Detecção de Intrusão sobre Pacotes Utilizando Algoritmos de Fluxos Contínuos de Dados

Gilberto Olímpio Júnior

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2021

Gilberto Olímpio Júnior

Detecção de Intrusão sobre Pacotes Utilizando Algoritmos de Fluxos Contínuos de Dados

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof[a]. Dr[a]. Elaine Ribeiro de Faria Paiva
Coorientador: Prof. Dr. Rodrigo Sanches Miani
Coorientador: Prof. Dr. Lásaro Jonas Camargos

Uberlândia
2021

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

O46 Olimpio Júnior, Gilberto, 1982-
2021 Detecção de Intrusão sobre Pacotes Utilizando Algoritmos de Fluxos Contínuos de Dados [recurso eletrônico] / Gilberto Olimpio Júnior. - 2021.

Orientadora: Elaine Ribeiro de Faria.
Coorientador: Rodrigo Sanches Miani.
Coorientador: Lasaro Jonas Camargos.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2021.634
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Faria, Elaine Ribeiro de,1980-, (Orient.). II. Miani, Rodrigo Sanches,1983-, (Coorient.). III. Camargos, Lasaro Jonas,1978-, (Coorient.). IV. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. V. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Mestrado Acadêmico, 24/2021, PPGCO				
Data:	28 de outubro de 2021	Hora de início:	14:00	Hora de encerramento:	16:30
Matrícula do Discente:	11912CCP011				
Nome do Discente	Gilberto Olímpio Júnior				
Título do Trabalho:	Detecção de Intrusão sobre Pacotes Utilizando Algoritmos de Fluxos Contínuos de Dados				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Inteligência Artificial				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Renan Gonçalves Cattelan - FACOM/UFU; Hermes Senger - DC/UFSCAR; Lásaro Jonas Camargos - FACOM/UFU (coorientador); Rodrigo Sanches Miani - FACOM/UFU (coorientador) e Elaine Ribeiro de Faria Paiva - FACOM/UFU, orientadora do candidato.

Os examinadores participaram desde as seguintes localidades: Hermes Senger - São Carlos/SP; Renan Gonçalves Cattelan, Lásaro Jonas Camargos, Rodrigo Sanches Miani e Elaine Ribeiro de Faria Paiva - Uberlândia/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos a presidente da mesa, Prof.ª Dr.ª Elaine Ribeiro de Faria Paiva, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir a senhora presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Renan Gonçalves Cattelan**, **Professor(a) do Magistério Superior**, em 29/10/2021, às 08:55, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Elaine Ribeiro de Faria Paiva**, **Professor(a) do Magistério Superior**, em 29/10/2021, às 09:08, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Lásaro Jonas Camargos**, **Professor(a) do Magistério Superior**, em 29/10/2021, às 09:15, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rodrigo Sanches Miani**, **Professor(a) do Magistério Superior**, em 29/10/2021, às 10:24, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Hermes Senger**, **Usuário Externo**, em 04/11/2021, às 15:41, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3132311** e o código CRC **2B6C5FB5**.

Referência: Processo nº 23117.071442/2021-61 SEI nº 3132311

À Bell e ao Bê, vocês sempre foram o meu maior incentivo.

Agradecimentos

Os anos de mestrado foram desafiadores, com muito estudo, esforço e empenho. Tenho comigo que os grandes desafios nunca podem ser conquistados solitariamente e, por isso, gostaria de agradecer algumas pessoas que sempre estiveram ao meu lado.

Primeiramente agradeço à Fabíola, minha amada esposa, que com seu jeito simples e afetuoso sempre me incentivou, mesmo nos momentos mais difíceis que passamos, nunca me deixou dar um passo atrás. Obrigado por ter sido pai e mãe em tantas ocasiões que estive ausente. Obrigado por ficar ao meu lado em tantos finais de semana e feriados, sem que pudéssemos nos divertir e descansar. Sou grato por tê-la ao meu lado, te amo, você foi fundamental e essa jornada só foi possível graças a você! Agradeço ao meu amado filho Bernardo, que com sua alegria em viver me ensinou que a persistência, paciência e amor são fundamentais em qualquer projeto em nossas vidas! Te amo, Bê!

Gostaria de agradecer também a minha mãe, Dona Neném a quem me deu a vida! Obrigado por ter me ensinado a ser uma boa pessoa, obrigado por ter me dado a oportunidade de estudar, mesmo com tantas dificuldades, a senhora nunca desistiu de mim! Às minhas irmãs, Paula e Titi, a quem devo muito do que sou hoje. Vocês que, por vezes abriram mão de várias coisas para que eu pudesse ter como estudar, foram fundamentais nessa conquista! Vocês três foram e sempre serão importantes em minha vida!

Agradeço, com muito carinho, àqueles que transformaram a minha vida através do ensino! Os meus mestres: Elaine, Miani e Lásaro! A missão de vocês não é simples, mas é fantástica! E vocês sempre a cumpriram de forma extraordinária, vejo paixão pelo que fazem! Elaine, quando me aceitou como o seu aluno foi o início da transformação, o divisor de águas em minha vida profissional! Você com a sabedoria, que somente os melhores mestres possuem, me fez sentir capaz! Terei saudades das reuniões semanais onde, com muita paciência me ensinou, deu sugestões e conselhos. Você foi muito importante em tudo isto! Miani, obrigado por ter me aceitado como seu orientando, você é um cara fantástico, cheio de ideias e sempre me incentivou. Sua sabedoria é de encher os olhos. Sempre me atendeu com toda a disposição e alegria! Lásaro, obrigado pelo incentivo e pela parceria! Você acreditou, quando eu duvidei que seria possível! As trocas de ideias,

as palavras incentivadoras e o seu empenho em me ajudar fizeram ser possível! Tenho o maior orgulho de dizer a todos: esses são os meus mestres! Como respondi certa vez ao Miani, se fosse possível escolher novamente os meus orientadores, escolheria por 1.000 vezes vocês! Além de mestres, vocês se tornaram meus amigos!

Agradeço a UFU e, em especial, a FACOM! Que sempre prezou por um ensino de qualidade e que, mesmo em tempos tão difíceis como o que passamos, se manteve firme diante dos desafios. Obrigado aos professores, que sempre estiveram dispostos a nos ensinar com qualidade! Terei saudades desse lugar!

Por fim, gostaria de agradecer àqueles que me aguentaram por alguns anos, falando sempre sobre o mesmo assunto. Juliano Josceli, meu amigo, que soube entender o tamanho do desafio, obrigado pelo incentivo, ensinamentos e pelas sugestões. A Vanessa Pereira pelo incentivo e por estar sempre disposta a ajudar neste grande projeto. Ao Lucas Chaves pelas correções, sugestões e explicações. Aos amigos da Primo Investiments que não se cansaram (eu acho!) de meu ouvir, principalmente quando eu tentava explicar dezenas de vezes o que eu estava estudando! Obrigado a Dayana, Helen, Joyce, Larissa, Guilherme e João. A torcida de vocês foi incentivadora! Aos amigos que, em cada conversa, me ensinaram bastante sobre ciência de dados: Cássio, Matheus e George (Z0del). Obrigado!

Por fim, agradeço aqueles que contribuíram direta ou indiretamente para que essa dissertação fosse possível! Meus sinceros agradecimentos!

"For small creatures such as we the vastness is bearable only through love."
(Carl Sagan)

Resumo

Os Sistemas de Detecção de Intrusão (IDS) auxiliam na proteção das redes de computadores, pois identificam e detectam tentativas de obter acesso não autorizado aos dados, inspecionando pacotes individualmente ou no contexto de fluxos. Considerando que o processo de detecção de intrusão é uma tarefa de classificação de um fluxo de pacotes gerados continuamente em uma distribuição não-estacionária, os modelos de decisão devem sofrer atualizações a fim de identificar mudanças no comportamento dos ataques e do tráfego normal da rede. A atualização dos modelos, em geral, requer instâncias rotuladas, o que exige grande esforço de especialistas. Este trabalho contribuiu com o desenvolvimento de um IDS para o mundo real, propondo: i) comparar o uso de pacotes individuais e fluxos de pacotes na tarefa de detecção de intrusão, por meio da análise do desempenho preditivo de classificadores de fluxos contínuo de dados construídos a partir desses dois tipos de dados; ii) analisar o impacto no desempenho dos classificadores quando há atraso na entrega dos rótulos das instâncias para atualização dos modelos; e iii) avaliar o impacto que as estratégias de aprendizado ativo causam nos classificadores quando somente as melhores instâncias são rotuladas e usadas na atualização dos modelos. Neste sentido, os experimentos foram realizados usando o conjunto de dados CICIDS2017, diferentes algoritmos de classificação de fluxos contínuos de dados e medidas de avaliação. Eles mostraram que inspecionar os pacotes individualmente tem um desempenho semelhante à inspeção de fluxos na detecção de intrusão. De modo que, a partir desse resultado, os pacotes individuais foram usados no estudo de técnicas de aprendizado ativo e atraso na entrega dos rótulos. O desempenho dos classificadores sofreu queda à medida que se aumentava o atraso na entrega dos rótulos verdadeiros. Por fim, as estratégias de aprendizado ativo permitiram manter o desempenho preditivo na classificação de tráfego normal e malicioso, usando um conjunto reduzido de instâncias rotuladas.

Palavras-chave: Sistemas de Detecção de Intrusão, Aprendizado de Máquina, Fluxos Contínuos de Dados, Pacotes de Rede.

Abstract

Intrusion Detection Systems (IDSs) help protect computer networks by identify and detect attempts to obtain unauthorized access to data via computer networks by inspecting packets separately or in the context of flows. Considering that the intrusion detection process is a classification task of continuously stream-generated packets in a non-stationary distribution, security analysis must constantly update decision models to identify changes in attack behaviors and normal traffic of a network. Since improving models usually requires labeled instances, which demands significant effort from security specialists, the purpose of this work is to contribute to the development of real-world IDSs. Therefore, our goal is to: i) compare the use of individual packets and network flows in the intrusion detection task by analyzing the predictive performance of data stream classifiers; ii) analyze the impact of delayed labelling for updating the models on the classifiers' performance; and iii) evaluate the impact of active learning strategies on the classifiers' performance. Our experimental evaluation used the CICIDS2017 dataset, different data stream classification algorithms, and five evaluation measures. Experiments have shown packet-based IDSs perform similarly to flow-based IDSs. Based on this result, we studied different active learning techniques to estimate the impact of delayed labelling on packet-based IDSs. The performance of the classifiers is inversely proportional as the label delivery rate. Besides, the active learning strategies helped keep the performance at a satisfactory level, even with a small set of labeled instances.

Keywords: Intrusion Detection Systems, Machine Learning, Data Streams, Packets Header.

Lista de ilustrações

Lista de tabelas

Lista de siglas

ADWIN *Adaptive Windowing*

ARFF *Attribute-Relation File Format*

AWE *Accuracy Weighted Ensemble*

ASHT *Adaptive Size Hoeffding Tree*

CSV *Comma-separated Values*

DARPA *Defence Advanced Research Project Agency*

DC Diferença Crítica

DoS *Denial of Service*

DDoS *Distribuited Denial of Service*

FAR *False Alarm Rate*

IDS *Intrusion Detection Systems*

IP *Internet Protocol*

IoT *Internet of Things*

KDD *Knowledge Discovery in Databases*

LSTM *Long Short Term Memory*

MOA *Massive Online Analysis*

OSI *Open Systems Interconnection*

PCAP *Packet Capture*

SQL *Structured Query Language*

TCP *Transfer Control Protocol*

TLS *Transport Layer Security*

UDP *User Datagram Protocol*

VPN *Virtual Private Network*

Sumário

Capítulo 1

Introdução

As redes de computadores são infraestruturas essenciais dos serviços computacionais e permeiam as atividades diárias de pessoas e empresas (HAMID; MUTHUKUMARASAMY; RANGANATHAN, 2016). Segundo a CISCO, multinacional do setor de redes e comunicações, o tráfego total de dados nas redes em 2016 era de 96 Exabytes/mês e, em 2021, espera-se que atinja 276 Exabytes/mês (VIEGAS et al., 2019), um aumento de 187, 5%. Consequentemente, o tráfego nas redes de computadores, em particular da Internet, a qual suporta um intenso e sigiloso fluxo de dados, desperta o interesse daqueles que desejam obtê-los de forma ilegal utilizando-os de maneira maliciosa (WAITZMAN, 2001). Deste modo, com o desenvolvimento e popularização da Internet, a necessidade de garantir a proteção e segurança dos dados que trafegam nessa rede também aumenta e, por isso, a área de pesquisa em segurança da informação tem se tornado cada vez mais ativa e necessária (LI et al., 2012).

Uma intrusão é qualquer tentativa de acessar ou manipular dados indevidamente através das redes de comunicação como, por exemplo, a Internet. Com um volume de tráfego crescente, as intrusões estão evoluindo na sua quantidade e capacidade (VIEGAS; SANTIN; OLIVEIRA, 2017). São vários os propósitos e características dos ataques e, por vezes, seus sintomas não são tão óbvios devido às diversas formas de intrusão existentes. Neste contexto, os IDSs são *hardware* ou *software* que desempenham papel fundamental, uma vez que os mesmos monitoram continuamente a rede buscando identificar e alertar os administradores de segurança sobre as tentativas de intrusão (SHARAFALDIN; LASHKARI; GHORBANI, 2018). Eles também coletam, analisam e armazenam informações que pertencem ao tráfego de rede (NAKAMURA; GEUS, 2007). Desta maneira, devido às diversas formas de intrusões existentes, diferentes estratégias devem ser consideradas durante o desenvolvimento de um IDS. Frequentemente, dois métodos são utilizados por eles para a detecção de comportamentos maliciosos, os baseados em assinaturas e os baseados em detecção de anomalias (BHUYAN; BHATTACHARYYA; KALITA, 2017).

Os métodos baseados em assinaturas comparam o tráfego analisado com uma base de dados de ataques anteriormente catalogados. Caso algum ataque presente na base de

dados seja identificado no tráfego, o IDS caracteriza-o como malicioso. Por utilizarem bases de dados de assinaturas dos ataques, estes sistemas são precisos, porém podem não identificar ataques recém desenvolvidos. Isso ocorre pois as assinaturas dos ataques tem um período entre sua identificação, inserção na base e sua difusão na Internet. Já nos métodos baseados em anomalias, a identificação é feita baseando-se em um comportamento anteriormente reconhecido como normal, ou seja, em um padrão não-malicioso. Assim, caso um comportamento saia deste padrão pré-analisado ele será classificado como um ataque (LIAO et al., 2013). Desta forma, tais sistemas podem identificar ataques novos e/ou desconhecidos devido à sua natureza de identificação, tomando por base o comportamento fora do padrão de normalidade da rede de computadores (BHUYAN; BHATTACHARYYA; KALITA, 2017).

Para os métodos baseados em anomalias, uma proposta comumente usada são os algoritmos de aprendizado de máquina, já que estes podem ser usados para identificar comportamentos fora de um padrão pré-estabelecido (GARCÍA-TEODORO et al., 2009; KANIMOZHI; JACOB, 2019; AL-UTAIBI; EL-ALFY, 2018; ATLI et al., 2018; PAPAMARTZIVANOS; MáRMOL; KAMBOURAKIS, 2019). O *Knowledge Discovery in Databases* (KDD) é um processo habitualmente utilizado para encontrar padrões, permitindo a interpretação dos dados ou a previsão de eventos futuros (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996b). Ele é iterativo e abrange as seguintes etapas: coleta de dados; pré-processamento dos dados coletados; transformação dos dados; mineração e descoberta dos padrões e; a interpretação e avaliação dos resultados.

Quanto à coleta dos dados, os dispositivos de rede podem apresentá-los tanto individualmente na forma de pacotes quanto agrupados em fluxo destes pacotes. O fluxo é uma agregação de pacotes de redes transmitidos que compartilham algumas características em comum, tais como: o mesmo endereço *Internet Protocol* (IP) de origem e destino, porta de origem e destino e protocolo de transporte dentro de uma janela de tempo (RING et al., 2019a). Os trabalhos de Viegas et al. (2019), Atli et al. (2018), Papamartzivanos, Mármol e Kambourakis (2019) utilizam algoritmos de aprendizado de máquina e analisam os fluxos de pacotes para detectar anomalias. Embora uma abordagem baseada em fluxo faça a suposição coerente de que os ataques provavelmente abrangem vários pacotes, esta perspectiva deve limitar a porção de informações mantidas para cada fluxo, dada a quantidade deles que podem haver na rede e os recursos computacionais que são necessários para armazenar grande quantidade de informações para cada um.

Outros trabalhos como Zhong et al. (2020), Kim, Park e Lee (2020) e Bortolameotti et al. (2020), propõem uma outra abordagem na qual a classificação é baseada exclusivamente nos pacotes de redes. Essa abordagem preconiza a característica de avaliação individual dos pacotes, de modo a identificar pacotes maliciosos.

A análise dos pacotes pode considerar a inspeção da carga útil ou restringe-se somente aos cabeçalhos e rodapés. Ainda que a exploração da carga útil possa criar um perfil do

tráfego mais detalhado e, consequentemente, levar a uma melhor detecção de diferentes tipos de ataque, o desempenho preditivo pode ser reduzido em cenários com alto volume de tráfego (UHM; PAK, 2021). Além disso, qualquer tentativa de análise da carga útil torna-se impraticável se as informações estiverem cifradas.

Para que os classificadores possam utilizar os dados de forma adequada, é necessária uma etapa de pré-processamento para que, dessa forma, os cabeçalhos dos pacotes possam ser transformados em um conjunto de atributos para os diferentes protocolos existentes (por exemplo *Transfer Control Protocol* (TCP), IP, *User Datagram Protocol* (UDP), etc.). No entanto, muitos tipos de intrusão podem ser distribuídos ao longo da carga útil de vários pacotes de rede, permanecendo inacessíveis em uma análise por pacote.

Na etapa de mineração de dados, em geral, diferentes algoritmos de classificação são utilizados para distinguir tráfego entre malicioso e normal. Alguns exemplos de algoritmos de classificação utilizados nesta etapa são: árvores de decisão (Ariyaluran Habeeb et al., 2019), máquinas de vetores de suporte (RING et al., 2019b) e métodos probabilísticos (LIAO et al., 2013). Devido à evolução contínua das redes e do tráfego de redes, as características de um tráfego normal mudam constantemente (SOMMER; PAXSON, 2010) e essa natureza de inconstância do tráfego dificulta a identificação de novos ataques, exigindo uma flexibilidade e adaptabilidade dos algoritmos de classificação (LEE; STOLFO, 2000). Com o objetivo de permitir adaptabilidade dos IDSs, trabalhos recentes consideram a tarefa de detecção de intrusão como uma classificação em fluxo contínuos de dados (VIEGAS et al., 2019; RIBEIRO; PAIVA; MIANI, 2020; COSTA et al., 2018; CASSALES et al., 2019). Fluxos contínuos de dados (do inglês *data streams)* são sequências de dados geradas continuamente, em geral, em alta velocidade (KNOWLEDGE. . . , 2010). Nesse tipo de cenário, a distribuição que gera os dados pode mudar, o que exige que a tarefa de classificação de fluxos contínuos de dados frequentemente adapte o modelo de decisão para representar tais mudanças usando estratégias de baixo custo computacional. Sendo assim, o uso de técnicas de classificação para fluxos contínuos de dados em IDSs torna-se uma interessante estratégia para manter os modelos adaptados às condições atuais do tráfego de rede.

Apesar de trabalhos recentes considerarem o uso de técnicas de classificação que se adaptam às condições recentes do fluxo (VIEGAS et al., 2019; UHM; PAK, 2021; KOZIK; PAWLICKI; CHORAŚ, 2021), uma importante questão a ser tratada é o uso de uma grande quantidade de instâncias rotuladas para que os classificadores consigam determinar o que é um tráfego normal ou ataque. Ainda que existam vários tipos de processos de categorização de instâncias, o mais utilizado é aquele onde elas são rotuladas manualmente por um especialista de domínio. O qual analisa o conjunto de dados dentro de um contexto e, então, decide se o dado é malicioso ou não. No entanto, este processo é custoso e demanda tempo para a sua execução podendo causar atrasos na rotulagem das instâncias. Muitos trabalhos da literatura (VIEGAS et al., 2020; HAIDER et al., 2021; UHM; PAK,

2021) consideram que todas as instâncias do fluxo de dados serão rotuladas imediatamente após a sua chegada para atualização do modelo de decisão. Apesar do alto desempenho preditivo de tais classificadores na detecção de intrusão, esta proposta é hipotética e não permite sua aplicação em IDSs a serem utilizados no mundo real.

Assim, diante dos desafios em classificar os pacotes de redes, este trabalho visa estabelecer a utilização de técnicas de classificação de fluxos contínuos de dados como alternativa relevante para a construção de um IDS, utilizando somente os pacotes individuais no processo de classificação. Isso traz algumas vantagens para o processo de detecção, já que não é necessário processar os pacotes para, só então, gerar o fluxo com os resumos de informações e a partir daí executar a tarefa de classificação. Mais especificamente, é investigado se existem diferenças significativas na identificação de tráfego malicioso quando são empregados os dados sumarizados nos fluxos ou os dados de cada um dos pacotes do tráfego da rede. Além disso, com a utilização de algoritmos de classificação de fluxos contínuos de dados, não há a necessidade de retreinar o modelo a cada mudança na distribuição dos dados, como frequentemente ocorre em modelos tradicionais de classificação (CASSALES et al., 2019).

Por fim, o presente trabalho também considera que a atualização do modelo de decisão não pode ser feita supondo que todas as instâncias do fluxo serão rotuladas. Para isso, técnicas de aprendizado ativo (ZHU et al., 2007; KAVITHA; MAHESWARI et al., 2021; YANG et al., 2018) que escolhem um subconjunto de instâncias a serem rotuladas são investigadas, em especial, aquela a qual supõe que existe um atraso para a entrega das instâncias após receber um rótulo.

1.1 Objetivos e Desafios da Pesquisa

Este trabalho tem como objetivo colaborar para o desenvolvimento de um IDS que atenda aos requisitos reais do problema de detecção de intrusão, tais como: geração contínua de dados e constante atualização do modelo de detecção de intrusão. Para isto, este trabalho visa comparar, inicialmente, o desempenho preditivo de classificadores treinados com dados de fluxos com aqueles treinados a partir de pacotes individuais, a fim de identificar se as duas abordagens produzem resultados semelhantes. Pretende também, analisar o impacto do atraso da entrega de instâncias rotuladas para a atualização do modelo e, por fim, analisar o impacto da seleção de um conjunto reduzido de amostras para receberem o rótulo para a atualização do modelo de classificação.

Os objetivos específicos são:

- ❑ comparar o desempenho preditivo de classificadores induzidos a partir de dados do fluxo e de dados dos pacotes da rede;

- ❏ comparar o desempenho preditivo de um conjunto de algoritmos de classificação de fluxos contínuos de dados na detecção de tráfego malicioso;
- ❏ analisar como o atraso do rótulo para a atualização dos modelos de classificação impacta no desempenho de classificação de novos dados da rede, e;
- ❏ avaliar como as estratégias de aprendizado ativo podem auxiliar na escolha das melhores instâncias a serem rotuladas de forma a manter o desempenho do classificador, mas diminuindo o número de instâncias rotuladas.

1.2 Hipótese

A seguir são apresentadas as hipóteses estabelecidas para a execução do presente trabalho.

- ❏ **H1:** A utilização dos pacotes individuais de redes para a identificação de ataques pelos classificadores, apresentará resultados estatisticamente semelhantes quando comparados à utilização dos fluxos.
- ❏ **H2:** Quando houver atraso na apresentação dos rótulos verdadeiros das instâncias aos classificadores, o desempenho de classificação dos pacotes de rede será impactado.
- ❏ **H3:** A utilização de técnicas de aprendizado ativo, que escolhem um subconjunto de instâncias a serem rotuladas pelo especialista para atualização do modelo, alcança um desempenho preditivo semelhante aos modelos atualizados usando todas as instâncias rotuladas do fluxo rotuladas na detecção de intrusão.

1.3 Organização do Trabalho

Esta dissertação está organizada da seguinte forma. O Capitulo 2 trata da fundamentação teórica para embasar os métodos propostos. O Capítulo 3 apresenta os trabalhos relacionados utilizados como base bibliográfica para esta dissertação. O Capítulo 4 descreve detalhadamente os métodos utilizados para testar as hipóteses propostas. O Capítulo 5 apresenta os experimentos e a discussão dos resultados obtidos. Por fim, o Capítulo 6 apresenta a conclusão desta dissertação, as principais contribuições e sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo, serão apresentados os conceitos teóricos fundamentais à compreensão desta pesquisa, sendo o mesmo dividido nas seções descritas a seguir. Na Seção 2.1 são estabelecidos os conceitos de redes de computadores e seus protocolos, especialmente aqueles utilizados neste trabalho, bem como os conceitos sobre segurança de redes de computadores e os sistema de detecção de intrusão. Na Seção 2.2 é feita uma apresentação do método de descoberta de conhecimento em bases de dados. Na Seção 2.3 é apresentada a mineração em fluxos contínuos de dados, seus desafios, bem como algumas das estratégias e algoritmos de classificação usados na literatura de detecção de intrusão. Também é apresentado o conceito de aprendizado ativo, que foi usado neste trabalho. Na Seção 2.4 são apresentados os avaliadores de desempenho preditivo mais utilizados na literatura de classificação de fluxos contínuos de dados e detecção de intrusão.

