52%	VP/FP(Ø)
	87.58%	93.38%	49.69%	65.28%	8.00%	14.82%	0.00%	Indef.

Uso, executamos também o experimento das Tabelas 7 e 8. Os mesmos esquemas de cores do experimento anterior foram usados, onde todas as combinações entre cada UC entre treino e teste foram executadas. Podemos observar que o UC1 é facilmente encontrado com o aprendizado sobre o UC2 e UC7, ou seja, o ataque *Random Replay Attack* pode ser identificado a partir dos ataques *Inverse Replay Attack* e *High-Rate Flooding Attack*. O UC2 é facilmente encontrado com UC7, desta forma, o ataque *Inverse Replay Attacks* pode ser identificado a partir do ataque *High-Rate Flooding Attack*.

Concluindo, voltando para a análise temporal montada inicialmente no experimento das Figuras 17 e 18, podemos observar que o UC 6 é facilmente encontrado com o aprendizado dos Casos de Uso 2 e 5. Logo, o ataque *High-Rate Flooding Attack* (UC6) possui uma forte relação com os ataques *Inverse Replay Attacks* (UC2) e *Random Message Injection attacks* (UC5). Já o UC 7 é facilmente encontrado com o aprendizado sobre os Casos de Uso 2 e 3. Desta forma, o ataque *High-Rate Flooding Attack* (UC7) possui uma forte relação com os ataques *Inverse Replay Attacks* (UC2) e *Masquerade Attacks - Outage* (UC3). Para uma análise mais aprofundada da relevância das *features* com o classificador J48 recomendamos a leitura do trabalho de Silvio Quincozes (2022), a fim de identificar quais estão mais correlacionadas com cada tipo de ataque. Apontamos, como um trabalho futuro, a análise com as *features* com a árvore de decisão do classificador *REPTree*.

4.3.5.2 Análise no *Framework* MOA

No experimento com o *framework* MOA, as métricas foram tiradas com uma janela de tamanho igual ao Caso de Uso em avaliação, logo o método prequencial foi executado em todo o *dataset* porém os resultados são sempre referentes à última janela. As Figuras 19

Tabela 8: Classificador *REPTree* com Treino e Teste (UC5, UC6 e UC7) entre os Casos de Uso do *dataset* ERENO.

Treino\Teste	**UC5**		**UC6**		**UC7**	
UC1	90.50%	VP/FP(Ø)	91.18%	100.00%	76.78%	16.97%
	0.00%	Indef.	7.19%	13.42%	99.54%	28.99%
UC2	90.50%	100.00%	99.97%	100.00%	99.69%	93.90%
	0.04%	0.07%	99.66%	99.83%	100.00%	96.86%
UC3	92.42%	100.00%	94.57%	100.00%	100.00%	100.00%
	20.25%	33.68%	42.90%	60.05%	100.00%	100.00%
UC4	90.50%	100.00%	95.25%	99.93%	94.92%	0.00%
	0.01%	0.03%	50.01%	66.66%	0.00%	0.00%
UC5	**Accuracy**	**Precision**	100.00%	100.00%	95.24%	VP/FP(Ø)
	Recall	**F1Score**	100.00%	100.00%	0.00%	Indef.
UC6	92.88%	100.00%	**Accuracy**	**Precision**	95.24%	VP/FP(Ø)
	25.05%	40.06%	**Recall**	**F1Score**	0.00%	Indef.
UC7	90.63%	100.00%	91.19%	100.00%	**Accuracy**	**Precision**
	1.40%	2.75%	7.28%	13.58%	**Recall**	**F1Score**

e 20 mostram a evolução do classificador HAT ao longo dos Casos de Uso em sequência. Podemos ver que a sua acurácia é elevada, acima de 99,96% durante todos os Casos de Uso e é finalizado em 99,98%. A TFP se manteve baixa praticamente em todos os Casos de Uso com exceção do UC2 onde ficou com 0,02%, com 90 FN para 56784 VP. A TFN apresentou valores maiores, porém ainda pequenos de forma a influenciar muito pouco na acurácia do classificador. O gráfico mostra que o ML *On-line* teve um desempenho superior em todos os instantes, porém com dois casos, onde o ML *Off-line* e MOA apresentaram acurácias semelhantes, em UC1 (99,99% e 99,98%, respectivamente) e UC6 (99,99% e 99,98%, respectivamente). Desta forma, o MOA conseguiu manter seu alto nível de acerto nas predições ao longo de todas as instâncias, o que mostra a sua adaptabilidade a novos ataques ainda não treinados.

Em resposta à pergunta 5: Qual é a diferença entre os ML *On-line* e ML *Off-line* na detecção de novos ataques em sequência? Conforme as Figuras 17 a 20, podemos constatar que o ML *On-line* apresenta uma capacidade maior em detectar novos ataques. O ML *Off-line* apresenta valores inferiores de acurácia, TFN e TFP ao longo de todas as instâncias quando comparado com o ML *On-line*. Este resultado mostra a capacidade maior do ML *On-line* em se adaptar a novos ataques, apresentados em cada Caso de Uso ao longo das instâncias.

Desta forma, com os experimentos executados até aqui, buscamos responder às principais questões de pesquisa apresentadas na Introdução, Capítulo 1, e detalhadas no Capítulo 4, a fim de investigar qual é o método de ML mais adequado para CPS.

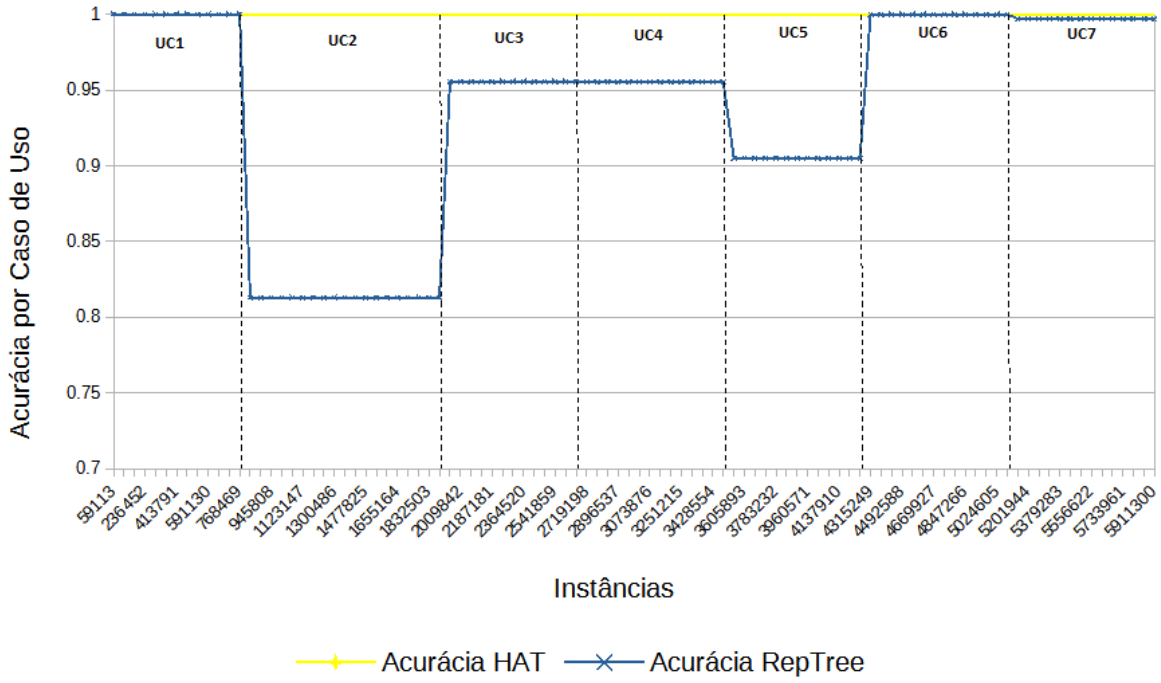

Figura 19: Acurácia do *dataset* ERENO entre os ML *Off-line* e ML *On-line* por Instância no eixo x.

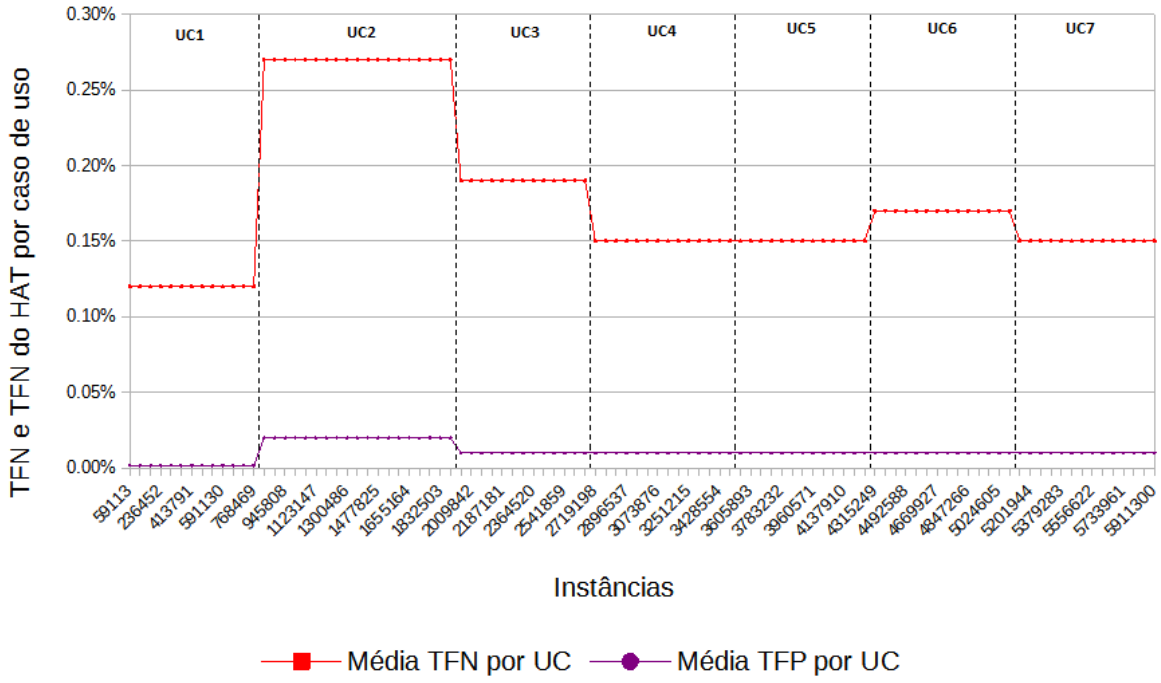

Figura 20: Detalhamento da Taxa de Falso Negativo e Taxa de Falso Positivo do classificador *Hoeffding Adaptive Tree* do *dataset* ERENO entre o ML *Off-line* e ML *On-line* por Instância no eixo x.