2.1 Redes de computadores

Uma rede de computadores interliga os computadores com o objetivo de transportar informações entre eles. As mensagens são enviadas entre os computadores utilizando a infraestrutura de rede (NEWMAN, 2018).

As redes de computadores criaram mudanças na economia mundial. Com elas, as empresas puderam modificar a forma como se comunicavam, criaram serviços para os seus clientes e diminuíram seus custos devido à forma como executavam as atividades, utilizando o acesso aos computadores dispostos remotamente (COMER, 2017). Também permitiram criar um setor inteiro que desenvolve tecnologia, produtos e serviços voltados para as redes de computadores. Quanto à sua aplicação, as redes de computadores são utilizadas em sistemas de telecomunicações, entretenimento, sistemas de navegação, redes de sensores, acesso a dados remotos, dentre outras funcionalidades (COMER, 2017).

2.1.1 Protocolos IP, TCP e UDP

O protocolo IP, que pertence à camada de rede segundo o padrão *Open Systems Interconnection* (OSI), é utilizado pelos comutadores para o endereçamento e o encaminhamento dos pacotes de redes. Em uma rede IP, um endereço único de origem e destino é atribuído a cada datagrama trocado na rede, conforme representado na Figura 1.

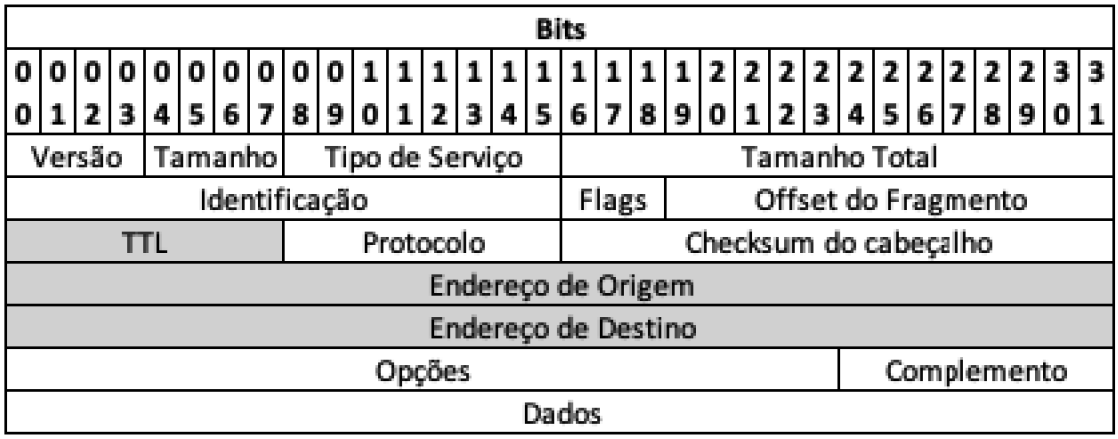


Figura 1 – *Datagrama* do protocolo IP com 32 bits de tamanho - Adaptado de Kurose e Ross (2011).

Já o protocolo TCP pertence à camada de transporte do padrão OSI. Ele é orientado a conexão e, assim, antes de enviar os dados para um servidor os dois atores (cliente/servidor) devem executar um *"handshake"* (KUROSE; ROSS, 2011), procedimento que estabelece a conexão. Somente após esse procedimento é que os pacotes, representados na Figura 2, passam a trafegar entre as partes. O estabelecimento da conexão, associado a diversos mecanismos, permite ao TCP garantir confiabilidade na troca de mensagens, bem como controle de fluxo e de congestionamento (KUROSE; ROSS, 2011).

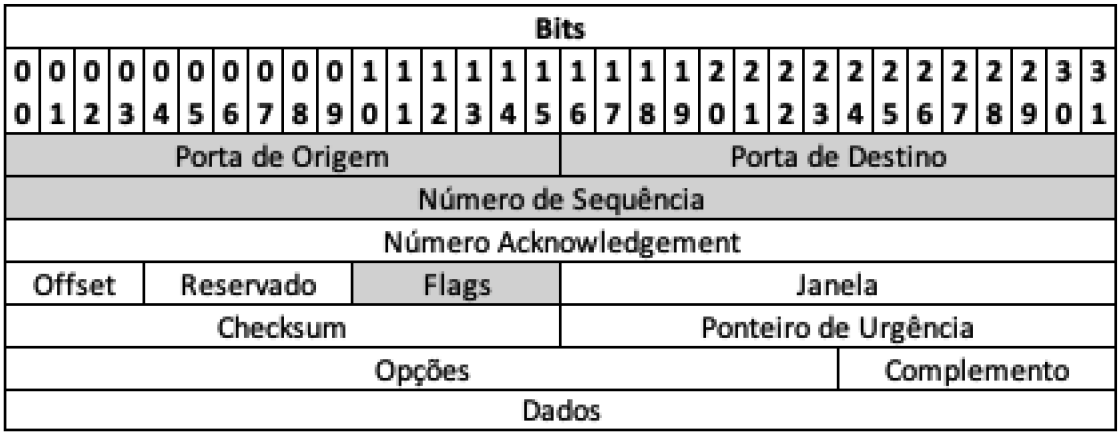


Figura 2 – *Datagrama* do protocolo TCP com 32 bits de tamanho - Adaptado de Kurose e Ross (2011).

O UDP é um protocolo mais simples que também pertence à camada de transporte, e sua organização pode ser vista na Figura 3. Como é um protocolo orientado para

transações (POSTEL et al., 1980), ou seja, sem a necessidade de estabelecer uma conexão como no protocolo TCP, não há a garantia de entrega dos pacotes (FALL; STEVENS, 2011). Considerando suas utilizações, o protocolo UDP é extremamente útil em situações de cliente/servidor nas quais seja necessário o envio de uma solicitação ao servidor sem necessidade de resposta. Assim, caso a resposta não chegue de volta, o cliente apenas receberá uma mensagem de tempo esgotado, sendo preciso somente o envio de uma nova solicitação sem onerar o tráfego da rede (TANENBAUM; WETHERALL et al., 2014).

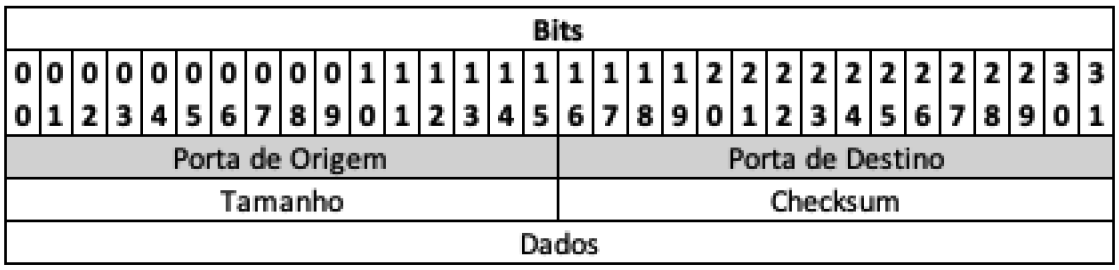


Figura 3 – *Datagrama* do protocolo UDP com 32 bits de tamanho - Adaptado de Kurose e Ross (2011).

2.1.2 A Segurança nas Redes de Computadores e os Sistemas de Detecção de Intrusão

As redes de computadores são tipicamente recursos compartilhados, usados por várias aplicações e diversos interesses. Considerando esse compartilhamento, os seus diferentes usos trazem oportunidades de invasões e ataques a tais recursos. Portanto, faz-se necessária a criação de mecanismos que permitam o desenvolvimentos de canais seguros de comunicação (WAITZMAN, 2001).

A tríade básica de uma comunicação segura envolve as seguintes propriedades (KUROSE; ROSS, 2011):

- **Confidencialidade**: garantia de que somente o emissor e o receptor da mensagem entendam o que está sendo enviado, implementada pelo uso de algoritmos de criptografia;

- **Integridade**: garantia de que mensagens sejam entregues aos destinatários de forma íntegra, não sendo alteradas por problemas durante seu transporte ou mesmo por agentes maliciosos; e,

- **Disponibilidade**: garantia de que uma informação ou recurso estará acessível quando requerido, tal que a velocidade de acesso seja rápida o suficiente para que o sistema possa executar uma tarefa satisfatoriamente.

Aspectos como autenticidade, não-repúdio e a segurança operacional complementam a tríade básica. A autenticidade garante que a pessoa ou sistema que está acessando o serviço é quem realmente diz ser e, geralmente, é feita por um sistema de autenticação de usuários. O não-repúdio procura garantir que o usuário emissor de alguma informação ou, ainda, um executor de alguma ação sobre a informação não consiga negar, posteriormente, que essa ação tenha sido executada. Por fim, a segurança operacional visa garantir que haja toda uma infraestrutura disponível para garantir a segurança da rede como, por exemplo, a utilização de *firewalls* e sistemas de detecção de intrusão. Consequentemente, um ataque é uma tentativa bem sucedida de comprometer pelo menos uma das propriedades anteriores.

Os IDSs são sistemas computacionais que automatizam o processo de monitoramento de eventos, ocorridos em um sistema de computação ou rede de computadores. Seu principal papel é verificar a existência de um ataque em curso. Assim, os IDSs desempenham um papel vital, sendo uma linha de defesa contra intrusões às redes.

Os IDSs podem se localizar em vários pontos da rede, e sua localização indica um tipo específico de proteção. Na Figura 4, é possível identificar essas posições. Nakamura e Geus (2007) descrevem cada uma delas da seguinte forma:

- **IDS 1**: nessa posição, o IDS oferece uma proteção que detecta todas as tentativas de ataques contra a rede. Esse IDS oferece uma fonte de informação rica sobre os tipos de tentativas de ataques a rede.

- **IDS 2**: nessa posição, o IDS pode detectar ataques contra o *firewall*.

- **IDS 3**: nessa posição, detecta as tentativas de ataque contra os servidores localizados na zona desmilitarizada (do inglês *DeMilitarized Zone* — DMZ).

- **IDS 4**: nessa posição, pode detectar as tentativas de ataques contra os recursos internos da rede que passaram pelo *firewall* e que podem ocorrer via VPN (*Virtual Private Network*.

- **IDS 5**: nessa posiçã, pode detectar as tentativas de ataques contra servidores na DMZ 2. Nesse cenário as tentativas de ataques passaram pelo *firewall*, pela *Virtual Private Network* (VPN) ou por algum outro serviço na DMZ 1.

- **IDS 6**: nessa posição, pode detectar as tentativas de ataques internos na organização.

Além do posicionamento, uma outra característica dos IDSs está relacionada à forma com que as detecções são realizadas. Existem dois sistemas principais de detecção de intrusão: os sistemas baseados em assinaturas e os sistemas baseados em anomalias (BHUYAN; BHATTACHARYYA; KALITA, 2017).

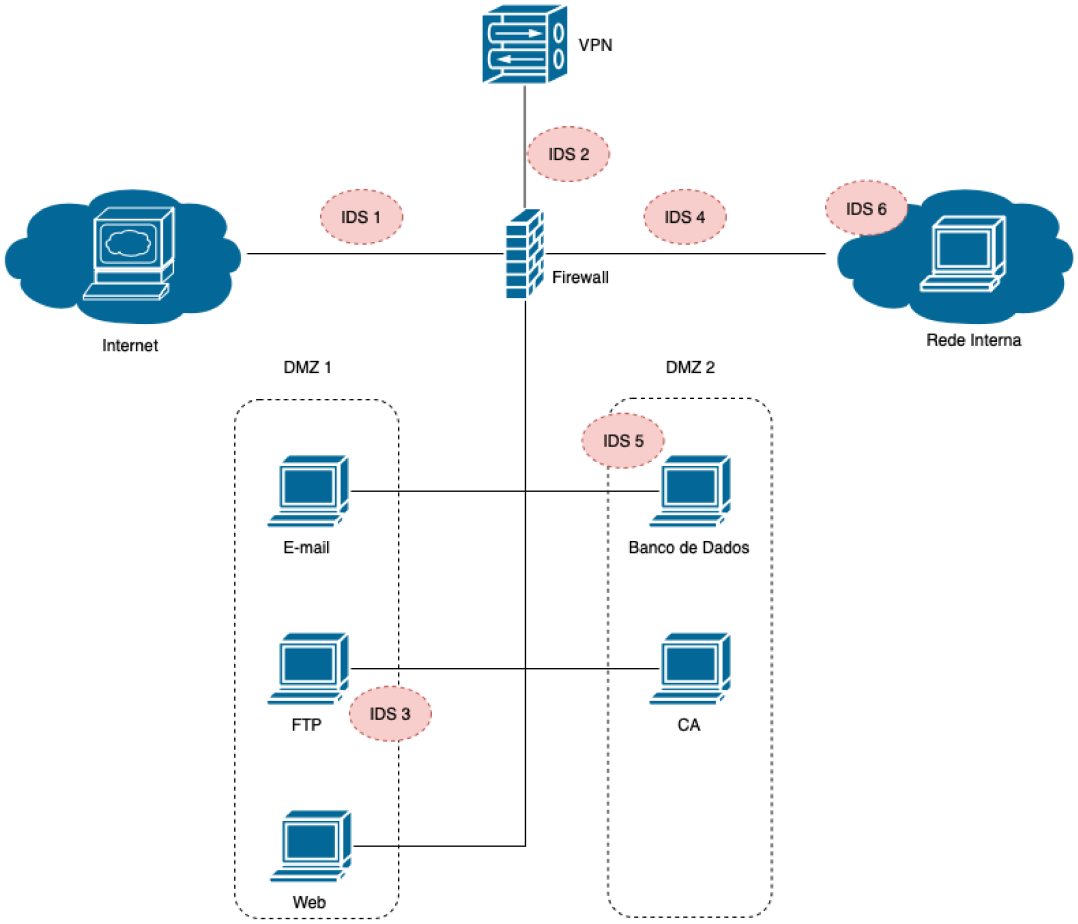


Figura 4 – Possíveis posições de um IDS em uma rede - Adaptado de Nakamura e Geus (2007).

Quando um ataque é identificado pela comunidade de segurança, é desenvolvido um conjunto de regras chamadas de assinaturas, e que podem ser usadas por IDSs para identificar uma ameaça através de comparação. A abordagem baseada em assinatura é conhecida por ter um baixo número de falsos positivos e falsos negativos, pois a sua assinatura é conhecida pela base de ataques. Porém, no reconhecimento de ataques baseado em assinaturas, não há como distinguir novos ataques até que a base seja atualizada e distribuída na Internet (GARCÍA-TEODORO et al., 2009).

Já as técnicas baseadas em anomalias, visam caracterizar uma alteração no padrão normal de comportamento da rede. Esse padrão é estabelecido anteriormente, analisando o fluxo de pacotes normais e criando assim, uma referência que será utilizada para a identificação do desvio de comportamento da rede. Esse tipo de técnica é também conhecida como detecção baseada em comportamento (BHUYAN; BHATTACHARYYA; KALITA, 2017). Geralmente, para construir a base de comportamento normal e ataques, são utilizadas técnicas estatísticas ou de aprendizado de máquina (ZARPELãO et al., 2017). Elas podem identificar ataques desconhecidos, apesar de poderem atingir uma alta taxa de falsos positivos e falsos negativos (MAHONEY; CHAN, 2001).

Diante disso, a pesquisa acadêmica sobre detecção de intrusão vem aumentando ao

longo dos anos, dadas as promissoras capacidades dos IDSs baseados em anomalias, visando o desenvolvimento de novas abordagens para esses sistemas (SHARAFALDIN; LASHKARI; GHORBANI, 2018). Como os sistemas baseados em anomalias trabalham com a busca de um comportamento fora de um padrão conhecido, é possível utilizar o aprendizado de máquina para que se possa treinar um modelo e, assim, identificar novas ameaças diferentemente do modelo de detecção por assinaturas (GARCÍA-TEODORO et al., 2009).

2.2 Descoberta de Conhecimento em Bases de Dados

Com o desenvolvimento e avanço da Internet, a quantidade de dados trafegados no mundo é sem precedentes (VIEGAS et al., 2019), desta forma as redes sociais, sites governamentais, bancos, hospitais, além do uso de aparelhos como *smartphones*, geram diariamente uma grande quantidade de dados sobre comportamentos, perfis de compras, saúde, finanças, dentre outras informações. Todos esses registros gerados e que antes não eram totalmente analisados devido à falta de métodos para sua manipulação e compreensão, agora podem ser utilizados de uma forma estratégica pelas empresas a fim de encontrar, por exemplo, comportamentos e tendências dos seus consumidores. Essa inteligência competitiva cria valor para as empresas, que usam os seus dados de forma ativa (OUSSOUS et al., 2018) para extraírem informações úteis ao negócio (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996b).

A descoberta de conhecimento usando grandes bases de dados, chamada também de KDD, vem sendo estudada desde a década de 90. O seu processo clássico, apresentado na Figura 5, foi proposto por Fayyad, Piatetsky-Shapiro e Smyth (1996a) e é definido como sendo um processo não trivial de identificar padrões válidos, novos, potencialmente úteis e, em última instância, compreensíveis para um dado fim. Esse processo é interativo e iterativo, no qual o usuário deverá tomar várias decisões a fim de estabelecer os objetivos da mineração. Para se chegar a uma descoberta de fato, são utilizadas várias etapas, tais como: a preparação, a seleção e a limpeza dos dados, além de incorporação de conhecimento prévio útil e a interpretação adequada da mineração dos dados. Elas garantem que um conhecimento útil possa ser extraído das bases de dados (BUCZAK; GUVEN, 2016).

Inicialmente, é necessário entender o domínio de conhecimento dos dados a serem trabalhados (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996b). Portanto, o conhecimento prévio e experiência com os dados é fundamental para se estabelecer o objetivo claro do problema a ser solucionado (KANTARDZIC, 2011). Partindo desse ponto, após a análise dos dados, podem ser geradas várias hipóteses formuladas para um único problema (KANTARDZIC, 2011), assim é possível que, conforme um projeto de KDD avance, essa etapa possa ser revisitada para ajustes (MAIMON; ROKACH, 2005).

A primeira etapa do processo está relacionada com a geração e coleta dos dados (KAN-

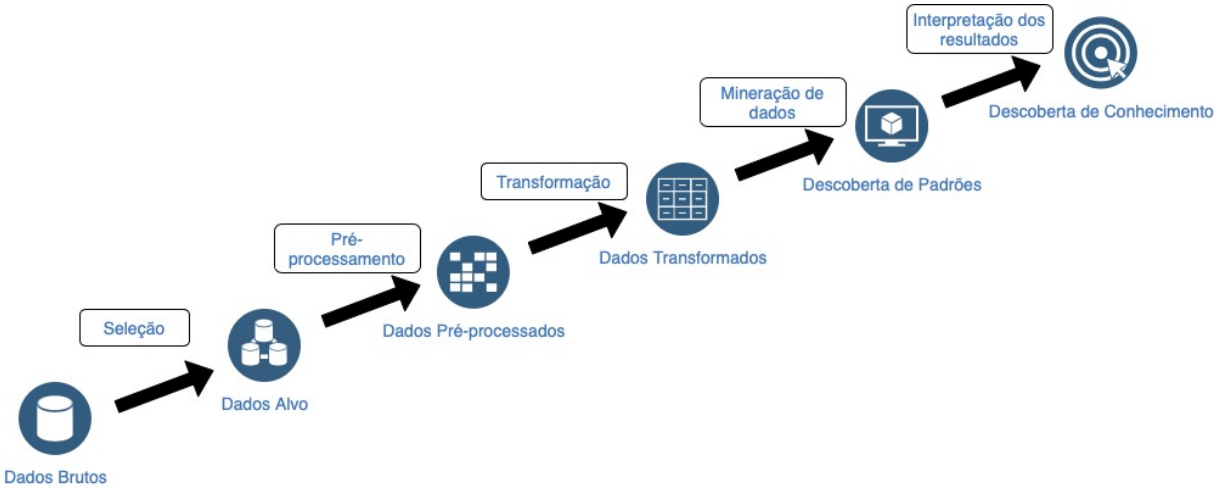


Figura 5 – Etapas do processo de KDD - Adaptado de Fayyad, Piatetsky-Shapiro e Smyth (1996a).

TARDZIC, 2011). Deve-se selecionar um conjunto de dados dentro do objetivo proposto para, assim, focar em um subconjunto de variáveis nas quais será feita a tentativa de descoberta de conhecimento (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996a). Nessa etapa, também é necessário identificar quais dados estão disponíveis, obter informações adicionais necessárias para uma melhor qualidade da descoberta e, em seguida, integrar todos esses dados em um único conjunto que será utilizado para executar a descoberta de conhecimento.

A segunda etapa é o pré-processamento, no qual se executa a limpeza dos dados, exclusão de ruídos, definição e execução de estratégias para lidar com valores ausentes e seleção de atributos (MAIMON; ROKACH, 2005; FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996b). A seleção de atributos de alta correlação com a classe alvo, pode reduzir a quantidade de dados a serem analisados pelos algoritmos de aprendizado de máquina e trazem consigo uma melhoria do desempenho preditivo, pois os dados utilizados serão aqueles que possuem uma maior importância para a caracterização da classe alvo.

A terceira etapa está relacionada à transformação dos dados. Nela ocorre a preparação dos melhores atributos para a tarefa de descoberta de conhecimento (MAIMON; ROKACH, 2005). Uma destas técnicas é a redução de dimensionalidade, que trata de encontrar características redundantes e, para isso, há na literatura várias técnicas que podem ser utilizadas, como por exemplo a análise de componente principal (PCA do inglês *Principal Component Analysis*) (GARCÍA-TEODORO et al., 2009).

Na quarta etapa, um método de mineração de dados, como classificação, regressão ou agrupamento, poderá ser utilizado para atingir os objetivos definidos ainda na primeira etapa (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996a; MAIMON; ROKACH, 2005). Tais métodos podem ser supervisionados, não-supervisionados ou mesmo semi-supervisionados, conforme a disponibilidade dos rótulos dos exemplos na base selecionada.

A quinta etapa é uma análise exploratória, na qual o resultados dos modelos são

interpretados e são identificados os ganhos de conhecimento gerados por eles (ZHANG; ZHANG; YANG, 2003). Nesse ponto há a seleção dos melhores modelos para a hipótese discutida no início do processo do KDD.

Após a aplicação de todas as etapas é feita a implementação do algoritmo para mineração dos dados. Assim, poderão ser encontrados padrões de interesse em uma forma particular de representação ou, então, um conjunto dessas representações. Nesse ponto, às vezes, é necessário aplicar o algoritmo várias vezes, ajustando os seus parâmetros e analisando os resultados através de medidas de avaliação (MAIMON; ROKACH, 2005).

Há, ainda, a necessidade de visualizar, interpretar e avaliar os dados minerados. Nesse ponto, o resultado começa a aparecer de fato. Porém, possivelmente será necessário retornar em uma das etapas anteriores de forma iterativa, a fim de executar ajustes para melhorar a interpretabilidade dos resultados (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996b).

2.2.1 Taxonomia em Mineração de dados

Em geral, os métodos de mineração de dados são divididos em supervisionados e não supervisionados, de acordo com a presença ou não de rótulos para a geração de modelos. A Figura 6 mostra uma taxonomia que categoriza as tarefas de mineração de dados em supervisionados e não-supervisionados.

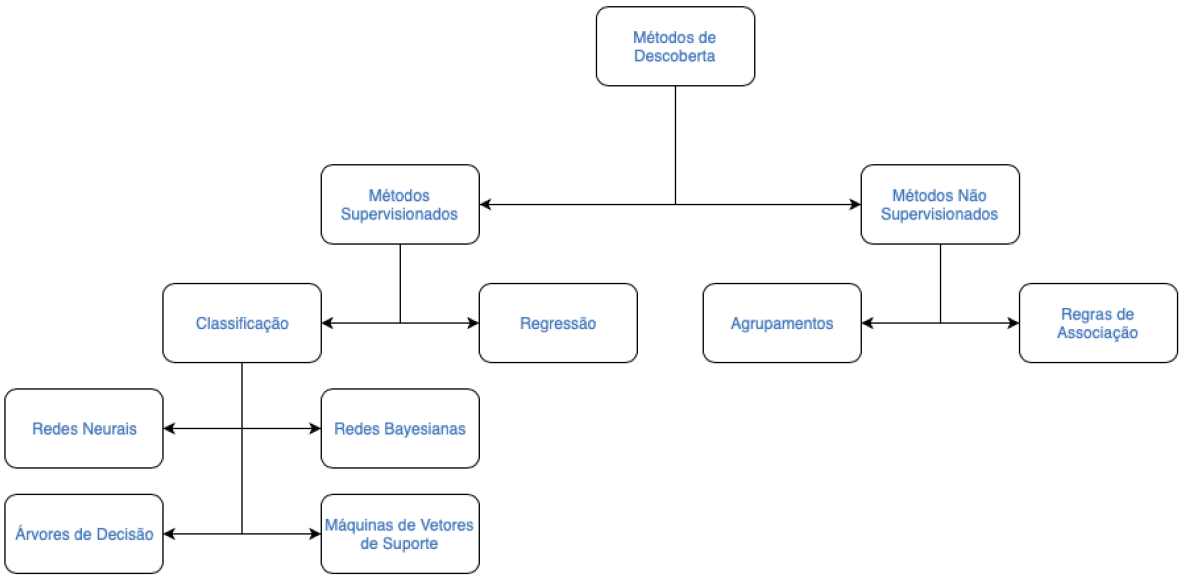


Figura 6 – Taxonomia de Métodos para a descoberta de conhecimento - Adaptado de Maimon e Rokach (2005).

Os métodos supervisionados visam construir um modelo de comportamento padrão para os dados de entrada prevendo, assim, o resultado de uma ou mais variáveis de acordo com uma base de treinamento na qual há rótulos para as instâncias. Em particular, a aprendizagem supervisionada é composta por métodos que buscam descobrir o

relacionamento entre os atributos de entrada (variáveis independentes) com um atributo alvo (variável dependente). Há duas subdivisões quanto aos métodos supervisionados, os modelos de classificação e de regressão (MAIMON; ROKACH, 2005).

Já os métodos não-supervisionados utilizam algoritmos de aprendizado de máquina para analisar e agrupar conjuntos de dados não rotulados. Esses algoritmos buscam descobrir padrões ocultos ou agrupamentos de dados de forma autônoma, através de semelhanças e diferenças nas informações dos conjuntos de dados. Porém, para a interpretação do resultado final, há a necessidade de um especialista de domínio (MAIMON; ROKACH, 2005). Este trabalho irá focar nos métodos de classificação, que serão melhor explicados na próxima seção.

2.2.2 Métodos de classificação

Problemas de classificação utilizam métodos que analisam os dados de entrada, os quais possuem sua classe rotulada. Dessa forma, conseguem aprender modelos de classificação, chamados também de classificadores ou modelos de decisão. Eles descrevem certas classes alvo e, assim, através desses modelos, conseguem estimar os rótulos das classes de instâncias ainda desconhecidas (AGGARWAL, 2015; HAN; PEI; KAMBER, 2011).

Genericamente, um problema de classificação pode ser definido como um conjunto de K instâncias n-dimensionais que possuem um atributo y chamado de rótulo. O objetivo é aprender uma função $y = f(x)$ (modelo de decisão) a partir desses exemplos e, então, irá predizer o rótulo y de exemplos futuros.

A detecção de intrusão em uma rede de computadores tem sido tratada na literatura como uma tarefa de classificação, onde as instâncias de dados representam os pacotes de dados trafegados na rede e as classes indicam se aquele pacote é classificado como malicioso (ataque) ou normal. Várias técnicas de aprendizado de máquina estão sendo utilizadas para implementar sistemas de detecção de intrusão (UHM; PAK, 2021), tais como: florestas aleatórias (BREIMAN, 2001), árvores de decisão (QUINLAN, 1986) e redes neurais profundas (LAROCHELLE et al., 2009).

Um dos métodos de classificação mais eficientes e amplamente utilizados na literatura são as árvores de decisão (LOH, 2014). Considerando a detecção de intrusão, vários trabalhos alcançaram bons resultados usando este tipo de classificador (SARKER et al., 2020; VIEGAS; SANTIN; OLIVEIRA, 2017; GU; LU, 2021; HSU et al., 2019). Classificadores desse tipo criam o modelos na forma de árvores, onde cada nó contém um teste em um atributo. Cada ramificação de um nó corresponde então a uma possibilidade de resultado do teste feito pelo nó e, então, cada folha contém uma classe y predita. A classificação de uma instância, é feita executando testes de forma recursiva em todos os nós, até o momento que atingirá o fim da árvore que receberá o rótulo representado pela folha.

2.3 Mineração em Fluxos Contínuos de Dados

Os algoritmos de mineração de dados evoluíram nas últimas duas décadas, partindo dos cenários conhecidos como *batch*, no qual o conjunto de dados era pequeno, sendo possível fazer várias varreduras na base e os modelos induzidos não precisavam ser retreinados. Atualmente, a mineração de dados enfrenta um cenário marcado pela dinamicidade, onde os dados são gerados de forma contínua. Deve-se então considerar que os dados estão em movimento, frequentemente em expansão e mudando suas propriedades ao longo do tempo (MORALES et al., 2016). Essas mudanças podem afetar, por exemplo, a segurança nas redes de computadores (FAISAL et al., 2015), uma vez que os padrões aprendidos sobre comportamentos normais e ataques nas redes podem se modificar.

Esses cenários dinâmicos, em que fluxos contínuos de dados são gerados, representam um grande desafio para a área de aprendizado de máquina, principalmente quanto a sua capacidade de manter um alto poder preditivo e operar em tempo real (CANO; KRAWCZYK, 2019). Para acomodar tais características, os fluxos contínuos de dados influenciam o desenvolvimento de novas famílias de algoritmos, capazes de promover integração contínua de novos dados ao mesmo tempo em que proporcionam robustez à sua natureza em evolução (CANO; KRAWCZYK, 2020).

Formalmente, um fluxo contínuo de dados é representado por um conjunto de dados multidimensional, potencialmente infinito $\overline{X}_1 ... \overline{X}_k ...$, chegando nos momentos $T_1 ... T_k ...$, nos quais cada $\overline{X}_i$ é uma instância d-dimensional que é denotada por $\overline{X}_i = (x_i^1 ... x_i^d)$ (AGGARWAL et al., 2003).

Segundo Faria, Carvalho e Gama (2015), a mineração em fluxo contínuo de dados é a extração de conhecimento de grandes conjuntos de dados, gerados continuamente em um ambiente não-estacionário, isto é, em que a distribuição que gera os dados pode mudar. A abordagem de mineração em fluxos contínuos de dados oferece uma alternativa aos altos custos de armazenamento de dados para a análise posterior, pois essa avaliação pode ser feita em tempo real por algoritmos especificamente desenvolvidos para essa tarefa (AGGARWAL, 2015).

Ainda sobre os algoritmos de mineração de fluxos contínuos de dados, Aggarwal (2015) destaca que estes devem trabalhar sob as restrições descritas a seguir.

- **Restrição de varredura única**: devido ao alto volume de dados que são gerados continua e rapidamente, assume-se que o dado pode ser processado somente uma vez. Portanto, as informações nunca serão armazenadas para um processamento posterior.

- **Restrições de recursos**: devido à continuidade dos dados, que normalmente não estão sob o controle do usuário, podem ocorrer picos de processamento devido ao alto volume em que os dados chegam, não sendo possível assim, fazer todo o processamento em tempo real.

Knowledge... (2010) apresenta mais desafios quando se trata de fluxos contínuos de dados. O primeiro é a mudança contínua da distribuição em que eles são gerados, conhecida na literatura como mudança de conceito (do inglês *concept drift*). O segundo, conhecido como evolução de conceito, diz respeito ao surgimento de novas classes do problema ao longo do fluxo (MASUD et al., 2010). O presente trabalho enfatiza a mudança de conceito, dado o comportamento dos pacotes na comunicação através da Internet. Considerando que uma instância $\overline{X}_t$ é gerada por uma distribuição de probabilidade D_t, formalmente, uma mudança de conceito ocorre quando, para quaisquer exemplos x_1^d e x_2^d, no tempo T_1 e T_2, a distribuição $D_1 \neq D_2$ (FARIA; CARVALHO; GAMA, 2015). Na literatura, encontram-se dois tipos distintos de mudança de conceito, o abrupto e o incremental (TSYMBAL, 2004). No cenário abrupto, a mudança ocorre de forma repentina, como por exemplo o aumento de acesso a um *site* de compras de ingressos para uma determinada apresentação. Já o tipo incremental, ocorre de forma lenta, como por exemplo, os dados gerados por sensores a fim de analisar o desgaste de um equipamento em uma fábrica.

Além desses dois conceitos básicos, Gama et al. (2014) discutem ainda o tipo gradual, onde a mudança de conceito ocorre progressivamente voltando algumas vezes ao conceito anterior, até que se altere definitivamente. Há também a recorrência, onde em certas janelas de tempo a mudança de conceito ocorre como, por exemplo, compras em datas festivas como o natal.

A tarefa de classificação de fluxos contínuos de dados tem o mesmo objetivo que a tarefa de classificação em cenários *batch*, mas apresenta mais desafios pelo contexto de continuidade em que os dados são gerados. Os desafios estabelecidos são: a alta velocidade da chegada do fluxo, a quantidade de memória necessária para a classificação de um grande volume de dados e a mudança e evolução de conceito às quais os dados estão sujeitos. Diante de todos esses desafios, há a necessidade da utilização de várias técnicas exclusivas para fluxos contínuos de dados (GABER; ZASLAVSKY; KRISHNASWAMY, 2007).

2.3.1 Estratégias para Classificação de Fluxos Contínuos de Dados

Uma das estratégias comumente usada para classificação de fluxos contínuos de dados é usar um único classificador, o qual é atualizado ao longo do processamento de dados sempre que é detectada uma mudança de conceito na distribuição.

No entanto, vários trabalhos recentes têm utilizado uma estratégia diferente, na qual um comitê (conjunto) de classificadores (ROKACH, 2009) é utilizado ao invés de um único classificador. Os comitês permitem que os resultados individuais, de um conjunto de classificadores, sejam combinados de alguma maneira para classificar novas instâncias (GAMA et al., 2004). Estudos mostram que os comitês de classificadores, frequente-

mente, possuem uma acurácia maior que os classificadores individuais (DIETTERICH, 2000). Segundo Hansen e Salamon (1990), a condição necessária e suficiente para o comitê de classificadores ser mais preciso do que qualquer um de seus membros individuais, é que o comitê seja diversificado e com uma alta acurácia. Quanto à diversidade dos classificadores, diz-se que eles são diversificados se cometerem erros diferentes para novas instâncias (GAMA et al., 2004). Por fim, um classificador com alta acurácia é aquele que tem uma taxa de erro melhor do que a aleatoriedade para instâncias desconhecidas (DIETTERICH, 2000).

Para entender porque a alta acurácia e a diversidade entre os classificadores é uma boa opção, imagine um conjunto de três classificadores: $C1$, $C2$ e $C3$, e considere uma nova instância x. Se os três classificadores forem idênticos (ou seja, não diversos), quando $C1(x)$ estiver errado consequentemente $C2(x)$ e $C3(x)$ também estarão. No entanto, se os erros cometidos pelos classificadores não forem correlacionados, quando $C1(x)$ estiver errado, existe a probabilidade de $C2(x)$ e $C3(x)$ estarem corretos. Assim, uma simples votação majoritária entre os classificadores $C1$, $C2$ e $C3$ poderá classificar a instância x corretamente.

Dentre as técnicas de comitês de classificadores duas se destacam: o *boosting* e o *bagging*. Como pode ser visto na Figura 7, a principal ideia por trás da técnica de *boosting* consiste em melhorar a acurácia de um classificador (KRAWCZYK et al., 2017), mantendo um peso para cada instância no conjunto de treinamento que reflita sua importância. Ao final da indução de cada classificador base, quando uma instância é classificada incorretamente seu peso é incrementado. O ajuste dos pesos das instâncias, faz com que o classificador se concentre em diferentes exemplos que levam a diferentes resultados. O classificador final agrega os resultados em cada iteração por votação ponderada, onde o peso de cada um deles é uma função de sua acurácia.

Já o método *bagging* pode ser visto na Figura 8, consiste em construir um classificador a partir de uma amostra de dados que contenha o mesmo número de instâncias do conjunto de treinamento original, mas que foram selecionadas de forma aleatória e que podem ser repostas ao longo do processo. Assim, uma mesma instância pode aparecer mais de uma vez numa mesma amostra ou aparecer em apenas algumas delas. Cada amostra é usada para treinar um classificador diferente, elas são formalmente chamadas de replicação *bootstrap* do conjunto de treinamento original. A técnica primária implica em uma replicação que contém, em média, 63% do conjunto de treinamento original, assim é possível que uma mesma instância apareça várias vezes no conjunto de treinamento (BREIMAN, 1996; DIETTERICH, 2000; KNOWLEDGE. . . , 2010).

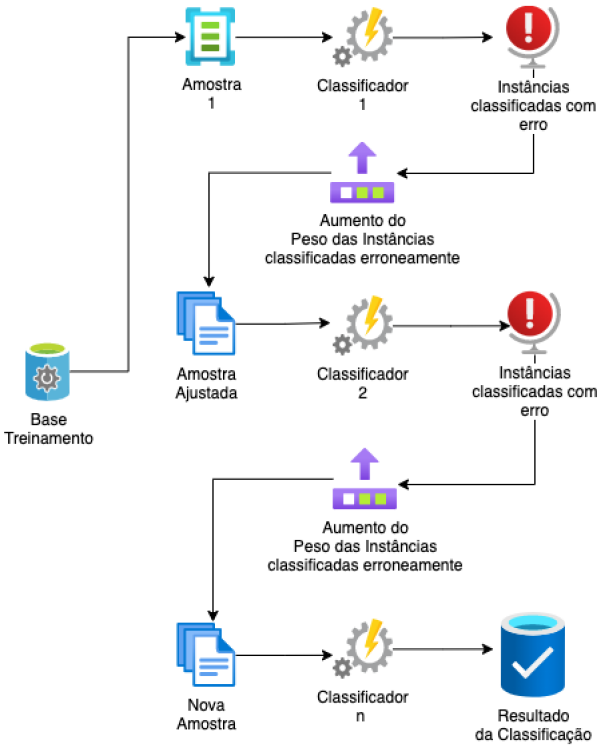


Figura 7 – Exemplo do funcionamento da técnica de *boosting* (Fonte: O autor (2021)).

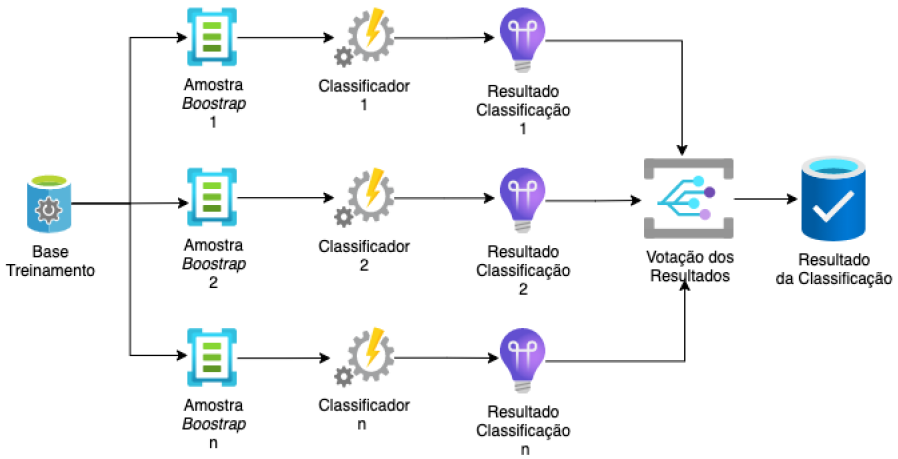


Figura 8 – Exemplo do funcionamento da técnica de *bagging* (Fonte: O autor (2021)).

2.3.2 Algoritmos para Classificação de Fluxos Contínuos de Dados

Ainda que vários algoritmos para classificação de fluxos contínuos de dados tenham sido propostos, esta seção irá descrever os algoritmos de fluxos contínuos de dados comumente usados na literatura de detecção de intrusão em redes de computadores (VIEGAS et al., 2019; FAISAL et al., 2015). Em geral, estes trabalhos usam estratégias diversas, sejam aquelas baseadas em único classificador ou então as que utilizam comitês de classificadores. Adicionando, em ambos os casos, técnicas que permitem identificar e reagir a mudanças de conceito nos dados. Em sua maioria, esses trabalhos usam como algoritmo

de classificação base a *Hoeffding Tree* (DOMINGOS; HULTEN, 2000). Assim, nos comitês, um conjunto de classificadores do tipo *Hoeffding Tree* pode ser utilizado, já na técnica classificador único uma única *Hoeffding Tree* é usada com um mecanismo de detecção de mudança.

Esta seção irá inicialmente apresentar o algoritmo de classificação base *Hoeffding Tree* e a seguir os seguintes algoritmos de classificação usados neste trabalho: *Single Classifier Drift* que baseia-se em classificador único e; *Accuracy Updated Ensemble*; *Leveraging Bagging*; *Limited Attributes Classifier*; *Oza Bagging with ADWIN* e; *Oza Bagging with Adaptive Size Hoeffding Tree* que são baseados em comitês de classificadores.

2.3.2.1 *Hoeffding Tree*

É um classificador baseado em árvores de decisão e que classifica fluxos contínuos de dados utilizando o *Hoeffding bound* (HOEFFDING, 1994). Quando o número de exemplos em um nó da árvore satisfaz o *Hoeffding bound*, o melhor atributo disponível é escolhido como um nó de divisão da árvore. Nesse algoritmo não é necessário nenhum tipo de armazenamento, pois as instâncias são lidas somente uma vez permitindo que a predição ocorra a qualquer momento. O algoritmo também possui a capacidade de adaptar o modelo em caso de alterações na distribuição dos dados (LEMAIRE; SALPERWYCK; BONDU, 2015).

Para um melhor entendimento, o Algoritmo 1 detalha o funcionamento do classificador *Hoeffding Tree*. Ele consiste em gerar, inicialmente, uma árvore com uma única folha (Algoritmo 1 linha 11). É necessário também uma função que analisará o ganho de informação $\overline{G}$ (Algoritmo 1 linha 12). Um exemplo de função ganho de informação pode ser a entropia que é dada pela equação:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i), \tag{1}$$

onde p é a probabilidade de observar o valor x_i.

Após a árvore ser inicializada, para cada instância é executado o procedimento descrito a seguir. Cada instância é propagada pela árvore, executando a análise de cada nó até que a instância seja classificada ao atingir uma folha. Nesse momento, as estatísticas disponíveis da folha são atualizadas (Algoritmo 1 linha 20). Após a classificação, se as instâncias analisadas até o momento não são de uma mesma classe, é utilizada a função de ganho de informação $\overline{G}$ para cada uma das classes. Duas estatísticas importantes são extraídas, a primeira é o atributo com o maior ganho de informação $\overline{G}_{l1}$ (Algoritmo 1 linha 26) e a segunda é o atributo com o segundo maior ganho de informação $\overline{G}_{l2}$ (Algoritmo 1 linha 27). Posteriormente, o cálculo do *Hoeffding Bounds* é feito (Algoritmo 1 linha 28) e então, se o ganho de informação $\overline{G}_{l1}$ menos $\overline{G}_{l2}$ for maior que o *Hoeffding Bounds* calculado, é feita uma divisão do nó gerando duas novas folhas e as suas estatísticas são

atualizadas. Ao final do processamento de todas as instâncias, uma árvore com o modelo criado é retornada.

Algoritmo 1 Pseudocódigo do *Hoefdding Tree*. Adaptado de Domingos e Hulten (2000)

Entrada
  S sequência de instâncias
  $\mathbf{X}$ conjunto de atributos discretos
  $G(.)$ avalia o ganho de informação
  δ probabilidade mínima desejada para seleção do atributo do nó

Saída
  HT árvore de decisão gerada

HoeffdingTree$(S, \mathbf{X}, G, \delta)$
  Sendo HT uma árvore com uma única folha l_1 (raiz)
  Sendo $X_1 = X \cup \{X_\emptyset\}$
  Sendo $\overline{G}_1(X_\emptyset)$ o ganho de informação $\overline{G}$ obtido pela previsão da classe mais frequente em S
  Para cada classe y_k **Faça**
    Para cada valor x_{ij} de cada atributo $X_i \in X$ **Faça**
      $n_{ijk}(l_1) = 0$
  Para cada exemplo $(x, y_k) \in S$ **Faça**
    Insira (x, y) em uma folha l usando HT
    Para cada x_{ij} em $x \mid X_i \in X_l$ **Faça**
      Incremente $n_{ijk}(l)$
    Classifica l com a classe majoritária entre os exemplos vistos até o momento em l
    Se os exemplos observados até o momento em l não são todos da mesma classe **Então**
      Calcula o ganho de informação $\overline{G_l}(X_i)$ para cada atributo $X_i \in X_l - \{X_\emptyset\}$ usando a contagem de $n_{ijk}(l)$
      Dado que X_a é o atributo com o maior ganho de informação $\overline{G_l}$
      Dado que X_b é o atributo com o segundo maior ganho de informação $\overline{G_l}$
      Hoeffding Bounds calculado por $\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$
      Se $\overline{G}_l(X_a) - \overline{G}_1(X_b) > \epsilon$ e $X_a \neq X_\emptyset$ **Então**
        Atualize l por um nó interno que foi dividido em X_a
        Para cada ramificação da divisão **Faça**
          Adicione uma nova folha l_m e então $X_m = X - \{X_a\}$
          Sendo $\overline{G}_m(X_\emptyset)$ seja o $\overline{G}$ obtido na predição da classe mais frequente em l_m
          Para cada classe y_k e valor $x_i j$ do atributo $X_i \in X_m - \{X_\emptyset\}$ **Faça**
            $n_{ijk}(l_m) = 0$
  Retorna HT

2.3.2.2 *Single Classifier Drift* (*SingleClassifierDrift*)

O *Single Classifier Drift* é uma estratégia baseada em apenas um classificador (FAISAL et al., 2015). Devido à presença de mudança de conceito nos dados, o modelo de classificação precisa evoluir e, para identificá-las além de indicar os momentos em que o modelo precisa ser atualizado, vários métodos de detecção de mudança podem ser utilizados. Um deles é o DDM (do inglês *Drift Detection Method*) que também foi usado neste trabalho. Esse método foi empregado em outros trabalhos de detecção de intrusão, como Faisal et al. (2015), Faisal et al. (2012) e Gomes et al. (2014).

O **DDM** (GAMA et al., 2004), funciona com o controle do número de erros produzidos pelo modelo de aprendizagem durante a predição. Duas janelas são comparadas, onde a primeira contém todos os dados e a segunda contém apenas os dados do início até o momento em que o número de erros aumenta. Ao invés de armazenar essas janelas na memória, esse método apenas mantém as estatísticas necessárias e uma janela de erros recentes. Além disso, esse método considera que o número de erros em uma amostra de instâncias é modelado por uma distribuição binomial. Um aumento considerável no erro do algoritmo sugere que a distribuição de classes está mudando e, portanto, o modelo de decisão original é considerado inadequado. Indicadores de nível de mudança de conceito e de contexto são consideradas para a atualização do modelo.

2.3.2.3 *Accuracy Updated Ensemble* (*AccUpdatedEnsemble*)

Em um processo de classificação de fluxos contínuos de dados utilizando comitês, os classificadores são constantemente avaliados e, quando necessário, são atualizados posteriormente, removidos ou alterados de acordo com a avaliação proposta. Um exemplo desse tipo de algoritmo é o *Accuracy Weighted Ensemble* (AWE) (WANG et al., 2003). Contudo, esse algoritmo possui um problema para definir o tamanho do bloco de instâncias utilizados para o treinamento do modelo, impactando diretamente em sua acurácia (BRZEZIŃSKI, 2010).

Uma versão aprimorada do AWE (WANG et al., 2003) é o *AccUpdatedEnsemble* que o aprimora, classificando os atributos de forma *online* atualizando os classificadores básicos em vez de somente ajustar seus pesos, e também os atualiza de acordo com a distribuição mais recente. O *AccUpdatedEnsemble* determina o peso para o classificador de atributos correspondente medindo a taxa de erro no bloco de dados mais recente. Com isso, é possível atualizar os classificadores do comitê de acordo com a distribuição atual. Os membros do comitê são classificados pelo seu peso e podem ser removidos sempre que necessário, com isso nem sempre eles serão atualizados, ao contrário do que ocorre com o *bagging* (KNOWLEDGE..., 2010). É considerável compreender que não há limites ao tamanho do classificador base, também não é utilizado nenhum tipo de janela e a atualização dos membros do comitê somente é feita se eles forem precisos o suficiente

para a distribuição atual.

2.3.2.4 *Leveraging Bagging* (*LeveragingBag*)

Tomando como base o método de *bagging*, Oza e Russell (2001) propuseram o *Online Bagging*, que é um método *online* que, ao invés de amostrar com reposição, dá a cada exemplo um peso de acordo com a distribuição de *Poisson*. Considerando então o *Online Bagging*, Bifet, Holmes e Pfahringer (2010) propuseram o *LeveragingBag* com a intenção melhorar o método que trabalha a aleatoriedade das entradas das instâncias.

Embora a re-amostragem do *Online Bagging* seja feita utilizando a distribuição de *Poisson*, Bifet, Holmes e Pfahringer (2010) sugeriram aumentar os pesos da re-amostragem aumentando o valor de λ para calcular a distribuição. Em um segundo aprimoramento, a aleatoriedade foi adicionada na saída do conjunto usando estratégias para correção de erros. Uma *string* binária de comprimento n é atribuída a cada classe e, portanto, um conjunto de n classificadores binários são construídos. Cada classificador aprende um *bit* para cada posição nesta *string* binária. Quando uma nova instância x chega, ela é atribuída à classe com o código binário mais próximo. Um código de correção de erros pode ser visto como uma forma de votação, em que um número de votos incorretos pode ser compensado. A principal motivação para usar código aleatório em vez de código determinístico é que cada membro do comitê irá prever uma função diferente podendo reduzir assim, os efeitos de correlação entre os classificadores. Posteriormente, a diversidade do conjunto será aumentada e cada classificador m e classe c são atribuídos valores binários de uma forma uniforme, independente e aleatória. Metade exata das classes é mapeada para 0. A saída do classificador para uma instância é a classe que tem mais votos de suas classes do mapeamento binário.