5 Conclusões

Este trabalho fez um estudo detalhado sobre Detecção de Intrusão em CPS, com o objetivo de identificar as melhores técnicas de ML *On-line* e ML *Off-line* para dispositivos com baixa capacidade computacional diante de um grande fluxo de dados. Ao longo da dissertação foram descritos conceitos básicos de IDS, assim como as abordagens mais atuais tanto em métodos amplamente empregados de Aprendizado de Máquina *Off-line*, quanto uma breve descrição do funcionamento das técnicas de aprendizado de máquina *On-line*. Com a clara importância da utilização de *datasets* na avaliação dos IDS, se buscou elencar também os principais *datasets* disponíveis no meio acadêmico focados em CPS.

Uma Revisão Sistemática da Literatura foi realizada sobre o MOA para descrever o estado da arte quanto ao aprendizado de máquina *On-line*, as principais técnicas de predição utilizadas, assim como os principais *datasets* empregados. Foi a partir dessa RSL que descobrimos que apenas o trabalho de Dahal et al. (2015) fez testes com *dataset* específico de CPS de forma a extrair dados de memória e tempo para comparar os classificadores *Hoeffding Tree*, J48 e *REPTree*. Desta forma, foi possível identificar uma ausência de estudos mais específicos nessa área sobre *datasets* reais específicos de CPS que possibilitasse comparar o desempenho entre os métodos de ML *Off-line* e ML *On-line*, o que foi um dos objetivos dessa dissertação.

Para a análise dos dois métodos de Aprendizado de Máquina *On-line* e *Off-line*, foram utilizados os *frameworks* MOA e WEKA, respectivamente, a fim de comparar e verificar as suas principais características diante dos *datasets* apresentados. Para o ML *Off-line* foram utilizados os classificadores *Random Tree*, J48, *REPTree*, *Naive Bayes* e *Random Forest*, a fim de dar continuidade ao trabalho desenvolvido em conjunto em Quincozes, Mossé et al. (2021). Para o ML *On-line*, decidimos abordar os classificadores NB, HT, HAT com o método prequencial, por serem os classificadores mais encontrado na RSL. No final, os *datasets* foram submetidos aos sete classificadores (três do ML *On-line* e cinco do ML *Off-line*) a fim de fazer uma análise das suas principais características de VP, VN, FP e FN ao longo do tempo.

Desta forma, as perguntas iniciais, descrita na introdução, foram respondidas.

5.1 Contribuições

Desta forma, as principais contribuições são:

- Levantamento e análise de *datasets* disponíveis no meio acadêmico para CPS. Foi possível observar as principais características de cada *dataset* e a forte dependência dos classificadores *On-line* quanto à ocorrência dos ataques ao longo do tempo.
- Revisão Sistemática da Literatura de artigos relacionados aos Sistemas de Detecção de Intrusão com Aprendizado de Máquina *On-line* com uso do *framework* MOA, a fim de identificar as contribuições para o meio acadêmico que puderam ser exploradas nessa dissertação.
- Experimento com os sete *datasets* analisados a fim de extrair métricas de desempenho dos diferentes classificadores (*Random Tree*, J48, *REPTree*, *Random Forest*, *Naive Bayes*, *Hoeffding Tree* e *Hoeffding Adaptive Tree*). Nestes experimentos foi possível identificar, por exemplo, o HAT, um classificador adaptativo com janela ADWIN, com capacidade de detectar os ataques em menor tempo, quando comparado com os classificador *REPTree*, e com menos ocorrência de erros FP e FN, quando comparado com o classificador *Naive Bayes*.
- Comparação direta entre o ML *Off-line* e ML *On-line* quanto a métricas de Acurácia, *Recall*, Precisão, F1*Score*, TFN e TFP, assim como comparações quanto ao consumo de recursos de memória e tempo. Foi possível observar que o ML *On-line* apresenta menor consumo de recursos de memória e tempo de processamento quanto comparado com o ML *Off-line*, porém este último apresenta maior capacidade em identificar os ataques.
- Demonstração comparativa, com o *dataset* ERENO, entre o processo de predição de ML *Off-line* com ML *On-line*, onde o último teve um desempenho superior ao longo do tempo, o que mostra a sua capacidade em se adaptar a novos ataques, apesar de apresentar piores métricas de identificação de ataques.

5.2 Trabalhos Futuros

Estudos futuros a serem desenvolvidos incluem:

- Ampliar a comparação entre o ML *Off-line* e ML *On-line*, como uma comparação detalhada quanto ao uso de memória. O tamanho das árvores de decisão pode ser um indício para esta comparação, conforme o trabalho de Dahal et al. (2015).

- As instâncias observadas individualmente e progressivamente ao longo de um fluxo desbalanceado podem reduzir a quantidade de classes minoritárias disponíveis para o aprendizado, o que pode gerar no método prequencial um classificador tendencioso. Logo, um balanceamento do fluxo de dados online antes da fase de treinamento (BERNARDO; DELLA VALLE; BIFET, 2020) pode ser feito sobre os *datasets* desbalanceados, como o BATADAL, SWaT e ERENO, para estudar estes efeitos.

- Devido ao fato dos *datasets* utilizados nesta dissertação serem estáticos, uma análise prática temporal com grandes volumes de dados pode ser elaborada de forma a fazer predições de ataques em um fluxo de dados online. Uma comparação temporal pode ser feita quanto ao atraso entre a predição e a confirmação da classificação diante de diferentes ataques e de Mudanças de Conceito.

REFERÊNCIAS

ABREU, Diego; CARVALHO, Igor; ABELÉM, Antônio. Seleção de Caracterıésticas em Stream para a Detecção de Ataques de Rede. In: SBC. ANAIS do XXI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. [S. l.: s. n.], 2021. p. 197–210.

ADHIKARI, Uttam. **Event and intrusion detection systems for cyber-physical power systems**. [S. l.]: Mississippi State University, 2015.

ADHIKARI, Uttam; MORRIS, Thomas; PAN, Shengyi. WAMS cyber-physical test bed for power system, cybersecurity study, and data mining. **IEEE Transactions on Smart Grid**, IEEE, v. 8, n. 6, p. 2744–2753, 2016.

ADHIKARI, Uttam; MORRIS, Thomas H; PAN, Shengyi. Applying hoeffding adaptive trees for real-time cyber-power event and intrusion classification. **IEEE Transactions on Smart Grid**, IEEE, v. 9, n. 5, p. 4049–4060, 2017.

AGRAWAL, Anand et al. Towards Robust Power Grid Attack Protection using LightGBM with Concept Drift Detection and Retraining. In: PROCEEDINGS of the 2020 Joint Workshop on CPS&IoT Security and Privacy. [S. l.: s. n.], 2020. p. 31–36.

ALJAMAL, Ibraheem et al. Hybrid intrusion detection system using machine learning techniques in cloud computing environments. In: IEEE. 2019 IEEE 17th international conference on software engineering research, management and applications (SERA). [S. l.: s. n.], 2019. p. 84–89.

ALMOMANI, Iman; AL-KASASBEH, Bassam; AL-AKHRAS, Mousa. WSN-DS: A dataset for intrusion detection systems in wireless sensor networks. **Journal of Sensors**, Hindawi, v. 2016, 2016.

ANTON, Simon Duque et al. Two decades of SCADA exploitation: A brief history. In: IEEE. 2017 IEEE Conference on Application, Information and Network Security (AINS). [S. l.: s. n.], 2017. p. 98–104.

ASSIS, Danilo. **Uma visão geral do padrão IEEE 2030.5-2018 com abordagem de segurança**. Mar. 2021. Diss. (Mestrado) – Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, Brasil.

AUNG, Yan Lin et al. Scalable VPN-forwarded Honeypots: Dataset and Threat Intelligence Insights. In: SIXTH Annual Industrial Control System Security (ICSS) Workshop. [S. l.: s. n.], 2020. p. 21–30.

BAEK, Sunhee et al. Unsupervised labeling for supervised anomaly detection in enterprise and cloud networks. In: IEEE. 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud). [S. l.: s. n.], 2017. p. 205–210.

BEAVER, Justin M et al. An evaluation of machine learning methods to detect malicious SCADA communications. In: IEEE. 2013 12th international conference on machine learning and applications. [S. l.: s. n.], 2013. v. 2, p. 54–59.

BENKHELIFA, Elhadj; WELSH, Thomas; HAMOUDA, Walaa. A critical review of practices and challenges in intrusion detection systems for IoT: Toward universal and resilient systems. **IEEE communications surveys & tutorials**, IEEE, v. 20, n. 4, p. 3496–3509, 2018.

BERNARDO, Alessio; DELLA VALLE, Emanuele; BIFET, Albert. Incremental Rebalancing Learning on Evolving Data Streams. In: IEEE. 2020 International Conference on Data Mining Workshops (ICDMW). [S. l.: s. n.], 2020. p. 844–850.