Por fim, para lidar com a mudança de conceito, o *Adaptive Windowing* (ADWIN) (BIFET; GAVALDà,) foi usado. ADWIN é um detector de mudanças e um classificador que resolve especificamente o problema de rastreamento da média de um fluxo de *bits*. O ADWIN funciona, mantendo uma janela de comprimento variável dos atributos vistos recentemente, com a propriedade de que a janela tem o comprimento máximo estatisticamente consistente com a hipótese de não ter ocorrido alterações nos valores médios da janela. Ele é capaz de detectar e se adaptar automaticamente a taxa de mudança atual utilizando um único parâmetro, que configura o limite de confiança δ que indica o quão confiante será a saída do algoritmo. Em geral, esta estratégia é característica de algoritmos que lidam com processos aleatórios. Dentre diversos classificadores base, o *Leveraging Bagging* também aceita o *Hoeffding Tree*, que é uma ótima opção para a construção de um classificador utilizado em IDSs.

2.3.2.5 *Limited Attributes Classifier* (*LimAttClassifier*)

A proposta de Bifet et al. (2010) chamada *Limited Attributes Classifier*, consegue produzir um modelo de classificação baseado em um comitê de classificadores de árvores de decisão restritas, onde cada árvore é produzida a partir de um sub-conjunto distinto de atributos. O modelo final é formado pela combinação das probabilidades logarítmicas da classe preditas nessas árvores, usando *perceptrons* sigmoides com um *perceptron* por classe. Em contraste com a abordagem de *boosting* padrão, que forma um comitê de classificadores utilizando um algoritmo guloso, construindo cada árvore em sequência e atribuindo pesos correspondentes como um subproduto, o método proposto gera cada árvore em paralelo e as combina usando *perceptron*, adotando a abordagem de empilhamento (ou *stacking* em inglês). Como o cenário é um fluxo contínuo de dados, as *Hoeffding Tree* são utilizadas como os membros do comitê de classificadores. A vantagem de utilizá-las é que, assim como os *perceptrons*, elas podem ser treinadas incrementalmente utilizando o ADWIN como detector de mudança de conceito.

2.3.2.6 *Oza Bagging with ADWIN* (*OzaBagAdwin*)

Em seu trabalho, Oza e Russell (2001) propõem a construção de um algoritmo de *bagging* com funcionamento *online*, ou seja, processa cada uma das instâncias de treinamento no momento em que chega ao classificador, sem a necessidade de amarzená-la para então executar o reprocessamento. Também mantém o atual modelo, que reflete todas as instâncias processadas até o momento.

Um versão *online* do algoritmo de *bagging* requer uma maneira de espelhar suas técnicas para gerar múltiplos modelos base. Porém, uma dificuldade é a necessidade de se prever o tamanho do conjunto de treinamento, o que no caso de contexto de fluxo contínuos de dados não é uma tarefa trivial. Tem-se como exemplo o trabalho de re-amostragem do conjunto de treinamento com tamanho n para então, produzir m conjuntos de treinamento para o *bootstrap* com tamanho n, onde cada um deles é utilizado por um classificador base. A proposta de Oza e Russell (2001) visa treinar todos os m classificadores base de forma *online* de forma que, para isso, ele envia para os classificadores base K cópias de cada um das novas instâncias de treinamento para atualizar o classificador base, onde K é uma variável aleatória baseada na distribuição de *Poisson*. Esta ação gera um comportamento similar àquele visto no *bagging* em lotes, soma-se isso, a necessidade de identificar as mudanças de conceito para que possa executar a atualização dos classificadores base. Essa atualização é feita através do algoritmo ADWIN e o classificador base utilizado é a *Hoeffding Tree*. Quando uma mudança é detectada, o pior classificador do comitê é removido e um novo é adicionado ao grupo.

2.3.2.7 *Oza Bagging with Adaptive Size Hoeffding Tree* (*OzaBagASHT*)

Diversos estudos focaram-se no desenvolvimento de adaptações dos algoritmos de *bagging*, como por exemplo Bifet et al. (2009a), Bifet et al. (2009b) e Gomes et al. (2017). Diante dos desafios em fluxos contínuos de dados, Bifet et al. (2009a) propôs um algoritmo de *bagging* com *Hoeffding Trees* de diferentes tamanhos chamado de "*Hoeffding Trees* de tamanho adaptável" (em inglês *Adaptive Size Hoeffding Tree* (ASHT)).

As *Hoeffding Trees*, por serem um algoritmo de indução de árvores de decisão incremental, são capazes de aprender a partir de fluxos de dados massivos. Porém, em sua forma tradicional, assumem que as instâncias chegam em uma distribuição constante e não se alteram no tempo. Por outro lado, as *Hoeffding Trees* exploram o fato de que uma pequena amostra pode, muitas vezes, ser suficiente para selecionar um atributo para a divisão ideal dos nós da árvore de decisão (BIFET et al., 2009b).

As principais diferenças da implementação do *Oza Bagging with ASHT* são: um número máximo de divisões ou um tamanho máximo e; após a divisão de um nó, se a quantidade total for maior do que o tamanho máximo da árvore, ele exclui alguns nós para reduzir seu tamanho. A ideia por trás desse método é: árvores menores se adaptam mais rapidamente às mudanças e árvores maiores se saem melhor durante períodos com nenhuma ou pouca mudança, simplesmente porque foram construídas com mais dados. Árvores limitadas a tamanhos s serão redefinidas com cerca de duas vezes mais frequência do que árvores com tamanho limite de $2s$. Isso cria um conjunto de diferentes velocidades de reinicialização para um conjunto de árvores e, portanto, um subconjunto que são uma boa aproximação para a taxa de mudança atuante no momento. É importante ressaltar que as redefinições acontecerão o tempo todo, mesmo para conjuntos de dados estacionários, mas esse comportamento não deve ter um impacto negativo no desempenho preditivo do conjunto como um todo (BIFET et al., 2009b). Então, quando uma árvore excede o seu tamanho máximo, existem duas opções: os nós mais antigos são deletados, a raiz e todos os seus filhos, exceto onde a divisão foi feita e assim, o nó não excluído se torna a nova raiz e; todos os nós são deletados e uma nova árvore se inicia. O fato de se diversificar as árvores causa um impacto positivo no desempenho preditivo geral do comitê de classificadores (BIFET et al., 2009a).

2.3.3 Aprendizado Ativo

Os algoritmos para classificação de fluxos contínuos de dados precisam constantemente atualizar o modelo de decisão a fim de tratar os fenômenos de mudança e evolução de conceito. Muitos desses algoritmos atualizam seus modelos de forma supervisionada e, para isso, supõem que o rótulo de todas as instâncias estará disponível para atualização do modelo. Assim, eles consideram que após uma instância ser classificada, o seu rótulo estará imediatamente disponível para atualização do modelo de classificação.

A tarefa de rotular as instâncias de um fluxo, em geral, é feita por um especialista de domínio. Considerando o alto volume de dados em um tráfego de uma rede de computadores, rotular todas as instâncias torna-se uma tarefa inviável, já que é extremamente custosa, cansativa e exige a presença de um especialista avaliando milhões de instâncias. Ainda que a ideia de rotular todas as instâncias de um fluxo de dados pareça ser uma proposta inviável para problemas reais, vários trabalhos de detecção de intrusão usaram tal proposta, tais como Yang et al. (2018), Kumari e Varma (2017), Li e Guo (2007). Para Zhu et al. (2007), considerando cenários de mudanças de conceito, estratégias diferentes do aprendizado tradicional devem ser utilizadas para para amenizar os problemas descritos.

Recentes algoritmos de classificação de fluxos contínuo de dados (como Faria, Carvalho e Gama (2015), Mohamad, Sayed-Mouchaweh e Bouchachia (2018)) têm utilizado a estratégia de aprendizado ativo com o objetivo de tornar os algoritmos mais apropriados para problemas do mundo real. O aprendizado ativo é uma alternativa para a solução de muitos problemas modernos de aprendizado de máquina, em que dados não rotulados podem ser abundantes, mas os rótulos são difíceis, demorados ou caros de se obter (SETTLES, 2009). A ideia principal por trás do aprendizado ativo é que um algoritmo de aprendizado de máquina pode alcançar maior precisão com menos instâncias de treinamento rotuladas mas, para isso, é necessário escolher as melhores instâncias para o seu treinamento (SETTLES, 2009).

A Figura 9, ilustra o ciclo de aprendizado ativo baseado em um conjunto de instâncias. Um classificador pode começar com um pequeno número de instâncias rotuladas no conjunto de treinamento, solicitar ao oráculo (que, em geral, é um especialista de domínio) o rótulo para uma ou mais instâncias cuidadosamente selecionadas, aprender com as novas instâncias rotuladas e, em seguida, aproveitar seu novo conhecimento para escolher quais instâncias consultar em seguida. Depois que uma consulta é feita, geralmente não há suposições adicionais por parte do classificador, pois a nova instância rotulada é simplesmente adicionada ao conjunto rotulado e o classificador prossegue a partir daí de uma forma supervisionada.

Existem vários cenários onde o aprendizado ativo pode fazer perguntas sobre quais instâncias rotular, também existem diversas estratégias de consultas diferentes que podem ser usadas para decidir quais instâncias são mais informativas (SETTLES, 2011), sendo que, as estratégias convencionais de aprendizado ativo, concentram-se em consultar as instâncias mais incertas da base (SETTLES, 2009).

Em fluxos contínuos de dados, o objetivo do aprendizado ativo é maximizar a precisão da predição ao longo do tempo, mantendo os custos de rotulagem fixos dentro de uma expectativa da quantidade de instâncias rotuladas (ZHU et al., 2007). Žliobaitė et al. (2011) afirma que estratégias de aprendizagem ativa em fluxos de dados, além de serem capazes de gerar um classificador com uma alta acurácia em situações estacionárias, precisam ser

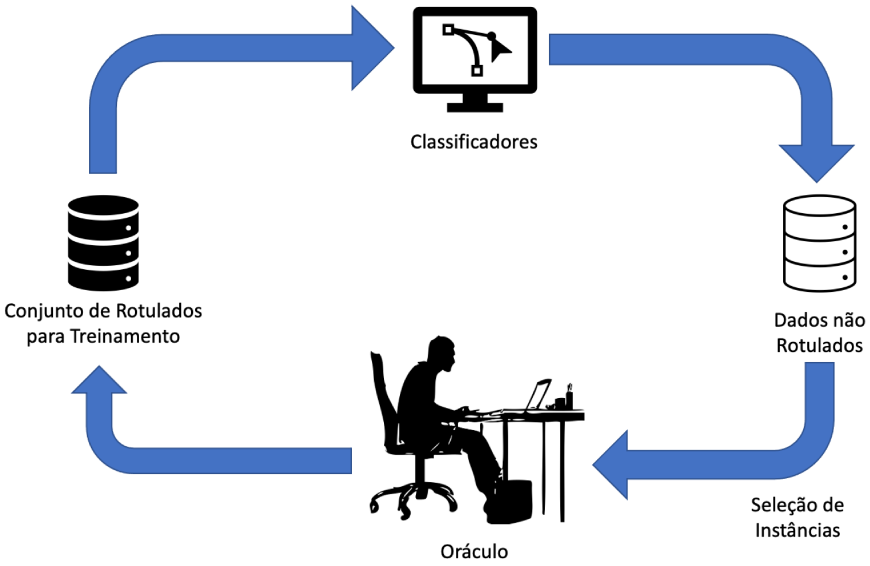


Figura 9 – Exemplo do funcionamento da técnica de aprendizado ativo. Adaptado de Settles (2009).

capazes de: i) equilibrar a expectativa da quantidade de instâncias que serão rotuladas ao longo do tempo; ii) observar as mudanças que podem ocorrer em qualquer lugar no espaço de soluções e; iii) preservar a distribuição dos dados recebidos para detectar mudanças de conceito.

O trabalho de Žliobaitė et al. (2011), apresenta algumas estratégias para seleção de instância a serem rotuladas, elas desenvolvidas para suprir as demandas citadas anteriormente. As mais relacionadas a este trabalho são descritas:

- A amostragem aleatória é uma estratégia que rotula as instâncias que chegam ao classificador de forma aleatória, em vez de decidir ativamente qual rótulo seria mais relevante. Para cada instância que chega ao classificador, o rótulo verdadeiro é solicitado com uma probabilidade $1/B$, onde B é a quantidade prevista de instâncias que serão rotuladas, aqui chamada de percentual de rotulagem.

- A amostragem por meio de incerteza fixa utiliza a ideia de rotular as instâncias em que o classificador possui menos confiança onde, em uma configuração online, ela corresponde a rotular as instâncias para as quais a certeza está abaixo de um limite pré-estabelecido. Uma maneira simples de medir a incerteza é usar estimadores de probabilidade *a posteriori* com o resultado do classificador. Assim, é introduzido o limite variável que é ajustado conforme a chegada de novos dados. Se um classificador aumenta sua confiança, o limite se expande para ser capaz de capturar o maior número de instâncias incertas. Se uma mudança acontecer e, de repente, aparecerem muitas solicitações de rotulagem, o limite é contraído para consultar primeiro as instâncias mais incertas.

A Figura 10, mostra um exemplo de funcionamento do aprendizado ativo e da quota de

rotulagem. Dentro do espaço amostral, que é o espaço entre todas as possíveis instâncias para uma dada tarefa de classificação (Figura 10(a)), existem as instâncias da classe de *Ataque* (que, no exemplo, pode ser atribuída à classe A) e as instâncias *Normais* (que pode ser atribuída à classe B). Suponto que a quota é um parâmetro que será configurado, em um primeiro caso, a quantidade selecionada para rotulagem é n (Figura 10(b)). Assim, o total de instâncias possíveis de serem rotuladas representa n/i onde i é o total de instâncias. O segundo exemplo, indica que o percentual de rotulagem das instâncias é $n*2$ e, assim, a quantidade de instâncias será representada por $n*2/i$ instâncias rotuladas, como pode ser observado na Figura 10(c). Finalmente, se o percentual sofrer um aumento na ordem de n^2 como visto na Figura 10(d), o número de instâncias rotuladas será ainda maior, trazendo a possibilidade de mais instâncias selecionadas, porém, terá um maior custo para que essa rotulagem seja executada.

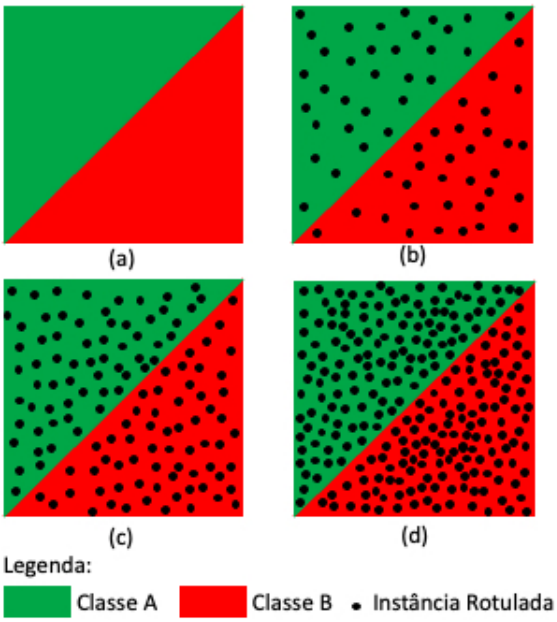


Figura 10 – Aprendizado Ativo (a) espaço amostral; (b) n instâncias rotuladas; (c) $n*2$ instâncias rotuladas e (d) n^2 instâncias rotuladas. Adaptado de Žliobaitė et al. (2011)

2.4 Método de Avaliação

Um ponto chave para os algoritmos de aprendizado de máquina é a estratégia usada para separar os dados de treinamento e teste, assim como os indicadores de desempenho preditivo aplicados aos classificadores. Nessa seção são apresentadas os métodos de avaliação, além dos indicadores de desempenho preditivo utilizados na literatura.

2.4.1 Métodos de Avaliação de Fluxos de Dados

Em um contexto estacionário, ou seja, onde a distribuição que gera os dados não sofre alterações, costumam-se usar duas técnicas para separar a base de dados nos conjuntos de treinamento e teste. A primeira chamada de *hold-out*, como pode ser visto na Figura 11(a) simplesmente divide o total de instâncias disponíveis em dois grupos um de treinamento e outro de testes. Ao final, uma avaliação de desempenho é calculada usando conjunto de teste e o modelo induzido a partir do conjunto de treinamento. Já na segunda estratégia, chamada de validação cruzada (do inglês *cross-validation*), exemplificada na Figura 11(b), o conjunto total de instâncias é subdividido em k conjuntos menores ou pastas (para ao exemplo da Figura 11(b) $k = 10$ pastas). A cada rodada, uma pasta é utilizada para o teste e as demais para o treinamento do modelo, até que todos os subconjuntos tenham sido utilizados. A medida de desempenho preditivo é então calculada como a média dos valores auferidos em cada rodada.

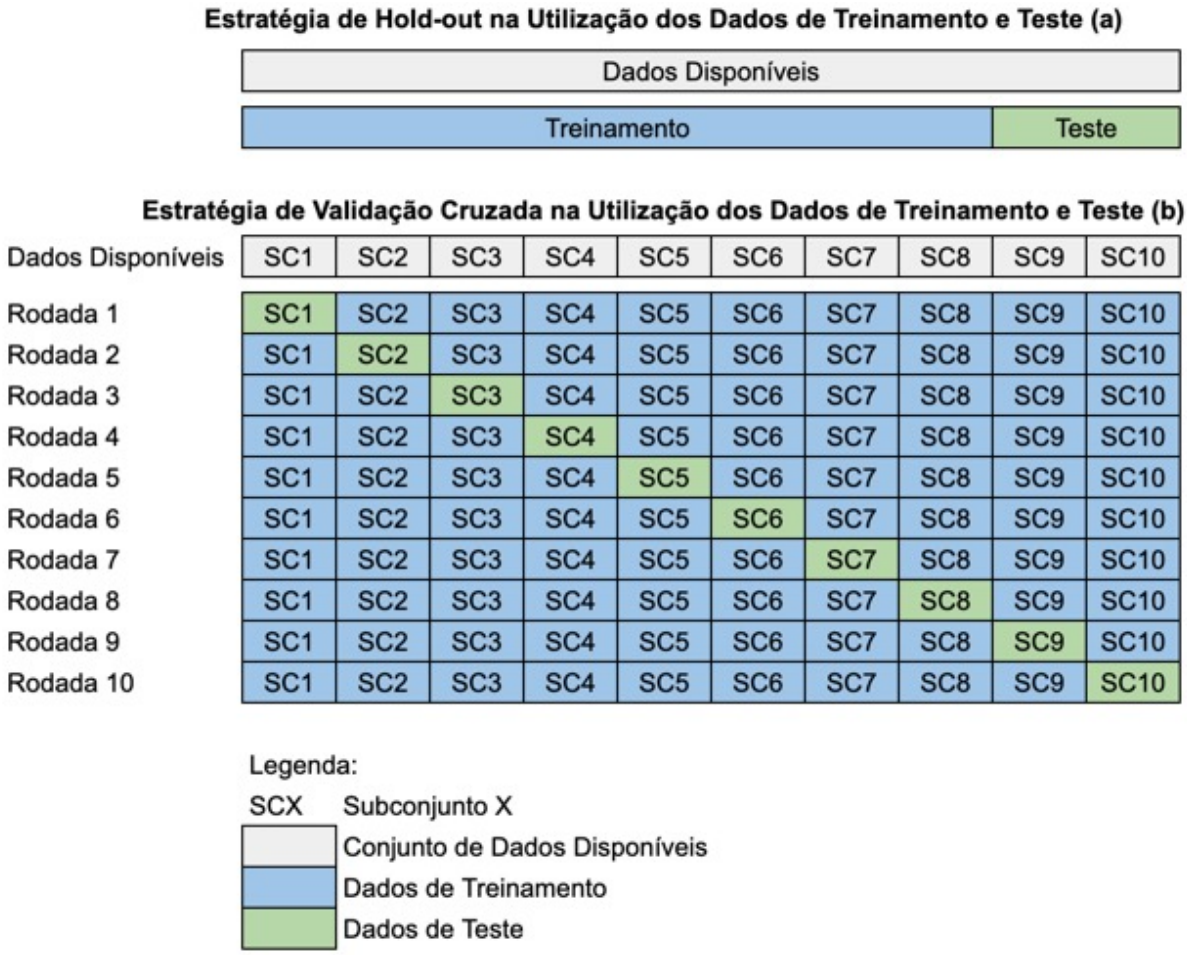


Figura 11 – Exemplo da Estratégia Hold-out (a) e Validação Cruzada (b). Adaptado de Pedregosa et al. (2011).

Considerando o cenário de fluxos contínuos de dados, onde novos dados estão continuamente chegando e o modelo de decisão está sendo constantemente atualizado, diferentes

métodos são usados para separar os dados de treinamento e teste. Dois métodos se destacam na literatura: o de fragmentos intercalados e o de sequenciamento preditivo.

O método de fragmentos intercalados (do inglês *Interleaved Chunks*), utiliza fragmentos da base de testes (*chunks*) na qual o modelo de decisão é aplicado em intervalos de tempos regulares, primeiro para testar e depois para treinar o modelo (CARELA-ESPAÑOL et al., 2016). Já a técnica de sequenciamento preditivo (do inglês *Prequential*) consiste em utilizar cada exemplo individualmente para testar antes de treinar o modelo (BIFET et al., 2010). O erro é calculado frequentemente, com base em uma soma acumulada de uma função de perda entre a predição e os valores reais observados na saída do modelo (KNOWLEDGE. . . , 2010).

2.4.2 Avaliadores de Desempenho Preditivo

O objetivo dos avaliadores de desempenho preditivo é calcular medidas que sejam capazes de proporcionar a comparação entre diferentes classificadores. É importante destacar que nem todas as métricas utilizadas em aprendizado de máquina são interessantes quando se trabalha com IDSs. Isso acontece porque o tráfego de rede é naturalmente muito desbalanceado, possuindo mais pacotes do tipo $Normal$ que *Ataques*. Assim, o uso de métricas como a acurácia, que foi utilizada em vários trabalhos sobre detecção de intrusão (JI et al., 2016; BARBARA; WU; JAJODIA, 2001; YIN et al., 2017), pode não refletir adequadamente o desempenho preditivo em cenários desbalanceados, não sendo recomendado nessas circunstâncias (AKOSA, 2017; BARBARA; WU; JAJODIA, 2001).

Em geral, para uma tarefa de classificação binária é utilizada a matriz de confusão para o cálculo dos avaliadores de desempenho preditivo. Cada linha da matriz indica a classe prevista e as colunas evidenciam a classe real da instância, conforme pode ser visto na Tabela 1. Assim, quando o modelo executa a classificação de uma instância, uma das células da matriz de confusão é atualizada. Em problemas de classificação binária e desbalanceados, a classe minoritária é considerada a classe positiva. Portanto, no problema de identificação de intrusão, as instâncias que são rotuladas como *Ataque* são interpretadas como instâncias positivas enquanto que, as instâncias rotuladas como $Normal$, são interpretadas como instâncias negativas.

- Verdadeiro Positivo (VP), quando o modelo executa corretamente a classificação de um pacote de ataque;
- Verdadeiro Negativo (VN), quando o modelo executa corretamente a classificação de um pacote normal;
- Falso Positivo (FP), quando o modelo classifica erroneamente uma instância como um ataque, quando na verdade é um pacote normal;

- ❑ Falso Negativo (FN), quando o modelo classifica erroneamente uma instância como um pacote normal, quando na verdade é um ataque.

Tabela 1 – Matriz Confusão para problemas binários. Fonte: O autor, 2021.

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

Os avaliadores mais comuns vistos na literatura sobre IDSs são: a revocação, a precisão e a taxa de alarmes falsos (em inglês *False Alarm Rate* - FAR).

A revocação, representada pela Equação 2, mede o quão bem o classificador identificou comportamentos anormais como *Ataque*.

$$Rev = \frac{VP}{VP + FN} \tag{2}$$

A precisão, dada pela Equação 3, quantifica o quanto de instâncias de *Ataque* foram classificadas corretamente dentre todas as que foram classificadas como *Ataque*.

$$Prec = \frac{VP}{VP + FP} \tag{3}$$

A Taxa de Alarme Falso (FAR) indica a proporção das observações de instâncias do tipo $Normal$ incorretamente sinalizadas como *Ataque*, que é dada pela Equação 4.

$$FAR = \frac{FP}{FP + VN} \tag{4}$$

De forma geral, o objetivo de um bom IDS é maximizar a revocação e a precisão além de minimizar o FAR.