BIFET, Albert; FRANCISCI MORALES, Gianmarco de et al. Efficient online evaluation of big data stream classifiers. In: PROCEEDINGS of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. [S. l.: s. n.], 2015. p. 59–68.

BIFET, Albert; GAVALDA, Ricard. Adaptive learning from evolving data streams. In: SPRINGER. INTERNATIONAL Symposium on Intelligent Data Analysis. [S. l.: s. n.], 2009. p. 249–260.

_____. Learning from time-changing data with adaptive windowing. In: SIAM. PROCEEDINGS of the 2007 SIAM international conference on data mining. [S. l.: s. n.], 2007. p. 443–448.

BIFET, Albert; HOLMES, Geoff et al. Moa: Massive online analysis, a framework for stream classification and clustering. In: PMLR. PROCEEDINGS of the First Workshop on Applications of Pattern Analysis. [S. l.: s. n.], 2010. p. 44–50.

CASE, Defense Use. Analysis of the cyber attack on the Ukrainian power grid. **Electricity Information Sharing and Analysis Center (E-ISAC)**, v. 388, p. 1–29, 2016.

CESCHIN, Fabrıécio et al. Machine Learning (In) Security: A Stream of Problems. **arXiv preprint arXiv:2010.16045**, 2020.

CHANDOLA, Varun; BANERJEE, Arindam; KUMAR, Vipin. Anomaly detection: A survey. **ACM computing surveys (CSUR)**, ACM New York, NY, USA, v. 41, n. 3, p. 1–58, 2009.

CHOI, Seungoh; YUN, Jeong-Han; KIM, Sin-Kyu. A comparison of ICS datasets for security research based on attack paths. In: SPRINGER. INTERNATIONAL Conference on Critical Information Infrastructures Security. [S. l.: s. n.], 2018. p. 154–166.

CORRÊA, Diego Guarnieri. **Detecção de Anomalias em Redes Utilizando Mineração em Fluxos Contínuos de Dados**. Mar. 2017. Diss. (Mestrado) – UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ CÂMPUS CORNÉLIO PROCÓPIO, CORNÉLIO PROCÓPIO, PR, Brasil.

CORRÊA, Diego Guarnieri; ENEMBRECK, Fabrıécio; SILLA, Carlos N. An investigation of the hoeffding adaptive tree for the problem of network intrusion detection. In: IEEE. 2017 International Joint Conference on Neural Networks (IJCNN). [S. l.: s. n.], 2017. p. 4065–4072.

DAHAL, Nischal et al. Event stream processing for improved situational awareness in the smart grid. **Expert Systems with Applications**, Elsevier, v. 42, n. 20, p. 6853–6863, 2015.

DESALE, Ketan Sanjay; KUMATHEKAR, Chandrakant Namdev; CHAVAN, Arjun Pramod. Efficient intrusion detection system using stream data mining classification technique. In: IEEE. 2015 International Conference on Computing Communication Control and Automation. [S. l.: s. n.], 2015. p. 469–473.

DOMINGOS, Pedro; HULTEN, Geoff. Mining high-speed data streams. In: PROCEEDINGS of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. [S. l.: s. n.], 2000. p. 71–80.

ESSEGHIR, Mohamed Amir. Effective wrapper-filter hybridization through grasp schemata. In: PMLR. FEATURE Selection in Data Mining. [S. l.: s. n.], 2010. p. 45–54.

FAISAL, Mustafa Amir et al. Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study. **IEEE Systems journal**, IEEE, v. 9, n. 1, p. 31–44, 2014.

FAISAL, Mustafa Amir et al. Securing advanced metering infrastructure using intrusion detection system with data stream mining. In: SPRINGER. PACIFIC-ASIA Workshop on Intelligence and Security Informatics. [S. l.: s. n.], 2012. p. 96–111.

FARHAT, Saida et al. Comparative Study of Classification Algorithms for Cloud IDS using NSL-KDD Dataset in WEKA. In: IEEE. 2020 International Wireless Communications and Mobile Computing (IWCMC). [S. l.: s. n.], 2020. p. 445–450.

FATTAHI, Javad et al. Transactive Demand Response Operation at the Grid Edge using the IEEE 2030.5 Standard. **Engineering**, Elsevier, v. 6, n. 7, p. 801–811, 2020.

FISHER, Brigitte. **Entwicklung eines Konzeptes für ein Framework zur teilautomatisierten Erstellung von prozessbasierten IDS-Regeln in ICS**. 2019. Tese (Doutorado) – Fernuniversität Hagen.

AL-FUQAHA, Ala et al. Internet of things: A survey on enabling technologies, protocols, and applications. **IEEE communications surveys & tutorials**, IEEE, v. 17, n. 4, p. 2347–2376, 2015.

GAMA, Joao; SEBASTIAO, Raquel; RODRIGUES, Pedro Pereira. Issues in evaluation of stream learning algorithms. In: PROCEEDINGS of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. [S. l.: s. n.], 2009. p. 329–338.

GHALIB, Marwan et al. Implementation of a smart grid communication system compliant with IEEE 2030.5. In: IEEE. 2018 IEEE International Conference on Communications Workshops (ICC Workshops). [S. l.: s. n.], 2018. p. 1–6.

GHARIB, Amirhossein et al. An evaluation framework for intrusion detection dataset. In: IEEE. 2016 International Conference on Information Science and Security (ICISS). [S. l.: s. n.], 2016. p. 1–6.

GOH, Jonathan et al. A dataset to support research in the design of secure water treatment systems. In: SPRINGER. INTERNATIONAL conference on critical information infrastructures security. [S. l.: s. n.], 2016. p. 88–99.

GREEN, Wyatt; JOHNSTEN, Tom; BENTON, Ryan G. TADS: Transformation of Anomalies in Data Streams. In: IEEE. 2020 IEEE International Conference on Big Data (Big Data). [S. l.: s. n.], 2020. p. 4284–4292.

HA, Taejin et al. Suspicious flow forwarding for multiple intrusion detection systems on software-defined networks. **IEEE Network**, IEEE, v. 30, n. 6, p. 22–27, 2016.

HARADA, Yoshiyuki et al. Log-based anomaly detection of CPS using a statistical method. In: IEEE. 2017 8th International Workshop on Empirical Software Engineering in Practice (IWESEP). [S. l.: s. n.], 2017. p. 1–6.

HIDALGO, Juan I González; MACIEL, Bruno IF; BARROS, Roberto SM. Experimenting with prequential variations for data stream learning evaluation. **Computational Intelligence**, Wiley Online Library, v. 35, n. 4, p. 670–692, 2019.

HIDALGO, Juan Isidro González. **Experiências com variações prequential para avaliação da aprendizagem em fluxo de dados**. Ago. 2017. Diss. (Mestrado) – Ciência da Computação, Universidade Federal de Pernambuco, Recife, PE, Brasil.

HINK, Raymond C Borges et al. Machine learning for power system disturbance and cyber-attack discrimination. In: IEEE. 2014 7th International symposium on resilient control systems (ISRCS). [S. l.: s. n.], 2014. p. 1–8.

HOI, Steven CH et al. Online learning: A comprehensive survey. **Neurocomputing**, Elsevier, v. 459, p. 249–289, 2021.

JACOBSEN, Rune Hylsberg; MIKKELSEN, Søren Aagaard. Infrastructure for intelligent automation services in the smart grid. **Wireless Personal Communications**, Springer, v. 76, n. 2, p. 125–147, 2014.

JOHANSSON, Karl Henrik. The quadruple-tank process: A multivariable laboratory process with an adjustable zero. **IEEE Transactions on control systems technology**, IEEE, v. 8, n. 3, p. 456–465, 2000.

KHAN, Rafiullah et al. Threat analysis of blackenergy malware for synchrophasor based real-time control and monitoring in smart grid. In: 4TH International Symposium for ICS & SCADA Cyber Security Research 2016 4. [S. l.: s. n.], 2016. p. 53–63.

LANGNER, Ralph. Stuxnet: Dissecting a cyberwarfare weapon. **IEEE Security & Privacy**, IEEE, v. 9, n. 3, p. 49–51, 2011.

LE NGUYEN, Minh Huong; GOMES, Heitor Murilo; BIFET, Albert. Semi-supervised Learning over Streaming Data using MOA. In: IEEE. 2019 IEEE International Conference on Big Data (Big Data). [S. l.: s. n.], 2019. p. 553–562.

LEMAY, Antoine; FERNANDEZ, José M. Providing {SCADA} network data sets for intrusion detection research. In: 9TH Workshop on Cyber Security Experimentation and Test (**{**CSET**}** 16). [S. l.: s. n.], 2016.

LIPPMANN, Richard P et al. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In: IEEE. PROCEEDINGS DARPA Information Survivability Conference and Exposition. DISCEX'00. [S. l.: s. n.], 2000. v. 2, p. 12–26.

LU, Yaqi et al. Upper-middleware development of smart energy profile 2.0 for demand-side communications in smart grid. In: IEEE. IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society. [S. l.: s. n.], 2018. p. 306–310.

LUO, Yuanyan; DU, Xuehui; SUN, Yi. Survey on real-time anomaly detection technology for big data streams. In: IEEE. 2018 12th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID). [S. l.: s. n.], 2018. p. 26–30.

MACEDO, Evandro LC et al. On the security aspects of Internet of Things: A systematic literature review. **Journal of Communications and Networks**, IEEE, v. 21, n. 5, p. 444–457, 2019.

MACIEL, Bruno IF; SANTOS, Silas GTC; BARROS, Roberto SM. MOAManager: a tool to support data stream experiments. **Software: Practice and Experience**, Wiley Online Library, v. 50, n. 4, p. 325–334, 2020.

MATER, James; KANG, Steve; SIMPSON, Robby. WHITE PAPER: IEC 61850 and IEEE 2030.5: A Comparison of 2 Key Standards for DER Integration: An Update. **PacWorld**, QualityLogic, 2019.

MELONI, Alessio; ATZORI, Luigi. A cloud-based and restful internet of things platform to foster smart grid technologies integration and re-usability. In: IEEE. 2016 IEEE International Conference on Communications Workshops (ICC). [S. l.: s. n.], 2016. p. 387–392.

MORRIS, Thomas; GAO, Wei. Industrial control system traffic data sets for intrusion detection research. In: SPRINGER. INTERNATIONAL Conference on Critical Infrastructure Protection. [S. l.: s. n.], 2014. p. 65–78.

MORRIS, Thomas H; THORNTON, Zach; TURNIPSEED, Ian. Industrial control system simulation and data logging for intrusion detection system research. **7th annual southeastern cyber security summit**, p. 3–4, 2015.

MOUSTAFA, Nour; SLAY, Jill. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: IEEE. 2015 military communications and information systems conference (MilCIS). [S. l.: s. n.], 2015. p. 1–6.

MUALLEM, Asmah et al. TDDEHT: Threat detection using distributed ensembles of hoeffding trees on streaming cyber datasets. In: IEEE. MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM). [S. l.: s. n.], 2018. p. 1–6.

AL-NASHIF, Youssif et al. Multi-level intrusion detection system (ML-IDS). In: IEEE. 2008 International Conference on Autonomic Computing. [S. l.: s. n.], 2008. p. 131–140.

NAZARI, Ziaeddin; NOFERESTI, Morteza; JALILI, Rasool. DSCA: an inline and adaptive application identification approach in encrypted network traffic. In: PROCEEDINGS of the 3rd International Conference on Cryptography, Security and Privacy. [S. l.: s. n.], 2019. p. 39–43.

NIXON, Christopher; SEDKY, Mohamed; HASSAN, Mohamed. Practical application of machine learning based online intrusion detection to internet of things networks. In: IEEE. 2019 IEEE Global Conference on Internet of Things (GCIoT). [S. l.: s. n.], 2019. p. 1–5.

NOORBEHBAHANI, Fakhroddin et al. An incremental intrusion detection system using a new semi-supervised stream classification method. **International Journal of Communication Systems**, Wiley Online Library, v. 30, n. 4, e3002, 2017.

OBERT, James et al. Recommendations for trust and encryption in DER interoperability standards. In: TECH. Report, Sandia National Laboratories. [S. l.]: SAND2019–1490, 2019.

PAN, Shengyi; MORRIS, Thomas; ADHIKARI, Uttam. Developing a hybrid intrusion detection system using data mining for power systems. **IEEE Transactions on Smart Grid**, IEEE, v. 6, n. 6, p. 3104–3113, 2015.

PARKER, Brandon S; KHAN, Latifur; BIFET, Albert. Incremental ensemble classifier addressing non-stationary fast data streams. In: IEEE. 2014 IEEE International Conference on Data Mining Workshop. [S. l.: s. n.], 2014. p. 716–723.

PORTNOY, Leonid. **Intrusion detection with unlabeled data using clustering**. 2000. Tese (Doutorado) – Columbia University.

PRIYA, S; UTHRA, R Annie. Comprehensive analysis for class imbalance data with concept drift using ensemble based classification. **Journal of Ambient Intelligence and Humanized Computing**, Springer, v. 12, n. 5, p. 4943–4956, 2021.

QUINCOZES, Silvio. **ERENO: A FRAMEWORK FORGENERATING AND PROCESSINGREALISTIC IEC–61850 INTRUSIONDETECTION DATASETS**. 2022. Tese (Doutorado) – Fluminense Federal University.

QUINCOZES, Silvio E; ALBUQUERQUE, Célio et al. A survey on intrusion detection and prevention systems in digital substations. **Computer Networks**, Elsevier, v. 184, p. 107679, 2021.

QUINCOZES, Silvio E; MOSSÉ, Daniel et al. On the Performance of GRASP-Based Feature Selection for CPS Intrusion Detection. **IEEE Transactions on Network and Service Management**, IEEE, 2021.

QUINCOZES, Silvio E; PASSOS, Diego et al. GRASP-based Feature Selection for Intrusion Detection in CPS Perception Layer. In: IEEE. 2020 4th Conference on Cloud and Internet of Things (CIoT). [S. l.: s. n.], 2020. p. 41–48.

QUINCOZES, Silvio E; RANIERY, Carlos et al. Counselors network for intrusion detection. **International Journal of Network Management**, Wiley Online Library, v. 31, n. 3, e2111, 2021.

QUINCOZES, Silvio E; SANTOS, Carlos Raniery Paula dos et al. A Counselors-Based Intrusion Detection Architecture. In: LANOMS. [S. l.: s. n.], 2019.

QUINCOZES, Sílvio Ereno. **Detecção e prevenção de intrusões em subestações elétricas inteligentes**. Proposta de Tese de Doutorado. Niterói, RJ, Brasil, dez. 2019.

RADOGLOU-GRAMMATIKIS, Panagiotis I; SARIGIANNIDIS, Panagiotis G. Securing the smart grid: A comprehensive compilation of intrusion detection and prevention systems. **IEEE Access**, IEEE, v. 7, p. 46595–46620, 2019.

RODOFILE, Nicholas R et al. Process control cyber-attacks and labelled datasets on S7Comm critical infrastructure. In: SPRINGER. AUSTRALASIAN Conference on Information Security and Privacy. [S. l.: s. n.], 2017. p. 452–459.

SCHNEIDER, Peter; BÖTTINGER, Konstantin. High-performance unsupervised anomaly detection for cyber-physical system networks. In: PROCEEDINGS of the 2018 workshop on cyber-physical systems security and privacy. [S. l.: s. n.], 2018. p. 1–12.

SERAPHIM, B Ida; POOVAMMAL, E. Adversarial Attack by Inducing Drift in Streaming Data. **Wireless Personal Communications**, Springer, p. 1–25, 2021.

SETHA, Sugandh; SINGHA, Gurwinder; CHAHALA, Kuljit Kaur. Drift-based approach for evolving data stream classification in Intrusion detection system, 2021.

SHARAFALDIN, Iman; LASHKARI, Arash Habibi; GHORBANI, Ali A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. **ICISSp**, v. 1, p. 108–116, 2018.

SONG, Jungsuk et al. Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation. In: PROCEEDINGS of the first workshop on building analysis datasets and gathering experience returns for security. [S. l.: s. n.], 2011. p. 29–36.

TAORMINA, Riccardo et al. Battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks. **Journal of Water Resources Planning and Management**, American Society of Civil Engineers, v. 144, n. 8, p. 04018048, 2018.

TAVALLAEE, Mahbod et al. A detailed analysis of the KDD CUP 99 data set. In: IEEE. 2009 IEEE symposium on computational intelligence for security and defense applications. [S. l.: s. n.], 2009. p. 1–6.

TEIXEIRA, André et al. Attack models and scenarios for networked control systems. In: PROCEEDINGS of the 1st international conference on High Confidence Networked Systems. [S. l.: s. n.], 2012. p. 55–64.

USTUN, Taha Selim; FAROOQ, Shaik Mullapathi; HUSSAIN, SM Suhail. A novel approach for mitigation of replay and masquerade attacks in smartgrids using IEC 61850 standard. **IEEE Access**, IEEE, v. 7, p. 156044–156053, 2019.

VEIT, Maxime Fabian. **ICS protocol dissectors for signature-based NIDS**. 2021. Diss. (Mestrado) – Karlsruhe Institute od Technology.

WISESA, Hanif Arief et al. Processing big data with decision trees: A case study in large traffic data. In: IEEE. 2016 International Workshop on Big Data and Information Security (IWBIS). [S. l.: s. n.], 2016. p. 115–120.

WITTEN, Ian H; FRANK, Eibe. Data mining: practical machine learning tools and techniques with Java implementations. **Acm Sigmod Record**, ACM New York, NY, USA, v. 31, n. 1, p. 76–77, 2002.

WOŹNIAK, Michał et al. Active learning classification of drifted streaming data. **Procedia Computer Science**, Elsevier, v. 80, p. 1724–1733, 2016.

YUSTA, Silvia Casado. Different metaheuristic strategies to solve the feature selection problem. **Pattern Recognition Letters**, Elsevier, v. 30, n. 5, p. 525–534, 2009.

ZANELLA, Andrea et al. Internet of things for smart cities. **IEEE Internet of Things journal**, IEEE, v. 1, n. 1, p. 22–32, 2014.

ZHANG, Yue et al. Cyber physical security analytics for transactive energy systems. **IEEE Transactions on Smart Grid**, IEEE, v. 11, n. 2, p. 931–941, 2019.

APÊNDICE A

A.1 Tabelas da Matriz Confusão dos *frameworks* WEKA e MOA

Neste apêndice estão a Matriz Confusão em WEKA e MOA de cada *dataset*, na sequência SWaT, BATADAL, ERENO, Morris-1, Morris-3 Gas, Morris-3 Water e Morris-4. As cores que azul mostram os valores da Matriz Confusão com o melhor classificador do WEKA, enquanto as cores em vermelho mostram os valores da Matriz Confusão com o melhor do MOA. As Tabelas de 9 à 15 foram utilizadas para gerar os gráficos das subseções 4.3.1 e 4.3.2. As duas últimas Tabelas 16 e 17 se referem a matriz confusão utilizadas na confecção das Figuras 17 à 20 do experimento da subseção 4.3.5.

Tabela 9: Matriz confusão do *dataset* SWaT com os classificadores *Random Forest* (RF) do WEKA e *Hoeffding Adaptive Tree* (HAT) do MOA.