2.5 Considerações Finais

Neste capítulo, foram abordados os conceitos necessários para o entendimento deste trabalho. Primeiramente foi apresentada uma visão geral sobre as redes de computadores, seus protocolos e sua segurança. Em seguida, foram explanados os conceitos e a taxonomia sobre descoberta de conhecimento em bases de dados. Também foram apresentados conceitos sobre métodos de classificação estáticos e em fluxo contínuo de dados, bem como exemplos de algoritmos de classificação usados na área e uma breve definição sobre aprendizado ativo e sua importância. Por fim, foram expostos os métodos de avaliação mais comuns na literatura, que serão também utilizados neste estudo.

No Capítulo 3, serão discutidos os trabalhos relacionados e o modo como esta pesquisa se relaciona com eles.

Capítulo 3

Trabalhos Relacionados

Neste capitulo, serão apresentados os principais trabalhos relacionados a esta pesquisa. O objetivo é fornecer uma visão geral sobre o estado da arte em métodos de detecção de intrusão baseados em anomalias e assinaturas usando tanto os dados dos pacotes individuais, quanto aqueles que utilizam os fluxos desses pacotes. Assim, na Seção 3.1, serão analisados os trabalhos que utilizam a detecção baseada em anomalias e aqueles que utilizam assinaturas. Já na Seção 3.2, serão analisados os trabalhos que utilizam a inspeção dos pacotes individuais ou o fluxo deles. Na Seção 3.3 serão analisados os trabalhos que utilizam algoritmos de fluxos contínuos de dados para a detecção de intrusão. Por fim, a Seção 3.4, fala sobre os trabalhos que utilizaram a base CICIDS2017 como fontes de informação para os seus métodos de classificação de tráfego malicioso.

3.1 Método de Detecção

A mudança constante das características dos ataques torna mais difícil resolver o problema de detecção de intrusão. Ferramentas tradicionais de detecção de intrusão, como o *Snort* (ROESCH et al., 1999) e *Bro* (PAXSON, 1999), baseados em regras, não são mais capazes de atender à crescente demanda por segurança de redes (ZHONG et al., 2020). Com isso, trabalhos recentes trazem alternativas aos métodos puramente baseados em regras.

Dentre os trabalhos analisados, Cheng et al. (2020) propõe um método de detecção de intrusão baseado exclusivamente na análise de assinatura dos pacotes. Em seu método, os autores desenvolveram um algoritmo capaz de extrair informações dos pacotes de rede para que a identificação de intrusão pudesse ser feita, classificando o tráfego através das assinaturas dos pacotes e dos protocolos. Além disso, as informações estatísticas coletadas são apresentadas de forma gráfica ao administrador de rede, e toda a parametrização dos indicadores estatísticos para análise são configuráveis. Porém, este trabalho traz limitações, como a necessidade de se identificar previamente o protocolo que será analisado. Também, é considerável a limitação do uso de análise estatística para a identificação de

um padrão de pacotes, o que pode ocasionar falhas do tipo falso negativo, pois caso o padrão da assinatura seja alterado a base necessita ser atualizada para atingir o objetivo proposto.

Alternativas a análise de assinaturas foram propostas em diversos trabalhos, por exemplo em (VIEGAS; SANTIN; OLIVEIRA, 2017). Nesse trabalho, os autores destacam o campo promissor da utilização de técnicas baseadas em anomalias. Eles ressaltam a importância de se usar bases de dados representativas e métodos de avaliação confiáveis que considerem as propriedades do mundo real de redes de comutadores. Diante disso, os autores propuseram três contribuições que podem auxiliar a construção dos IDSs baseados em anomalias. A primeira, é um método para a criação de banco de dados de intrusão baseado em ferramentas que visam produzir bancos de dados que podem ser facilmente atualizados, reproduzem tráfego real e válido, que sejam representativos quanto aos tipos de ataques. A segunda é um novo esquema de avaliação específico para o campo de detecção de intrusão, que permitiu a validação de cada uma das premissas comuns na literatura, como a detecção de novos eventos e novos serviços. Finalmente, apresentaram e avaliaram um método de seleção de características de objetivo múltiplo. A abordagem de avaliação permitiu, ao administrador de sistemas, estabelecer a capacidade real de um sistema para detectar cada uma das propriedades comuns em qualquer ambiente de produção. Apesar de apresentarem validações para identificar alterações nas bases analisadas, a construção do IDS utilizou algoritmos tradicionais para a classificação do fluxo de pacotes. O problema dessa abordagem está na necessidade de retreinar os algoritmos sempre que houver mudanças na distribuição dos dados.

Sarker et al. (2020) apresentam o *IntruDTree*, ele é um IDS que utiliza detecção baseada em anomalias utilizando técnicas de aprendizado de máquina, especificamente algoritmos baseados em árvores de decisão tradicionais. Na abordagem proposta primeiro considerou-se a classificação dos atributos de acordo com sua importância e, em seguida, foi treinado um modelo de detecção de intrusão generalizado baseado em árvore com base nos atributos selecionados. Finalmente, a eficácia do modelo *IntruDTree* foi examinada através da realização de uma série de experimentos em conjuntos de dados de segurança cibernética. Os resultados do modelo *IntruDTree* foram comparados com vários métodos tradicionais de aprendizado de máquina usados em IDS. Medidas de avaliação como acurácia, precisão, revocação e *F1-Score* foram utilizadas. Os experimentos são um ponto de atenção, pois os autores utilizaram uma base sintética que não reflete os principais e mais atuais ataques conhecidos.

Em Churcher et al. (2021), os algoritmos do estado-da-arte de aprendizado de máquinas são comparados em termos de acurácia, precisão, revocação, *F1-Score* e *Log-Loss*, utilizando o conjuntos de dados *Bot-IoT*(KORONIOTIS et al., 2019) balanceados e não-balanceados. Essa comparação é baseada na detecção de anomalias e tem sua importância, pois mostra que as árvores de decisão tradicionais possuem uma melhor acurácia e preci-

são para uma base de dados não-balanceada. Porém, o método de treinamento em lotes não prevê a realidade do funcionamento dos IDSs, pois a sua dinamicidade é um fator importante a ser levado em conta.

3.2 Análise por Pacotes Individuais ou Fluxo

Para a identificação de intrusão há a possibilidade de utilizar dois tipos diferentes de dados para a inspeção das informações. Conforme pode ser visto na Tabela 2, somente os trabalhos de Feng, Li e Chana (2017) e Cheng et al. (2020) utilizaram a abordagem de pacotes individuais, enquanto todos os demais analisaram o fluxo formado por um conjunto de pacotes, onde dados estatísticos deste conjunto também podem ser utilizados para o treinamento dos modelos de detecção de intrusão. É importante destacar que a análise da carga útil dos pacotes não é um atividade trivial para os IDSs, principalmente devido as suas informações poderem estar cifradas impedindo, muitas vezes, que se identifique o seu conteúdo.

Feng, Li e Chana (2017) desenvolveram em seu trabalho uma estrutura de detecção de anomalias que combina um detector de anomalias ao nível de pacotes individuais, utilizando o método chamado *Bloom Filter* (PARTHASARATHY; KUNDUR, 2012), e um detector de anomalias a nível de série temporal, baseado em redes neurais artificiais recorrentes com memória a curto prazo (HOCHREITER; SCHMIDHUBER, 1997) (em inglês, *Long Short Term Memory* - LSTM). Entre as suas vantagens, está justamente a utilização dos dados dos pacotes individuais. Nesse casos não é necessário que se crie o resumo destes pacotes em um fluxo de dados para a utilização no treinamento, podendo assim realizar a classificação próxima ao tempo real. Já uma desvantagem nessa abordagem é a necessidade de uma grande quantidade de dados rotulados (*Normal* e *Ataque*), necessários para o treinamento dos modelos de detecção. Apesar da estrutura proposta ser capaz de aprender somente o comportamento normal, para então identificar os desvios, é necessário o ajuste de uma grande quantidade de parâmetros, principalmente nas redes *Long Short Term Memory* (LSTM), o que exige uma grande quantidade de testes até a identificação de parâmetros ótimos. Outro ponto é que, apesar da capacidade preditiva das redes LSTM, caso a distribuição de dados ditos como normais mude, por algum motivo, a tendencia é que o desempenho preditivo do classificador diminua e haja a necessidade de executar todo o processo de treinamento novamente, o que pode não ser uma tarefa trivial. A proposta obteve uma revocação de 78%, o que em termos de IDSs não é um resultado tão satisfatório como se espera deste tipo de sistema.

Ainda na proposta de analisar somente os pacotes individualmente, Min et al. (2018) visa em seu trabalho extrair automaticamente recursos disponíveis na carga útil dos pacotes para melhorar a precisão dos IDSs. No trabalho, os autores combinam dois tipos de classificadores, o primeiro são as árvores aleatórias (comitês de classificadores que utilizam

exclusivamente árvores) que, segundo o autor, possuem desempenho preditivo superior em dados estruturados, o segundo classificador são as redes convolucionais, que são mais adequadas para trabalhar com dados não estruturados, como aqueles encontrados na carga útil dos pacotes. No caso da carga útil, os autores também usam técnicas de processamento de linguagem natural (CHOWDHURY, 2003), que consistem em interpretar o que está sendo enviado no pacote. Os resultados obtidos mostram que o FAR atingiu $1,18\%$, sendo melhor do que os $1,74\%$ atingidos pelo modelo de floresta aleatória sem a utilização dos dados da carga útil, o que a principio é um resultado satisfatório para os IDSs. A revocação também atingiu um valor expressivo, chegando a $99,26\%$, o que mostra que a técnica realmente pode ser melhor do que os métodos tradicionais comparados. Apesar dos resultados se mostrarem interessantes, é fato que a possibilidade de se obter sucesso em dados cifrados da carga útil pode implicar em um atraso na identificação ou mesmo na impossibilidade de se executar tal processo. Assim, é importante salientar que a utilização dos pacotes individuais e de comitês de classificadores é um caminho promissor, com resultados que devem ser explorados.

Por outro lado, trabalhos que utilizam o fluxo dos pacotes para detecção de anomalias também vêm obtendo resultados satisfatórios na área de identificação de intrusão. Haider et al. (2021) propõem um modelo que analisa o fluxo de pacotes com foco na identificação de intrusão próximo ao tempo real. O método proposto, chamado de *RTS-DELM-CSIDS*, visa diminuir a taxa de alarmes falsos que podem ser comuns em IDSs baseados em anomalias. Para tanto, é necessário que um especialista humano faça a rotulagem das instâncias, e assim, em um método iterativo é possível melhorar a qualidade do classificador. Um treinamento inicial é feito com uma base *offline* chamada NSL-KDD (DHANABAL; SHANTHARAJAH, 2015). Ela contém exemplos do comportamento tanto do tráfego *Normal* como de *Ataque*. O algoritmo de aprendizado utilizado chamado de modelo de aprendizado extremamente profundo (em inglês *Deep Extreme Learning Model* - DELM) necessita, assim como em outros modelos de aprendizagem profunda, de uma grande quantidade de dados rotulados para o treinamento. Um ponto de atenção é a necessidade de ser parametrizar os modelos, o que pode ter um alto custo de tempo, além da necessidade de um especialista de domínio para compreender se os parâmetros utilizados na classificação geraram uma boa resposta do modelo. Ainda, o fato da distribuição dos dados poder mudar durante o tempo implica, especificamente neste modelo, dois problemas. O primeiro é ter disponível um especialista para rotular as novas instâncias que foram classificadas erroneamente, outro fato é a necessidade de retreinar o modelo sempre que a distribuição dos dados for alterada. Por fim, a acurácia atingida pelo modelo foi de $92,73\%$ na base de validação o que apesar de alto para IDSs, pode ser um resultado não tão satisfatório se comparados com demais estudos na área.

O trabalho de Hsu et al. (2019) também utiliza um sistema de detecção de intrusão de rede baseado em anomalias, analisando o fluxo de pacotes e utilizando um modelo

de comitês de classificadores empilhados, que consiste em modelos de um *autoencoder* (AE), máquina de vetores de suporte e floresta aleatória. A metodologia proposta incluiu etapas para coleta e pré-processamento de dados, e é aplicável a diferentes ambientes de rede. Os autores mostraram também que a abordagem é aplicável em diferentes redes de computadores, avaliando primeiro a abordagem por meio de dois conjuntos de dados amplamente utilizados para a comparação de IDSs, NSL-KDD e UNSW-NB15. Além disso, aplicaram o modelo a uma escala de cem milhões de *logs* de rede de um sistema da Palo Alto que foram coletados do ambiente local. Para mostrar a viabilidade da abordagem, os autores compararam a proposta de comitê de classificadores a três modelos diferentes de aprendizado de máquina, um de floresta aleatória, outro de máquina de vetores de suporte e um *Perceptron* de multi camadas, bem como a resultados relatados de outros estudos relacionados. Os resultados experimentais mostraram que o modelo proposto fornece uma acurácia de classificação em torno 91% para as bases NSL-KDD e UNSW-NB15, com uma precisão em torno de 92% além de uma revocação de 93%. Apesar dos bons resultados apresentados com a utilização de comitês de classificadores, é importante destacar que o trabalho não lida com a limitação das mudanças de distribuição dos dados, além de não informar como os dados são rotulados a fim de se executar um novo treinamento, caso seja necessário, devido à mudança de conceitos nos dados.

Como discutido nos trabalhos anteriores, quanto a utilização de dados de pacotes e fluxo, ambas as abordagens possuem suas limitações. Quanto às estratégias e algoritmos de classificação, pode-se perceber que a utilização de comitês de classificadores é uma abordagem promissora para IDSs, aumentando o desempenho de classificação dos modelos. No entanto, há dois pontos importantes a serem discutidos que não foram considerados pelos trabalhos apresentados, o primeiro, é a forma com que eles lidam com a mudança na distribuição dos dados durante o funcionamento do IDS, o que exigiria retreinar um novo modelo. O segundo, é como a rotulagem das instâncias deve ocorrer, a fim de não sobrecarregar o especialista de domínio neste processo, em especial, quando há mudanças na distribuição. Por fim, os trabalhos descritos não citam como o atraso na entrega de rótulos poderiam influenciar o desempenho preditivo dos classificadores.

3.3 Detecção de Intrusão usando Fluxos Contínuos de Dados

Conforme visto na Tabela 2 trabalhos recentes como Viegas et al. (2019), Faisal et al. (2015), Cassales et al. (2019) e Costa et al. (2018) utilizam algoritmos de mineração de fluxos contínuos de dados para a construção de IDSs. A suposição destes trabalhos é que os pacotes de redes são gerados segundo uma distribuição de dados não-estacionária. Assim, é importante atualizar os modelos criados, a fim de se obter um bom desempenho preditivo em ambientes próximos ao mundo real.

Viegas et al. (2019) explora o uso de algoritmos para fluxos contínuos de dados para criar um sistema de detecção de intrusão. A proposta feita pelos autores é o *BigFlow*, que objetiva fornecer um sistema capaz de executar a classificação com um alto rendimento na detecção de anomalias, além de um mecanismo de atualização do modelo, que é computacionalmente modesto, visando tratar mudança de conceito. A proposta usa um mecanismo de classificação com opção de rejeição, baseado na confiabilidade da classificação, em que uma rejeição indica uma alta probabilidade de que o comportamento da rede está mudando. O modelo proposto atingiu uma acurácia (99%) algoritmos de classificação como *Hoeffding Tree*, *OzaBoosting*, *Leveraging Bag* e um comitê de classificadores composto por todos eles. A abordagem usada pelo *BigFlow* agrupa os dados trocados pela rede e os resume em intervalos de tempo, reduzindo significativamente o esforço computacional necessário e os requisitos de armazenamento no detector de intrusão. Finalmente, para fornecer um mecanismo de atualização leve, o *BigFlow* explora algoritmos tradicionais existentes, incorporando assistência especializada para rotular eventos rejeitados quando eles não são mais bem compreendidos. Destaca-se que apesar da acurácia ter obtido um bom desempenho, isso pode não refletir o desempenho preditivo real do *BigFlow*, visto que o fato de se ter uma base naturalmente desbalanceada como *MAWIFlow* (GROUP et al.,), onde somente $1,52\%$ das instâncias são do tipo *Ataque*, pode implicar em uma análise parcial quando a acurácia for utilizada isoladamente. Assim, os autores analisaram também o percentual de falsos negativos que atingiu, no melhor caso, 4% quando utilizado o comitê de classificadores.

Já Faisal et al. (2015) propõem um IDS baseado em fluxo contínuo de dados para redes inteligentes. Neste trabalho, os autores investigam o desempenho preditivo dos seguintes algoritmos baseados em comitê de classificadores: *Accuracy Updated Ensemble*, *Active Classifier*, *Leveraging Bag*, *Lim Att Classifier*, *Oza Bag Adwin*, *Oza Bag ASHT* e *Single Classifier Drift*. Esse trabalho abrangeu uma quantidade satisfatória de classificadores que foram comparados entre si e que puderam estabelecer uma linha de base para os algoritmos utilizados nesta dissertação. Os autores também propuseram uma arquitetura para IDSs que foi projetado para ser confiável, dinâmico e considerar a natureza em tempo real do tráfego para cada componente de uma rede inteligente. Em seguida, os autores conduziram um experimento de análise de desempenho preditivo dos sete algoritmos no conjunto de dados públicos *KDD Cup 1999* (TAVALLAEE et al., 2009). Foram identificados algoritmos que necessitam de um baixo poder computacional para executar as classificações, como o *Oza Bag Adwin* e *Oza Bag ASHT*. Este fato implica que, em princípio, é possível o processamento de uma grande quantidade de dados em espaço de tempo reduzido, bem próximo ao tempo real. Os resultados mostraram a melhor taxa de falsos negativos na ordem de $5,15\%$ e de falsos positivos de $0,78\%$ para o algoritmo *Leveraging Bagging*. Também quanto ao desempenho preditivo para a detecção de mudança, os autores destacaram o *Oza Bag ASHT*, que pode detectar todos as mudanças de

conceitos disponíveis na base.

Cassales et al. (2019) propõem uma nova arquitetura de implementação para um sistema de detecção de intrusão baseado em anomalias que, basicamente, combina a utilização de computação de borda com computação em nuvem. Também utiliza três técnicas de detecção de novidades para identificação de novos ataques. O trabalho faz uma extensa análise sobre a detecção de novidades aplicada diretamente a IDSs e também descreve a dificuldade de obter um especialista de domínio para poder rotular todas as instâncias, indicando que os resultados encontrados são uma análise do teto de desempenho preditivo para as técnicas utilizadas. Deixa claro que para um cenário próximo da vida real a utilização de técnicas de aprendizado ativo é uma provável solução. Um outra análise interessante feita no trabalho, foi a utilização de métodos de detecção de novidades como *MINAS* e *AnyNovel* que utilizam técnicas de agrupamento para a criação dos modelos de decisão. Ainda segundo os autores, a principal vantagem é que as atualizações dos modelos de decisão podem ser feitas com ou sem a utilização de um agente externo. Elas utilizam um único classificador com uma memória de curto prazo para armazenar os exemplos ainda não explicados pelo modelo de decisão atual. Os resultados dos experimentos mostraram que as três técnicas utilizadas foram satisfatórias para a identificação de intrusão em redes *Internet of Things* (IoT). Atingindo medida do *Fnew* e *Error rate* (MASUD et al., 2011) menor que 3% e 5% respectivamente para todos os experimentos.

Em Costa et al. (2018), os autores utilizaram uma abordagem de classificador único baseado em árvore de decisão, chamada de *Very Fast Decision Tree* (*VFDT*), para a detecção de *botnets*. Segundo os autores, a escolha da *VFDT* foi feita devido a característica de baixa necessidade de memória para a classificação de fluxos contínuos de dados, podendo assim, ser utilizada em equipamentos com limitação de memória. Ainda destacaram a importância da utilização de algoritmos de classificação para fluxo contínuos de dados, devido a sua capacidade de aprender incrementalmente sem a necessidade de armazenar os dados, sendo capaz então de aprender continuamente.

Eles propuseram um sistema de detecção baseado em rede para identificar *botnets*, analisando os fluxos de rede de uma maneira *online* e empregando a *VFDT*. Na seleção de atributos da base *CTU-13* os autores retiraram da análise os endereços IP de origem e destino a fim de evitar qualquer tipo de enviesamento. Criaram um novo atributo que é a média de *bytes* por pacote. Porém, os autores não mencionaram nenhum tipo de técnica específica para a seleção de atributos. As medidas de avaliação utilizadas foram a revocação, precisão e a taxa de falsos positivos. O método de avaliação utilizado foi o de sequenciamento preditivo onde cada instância é utilizada para testar e então para treinar o modelo.

Nos 13 cenários de testes, a quantidade de rótulos necessários para que o classificador pudesse obter um bom desempenho preditivo variou significativamente. Segundos os autores, isso significa que alguns *botnets* necessitam naturalmente de mais instâncias para

poderem ser identificados pelo classificador. Além disso, a taxa de falsos positivos foi baixa em todos os cenários testados, sendo que, o maior valor dessa medida foi em torno de $0,07\%$. A precisão e a revocação também obtiveram uma grande variação indo de $63,5\%$ a $99,9\%$ e de $42,2\%$ até $99,6\%$ respectivamente.

No trabalho de Ribeiro, Paiva e Miani (2020) os autores propuseram a comparação de algoritmos para classificação de fluxos contínuos de dados para a identificação de ataques de *botnets*. A proposta se concentra na análise entre comitê de classificadores e algoritmos de classificador único. Além disso, os autores analisaram como o comportamento dos classificadores quando a quantidade de rótulos das instâncias de treinamento é restrita.

Como pode ser visto na Tabela 2, a base utilizada foi no trabalho foi a *CTU-13* que traz 13 cenários diferentes de fluxos de rede para ataques *botnets*. Os classificadores utilizados foram *VFTD* e a *Hoeffding Adaptative Tree* que empregaram a estratégia de classificador único enquanto que, para o comitê de classificadores, os algoritmos selecionados foram: *OzaBag*, *OzaBoost* e *OzaBagAdwin*. Quanto a seleção de atributos, os autores utilizaram a mesma metodologia empregada em Costa et al. (2018) e para os experimentos utilizaram a *framework Massive Online Analysis* (MOA) que possui todos os algoritmos implementados. Houve a necessidade de implementar, no MOA, uma estratégia para limitar o número de instâncias a serem rotuladas conforme uma configuração inicial, chamada pelos autores de *ALLimitedInstances*. O método de avaliação utilizado foi o sequenciamento preditivo onde as instâncias individuais são utilizadas para testar e depois para treinar o modelo. Quanto as métricas, foi utilizada somente a precisão e revocação, além do "melhor n" que indica a menor quantidade possível de instâncias para treinar o modelo e atingir a média máxima entre precisão e revocação.

Os resultados obtidos quando comparados os algoritmos de classificador único e aqueles baseados em comitês indicou que, o *OzaBoost*, atingiu o resultados mais consistente em todas as medidas de avaliação e para todos os cenários dos experimentos comparados ao *baseline* indicado pelos autores. Por fim os autores concluíram que, conforme o esperado, o desempenho preditivo de classificadores baseados em comitês foi superior àqueles com classificador único.

Percebe-se, diante dos resultados apresentados pelos trabalhos analisados, que o uso de classificadores de fluxos contínuos de dados é uma opção para a construção de IDSs, porém eles analisaram somente o fato de se trabalhar com o fluxo ao invés de pacotes individuais, não houve uma comparação entre estes dois tipos de bases. Ainda que, o fluxo seja uma predileção de trabalhos recentes, talvez a utilização de pacotes individuais seja uma opção satisfatória.

3.4 Conjunto de Dados

A escolha dos conjuntos de dados a serem utilizados nos experimentos é vital para os trabalhos na área de detecção de intrusão, pois são eles que permitem avaliar a capacidade do método proposto em detectar comportamento intrusivo. Devido a questões de privacidade, conjuntos de dados utilizados para análise de pacotes de rede em produtos comerciais nem sempre estão facilmente disponíveis (KHRAISAT et al., 2019). Contudo, existem alguns conjuntos de dados disponíveis publicamente, e eles são amplamente usados como linha de base para as análises do desempenho preditivo dos IDSs conforme pode ser visto na Tabela 3.

Em geral, é importante que a base utilizada disponibilize o tráfego de pacotes mais próximo da realidade. Assim, ela deve possuir ataques reais, recentes, com rótulos dos dados. Também deverá possuir os dados tanto dos pacotes individuais quanto dos fluxos destes pacotes para que o desempenho preditivo dos classificadores possa ser comparado de forma satisfatória com respeito a dois conjuntos de dados.

Yang et al. (2018) em seu trabalho utilizaram a base *KDD Cup 1999*. Ela foi a primeira base para IDSs, criada a partir do esforço da *Defence Advanced Research Project Agency* (DARPA) em 1998. Embora este conjunto de dados tenha sido uma contribuição importante para a pesquisa sobre IDSs, sua precisão e capacidade de considerar as condições da vida real foram amplamente criticadas (CREECH; HU, 2014). Esse conjunto de dados foi coletado usando vários computadores conectados à Internet, para modelar uma pequena base da Força Aérea dos Estados Unidos. Pacotes de rede e arquivos de *log* do servidor foram coletados simulando uma rede real com pacotes TCP. Foram inseridos no conjunto de dados várias intrusões simuladas, dessa maneira, somente o arquivo do fluxo de pacotes está disponível, possuindo um total de 41 atributos. No entanto, este conjunto de dados está desatualizado, pois não contêm registros de ataques criados após 1998. Na atualidade, tanto o comportamento dos invasores, quanto topologia de rede são diferente dos coletados naquela época. Sistemas operacionais, *softwares* de criação e de proteção também se modificaram bastante ao longo do tempo. Para o caso deste trabalho, não é possível utilizá-la pois a intenção é construir uma condição mais próxima de ambientes reais atuais.