		Predito	
		Ataque	Normal
Real	Ataque	54200 53252	204 1369
	Normal	40 1914	892274 890184

Tabela 10: Matriz confusão do *dataset* BATADAL com os classificadores *REPTree* (ReT) do WEKA e *Hoeffding Adaptive Tree* (HAT) do MOA.

		Predito	
		Ataque	Normal
Real	Ataque	100 38	90 181
	Normal	16 20	12730 12699

Tabela 11: Matriz confusão do *dataset* ERENO com os classificadores *REPTree (ReT)* do WEKA e *Hoeffding Adaptive Tree* (HAT) do MOA.

		Predito	
		Ataque	Normal
Real	Ataque	400852 396268	166 554
	Normal	0 516	5510278 5514048

Tabela 12: Matriz confusão do *dataset* Morris-1 com os classificadores *Random Forest* (RF) do WEKA e *Hoeffding Adaptive Tree* (HAT) do MOA.

		Predito	
		Ataque	Normal
Real	Ataque	53954 47675	1632 7988
	Normal	6026 6254	16762 16460

Tabela 13: Matriz confusão do *dataset* Morris-3 gas com os classificadores J48 do WEKA e *Hoeffding Adaptive Tree* (HAT) do MOA.

		Predito	
		Ataque	Normal
Real	Ataque	35394 35034	406 829
	Normal	316 246	60902 60910

Tabela 14: Matriz confusão do *dataset* Morris-3 water com os classificadores *REPTree* (ReT) do WEKA e *Hoeffding Adaptive Tree* (HAT) do MOA.

		Predito	
		Ataque	Normal
Real	Ataque	63370 61650	786 2114
	Normal	1278 2008	170744 170407

Tabela 15: Matriz confusão do *dataset* Morris-4 *Random Tree* (RaT) do WEKA e *Hoeffding Adaptive Tree* (HAT) do MOA.

		Predito	
		Ataque	Normal
Real	Ataque	56624 17779	3442 42269
	Normal	2220 3149	212340 211431

Tabela 16: Métricas do *dataset* ERENO com o classificador *REPTree* em diferentes sequências de Casos de Usos de Treino e Teste no *framework* WEKA.

WEKA REPTree	**VP**	**VN**	**FP**	**FN**
UC1	77998	742794	0	2
UC1/UC2	29061	876844	180956	27291
UC1-2/UC3	0	742794	0	34487
UC1-3/UC4	0	742794	0	34839
UC1-4/UC5	94	742794	0	77906
UC1-5/UC6	77944	742794	0	56
UC1-6/UC7	37144	740383	2411	0

Tabela 17: Matriz Confusão de Multiclasse do *dataset* ERENO no *framework* MOA

MOA HAT ERENO All	**Normal**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Normal	5514048	70	202	48	37	114	38	7
1	115	77885	0	0	0	0	0	0
2	200	0	56152	0	0	0	0	0
3	78	0	0	34409	0	0	0	0
4	43	0	0	0	34796	0	0	0
5	53	0	0	0	0	77947	1	0
6	44	0	0	0	0	0	77955	0
7	20	0	0	0	0	0	0	37124

APÊNDICE B

B.1 Importância de Sistemas de Detecção de Intrusão para padrões de rede *Smart Grid*

Há dois padrões de rede SG em desenvolvimento no mercado em destaque, o padrão IEC 61850, do *International Electrotechnical Commission*, e o IEEE 2030.5, do Instituto de Engenheiros Eletricistas e Eletrônicos. Os dois padrões fazem uso de diversos protocolos de rede, desde os largamente utilizados na Internet a até protocolos específicos de controle. Toda esta variedade de protocolos abre diversas brechas de segurança no padrão, que podem ser mitigadas com a utilização de um IDS que faça uso de regras para identificar os ataques.

Apesar dos padrões terem funcionalidades semelhantes, há poucas referências que façam a comparação entre os dois. Um dos poucos que podemos citar é o artigo Mater, Kang e Simpson (2019) que busca descrever um resumo dos dois padrões, desde a suas origens e evolução até a fase atual de desenvolvimento, no final são descritas as suas implementações quanto a regulamentação do estado da Califórnia, EUA, conhecida como CA Rule 21. Nenhuma comparação ou análise técnica é feita entre os padrões, o que demonstra que são duas iniciativas distintas de padronização de uma nova tecnologia que vem surgindo, um de iniciativa do mercado Norte-Americano (IEEE 2030.5) e outro do mercado Europeu (IEC 61850).

B.1.1 Padrão IEEE 2030.5

O padrão IEEE 2030.5 foi definido para Rede de Área Doméstica, do inglês, *Home Area Network* (HAN) para equipamentos IED em uma rede domiciliar para comunicação entre empresas fornecedoras de serviço e usuários, que funciona com o protocolo SEP 2.0 (*Smart Energy Profile 2.0*) em uma arquitetura RESTful (*Representational State Transfer*) que utiliza HTTP (*Hyper Transfer Protocol*) sobre TCP/IP, com troca de mensagens XML,

utilizando TLS 1.3 (*Transport Layer Security* versão 1.3). O padrão contém perfis de aplicação compatíveis com outros padrões como IEC 61968 e IEC 61850.

Apesar do padrão IEEE 2030.5 ser de fácil implementação e possuir interoperabilidade com diversos protocolos, conforme pode ser visto nos experimentos práticos realizados por Ghalib et al. (2018), Fattahi et al. (2020) e Lu et al. (2018), poucas pesquisas fazem análises quanto à segurança do padrão, conforme feito por Assis (2021) e Obert et al. (2019). Estes trabalhos apontam uma falha de implementação na geração dos certificados (*Public Key Infrastructure*) com validade indeterminada. Uma vez gerado o certificado para um equipamento, ele será válido por tempo indeterminado, logo se a chave privada for comprometida o seu certificado não poderá ser revogado, o que abre brecha para diversos ataques.

B.1.2 Padrão IEC 61850

O padrão IEC 61850 foi desenvolvido com o objetivo de prover interoperabilidade, estabilidade no longo prazo e configuração simplificada. A especificação define três camadas: topologia física, protocolos de rede e modelagem de objetos (QUINCOZES; ALBUQUERQUE et al., 2021).

A topologia física do padrão possui três níveis de infraestrutura, que são: **Estação**, interface humana para gerenciamento de subestações, como sistema SCADA ou *Remote Terminal Unit* (RTU); **Compartimento**, nível intermediário com funções automáticas em tempo-real que não necessitam de intervenção humana, como controle e medição, proteção e sincronização de equipamentos IED; **Processo**, equipamentos convencionais do domínio elétrico com interfaces entre os domínios físico e cibernético, como unidade de medições e terminais inteligentes.

Os protocolos de rede possuem dois tipos de canais de comunicação: horizontal, que ocorre entre equipamentos de mesmo nível; e vertical, que ocorre entre dispositivos de diferentes níveis. Além dos protocolos tradicionais, como FTP (*File Transfer Protocol*) e HTTP, o padrão introduz dois novos protocolos: GOOSE (*Generic Object Oriented Substation Events*) para comunicações intra-IED e *Sampled Value* (SV) para comunicações entre Unidade de União (*Merging Units - MU*) e IED, além do protocolo MMS (*Manufacturing Message Specification* - ISO 9506).

O protocolo SV permite que correntes e tensões digitalizadas sejam transmitidas para os IEDs utilizando cabos Ethernet a uma taxa alta de transmissão. O protocolo GOOSE

é utilizado para troca de mensagens de eventos entre subestações, como notificações, alarmes e comandos de controle, que são encapsuladas em quadros Ethernet. As mensagens são transmitidas periodicamente em intervalos fixos de tempo em situações normais de operação, caso ocorra algum evento, o intervalo de tempo é reduzido, a fim de melhorar o tempo de resposta ao evento. O protocolo MMS é executado no topo da pilha TCP/IP ou modelo OSI e faz a comunicação entre estações e IED por meio Ethernet. O protocolo é dividido em perfil de aplicação (*A-Profile*) e perfil de transporte (*T-Profile*), que se referem aos níveis 3 superior e 4 inferior do modelo OSI, respectivamente. Pode-se notar que o protocolo não tem nenhuma técnica de segurança, porém, por ser um protocolo mais completo que o GOOSE e SV, ele pode suportar diversos mecanismos extras de segurança. Outros protocolos legados podem ser utilizados no padrão como MODBUS (protocolo de comunicação serial desenvolvido e publicado pela Modicon em 1979 para utilização em PLC) e DNP3 (*Distributed Network Protocol* versão 3 para comunicações entre SCADA e IED, por exemplo), além dos protocolos PTP (*Precision Time Protocol*) e LLDP (*Link Layer Discovery Protocol*) para sincronização entre dispositivos IED.

Quanto à segurança deste padrão, os ataques de repetição e de disfarce são apontados como os mais preocupantes devido ao seu impacto iminente na operação das mensagens, GOOSE e SV, e a restrições de tempo de resposta de 4 ms do padrão (USTUN; FAROOQ; HUSSAIN, 2019). Apesar de não abordar soluções de segurança de IDS, o artigo apresenta uma primeira linha de defesa com a utilização de algoritmos de assinatura digital (*Digital Signature Algorithm* - DSA) com autenticação de chaves assimétricas RSA (*Rivest-Shamir-Adleman*) ou ECDSA (*Elliptic Curve Digital Signature Algorithm*). A proposta de defesa apresentada mostra que o tempo de processamento da estrutura alcança o máximo de atraso, com a assinatura ECDSA, de 1,2 ms e para verificação das mensagens de 1,5 ms, o que representa um tempo restante de 2,5 ms para a operação de detecção ou de prevenção de ataques.

Uma busca na literatura aponta que a maioria das ações de detecção de IDS desenvolvidos para os dois padrões são baseadas em especificações (QUINCOZES; ALBUQUERQUE et al., 2021), sendo os baseadas em assinaturas os menos explorados na literatura. Há poucas propostas de trabalhos que utilizem técnicas de análise em tempo-real para detectar intrusões no padrão com restrição de tempo descrito acima, o que é a proposta desta dissertação.

APÊNDICE C

C.1 *Datasets* não apropriados para *Cyber-Physical System*

Iremos agora descrever alguns dos principais *datasets* largamente utilizados em sistemas de detecção de intrusão estudados durante o período do mestrado, que são o KDD Cup, CIC-IDS, UNSW-NB15 e WSN-DS. Estes *datasets* foram escolhidos por possuírem uma grande quantidade de referências bibliográficas.

O KDD CUP 99[1] é muito utilizado para avaliação de IDS baseados em assinatura. Desde 1999 este *dataset* tem sido uma importante referência para comparação entre diversas pesquisas na área. Ele foi construído a partir do programa de avaliação DARPA IDS 98 (LIPPMANN et al., 2000). O tráfego de fundo do *dataset* foi gerado com a simulação de 100 usuários durante dois meses com 38 tipos de ataques diferentes. No total foram gerados sete semanas de treinamento com ataques rotulados e duas semanas de ataques não rotulados gerados para a execução de testes de detecção de intrusão. Desta forma, o *dataset* de treinamento do KDD ficou com aproximadamente 4.900.000 amostras com 41 atributos classificados em normal ou ataque.

Os atributos são agrupados em três grupos: recursos básicos, que são atributos extraídos das conexões TCP/IP; recursos de tráfego, que são atributos coletados em uma janela de intervalo de tempo divididos em grupos de atributos do mesmo *host* (examina conexões nos últimos 2 segundos do mesmo *host* da conexão atual) e atributos do mesmo serviço (examina conexões nos últimos 2 segundos do mesmo serviço da conexão atual); e recursos de conteúdo, que são atributos de conteúdo das conexões que indicam ataques nos níveis de aplicação.

Os ataques simulados são provenientes de quatro categorias: DoS (*Denial of Service*), ataque de negação de serviço; *User to Root Attack* (U2R), ataque de uma conta de usuários

[1]KDD Cup 1999. Site: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, setembro 2021.

normais que tentam ganhar acesso da conta raiz; *Remote to Local Attack* (R2L), ataque remoto pela rede que tenta ganhar acesso local a máquina; *Probing Attack*, tentativa de coletar informações sobre a rede dos computadores na tentativa de encontrar falhas de segurança. Propositalmente o *dataset* de teste não possui a mesma distribuição de probabilidade do *dataset* de treinamento, isso se deve a ausência de 14 ataques na fase de treinamento.

O *dataset* KDD CUP 99 apresenta alguns problemas como: grande número de dados redundantes de treinamento e de teste que levam a resultados tendenciosos; e a falta de uma representatividade dos dados de uma rede real. Posteriormente os dados redundantes foram removidos e o novo *dataset* passou a ser conhecido como NSL-KDD (TAVALLAEE et al., 2009).

Uma das dificuldades na área de detecção de intrusão é a disponibilidade de *datasets* que contenham cenários atuais de um tráfego de rede. Muitos destes dados, como por exemplo, o KDD 98, KDD CUP 99 e NSL-KDD são utilizados como referência em diversas pesquisas atuais, porém foram originados a mais de 10 anos atrás, que por mais que tenham sido atualizados, já não refletem o fluxo atual das redes de dados.

O CICIDS 2017[2] é um *dataset* gerado de maneira confiável, com tráfegos normais e características reais, que contém sete tipos atuais de ataques: DoS, *Distributed* DoS, *Brute Force*, XSS, SQL *Injection*, *Infiltration*, *Port scan* e Botnet. O *dataset* foi extraído de uma interface de pacotes capturados (PCAP) e analisado por um analisador de tráfego de rede CICFlowMeter[3] com os rótulos de classificação baseados em informações de tempo, fonte, IP de destino, portas de origem e destino, protocolos e ataques (SHARAFALDIN; LASHKARI; GHORBANI, 2018).

Os arquivos estão no formato CSV (*Comma Separated Values*) e foram construídos com comportamentos artificiais de 25 usuários baseados em HTTPS, FTP, SSH e protocolos de email. Para construir o *dataset* de forma confiável e atual foram considerados os 11 critérios propostos por Gharib et al. (2016), que foram atendidos pelas seguintes abordagens: (1) configuração de rede completa com equipamentos de modem, *firewall*, *switches*, roteadores e presença de diversos SO; (2) tráfego completo com presença de máquinas virtuais de vítimas e atacantes; (3) ataques rotulados para confiabilidade dos dados; (4) interação completa com rede LAN e conexão com a internet; (5) captura completa com ferramentas apropriadas e servidores de armazenamento; (6) variedade de protocolos dis-

[2]Site:https://www.unb.ca/cic/datasets/ids-2017.html, setembro de 2021.
[3]Site: https://github.com/ahlashkari/CICFlowMeter, setembro de 2021.

poníveis, como protocolos tradicionais (HTTP e FTP) e protocolos com interações em tempo real (VOIP); (7) diversidade de ataques baseados em relatórios confiáveis e recentes[4]; (8) anonimato com a remoção da carga útil do pacote IP; (9) heterogeneidade na coleta de dados para execução completa de testes; (10) conjunto de atributos retirados CICFlowMeter; (11) metadados com informações completas no *dataset*.

Para comprovar a qualidade deste *dataset*, no artigo Sharafaldin, Lashkari e Ghorbani (2018) são extraídas métricas de avaliação (*Precision*, *Recall* e F1*Score*) a partir de sete algoritmos de aprendizado de máquina, que são: *K-Nearest Neighbors* (KNN), *RadomForest* (RF), ID3, *Adaboost*, *Multilayer Perceptron* (MLP), *Naive-Bayes* (NB) e *Quadratic Disceptron* (QDA). Estas métricas servem como referência para trabalhos futuros que venham a utilizar o mesmo *dataset*. No final do trabalho é apontado que o CIC-IDS possui todos os critérios de avaliação que muitos outros *datasets* não cumprem.

Uma versão mais atualizada deste *dataset* pode ser encontrada no site da Universidade de Nova Brunswick (UNB), conhecido pelo nome CSE-CIC-IDS2018[5], elaborado em conjunto com o Instituto Canadense de Cibersegurança, do inglês, *Canadian Institute for Cybersecurity* (CIC) e o Estabelecimento de Segurança de Comunicação, do inglês, *Communications Security Establishment* (CSE).

Outro *dataset* novo muito utilizado no meio acadêmico é o UNSW-NB15, ele foi apresentado pelo trabalho de Moustafa e Slay (2015). O *dataset* apresenta os novos ataques que são atualizados a partir do site Exposições e Vulnerabilidades Comuns, do inglês, *Common Vulnerability and Exposures* (CVE)[6]. Desta forma, o UNSW-NB15 é um dataset híbrido com dados modernos reais e ataques sintetizados de tráfego de rede criado no laboratório de cibersegurança da Universidade de Nova Gales do Sul (*University of New South Wales* - UNSW), Austrália (MOUSTAFA; SLAY, 2015). O *dataset* possui 49 atributos armazenados em um servidor SQL 2008 extraídos a partir da ferramenta *tcpdump*[7], presente nos roteadores, para captura de arquivos PCAP, que são coletados a partir de 12 algoritmos próprios desenvolvidos em C# para coleta de pacotes e softwares de monitoramento de rede Argus[8] e Bro-IDS[9]. A ferramenta IXIA PerfectStorm[10] foi

[4]Site: https://www.mcafee.com/enterprise/pt-br/threat-center/mcafee-labs/reports.html, setembro de 2021.
[5]Site: https://www.unb.ca/cic/datasets/ids-2018.html, janeiro de 2022
[6]Site: https://cve.mitre.org/, setembro de 2021.
[7]Site: http://www.tcpdump.org/, setembro de 2021.
[8]Site: http://argus.tcp4me.com/, setembro de 2021.
[9]Atualmente conhecido como Zeek, site: https://zeek.org/, setembro de 2021.
[10]Site: https://www.keysight.com/br/pt/products/network-test/network-test-hardware/perfectstorm.html, setembro de 2021.

utilizada para a geração de tráfego de rede normal e geração de tráfego anormal em um ambiente sintético. Os dados são classificados a partir de uma tabela verdade gerada da ferramenta de geração de tráfego.

O artigo no final faz uma comparação entre o UNSW-NB15 com o KDD CUP 99, onde é apontado que o *dataset* possui 3 redes distintas, ao invés de 2 redes do KDD, 45 endereços IP distintos, comparado com 11 no KDD, e possui 9 tipos de ataque, onde o KDD possui 4 tipos principais. Desta forma, os ataques do novo dataset representam uma situação mais real das redes de dados existentes atualmente.

C.1.1 Dataset para Redes Sem Fio

As Redes de Sensores Sem Fio, do inglês *Wireless Sensor Network* (WSN), estão cada vez mais empregadas em aplicações em tempo real, como aplicações militares de vigilância, monitoramento de incêndios em florestas, cuidados de saúde e monitoramento de segurança de edifícios. A criptografia é um bom mecanismo de defesa que pode ser utilizado para proteger WSN contra ataques externos, porém o mesmo não pode ser garantido de ataques internos onde as chaves de segurança estão expostas ao atacante. Desta forma, para prover a segurança nessas redes é necessário utilizar um IDS, o que leva a diversos desafios na área devido aos recursos limitados dos dispositivos quanto a energia das baterias, memória e capacidades de processamento. Outro desafio é a alta vulnerabilidade desses sistemas devido a sua natureza distribuída e a natureza dos canais de eletromagnéticos a interferências que podem gerar grandes perdas de pacotes em instantes aleatórios.