Na busca de bases de dados mais atuais, os trabalhos de Zhong et al. (2020), Gu e Lu (2021) utilizaram a base CICIDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2018). Esse conjunto de dados inclui comportamento *Normal* e também detalhes de novos ataques como força bruta, negação de serviço, *heartbleed*, ataque web, infiltrações, *botnet* e negação de serviço distribuído. Ele foi rotulado com base na data e hora, IP de origem e destino, portas de origem e destino, protocolos e ataques. Uma topologia de rede completa foi configurada para coletar este conjunto de dados, que contém *modem*, *firewall*, *Switches*, roteadores e nós com diferentes sistemas operacionais (Microsoft Windows - com Windows 10, Windows 8, Windows 7 e Windows XP-, Apple macOS iOS e sistema operacional de

código aberto Linux). Este conjunto de dados contém 80 atributos de fluxo de rede do tráfego capturado. Também são disponibilizados arquivos com dados brutos dos pacotes do tipo PCAP, que foram bastante úteis para este trabalho. O contexto em que a base foi criada traz vantagens através de seus ataques atuais e simulação de ataques, permitindo uma maior realidade para os testes que foram propostos neste trabalho.

3.5 Comparação entre os trabalhos relacionados

A Tabela 2 apresenta uma visão comparativa entre os trabalhos relacionados discutidos nas seções anteriores.

Tabela 2 – Trabalhos relacionados sobre sistemas de detecção de intrusão (Fonte: O autor (2021)).

Referência	Tipo do Dado	Tipo de IDS	Tipo Algoritmo	Treinamento	Conjunto de Dados	Avaliadores	Aprendizado Ativo	Mudança de Conceito
Zhong et al. (2020)	Fluxo	Anomalia	Comitê	Lote	MAWILab CICIDS2017	Precisão Revocação F1-Score Taxa de Falso Positivos	N	N
Yang et al. (2018)	Fluxo	Anomalia	Comitê	Lote	AWID KDD Cup 1999	Precisão Revocação	S	N
Viegas et al. (2019)	Fluxo	Anomalia	Fluxos Conínuos de Dados	Online	MAWIFlow	Acurácia	N	S
Faisal et al. (2015)	Fluxo	Anomalia	Fluxos Conínuos de Dados	Online	KDD Cup 1999	Acurácia Taxa de Falsos Negativos Taxa de Falsos Positivos Coeficiente Kappa	S	S
Churcher et al. (2021)	Fluxo	Anomalia	Tradicional	Lote	Bot-IoT	Acurácia Precisão Revocação F1-Score Log-Loss Área sob a Curva (AUC) Coeficiente Kappa	N	N
Gu e Lu (2021)	Fluxo	Anomalia	Tradicional	Lote	UNSW-NB15 CICIDS2017	Acurácia Revocação FAR	N	N
Sarker et al. (2020)	Fluxo	Anomalia	Tradicional	Lote	Kaggle - IDS	Acurácia Precisão Revocação F1-Score	N	N
Hsu et al. (2019)	Fluxo	Anomalia	Comitê	Lote	NSL-KDD UNSW-NB15 Palo Alto SLD	Acurácia Precisão Revocação	N	N
Haider et al. (2021)	Fluxo	Anomalia	Tradicional	Online	NSL-KDD	Acurácia Taxa de Falsos Negativos Revocação FAR	N	N
Viegas, Santin e Oliveira (2017)	Fluxo	Anomalia	Tradicional	Lote	DARPA1998	Acurácia Taxa de Falsos Negativos Taxa de Falsos Positivos	N	N
Cassales et al. (2019)	Fluxo	Anomalia	Fluxos Contínuos de Dados	Lote + Online	Kyoto 2006+	Fnew Mnew Err	N	S
Costa et al. (2018)	Fluxo	Anomalia	Fluxos Contínuos de Dados	Online	CTU-13	Revocação Precisão Taxa de Falso Positivos	S	S
Ribeiro, Paiva e Miani (2020)	Fluxo	Anomalia	Fluxos Contínuos de Dados	Online	CTU-13	Revocação Precisão Melhor 'n'	S	S
Min et al. (2018)	Pacotes	Anomalia	Comitê	Lote	ISCX2012	Acurácia Revocação FAR	N	N
Feng, Li e Chana (2017)	Pacotes	Anomalia	Tradicional	Online	*Gas Pipeline*	Acurácia Precisão Revocação F1-Score	N	N
Cheng et al. (2020)	Pacotes	Assinatura	Tradicional	Online	Próprio	Não analisado	N	N

Na Tabela 3 estão listados todos os conjuntos de dados utilizados nos trabalhos analisados. Nela podemos identificar o nome, autor e ano de criaçao.

Tabela 3 – Lista de base de dados de ataques utilizadas nos trabalhos analisados (Fonte: O autor (2021)).

Base de Dados	**Autor**	**Ano**
AWID	Kolias et al. (2016)	2015
Bot-IoT	Koroniotis et al. (2019)	2019
CICIDS2017	Sharafaldin, Lashkari e Ghorbani (2018)	2017
CTU-13	Garcia et al. (2014)	2013
DARPA1998	DARPA (1998)	1998
Gas Pipeline Log	Morris, Thornton e Turnipseed (2015)	2015
ISCX2012	Shiravi et al. (2012)	2012
Kaggle - IDS	Bhosale (2018)	2018
KDD Cup 1999	Tavallaee et al. (2009)	1999
Kyoto 2006+	Song et al. (2011)	2011
MAWIFlow	Viegas, Santin e Jr (2021)	2020
MAWILab	Fontugne et al. (2010)	2010
NSL-KDD	Dhanabal e Shantharajah (2015)	2009
Palo Alto SLD	PaloAlto (2019)	2019
UNSW-NB15	Moustafa e Slay (2015)	2015

3.6 Considerações Finais

Neste capítulo, foram destacados os principais trabalhos da literatura que estão relacionados aos estudos desta pesquisa. Foi dado um destaque aos trabalhos que utilizaram os métodos de detecção de anomalias, além daqueles que utilizam os pacotes individuais para a identificação de intrusão. Além disso, os trabalhos que empregaram algoritmos para fluxos contínuos de dados foram discutidos e, por fim, as bases que se adéquam ao objetivo desta pesquisa.

De modo geral, na literatura não tratam a detecção de intrusão como um problema de fluxos contínuos de dados. O mais comum é executar treinamentos em lote para a geração dos modelos de classificação. É necessário compreender que os IDSs enfrentam um ambiente dinâmico onde é necessário se adaptar as alterações na distribuição dos dados a fim de manter um bom desempenho preditivo, inclusive para novos tipos de ataques. Também, em muitos casos, as técnicas de aprendizado ativo não são consideradas como uma opção para a rotulagem dos dados. Devido ao grande volume de pacotes em um ambiente de redes de computadores, pode-se tornar inviável o trabalho de rotulagem de todo o tráfego.

O Capítulo 4, apresentará a metodologia desenvolvida para a detecção de intrusão utilizando como conjunto de dados os pacotes de redes e os algoritmos de fluxo contínuo de dados, além das demais etapas necessárias para a análise das hipóteses propostas.

Capítulo 4

Método de Validação das Hipóteses

Neste capítulo será abordado o método adotado para a detecção de pacotes maliciosos nas redes de computadores. A Seção 4.1 apresenta uma visão geral do método proposto. As Seções 4.2, 4.3 e 4.4 descrevem os métodos que testará as Hipóteses $H1$, $H2$ e $H3$, respectivamente. Elas também justificam a utilização dos métodos de avaliação em cada uma das hipóteses. A Seção 4.5 apresenta as justificativas para a seleção dos algoritmos de classificação que serão utilizados nos experimentos. Na Seção 4.6, as medidas para avaliação de desempenho preditivo dos algoritmos de classificação são descritas no contexto de detecção de pacotes maliciosos em redes de computadores. A Seção 4.7 descreve as características necessárias de uma base de dados para que o método proposto seja viável.

4.1 Visão Geral

O presente trabalho, possui o objetivo de colaborar com o desenvolvimento de um IDS que atenda os requisitos reais do problema, tais como geração contínua de dados e constante atualização do modelo de detecção de intrusão. Para isso, alguns pontos serão investigados:

1. comparar o uso de fluxos de pacotes e dos pacotes individuais na tarefa de detecção de intrusão através dos algoritmos de classificação de fluxos contínuos de dados;

2. identificar, dentre os algoritmos selecionados, qual possui o melhor desempenho preditivo quanto à classificação dos pacotes de rede;

3. avaliar como a entrega atrasada de instâncias rotuladas para atualização do modelo de classificação pode impactar no desempenho preditivo do classificador; e

4. avaliar como as estratégias de aprendizado ativo, que escolhem as melhores instâncias a serem rotuladas por um especialista, podem auxiliar a manter o desempenho preditivo do classificador enquanto requerem menos trabalho dos especialistas.

Este trabalho propõe utilizar algoritmos de classificação para fluxos contínuos de dados no problema de detecção de intrusão em redes de computadores, pois esses algoritmos conseguem classificar dados contínuos e em alta velocidade (KNOWLEDGE..., 2010), sendo uma alternativa para a construção de IDSs. Esses algoritmos são capazes de se adaptar a mudanças de conceito na distribuição dos dados, reagindo a essas mudanças sem a necessidade de executar um novo treinamento para a geração de um modelo (KNOWLEDGE..., 2010). Já os algoritmos tradicionais de classificação foram desenvolvidos para o cenário estático (*batch*), onde se supõe que a distribuição que gera os dados não muda e o modelo gerado não evolui a não ser que seja treinado novamente. Além disso, é necessário que a amostra de treinamento contenha todos os tipos de ataques possíveis. Devido a dinamicidade dos pacotes de rede que necessitam ser classificados, o uso de classificadores tradicionais em IDSs pode não ser a escolha mais adequada para o problema.

Este trabalho também propôs comparar o uso dos pacotes individualmente ao invés dos dados de um fluxo de pacotes. Esta abordagem é interessante porque devido a geração do fluxo os dados de vários pacotes de rede são resumidos e, portanto, generalizados, tornando difícil para os IDSs baseados em fluxo detectar formas distintas de uma tentativa de ataque (UHM; PAK, 2021). Embora a abordagem com os pacotes individuais gere bases de dados maiores a serem classificadas, isso não é um empecilho para as técnicas de fluxo contínuos de dados, que conseguem classificar fluxos de dados em alta velocidade, sem a necessidade de revisitar dados já classificados.

Um dos importantes desafios para se lidar com modelos adaptativos que consigam lidar com fluxos de dados não-estacionários é a atualização do modelo de decisão (KNOWLEDGE..., 2010). Em geral, essa atualização supõe a disponibilidade de instâncias de dados rotuladas, o que exige a presença de um especialista de domínio dedicado a essa tarefa. Para bases com uma grande quantidade de dados, essa presença torna-se impraticável pelo alto custo despendido a esta tarefa.

Na busca pelo desenvolvimento de IDSs que sejam mais adequados às limitações impostas por cenários reais, este trabalho utiliza a estratégia denominada aprendizado ativo. O aprendizado ativo analisa como rotular seletivamente os dados ao invés de solicitar todos os rótulos verdadeiros das instâncias. Esta estratégia utiliza métodos como amostragem aleatória, amostragem de incerteza fixa, entre outras, para consultar as instâncias mais incertas e, assim, serem rotuladas pelo especialista de domínio. O aprendizado ativo, em geral, emprega um percentual de rotulagem que limita a quantidade de instâncias rotuladas, ou seja, cada instância rotulada é somada ao total até atingir o limite estabelecido. Um percentual maior indica mais instâncias rotuladas, enquanto o contrário exige que menos instâncias sejam rotuladas. Estabelecer um equilíbrio entre o número de instâncias rotuladas e o desempenho preditivo do modelo é uma das questões a serem investigadas neste trabalho.

Diante dos desafios para construção de IDSs baseados em algoritmos de classificação

de fluxos contínuos de dados, este trabalho propõe três hipóteses conforme visto na Seção 1.2:

- o objetivo da hipótese $H1$ é verificar se a utilização dos pacotes de redes individualmente para a identificação de ataques apresenta resultados estatisticamente semelhantes aos encontrados usando os fluxos de pacote de redes;
- o objetivo da hipótese $H2$ é verificar se o atraso na disponibilização de instâncias rotuladas para atualização do modelo afeta o desempenho preditivo na classificação de novas instâncias, usando somente os dados dos pacotes de redes; e
- o objetivo da hipótese $H3$ é analisar se as técnicas de aprendizado ativo, ao reduzirem o número de instâncias rotuladas para atualização do modelo, impactam no desempenho preditivo se comparado à utilização de todas as instâncias rotuladas.

Como este trabalho, por vezes, utilizará a comparação de pacotes analisados individualmente com o fluxo destes pacotes, é oportuno padronizar a terminologia. Assim, quando no texto for citado o termo "pacotes", refere-se aos pacotes sendo analisados individualmente, enquanto que o termo "fluxo" refere-se a agregação dos pacotes individuais.

4.2 Método para a validação de $H1$

A hipótese $H1$ pretendeu comparar os resultados obtidos pela análise de pacotes *versus* o seu fluxo. Ela foi proposta pois a utilização dos pacotes, ao invés dos fluxos, pode trazer ganhos para a velocidade da identificação da intrusão de rede. Assim, é possível executar a classificação de forma *online*, sem a necessidade de um tempo de processamento elevado para a geração do fluxo.

Para que esta hipótese fosse verificada, os pacotes e o fluxo de rede foram submetidos a um conjunto de algoritmos de classificação de fluxos contínuos de dados para construção dos modelos de decisão e para avaliação, conforme pode ser visualizado na Figura 12. Diante da utilização de um conjunto de classificadores, esperava-se que eles obtivessem um desempenho preditivo estatisticamente semelhante, tanto para a classificação do fluxo quanto dos pacotes.

Inicialmente, foi necessária a seleção de um conjunto de dados com os requisitos necessários para avaliar a hipótese. Dentre esses requisitos, era fundamental que a base contivesse os dados tanto dos pacotes quanto dos fluxos, denominado a partir de agora de *PkData* e *FlData* respectivamente. Em uma segunda etapa, foi necessário pré-processar essas bases visando a adequação dos dados *PkData* e *FlData* para o treinamento e geração do modelo de decisão.

Após a execução do pré-processamento para ambas as bases, foi iniciado o processamento dos dados através dos quatro passos a seguir. A primeira etapa, consistiu em

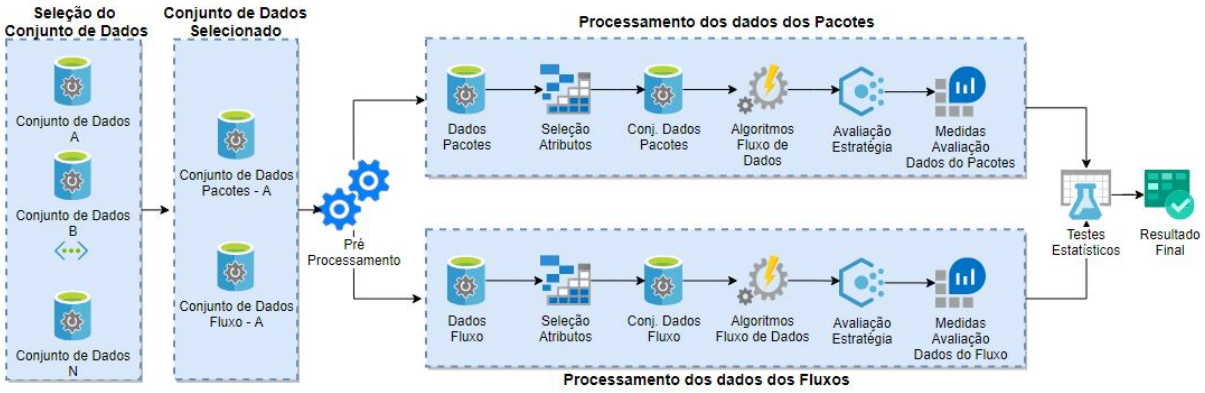


Figura 12 – Método Proposto para avaliar a hipótese $H1$ (Fonte: O autor (2021)).

executar uma seleção dos atributos com uma maior correlação com a classe alvo, ela foi executada inicialmente na base *PkData*, onde somente os dados referentes aos pacotes foram utilizados. A seguir, foram selecionados os atributos correspondentes na base *FlData*. Ao final, foram geradas novas bases as quais possuíam somente os atributos selecionados nessa etapa.

A segunda etapa do processamento consistiu na aplicação dos algoritmos de classificação nas bases resultantes da etapa anterior. Os algoritmos selecionados foram os seguintes:

- ❏ *Single Classifier Drift*
- ❏ *Accuracy Updated Ensemble*
- ❏ *Leveraging Bagging*
- ❏ *Limited Attribute Classifier*
- ❏ *Oza Bagging with ADWIN*
- ❏ *Oza Bagging with Adaptative Size Hoeffding Tree*

Destes, somente o *Single Classifier Drift* utiliza um único classificador, os demais utilizam comitê de classificadores. Uma discussão sobre a escolha dos algoritmos e o seu funcionamento foi apresentada na Seção 2.3.2.

A terceira etapa foi a seleção da estratégia de avaliação onde, para o presente trabalho, foi selecionada a estratégia chamada sequenciamento preditivo. Nessa técnica, cada instância é utilizada individualmente, primeiro para testar e depois treinar o modelo. Na quarta etapa, após os algoritmos selecionados terem executado a classificação de todas as instâncias das bases *PkData* e *FlData*, foi feita a avaliação por meio de medidas de desempenho preditivo como: revocação, precisão e taxa de alarmes falsos.

Com os resultados gerados, foi possível avaliar os classificadores tanto para base *PkData* quanto para a *FlData* comparando-os entre si. Baseado nestes resultados, o

teste de *Wilcoxon* foi aplicado a fim de verificar se a hipótese nula, que significa não haver diferença significativa entre os classificadores para as bases *PkData* e *FlData*, seria rejeitada. A expectativa, era que a hipótese nula não fosse rejeitada, indicando não haver diferença estatística significativa entre os resultados obtidos. Desta maneira, os resultados mostrariam que o desempenho de classificação de *PkData* se assemelharia com *FlData*. Porém, não rejeitar a hipótese nula não significa, necessariamente, que ela é aceita. Dessa maneira, houve a necessidade de se aplicar um teste para analisar o intervalo de confiança para cada um dos classificadores, diante dos resultados das medidas de avaliação utilizadas neste trabalho. A partir desse ponto, com a hipótese nula não rejeitada e o intervalo de confiança analisado, os melhores classificadores para a base *PkData* puderam ser identificados.

4.3 Método para a validação de $H2$

A hipótese $H2$ visa avaliar os algoritmos de classificação em cenários de detecção de intrusão mais próximos de ambientes reais, no qual há um atraso para se obter instâncias rotuladas para atualizar o modelo. Considerando cenários de fluxos contínuos de dados não-estacionários, em que a distribuição que gera os dados pode mudar ao longo do tempo, o modelo de decisão precisa evoluir a fim de se adaptar às mudanças recentes no fluxo. Nesse caso, sabe-se que a fase de treinamento do modelo de decisão não pode ocorrer mais de uma vez. É necessário então, o uso de algoritmos que sejam adaptativos, os quais constantemente atualizem o seu modelo.

Nesse contexto, grande parte dos algoritmos de classificação para fluxos contínuos de dados supõem que os rótulos dos exemplos estarão disponíveis imediatamente após a classificação. No entanto, considerando ambientes reais, nem sempre é possível rotular todas as instâncias devido ao alto custo necessário para executar esta tarefa. Em cenários em que seja possível rotular as mesmas, por exemplo, usando um agente automático, a entrega do rótulo será feita com algum atraso. Levando-se em consideração esta problemática, a hipótese $H2$ avaliou o impacto no desempenho preditivo do classificador quando há atraso na entrega das instâncias rotuladas. Os resultados obtidos foram contrastados com a entrega imediata dos rótulos das instâncias.

A Figura 13 dá uma visão geral do processo utilizado para a validação da hipótese $H2$. Os testes para validação desta hipótese utilizaram somente a base *PkData* e os classificadores foram configurados com determinados atrasos na entrega dos rótulos. O resultado gerado foi então avaliado através das medidas de avaliação revocação, precisão e FAR e então tiveram os seus resultados comparados com aqueles obtidos nos testes da hipótese $H1$. A seguir, o método utilizado referente ao atraso dos rótulos será detalhado.

A Figura 14 apresenta o esquema de rotulagem aplicado a um fluxo contínuo de dados. Nesse caso, assume-se que as instâncias chegam continuamente e o classificador executa a

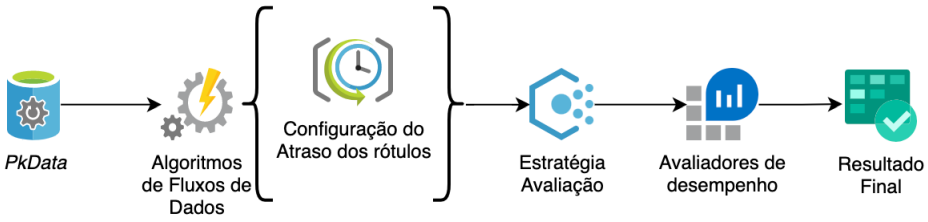


Figura 13 – Etapas do processo de validação da hipótese $H2$ (Fonte: O autor (2021)).

predição para cada uma das instâncias nos tempos t, $t+1$, $t+2$, ..., $t+n$. Em um cenário de atraso dos rótulos, o classificador não receberá o rótulo imediatamente após a classificação como ocorre em muitas técnicas de aprendizado de fluxos de dados. Nesse cenário de atraso, o tempo de recepção dos rótulos pode ser configurado em um tempo t no intervalo de $[0, \infty)$ que indicará quando o classificador receberá o rótulo verdadeiro da instância, simulando assim, diversos cenários. O modelo não será atualizado até que a instância classificada no tempo t tenha o seu rótulo verdadeiro entregue ao classificador no tempo $t+n$. Quando o tempo limite de atraso na entrega do rótulo de uma instâncias é atingido, o rótulo da instância é entregue ao classificador para que seja executado o treinamento do modelo e consequentemente sua atualização. A cada instância classificada pelo modelo, é executada a avaliação para contabilizar as instâncias classificadas corretamente.

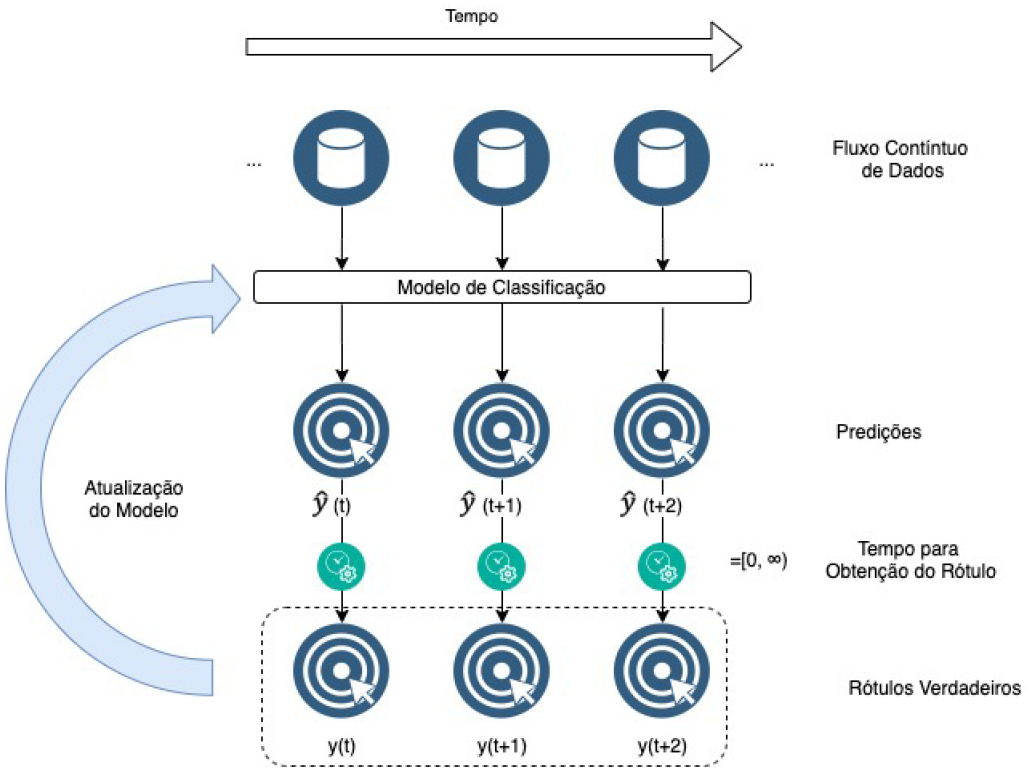


Figura 14 – Esquema de obtenção de rótulos em Fluxo Contínuo de Dados (Adaptado de Souza et al. (2015)

Para a análise de desempenho preditivo, existem alguns métodos de avaliação, como

visto na Seção 2.4.2 e a técnica sequenciamento preditivo pode ser adaptada com a possibilidade de inserção de atraso na entrega dos rótulos (GOMES et al., 2017). Esta técnica permitiu que o atraso fosse alterado conforme a necessidade dos experimentos, a fim de testar vários cenários.