Os *datasets* descritos até aqui não podem ser empregados de maneira genérica nas mais diversas redes. As WSN são uma dessas redes, onde a arquitetura distribuída e aberta levam a um comportamento diferente dos dados transmitidos de redes comuns. Por esta razão, o *dataset* (DS) WSN-DS (ALMOMANI; AL-KASASBEH; AL-AKHRAS, 2016) se mostra importante para o desenvolvimento de um IDS ou IDPS que funcione de forma eficiente e com custos adequados de processamento e energia em um equipamento WSN. Para alcançar este objetivo, o *dataset* construído possui quatro tipos de ataques: *Blackhole*, *Grayhole*, *Flooding* e *Scheduling attack*. Ele é formado de 374.661 instâncias contendo tráfego normal e malicioso com a assinatura destes quatro ataques com 23 atributos extraídos. Os dados foram coletados com o simulador NS-2 com uso do protocolo LEACH (*Low Energy Aware Cluster Hierarchy*), protocolo de roteamento mais utilizado em redes WSN.

No trabalho de Almomani, Al-Kasasbeh e Al-Akhras (2016) o *dataset* WSN-DS foi

validado através do modelo de classificação *Artificial Neural Network* (ANN) onde foi possível alcançar a acurácia de 92,8%, 99,4%, 92,2%, 75,6% e 99,8% para os ataques de *Blackhole*, *Flooding*, *Scheduling* e *Grayhole*, respectivamente. Infelizmente a validação se mostra incompleta por não incluir ataques diferentes nos dados de treinamento e de teste, pois a maior dificuldade dos classificadores se refere a detecção de novos ataques. Apesar de ser um importante *dataset* para a área de redes sem fio, há poucas citações a este trabalho, 96 referências, visto em setembro de 2021.

Assim como WSN, a adoção de dispositivos IoT tem levado a novos desafios aos sistemas IDS. Devido em parte também às restrições de hardware, o uso destes aparelhos levam a um aumento nos riscos de segurança, o que ficou evidente principalmente após o ataque malware Mirai 2016[11], que é constituído de ataques DDoS em larga escala explorando limitações dos dispositivos IoT.

A disponibilidade de *datasets* específicos para IoT também tem se mostrado escassa e limitada (AUNG et al., 2020), por este motivo foi elaborado o SUTD-IoT DS 2020[12]. Desenvolvido pela Universidade de Singapura de Tecnologia e Projetos, o *dataset* foi construído com uma variedade de dados de *honeypot*[13] durante 18 meses (2017-2018) com 40 endereços IPs públicos para comunicação com 11 dispositivos reais de IoT. Os arquivos foram gerados em formato JSON de mais de 250 mil arquivos capturados via interface PCAP que resultaram em mais de 81 milhões de registros. Os dados foram capturados com a utilização da ferramenta analisadora de pacotes de rede TShark[14] e Bro-IDS (Zeek) para gerar o dataset. A solução com *honeypot* foi proposta por fornecer dados confiáveis e reais referentes a ataques cibernéticos de forma a não comprometer a identidade e privacidade dos atores envolvidos.

Apesar do artigo Aung et al. (2020) descrever a geração de um *honeypot* para IoT, que é um tema que vem crescendo nos últimos anos, os autores não analisam a utilização do *dataset* em um IDS com classificadores de forma a validar a sua qualidade.

[11] Site: https://en.wikipedia.org/wiki/Mirai_(malware), setembro de 2021.
[12] Site: https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/, setembro de 2021.
[13] *Honeypot* é um mecanismo de segurança de armadilha virtual criado para iludir atacantes, através do uso intencional de sistemas de computadores com segurança danificada de forma a permitir que os atacantes explorem vulnerabilidades para que elas sejam estudadas para melhorar as políticas de segurança.
[14] Site: https://www.wireshark.org/docs/man-pages/tshark.html, setembro de 2021.

C.2 *Datasets* de *Cyber-Physical System* não encontrados

Os *datasets*, descritos aqui da Tabela 2, são referentes a CPS, porém não foram possíveis de encontra metadados que descrevessem a sua utilização imediata pelos *frameworks* empregados nos experimentos dessa dissertação.

A partir do *dataset* SWaT pelo artigo Goh et al. (2016), foi possível descobrir diversos trabalhos que fazem menção a *datasets* de CPS, porém poucos estão disponíveis para o meio acadêmico. Como exemplo, temos na área de controle de sistemas ciber-físicos e de controle de plantas industriais os trabalhos de Teixeira et al. (2012) e Harada et al. (2017), onde os autores tentam identificar anomalias nas operações de tanques aquáticos.

Em Teixeira et al. (2012) foi feita uma análise detalhada sobre os principais ataques na área de controle e automação de um sistema SCADA. Os ataques são descritos por modelos matemáticos de controle que são utilizados para criar uma estrutura de análise e detecção. No final, um experimento foi realizado sobre o controle de multivariáveis de um processo de tratamento de tanques quádruplos (JOHANSSON, 2000), onde ocorreram ataques de repetição, dinâmica zero e injeção de tendência. Nenhuma métrica é fornecida de forma a demonstrar a acurácia e precisão da estrutura proposta em identificar os ataques. O trabalho é muito bem citado, com 1.127 referências, visto em setembro de 2021, porém não foi localizado nenhum *dataset*, pois o experimento foi executado diretamente no sistema de tratamento de tanques, que se trata de um processo laboratorial multivariável na área de controle, que foi construído especificamente para este trabalho.

No artigo de Harada et al. (2017) foi descrito um método de detecção de anomalias em CPS utilizando métodos estatísticos em um sistema de gerenciamento automático de tanque, chamado de Aqua-tan[15]. Devido às características imprevisíveis do ambiente do aquário, como temperatura, umidade e nível de água, há uma falta de um modelo preciso para operação. Por isso, foram adotadas técnicas de controle largamente utilizadas conhecidas como Fator Discrepante Local, do inglês *Local Outlier Factor* (LOF), para identificação de falhas com a ferramenta ELKI[16] (*Environment for Developing KDD-Applications Supported by Index-Structures*). No final, o método proposto foi capaz de detectar muitos eventos de interesse, como falhas de funcionalidades e o reinício inesperado do sistema. Apesar do *dataset* ser disponibilizado online, há a necessidade de extração e tratamento da sua base de dados MySQL para serem processados por um IDS. Cabe enfatizar que nenhuma métrica foi extraída das anomalias quanto a precisão ou acurácia

[15]Site: https://se.is.kit.ac.jp/aquarium/, setembro de 2021.
[16]Site: https://elki-project.github.io/, setembro de 2021.

da detecção.

O *dataset* Morris-2 foi descrito no artigo Beaver et al. (2013), os dados foram retirados do fluxo de telemetria do Unidade de Terminal Remoto, do inglês, Remote Terminal Unit (RTU), durante um experimento de uma rede de gasoduto. O fluxo de telemetria contém exemplos de fluxos normais de dados e comandos e dados de ataque de injeção. O fluxo de telemetria contém pares de comandos e respostas, onde os comandos são usados para definir valores dos PLC, que controlam a pressão nos dutos. Muitos comandos se referem apenas à leitura da pressão atual em pontos determinados. Cada amostra do fluxo de telemetria contém o endereço do PLC e o código de função (*Function Code* - FC) que define o tipo de comando e resposta. Este *dataset* foi desconsiderado por não haver metadados que descrevessem os seus diversos arquivos CSV gerados, que são oito no total.

O *dataset* Lemay, está descrito no trabalho de Lemay e Fernandez (2016), ele contém pacotes capturados de tráfego MODBUS normais e maliciosos com rótulos em arquivo CSV, o que o torna muito útil para métodos de aprendizado de máquina supervisionados. Para gerar este *dataset*, os dados foram gerados em um *sandbox*[17] SCADA onde simuladores de rede elétricas foram utilizados para introduzir realismo ao componentes físicos. Porém, a ausência de metadados sobre este *dataset* torna difícil o seu pronto uso nas ferramentas que serão utilizadas nessa dissertação de mestrado (WEKA e MOA), pois só foi possível localizar os arquivos PCAP.

O *dataset* Rodofile, descrito no trabalho Rodofile et al. (2017), utiliza um simulador de uma planta de refinaria de mina. O processo principal consiste em três subprocessos executados em sequência: Transporte, Tanque de Lavagem e Canalização do Reator. A rede é constituída de três PLC escravos, cada um responsável por um dos subprocessos e é controlado por um dispositivo mestre. Todos os dispositivos são conectados por um *switch* de gerenciamento, onde o Sistema Global de Posicionamento (GPS), o monitor HMI e o atacante estão também conectados. Infelizmente, a ausência de metadados sobre este *dataset* torna inviável o seu pronto uso nas ferramentas WEKA e MOA, que são utilizados nessa dissertação, pois precisarem de arquivos ARFF ou CSV, e só foi possível localizar dados PCAP. Trabalhos futuros sobre IDS podem ser feitos em cima deste *dataset*, conforme pode ser visto nas referências Veit (2021) e Fisher (2019).

Não foi possível encontrar referências de artigos que tratassem especificamente dos se-

[17] *Sandbox* é uma plataforma de testes onde as aplicações podem ser alteradas sem interferir no meio de produção. Nela, os desenvolvedores podem executar todas as operações de mudanças experimentais que vão garantir o bom funcionamento da solução, evitando danos que possam prejudicar o sistema.

guintes *datasets*, apenas os repositórios com metadados insuficientes: Morris-5[18], 4SICS[19], S4x15CTF[20] e DEFCON23[21]. Esta limitação não impede o uso destes dados para o uso acadêmico, porém são dados a serem tratados em pesquisas futuras pois necessitam de uma série de tratativas afim de organiza os *datasets* de forma mais completa em arquivo único CSV para uso adequado em um *software* de mineração de dados. Uma descrição comparativa destes *datasets* pode ser encontrado em Choi, Yun e Kim (2018).

[18]Site: https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets, janeiro de 2022.
[19]Site: https://www.netresec.com/?page=PCAP4SICS, janeiro de 2022.
[20]Site: https://www.netresec.com/?page=DigitalBond_S4, jeniro de 2022.
[21]Site: https://media.defcon.org/DEF%20CON%2023/DEF%20CON%2023%20villages/DEF%20CON%2023%20ics%20village/, janeiro de 2022.

APÊNDICE D

D.1 Revisão Sistemática da Literatura sobre Aprendizado de Máquina *On-line*

Com o objetivo de fazer uma Revisão Sistêmica da Literatura (RSL), serão descritos os principais trabalhos na área de Aprendizado de Máquina *On-line* nos últimos 12 anos. Os tópicos foram definidos em áreas de modo a possibilitar a identificação das principais tendências. A RSL é baseada na escolha e análise de uma série de artigos científicos, de acordo com um protocolo bem definido, para que possam ser reproduzidos, verificados e auditáveis (MACEDO et al., 2019).

As etapas do RSL são divididas em:

- **Planejamento**, definição das perguntas de busca e desenvolvimento do protocolo de busca. Nesta etapa é identificada as necessidades do estudo, o que requer elaborar perguntas referentes à pesquisa.

- **Revisão da Literatura**, identificação de estudos relevantes, seleção de estudos primários, acesso à qualidade dos estudos, extração dos dados e sintetização das informações. Esta etapa consiste na definição da base de pesquisa, *string* de busca, critérios de inclusão e exclusão e critério de qualidade.

  Para a inclusão e exclusão de artigos é necessário estabelecer regras para seleção de estudos relevantes, como o tema abordado, o ano, a língua, o número de referências de um artigo. Cada filtro ou critério de inclusão e exclusão dos artigos deve ser documentado de maneira a possibilitar a reprodutibilidade.

  O critério de qualidade dos artigos leva em consideração a clareza quanto aos objetivos apresentados, apresentação de arquitetura, algoritmos ou protocolos viáveis e a apresentação dos resultados conforme o objetivo proposto. Desta forma, o objetivo desta fase é identificar as fontes de pesquisa, selecionar artigos, avaliar a qualidade e

extrair dados e síntese, para encontrar estudos primários que possam ser úteis para responder às perguntas iniciais da fase de planejamento.

- **Documentação**, relatório de revisão e validação. Nesta fase será feita a documentação com a descrição de todo o processo da RSL. O objetivo final é ter uma visão geral sobre o assunto e identificar o estado da arte na área de pesquisa.

Desta forma, o interesse é pesquisar artigos relacionados à área de Detecção de Intrusão que levam em consideração o *framework* MOA para o desenvolvimento de testes e provas de conceito de um grande fluxo de dados online. Para alcançar esse objetivo, as seguintes questões de pesquisa foram propostas:

- Qual a melhor técnica de IDS para dispositivos CPS que levem em consideração o limite de recursos e o grande fluxo de dados do sistema?
- Quais são as diferenças de performance de uma IDS offline para um IDS online que justifique a utilização de diferentes técnicas?

Para responder a estas perguntas foram utilizadas as bases de conteúdo acadêmico do *Google Scholar*, *Scopus* e *IEEE Xplore*, com as seguintes strings detalhadas na Tabela 18 abaixo:

Tabela 18: Número de trabalhos retornados na pesquisa da Revisão Sistemática da Literatura.

Base de busca	**String de busca**	**Trabalhos relacionados**
Google Scholar	"MOA: Massive Online Analysis"	1761
Scopus	"MOA: Massive Online Analysis"	44
IEEE Xplorer	"MOA: Massive Online Analysis"	15

As diferentes bases apresentaram resultados discrepantes, principalmente a base Google Scholar. Com objetivo de ter mais equidade nos resultados e garantir a reprodutibilidade, foram definidos filtros de seleção e exclusão como Critério de Exclusão (CE), conforme a Tabela 19.

Tabela 19: Filtros de critério de exclusão da Revisão Sistemática da Literatura.

Descrição	**Filtro**
O estudo não é em português ou inglês e foi escrito antes de 2010.	CE1
O estudo não é sobre a área de segurança de IDS.	CE2
O estudo não descreve o tema e faz apenas referência.	CE3
O estudo não descreve técnicas de aprendizado supervisionados.	CE4

O primeiro filtro utilizado (CE1) é quanto a data de publicação dos trabalhos e a linguagem das publicações português e inglês. A publicação inicial sobre a apresentação do *framework* MOA é de 2010, logo foi selecionado um filtro a fim de selecionar trabalhos a partir desse ano.

O segundo filtro aplicado (CE2) é referente ao uso do MOA com técnicas de detecção de intrusão voltados apenas para área de segurança. Este filtro se deve ao fato de que muitos trabalhos se referem a descrição de novos métodos de mineração de dados porém não tratam sobre segurança e não mencionam formas de detecção de intrusão. Cabe enfatizar aqui que o método utilizado, infelizmente, exclui bons trabalhos relacionados com mineração de fluxo de dados que não utilizam o MOA. Porém, como o objetivo desta dissertação é fazer testes práticos que possam ser recriados facilmente por este *framework*, não haverá prejuízo na RSL ao se aplicar este filtro. A Tabela 20 mostra os resultados após a aplicação dos dois primeiros filtros CE1 e CE2 aplicados até aqui automaticamente nas ferramentas das bases de busca.

O terceiro filtro (CE3) tem por objetivo eliminar redundância dos trabalhos entre cada base de busca e eliminar trabalhos que apenas mencionam ou referenciam o MOA, onde citamos aqui três exemplos como demonstração. No primeiro exemplo, temos o trabalho de Radoglou-Grammatikis e Sarigiannidis (2019) que faz referência ao MOA apenas para apresentar uma nova linha de um Sistema de Prevenção e Detecção de Intrusão (do inglês *Intrusion Detection and Prevention System* - IDPS). O foco do artigo está apenas em segurança para redes SG sem descrever técnicas de Aprendizado de Máquina. Outro exemplo é o trabalho de Nazari, Noferesti e Jalili (2019), ele faz a identificação do fluxo de dados de uma rede criptografada com VPN e Tor, logo são abordas técnicas de mineração de fluxo de dados com o MOA sem fazer referências a sistemas de detecção de intrusão. O último trabalho é o Wisesa et al. (2016) que faz uso do MOA para análise de trânsito de fluxo de dados e compara os resultados com outros *frameworks* como o WEKA e o SPARK MLib, também sem fazer referência a sistemas de detecção de intrusão.

O último filtro empregado (CE4) é sobre a utilização do MOA com técnicas de mineração não supervisionadas, ou seja, não fazem uso de rótulos para classificação das amostras. Podemos citar como exemplo o trabalho Luo, Du e Sun (2018), ele busca fazer a classificação de grande fluxo de dados com a abordagem da detecção de anomalias em tempo-real através de métodos não supervisionados de clusterização. Apesar dos autores apontarem que os algoritmos baseados nesta técnica são os que apresentam o maior progresso na literatura para a detecção de anomalias em fluxo de dados, não iremos trabalhar

com este tema.

Tabela 20: Número de trabalhos após filtragem com Critério de Exclusão 1 e 2.

Base de busca	**Filtro**	**Trabalhos relacionados**
Google Scholar	"IDS", "Intrusion Detection" e "A partir de 2010"	55
Scopus	"Intrusion Detection" e "A partir de 2010"	4
IEEE Xplorer	"Artigos de Conferências" e "A partir de 2010"	13

Estes dois últimos filtros foram executados com a análise individual de cada artigo, onde basicamente se buscou restringir as publicações referentes à área de pesquisa de segurança em detecção de intrusão. O resultado final pode ser visto na Tabela 3.

Dos artigos retirados podemos destacar sete principais temas, não relacionados com este trabalho, mas que merecem destaque por tratarem de temas atuais promissores sobre o assunto. Estes temas foram destacados pois podem ser possíveis temas de trabalhos futuros como continuação desta dissertação:

- propostas de novas técnicas de mineração para IDS nos trabalhos de Parker, Khan e Bifet (2014), Pan, Morris e Adhikari (2015), Woźniak et al. (2016) e Agrawal et al. (2020), que apesar de serem bons artigos para descrever os mais diversos algoritmos executados no MOA, os seus estudos levam a outro objetivo que foge ao tema desta dissertação;

- descrição de métodos híbridos de detecção em fluxo de dados nos artigos Noorbehbahani et al. (2017), Luo, Du e Sun (2018), Le Nguyen, Gomes e Bifet (2019) e Green, Johnsten e Benton (2020), onde os autores propõem novas técnicas semi-supervisionadas com utilização de técnicas de clusterização com o *framework* MOA;

- proposta de utilização de técnicas de seleção de atributos para otimização no desempenho dos algoritmos de classificação no trabalho de Abreu, Carvalho e Abelém (2021);

- desenvolvimento de ferramentas complementares para gerenciamento e métricas do MOA no artigo Maciel, Santos e Barros (2020);

- desenvolvimento de técnica para balanceamento de amostras de *dataset* em fluxo de dados no *framework* MOA mostrados por Bernardo, Della Valle e Bifet (2020);

- criação de técnica com classificadores em conjunto com uso de *framework* alternativo SPARK *Streaming*[1] no artigo Muallem et al. (2018);

- descrição teórica geral sobre Aprendizado de Máquina online e offline em segurança de fluxo de dados no artigo Ceschin et al. (2020), porém sem a utilização do *framework* MOA e sem testes práticos.

[1] Site:https://spark.apache.org/docs/latest/streaming-programming-guide.html, novembro de 2021.