É importante destacar que a estratégia comumente usada para avaliar algoritmos de classificação é iniciar o fluxo de dados e, à medida que novas instâncias rotuladas chegam, o classificador vai sendo atualizado. No entanto, sabe-se que existe um conhecimento prévio sobre os ataques, sendo que exemplos reais de dados de fluxos e pacotes que representam tanto um acesso normal quanto um ataque estão disponíveis. Assim, é comum que algoritmos de classificação de fluxos contínuos de dados tenham uma fase *offline* inicial, em que um modelo de decisão é criado com base em um conjunto de exemplos rotulados. Após essa fase, inicia-se a fase *online* na qual exemplos estão continuamente chegando e serão classificados pelo modelo. Nessa fase, também acontece a atualização do modelo à medida que instâncias rotuladas estão disponíveis.

Em geral, as bases de dados de ataques utilizadas para treinamento de modelos são desbalanceadas, contendo muito mais pacotes do tipo *Normal* do que *Ataque* (KURNIABUDI et al., 2021). Para mostrar ao classificador algumas instâncias do tipo *Ataque* na janela inicial de treinamento, esse experimento apresentou ao classificador algumas instâncias de *Ataque* com a intenção de que ele possa identificá-las com uma velocidade maior do que quando são apresentadas somente instâncias rotuladas como *Normal*. Diante disso, para testar a hipótese $H2$, foi usada a estratégia de selecionar uma janela inicial de dados de tamanho x, que foi então utilizada para o treinamento preliminar do modelo. Essa janela de tamanho x teve a mesma proporção de exemplos das classes *Ataque* e *Normal* da base original.

Por fim, foi analisado o desempenho preditivo dos classificadores nestes dois grupos de experimentos com base nas seguintes medidas de avaliação: revocação, precisão e taxa de alarmes falsos. Assim, foi possível verificar o comportamento dos classificadores e se houve uma perda significativa de desempenho preditivo comparado com os experimentos base executados para a verificação da Hipótese $H1$.

4.4 Método para a validação de $H3$

A hipótese $H3$ analisou o impacto de não se entregar todos os rótulos das instâncias ao classificador e, assim, avaliar se a atualização do modelo sofreu impactos significativos. Essa hipótese foi proposta por aproximar-se ainda mais de um ambiente real, no qual não é possível ter um especialista de domínio rotulando todas as instâncias do fluxo de dados para atualização do modelo.

Este trabalho propôs o uso de aprendizado ativo como uma estratégia para, dinamicamente, selecionar um conjunto de instâncias a serem rotuladas pelo especialista. Uma

visão geral do método pode ser vista na Figura 15. No aprendizado ativo, é estabelecido um limite de instâncias que serão rotuladas, que é um percentual de instâncias da base e, em ambientes reais, esse limite está diretamente relacionado à capacidade e disponibilidade do especialista em rotular novas instâncias. A avaliação dos resultados obtidos pelo uso das técnicas de aprendizado ativo foi feita pelo método de sequenciamento preditivo.

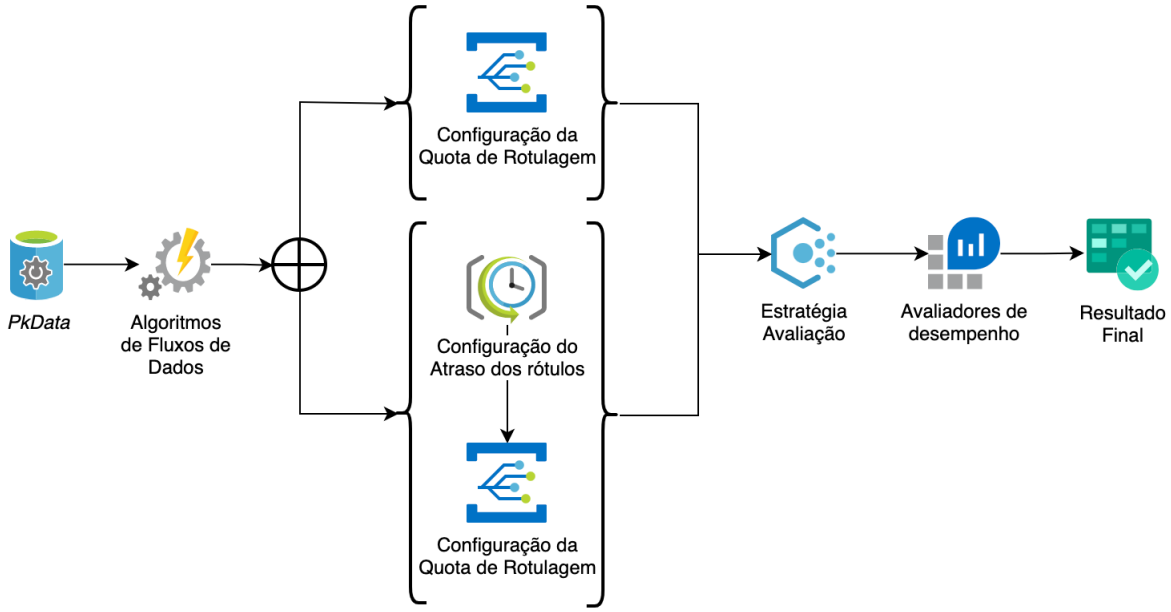


Figura 15 – Etapas do processo de validação da hipótese $H3$ (Fonte: O autor (2021)).

Dentre as possibilidades de estratégias de seleção dos rótulos, discutidas na Seção 2.3.3, foi selecionada a estratégia de amostragem aleatória. Nessa estratégia, os rótulos a serem entregues ao modelo de classificação são selecionados a partir de um processo aleatório, obedecendo sempre um percentual máximo de rótulos configurado previamente. A amostragem aleatória é um método clássico e possibilita que todas as instâncias possuam a mesma probabilidade uniforme de serem selecionadas (SETTLES, 2009). Como o fluxo de dados contínuos de pacotes de *Ataque* e *Normal*, em geral, é desbalanceado, dá-se a possibilidade de seleção proporcional a ambos os tipos de instâncias. Por este motivo, esta estratégia foi selecionada para este trabalho.

Seguindo a proposta de se aproximar a um ambiente real dos IDSs, foi apresentado um experimento complementar unindo as técnicas de atraso na entrega dos rótulos com a de aprendizado ativo. Assim, a entrega dos rótulos para o classificador foi atrasada e, ao momento de entregá-los, a técnica de aprendizado ativo é acionada, selecionando somente alguns rótulos aleatórios que serão entregues ao classificador. Diante disso, o impacto que essa proposta trouxe ao desempenho preditivo do classificador foi avaliada pelas medidas de avaliação utilizadas no presente trabalho.

A Figura 16 mostra o funcionamento do aprendizado ativo para a seleção de rótulos para a atualização do modelo de classificação. Após a predição feita pelo modelo, antes

dos rótulos verdadeiros serem enviados ao modelo para a atualização, é feita uma seleção aleatória desses rótulos com base em um percentual pré-estabelecido. A cada rótulo selecionado, este percentual é decrementado. Assim que o rótulo é selecionado, ele é enviado ao modelo de classificação para que a atualização seja executada.

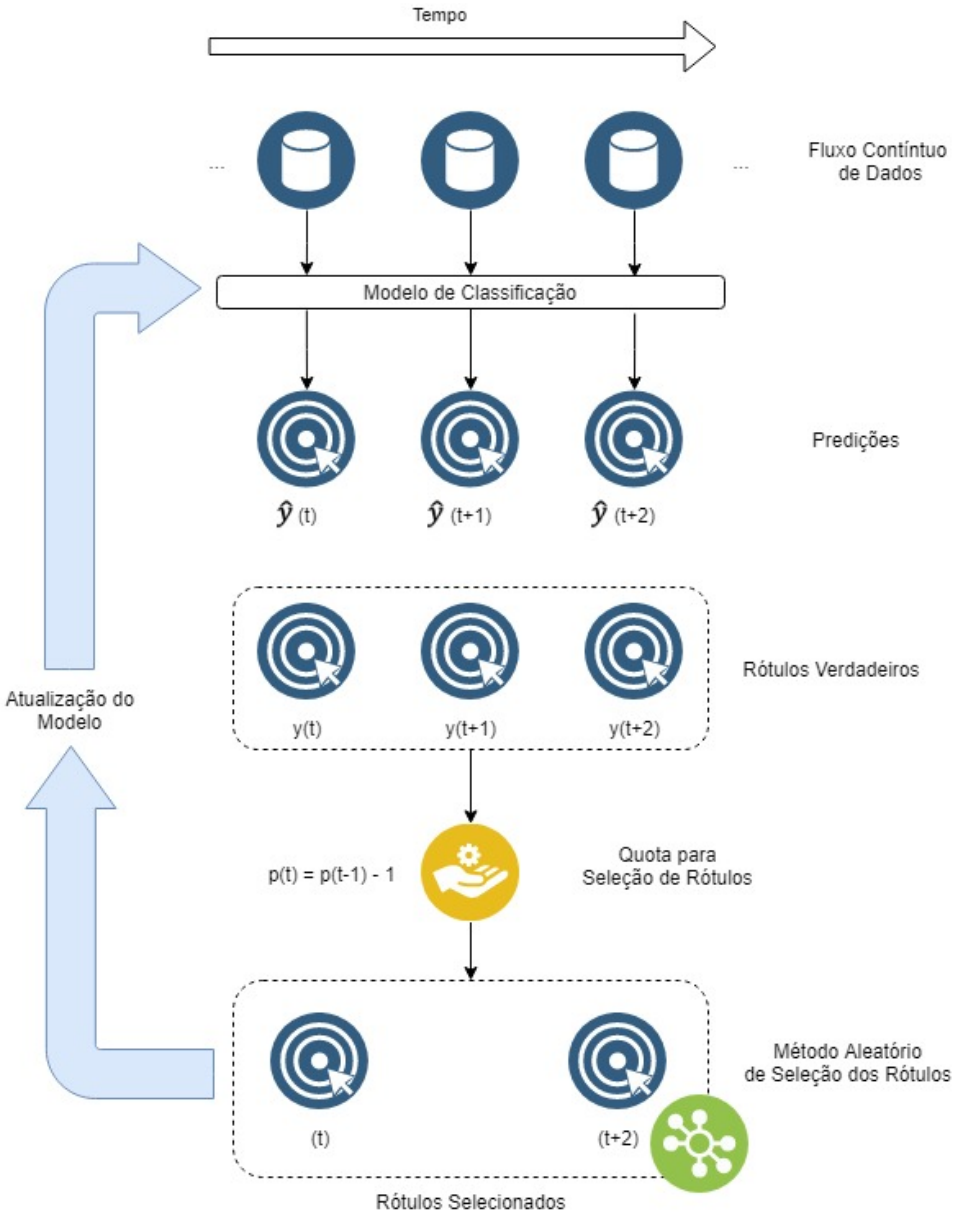


Figura 16 – Aprendizado Ativo utilizando o Método Aleatório para Seleção dos Rótulos (Fonte: O autor (2021)).

Por fim, para que os resultados sejam comparáveis com aqueles obtidos nos testes das hipóteses $H1$ e $H2$, foi utilizada a base *PkData*. Como nas hipóteses anteriores, todos os experimentos utilizaram as medidas de avaliação: revocação, precisão e taxa de alarme falso.

4.5 Estratégias e Algoritmos de Classificação

A escolha das estratégias e dos algoritmos de classificação utilizados para a verificação das hipóteses é uma etapa importante para a análise dos resultados. Dentre as estratégias de classificação, há aquelas baseadas em comitês de classificadores (*ensemble*) e as que trabalham com um classificador único, conforme discutido na Seção 2.3.2. Além disso, existem na literatura diversos algoritmos que são utilizados para a classificação de fluxos contínuos de dados (KRAWCZYK et al., 2017). A seguir, são listados os seis algoritmos selecionados para os experimentos no presente trabalho. Eles foram utilizados em trabalhos anteriores como em Viegas et al. (2019), Faisal et al. (2015), Ribeiro, Paiva e Miani (2020) e Noorbehbahani et al. (2017), em face de estarem diretamente ligados ao tema discutido aqui. As implementações desses algoritmos estão disponíveis em ferramentas como o MOA (BIFET et al., 2010), que é utilizado para trabalhos com fluxos contínuos de dados. Dentre os algoritmos selecionados, cinco utilizam a estratégia de comitês de classificadores. Os algoritmos baseados em comitê de classificadores são importantes pois, como visto na Seção 2.3.1, em ambientes dinâmicos como os de fluxos contínuos de dados, seu desempenho preditivo tende a ser melhor do que aqueles de classificadores únicos (BIFET et al., 2009a), já que combinam o resultado de vários classificadores a fim de melhorar o desempenho preditivo final.

- *Single Classifier Drift* - classificador único
- *Accuracy Updated Ensemble* - comitê de classificadores
- *Leveraging Bagging* - comitê de classificadores
- *Limited Attribute Classifier* - comitê de classificadores
- *Oza Bagging with ADWIN* - comitê de classificadores
- *Oza Bagging with Adaptative Size Hoeffding Tree* - comitê de classificadores

Neste trabalho, todos os classificadores citados utilizam a *Hoeffding Tree* ou suas variações como o classificador base, pois é um algoritmo extremamente rápido para a execução da classificação, possuindo uma complexidade computacional constante – $O(1)$ (HULTEN; SPENCER; DOMINGOS, 2001). Além disso, as árvores são modelos interpretáveis, que permitem ao especialista de domínio conhecer como as decisões foram tomadas pelo modelo.

O *Single Classifier Drift*, que dentre os selecionados é o único que possui a propriedade de trabalhar com um único classificador (GAMA et al., 2004), foi selecionado devido à sua capacidade de identificar mudanças na distribuição dos dados, sendo capaz de adaptar o modelo rapidamente.

4.6 Medidas de Avaliação

A fim de validar as hipóteses $H1$, $H2$ e $H3$, um importante passo é avaliar o desempenho preditivo dos classificadores em bases de dados de detecção de intrusão. Então, para avaliá-los, as seguintes medidas foram escolhidas: revocação, precisão e taxa de alarme falso.

Cada uma das medidas de avaliação, vistas na Seção 2.4.2, permitem a análise do desempenho preditivo dos classificadores sobre uma óptica diferente, complementando o resultado uma das outras. A revocação (Equação 2) indica a sensibilidade do modelo para classificar as instâncias, seu resultado indicará quantos pacotes de *Ataque* foram classificados dentre todos que realmente eram ataques. A precisão (Equação 3) indica a razão entre os pacotes classificados corretamente como *Ataque* dentre todos aqueles que foram classificados como *Ataque* (incluindo os falso positivos). Já a taxa de alarme falso (Equação 4) é geralmente utilizada para avaliar o desempenho preditivo dos IDSs, isso porque ela mede a quantidade de pacotes do tipo $Normal$ que foram classificados como *Ataque* indicando, assim, o percentual de alarmes falsos.

4.7 Escolha da Base de Dados para Validar as Hipóteses

A escolha da base de dados é um passo importante para que os métodos propostos fossem executados de forma satisfatória. A base de dados deveria contemplar algumas características essenciais como possuir tipos de ataques recentes e que sejam comumente utilizados em trabalhos correlatos além de possuir tanto os dados brutos dos pacotes quanto os respectivos fluxos. Por último, era importante que as instâncias estivessem rotuladas como as classes *Ataque* ou $Normal$ para que os treinamentos fossem possíveis de serem executados.

É considerável destacar que os dados brutos dos pacotes, geralmente, são disponibilizados no formato PCAP e isso exige um pré-processamento para transformá-los em um formato adequado para o processamento. Destaca-se também que técnicas de seleção de atributos fossem aplicadas à base, a fim de selecionar somente aqueles que possuíssem alguma correlação com a classe alvo. Isso auxiliou na melhoria da qualidade da base e consequentemente no desempenho preditivo dos classificadores.

4.8 Considerações Finais

Este capítulo mostrou o método proposto para verificar cada uma das hipóteses consideradas neste trabalho, incluindo: algoritmos de classificação utilizados e as medidas de avaliação e seleção da base de dados para a validação das hipóteses propostas. Também

mostrou a importância de se executar a seleção de atributos para que somente aqueles que possuem uma correlação com a classe alvo sejam selecionados. Por fim, indicou a importância da seleção de uma base de dados que possa refletir a real condição de funcionamento dos IDSs.

Capítulo 5

Experimentos e Análise dos Resultados

Neste capítulo, serão detalhados os experimentos realizados para validar o método proposto no Capítulo 4. Conforme apresentado anteriormente, este trabalho irá testar três hipóteses. A hipótese $H1$, a qual visa identificar se os classificadores obtiveram diferenças significativas de desempenho preditivo quando comparada a classificação do fluxo e a do pacote. A hipótese $H2$, a qual investigou se o atraso na entrega dos rótulos das instâncias aos classificadores impactou significativamente o seu desempenho preditivo. A hipótese $H3$, a qual investigou o impacto da entrega de apenas um subconjunto de instâncias rotuladas para a atualização do modelo, nessa hipótese, técnicas de aprendizado ativo foram utilizadas. Tendo sido todas as hipóteses testadas usando o mesmo conjunto de dados, o qual foi previamente pré-processado.

A Seção 5.1 descreve todas as características da base de dados CICIDS2017 utilizada neste trabalho. A Seção 5.2 descreve o processo de pré-processamento aplicado à base de dados selecionada. A Seção 5.3 descreve como os melhores atributos foram selecionados na tentativa de melhorar o desempenho preditivo da classificação. A Seção 5.4 discute sobre as características do *framework* MOA. Por fim, a Seção 5.5 descreve todos os experimentos realizados para testar as hipóteses $H1$, $H2$ e $H3$ bem como a análise dos resultados obtidos.

5.1 Base de Dados

A base de dados para os experimentos exigiu requisitos importantes como possuir os dados brutos dos pacotes e também do fluxo, e ser uma base atual que contivesse diferentes tipos de ataques, além de possuir os rótulos das instâncias identificando-as como *Normal* ou *Ataque*. Dentre as diversas bases analisadas, a CICIDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2018) foi selecionada para ser utilizada em todos os experimentos devido à sua completude diante dessas exigências para os experimentos. Originalmente, essa base possui dados do fluxo de pacotes disponibilizados em um arquivo *Comma-separated Values* (CSV), e os pacotes brutos em arquivos do tipo PCAP. A base

foi organizada em cinco conjunto de dados, um para cada dia da semana, sendo que cada um deles possui tipos de ataques distintos. Na tentativa de emular o funcionamento de uma rede real, seis perfis de ataques foram gerados pelos autores:

- **Ataque Força Bruta:** é um dos ataques mais comuns sendo bastante utilizado para decifrar senhas. Porém, pode ser utilizado também para descobrir páginas e conteúdos ocultos em aplicações na rede.
- **Ataque *HeartBleed*:** devido a um erro na biblioteca de criptografia *OpenSSL*, que é amplamente utilizada no protocolo *Transport Layer Security* (TLS), essa vulnerabilidade é explorada enviando uma solicitação malformada com uma pequena carga útil e um campo com comprimento exagerado para a área vulnerável (geralmente um servidor) a fim de obter uma resposta da "$Rede - Vitima$".
- **Ataque de Negação de Serviço (*Denial of Service* (DoS)):** nesse tipo de ataque o invasor busca deixar uma máquina ou recurso de rede indisponíveis temporariamente. Geralmente, isso ocorre com uma grande quantidade de requisições ao recurso de destino com solicitações sem função, a fim de sobrecarregar os sistemas e evitar que as solicitações legítimas sejam atendidas.
- **Ataque de Negação de Serviço Distribuído (*Distribuited Denial of Service* (DDoS)):** ocorre da mesma forma que o ataque DoS, porém com vários sistemas distribuídos executando o ataque de forma sincronizada.
- ***BotNet*:** é um tipo de ataque cibernético que utiliza um grupo de equipamentos conectados a Internet, controlados por um agente malicioso, a fim de prover ataques que vão desde o envio de spam e roubo de dados, atá ataques de negação de serviços.
- **Ataque Web:** são ataques que utilizam as páginas disponíveis na Internet para que, por exemplo, executem ataques de Injeção SQL, onde o invasor cria um comando *Structured Query Language* (SQL) e, em seguida, usa-o para forçar o banco de dados a responder as requisições diretamente.
- **Ataques de Infiltração:** esse tipo de ataque ocorre explorando vulnerabilidades de aplicações instalados na máquina da vítima. Caso o ataque seja bem sucedido, uma aplicação será executada sem consentimento, deixando assim a vítima totalmente vulnerável a ataques como varredura de IP, varredura de portas entre outros.

Os dados dos ataques da base CICIDS2017 foram capturados de forma contínua durante a semana de 3 a 7 de Julho de 2017. No primeiro dia de captura, obteve-se somente tráfego de pacotes do tipo $Normal$, sendo que, para os experimentos deste trabalho, o referido dia foi descartado. Todos os outros dias da semana possuem ataques, foram distribuídos em dois períodos, um pela manhã e outro à tarde. Sendo a distribuição destes ataques apresentada na Tabela 4.

Tabela 4 – Distribuição dos ataques por dia da semana e período do dia (Adaptado de: Sharafaldin, Lashkari e Ghorbani (2018)).

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

A base do fluxo de rede foi criada utilizando a aplicação *CICFlowMeter* (Habibi Lashkari. et al., 2017) que consegue extrair, na versão utilizada para este trabalho, cerca de 80 atributos baseados nos arquivos de pacotes de rede. Para a realização dos experimentos foram identificados dois macro grupos de arquivos, sendo o de fluxos designados aqui como *FlData* e o de pacotes nomeados como *PkData*. Para um melhor controle dos experimentos, foram criados quatro grupos, um para cada dia da semana. Os grupos e a quantidade de fluxos e pacotes são descritos na Tabela 5, como também o percentual que cada tipo de instância (*Normal* e *Ataque*) representa sobre as bases *FlData* e *PkData*. Os experimentos foram executados aplicando os algoritmos citados na Seção 4.5 tanto na bases *FlData* como na base *PkData*.

Tabela 5 – Resumo do conjunto de dados de fluxo e pacotes (Fonte: O autor (2021)).

Dia da Semana	***FlData***		***PkData***	
	Normal	**Ataque**	**Normal**	**Ataque**
Terça	432.074 (96,90%)	13.835 (3,10%)	9.931.455 (97,37%)	268.706 (2,63%)
Quarta	440.031 (63,52%)	252.672 (36,48%)	11.234.606 (81,52%)	2.546.737 (18,48%)
Quinta	456.734 (99,52%)	2.216 (0,48%)	6.797.681 (98,84%)	79.895 (1,16%)
Sexta	414.322 (58,92%)	288.923 (41,08%)	2.525.311 (65,83%)	1.310.760 (34,17%)

5.1.1 Distribuição dos Pacotes *Ataque* e *Normal* na Base de Dados

Nas Figuras 17, 18, 19 e 20 é apresentado o detalhamento da distribuição dos pacotes nos arquivos gerados para a base *PkData* para cada dia da semana. Nessas figuras, o eixo X representa as janelas de 100.000 pacotes e o eixo Y a distribuição dos pacotes, onde

as barras vermelhas indicam *Ataque* e as verdes os pacotes do tipo *Normal*. Nelas, é possível identificar o momento em que os pacotes de *Ataque* aparecem no fluxo em cada uma das bases.

A Figura 17 apresenta a distribuição da base *PkData* de terça-feira. A base é composta por 9.931.455 pacotes da classe *Normal* que representaram 97, 37% do total. Já a classe *Ataque* foi constituída de 268.706 pacotes, que significava 2, 63% do total de pacotes. Este detalhamento mostrou a existência de dois períodos distintos de ataques, os primeiros apareceram na janela de 5.100.000 pacotes persistindo até a janela de 6.900.000 pacotes e, logo em seguida, os ataques cessaram, voltando a ocorrer na janela de 9.200.000 pacotes e perdurando até o final.

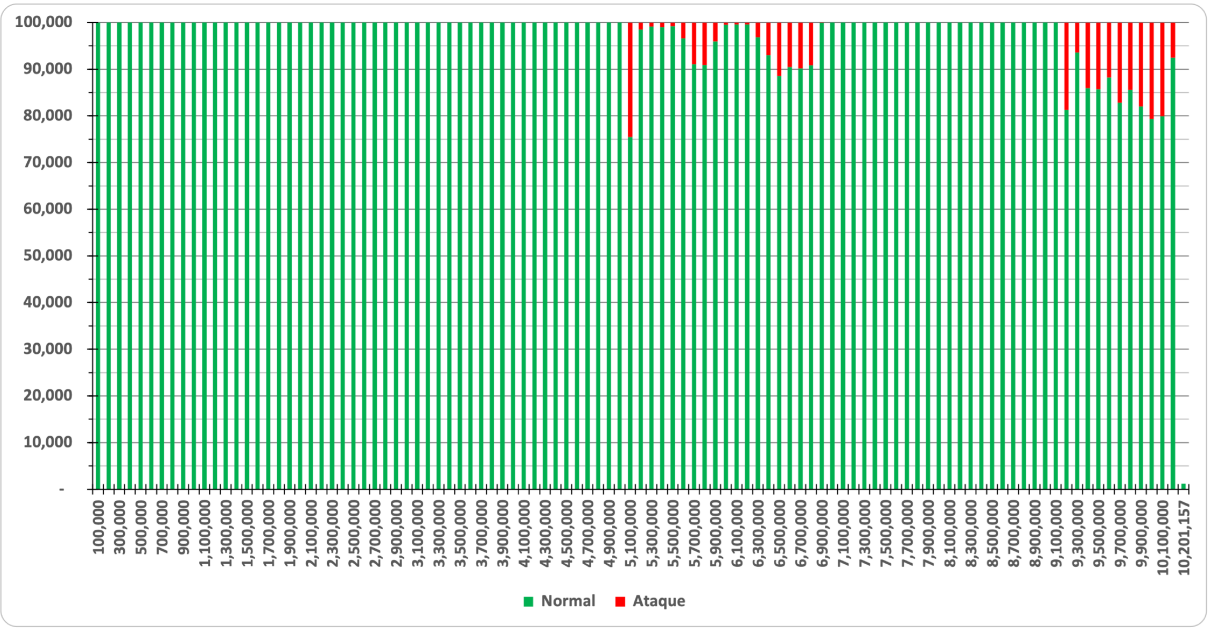


Figura 17 – Distribuição dos pacotes do tipo *Ataque* e *Normal* para a base de terça-feira (Fonte: O autor (2021)).

A Figura 18 apresenta a distribuição da base *PkData* de quarta-feira. Os pacotes da classe *Normal*, representaram 81, 52% do total da base, o que em termos absolutos correspondeu a 11.234.606 pacotes. Já a classe *Ataque* foi representada por 2.546.737 pacotes, que significou 18, 48% do total. Ao todo, foram 13.781.343 pacotes que fizeram com que a base de quarta-feira fosse a maior em termos de quantidade de instâncias. No detalhamento, observou-se entre as janelas 6.700.000 e 10.200.000 uma grande sequencia de pacotes do tipo *Ataque*. Ao finalizar-se a primeira onda de ataques, uma segunda foi observada entre as janelas 12.400.000 e 12.800.000 pacotes, porém essa com bem menos pacotes do tipo *Ataque* que a anterior.

A Figura 19 apresenta a distribuição da base de quinta-feira. Na qual, os pacotes do tipo *Normal* somaram um total de 6.797.681 pacotes, ou seja, 98, 84% do total. Já os pacotes do tipo *Ataque* somaram 79.895 ou 1, 16% dos 6.877.576 pacotes totais desta

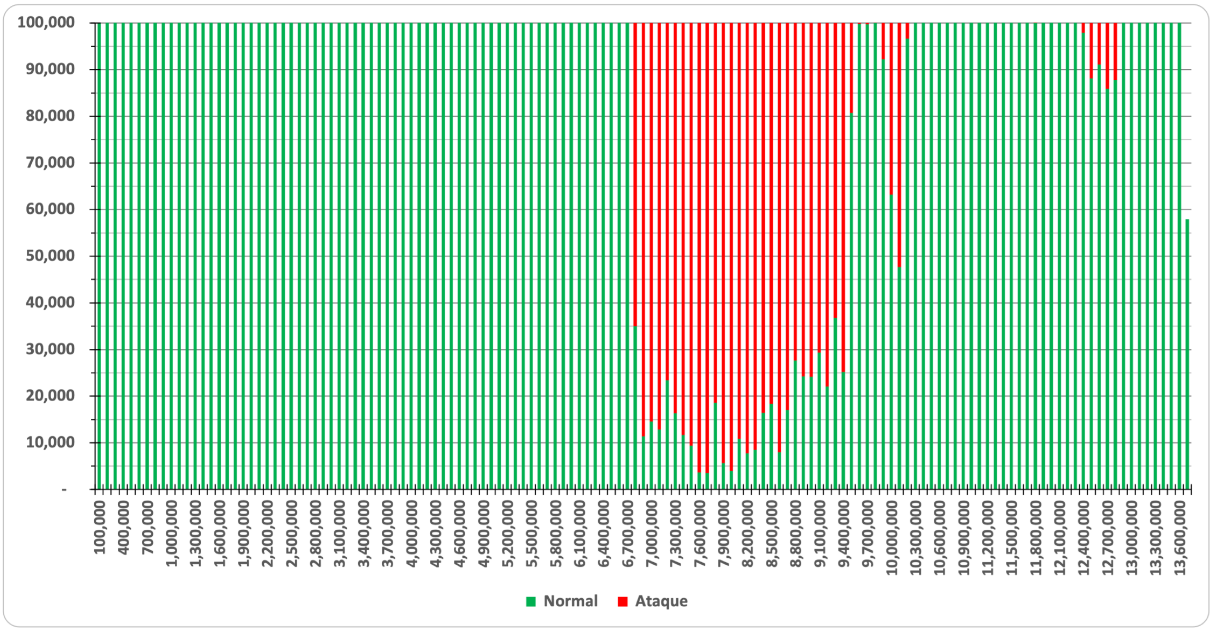


Figura 18 – Distribuição dos pacotes do tipo *Ataque* e *Normal* para a base de quarta-feira (Fonte: O autor (2021)).

base. É importante destacar, que esta base é a que apresentou o maior desbalanceamento entre as instâncias da classe *Normal* e *Ataque* dentre todas as demais do experimento.

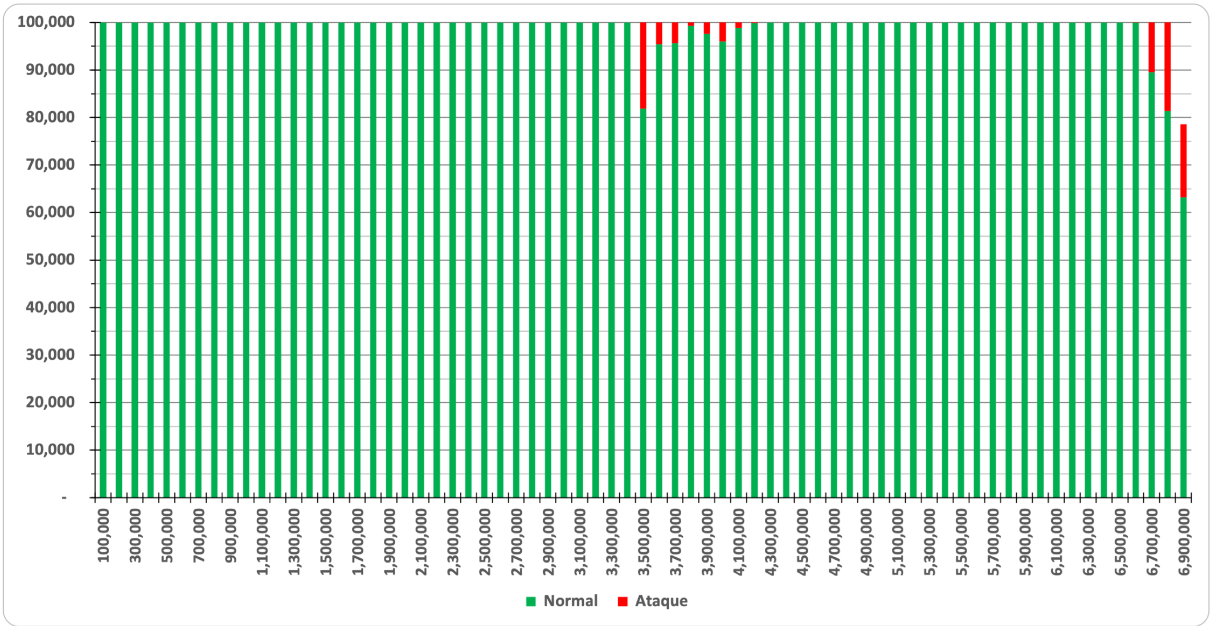


Figura 19 – Distribuição dos pacotes do tipo *Ataque* e *Normal* para a base de quinta-feira (Fonte: O autor (2021)).

Por fim, a Figura 20 apresenta o perfil da base de sexta-feira. Onde os pacotes do tipo *Normal* foram $65,83\%$ ou 2.525.311 dos 3.836.071 pacotes totais. Já os pacotes do tipo *Ataque* representaram $34,17\%$ ou 1.310.760 do total de pacotes. Possuindo a referida base duas importantes particularidades. A primeira relacionada ao tamanho e

a segunda associada ao balanceamento da mesma. A qual apresentou, dentre todas as bases utilizadas, a menor quantidade de dados disponíveis nos experimentos e também o conjunto mais balanceado. De modo que, duas grandes ondas de ataques puderam ser observadas. A primeira entre as janelas 1.600.000 e 2.000.000 pacotes e, a segunda relativamente maior, ocorreu entre as janelas 2.400.000 e 3.800.000 pacotes.

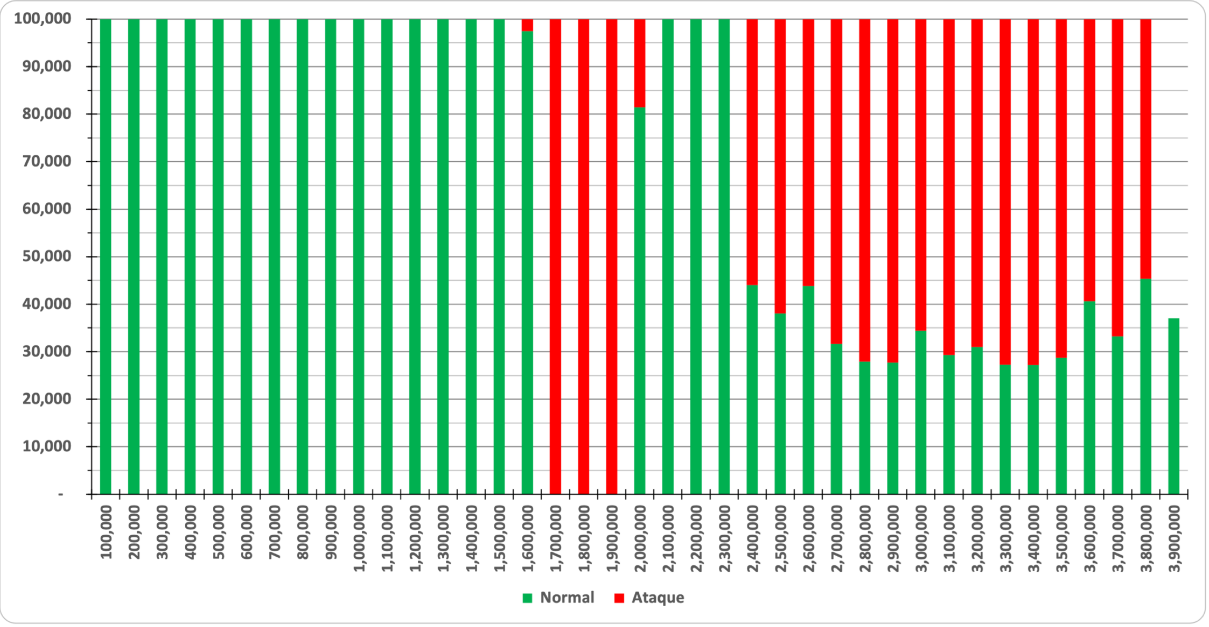


Figura 20 – Distribuição dos pacotes do tipo *Ataque* e *Normal* para a base de sexta-feira (Fonte: O autor (2021)).

5.2 Geração das Bases para os Experimentos

O processo utilizado para pré-processar a base pode ser visto na Figura 21 e consistiu em gerar os arquivos em um formato que os algoritmos de classificação pudessem processá-los.

O processo de conversão dos arquivos foi dividido em três etapas que serão detalhadas a seguir.

1. Na etapa 1, foi feita a leitura dos arquivos PCAP identificando campos como: protocolo, *Flags*, tamanho, data e hora de captura, portas e endereços de origem e destino além da carga útil destes pacotes. Feita essa identificação, foi necessário transformá-lo em um arquivo intermediário no formato CSV para que ele pudesse ser utilizado na etapa seguinte do processo.

2. A etapa 2 consistiu na leitura dos arquivos de fluxo no formato CSV e, a partir deles, foi possível identificar quais pacotes pertenciam aos respectivos fluxos, baseando-se na data e hora inicial e final do fluxo, além do protocolo e quantidade de pacotes.

Assim, foi possível rotular todos os pacotes como *Ataque* ou *Normal*, conforme a classe que o fluxo que ele pertence.

3. A etapa 3 consistiu na construção de uma aplicação que convertesse os arquivos *FlData* (disponibilizados em formato CSV) e *PkData* (disponibilizados em formato PCAP), para o formato *Attribute-Relation File Format* (ARFF). Os arquivos no formato ARFF são utilizados pelo *framework* MOA na tarefa de classificação e consequente avaliação. Em seguida realizou-se a geração do arquivo ARFF com todos os atributos disponíveis nos arquivos de pacotes e do fluxo de rede. Portanto, ao final do processo, para cada dia da semana foram gerados dois arquivos ARFF, um contendo as instâncias e atributos dos fluxos e outro contendo todo os atributos dos pacotes de rede.

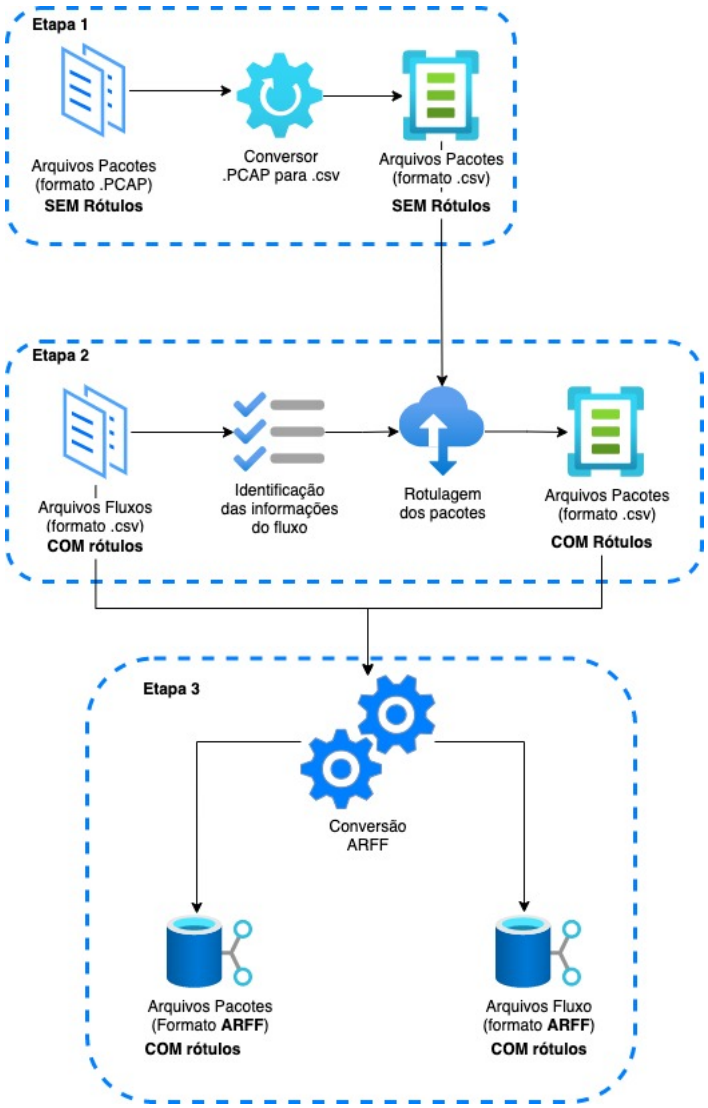


Figura 21 – Etapas do processo de conversão dos arquivos PCAP e fluxo (Fonte: O autor (2021)).

5.3 Seleção de Atributos

A seleção de atributos é considerada uma etapa importante na fase de pré-processamento dentro do processo do KDD, pois nela são eleitos os atributos de maior significância para o processo de classificação. Nessa etapa é permitida, por exemplo, a ordenação dos atributos por importância, de modo que aqueles menos significativos não participem do processo de classificação. Além de possibilitar que a dimensionalidade da base seja reduzida. Nesta etapa os seguintes passos foram executados: i) remoção dos atributos irrelevantes; ii) cálculo da correlação de *Pearson* (r^2) entre os atributos da base *PkData* e a classe alvo; e iii) remoção dos atributos com correlação menor ou igual a 0.

Assim, inicialmente selecionou-se a base *PkData* para o processo de seleção de atributos. Essa escolha se deu pois a base *FlData* possui os dados agregados da base *PkData*. Em seguida, os atributos então selecionados dentre os 21 possíveis na base *PkData* tiveram os seus correspondentes selecionados na base *FlData*. Os quais referem-se aos dados dos pacotes dos protocolos IP, TCP e UDP.

Adiante, na tentativa de remover qualquer tipo de enviesamento das informações, os atributos *EnderecoOrigem*, *EnderecoDestino* foram excluídos da base, uma vez que os mesmos permitiriam a identificação direta tanto do dispositivo de origem do ataque quanto o da vítima. Sendo esse tipo de procedimento usual em trabalhos de análise de tráfego. Já os atributos *TipoServico* e *VersaoIP* foram excluídos da base pois não possuíam significância para o processo de classificação.

Já para o estudo de correlação, desenvolveu-se uma aplicação na linguagem *Python* com o auxílio da biblioteca *Scikit-Learn* (PEDREGOSA et al., 2011) que analisou a correlação de *Pearson* entre cada atributo e a classe alvo. De modo que a aplicação analisou a base *PkData* com todos os dias da semana. Com o resultado, os atributos foram dispostos em ordem crescente de correlação. Posteriormente, foram selecionados somente aqueles com a correlação maior que 0. Como pode ser visto na Tabela 6, na qual os atributos *TCP Reserved*, *TCP Urgent Pointer*, *URG Flag* e Fragmento IP foram removidos por se enquadrarem na regra de corte estabelecida. Além disso, nenhum dos atributos obteve correlação negativa com a classe alvo.

Com o objetivo de manter o mesmo tipo de informação nas bases *PkData* e *FlData*, uma vez selecionados os atributos da base *PkData*, o seus correspondentes foram selecionados na base *FlData*, vide Tabela 7.

5.4 *Massive Online Analysis* - MOA

Para a execução dos experimentos, foi utilizado o *framework* MOA, desenvolvido por Bifet et al. (2010), que disponibiliza um conjunto de algoritmos para fluxos contínuos de dados para diferentes tarefas de aprendizado de máquina. O *framework* também

Tabela 6 – Resultado final da análise de correlação de *Pearson* entre os dados do pacote e a classe alvo (Fonte: O autor (2021)).

Atributo	**Correlação**
ACK *Flag*	0,17
RST *Flag*	0,16
TTL	0,16
SYN *Flag*	0,14
PSH *Flag*	0,14
Identificação IP	0,11
Protocolo	0,10
Porta de Origem	0,10
Porta de Destino	0,08
Tamanho do Cabeçalho	0,08
Tamanho Total do Pacote	0,06
FIN *Flag*	0,06
TCP Ack Number	0,02
TCP Reserved	0,00
TCP Urgent Pointer	0,00
URG *Flag*	0,00
Fragmento IP	0,00

permite que novos algoritmos desenvolvidos pela comunidade científica sejam incorporados ao mesmo, a fim de estender a sua utilização em diferentes problemas. O MOA foi desenvolvido em Java, com a possibilidade de uso tanto através de interface gráfica quanto através de linha de comando.

Entre os algoritmos disponíveis no MOA para diferentes tarefas, destacam-se os algoritmos de classificação, incluindo aqueles usados nesse trabalho. Além disso, há vários métodos de avaliação, funcionalidades de exportação dos resultados de análise e medidas de desempenho preditivo para avaliação de classificadores. Quanto aos métodos de avaliação, o método de sequenciamento preditivo está disponível, sendo o mesmo utilizado no presente trabalho e discutido na Seção 2.4.1.

Adicionalmente, o MOA permite a utilização de estratégias de avaliação como o sequencialmente preditivo com atraso, que possibilita o atraso dos rótulos, não entregando-os imediatamente ao classificador. É possível configurar esse atraso em número de instâncias, além da possibilidade do classificador utilizar uma janela inicial de treinamento. Essa estratégia foi utilizado para os experimentos que testaram a hipótese $H2$.

Tabela 7 – Relacionamento entre os campos dos pacotes (a) e do fluxo de rede (b) (Fonte: O autor (2021)).

Atributos do Pacote (a)	**Atributos do Fluxo de Rede(b)**
Porta de Origem	Porta de Origem
Porta de Destino	Porta de Destino
Protocolo	Protocolo
Tamanho Total do Pacote	Tamanho Total dos Pacotes Enviados
	Tamanho Total dos Pacotes Retornados
FIN *Flag*	Contagem Fin *Flag*
SYN *Flag*	Contagem SYN *Flag*
RST *Flag*	Contagem RST *Flag*
PSH *Flag*	Contagem PSH *Flag*
ACK *Flag*	Contagem ACK *Flag*
Tamanho do Cabeçalho	Tamanho do pacotes encaminhados
TTL	Não Disponível
Número TCP Ack	Não Disponível

Por fim, o MOA permite testar diferentes estratégias de aprendizado ativo, tais como: *ALUncertainty* e *ALRandom*, sendo esta última utilizada no presente trabalho. Ainda dentro das estratégias de aprendizado ativo, foi desenvolvido dentro do MOA um avaliador que mesclou o aprendizado ativo com o atraso dos rótulos. Nesta implementação, os rótulos selecionados pela estratégia *ALRandom* não foram entregues imediatamente ao classificador, ao invés disso, foi aplicado um atraso nos rótulos conforme um parâmetro pré-configurado.

5.5 Experimentos e resultados

Os resultados dos experimentos de validação das hipóteses propostas são apresentados e discutidos nesta seção. Na Tabela 8 é visto um resumo dos experimentos agrupados por hipóteses. Tanto as bases, quanto os avaliadores e os classificadores utilizados estão divididos por experimentos.

Tabela 8 – Resumo dos experimentos por hipótese (Fonte: O autor (2021)).

Hipótese	Experimento	Base de Dados	Avaliadores	Classificadores Utilizados
H1	Analisar o desempenho preditivo dos classificadores considerando fluxo e pacote de dados	FlData PkData	Sequenciamento Preditivo	*SingleClassifierDrift* *AccUpdatedEnsemble* *LeveragingBag* *LimAttClassifier* *OzaBagAdwin* *OzaBagASHT*
H2	Analisar se o atraso dos rótulos influenciam no desempenho preditivo do classificador	PkData	Sequenciamento Preditivo com atraso	*LimAttClassifier* *LeveragingBag* *OzaBagAdwin* *OzaBagASHT*
H3	Analisar o desempenho do classificador aplicando a técnica de aprendizado ativo	PkData	Sequenciamento Preditivo	*OzaBagASHT*

5.5.1 Analisar o Desempenho Preditivo dos Classificadores Utilizando Inspeção Individual de Pacotes

A hipótese $H1$ objetivou investigar se os classificadores obtêm um desempenho preditivo semelhante, tanto para a classificação do fluxo quanto para o pacotes individuais. Para isso, o experimento utilizou os classificadores apresentados na Seção 4.5, tanto para *FlData* quanto para *PkData* sendo que, todos os experimentos, foram executados utilizando o *framework* MOA e os valores dos parâmetros utilizados pelos classificadores foram os padrões definidos no *framework*.

É importante destacar que *FlData* é uma versão agregada do *PkData* e, portanto, possui um número menor de instâncias. Embora os conjuntos de dados não possam ser diretamente comparáveis, eles têm a mesma sequência de instâncias das classes $Normal$ e de *Ataque*. Além disso, eles dizem respeito aos mesmos acessos a uma rede de computadores, na mesma ordem, um de forma detalhada e outro de forma sumarizada.

As Figuras 22, 23, 24 e 25 sintetizam os resultados coletados do experimento. Nas colunas destas figuras estão dispostos os seis algoritmos de classificação analisados, enquanto as linhas apresentam as medidas de avaliação: precisão, revocação e taxa de alarmes falsos (FAR). Em cada célula, podem ser visualizados os resultados percentuais para a base *FlData* na cor azul e *PkData* na cor laranja para o avaliador de sequenciamento preditivo.

Considerando o conjunto de dados apresentados na Figura 22, os resultados da revocação e precisão são bastante similares entre *FlData* e *PkData*. Considerando a precisão, todos os algoritmos de classificação atingiram resultados aproximados. Uma exceção ocor-

reu com os classificadores *SingleClassifierDrift* e *OzaBagASHT* no qual a diferença foi em torno de 5 pontos percentuais. Considerando a medida de revocação, a maior diferença atingida, foi de 15 pontos percentuais para o classificador *AccUpdateEnsemble* a favor do conjunto de dados *FlData*. Contudo, quando utilizado o classificador *LimAttClassifier*, *PkData* superou o *FlData* em 8 pontos percentuais. Quando considerada a medida FAR, a maior diferença entre *PkData* e *FlData* é de $0,15$ pontos percentuais quando utilizado o classificador *OzaBagASHT*.

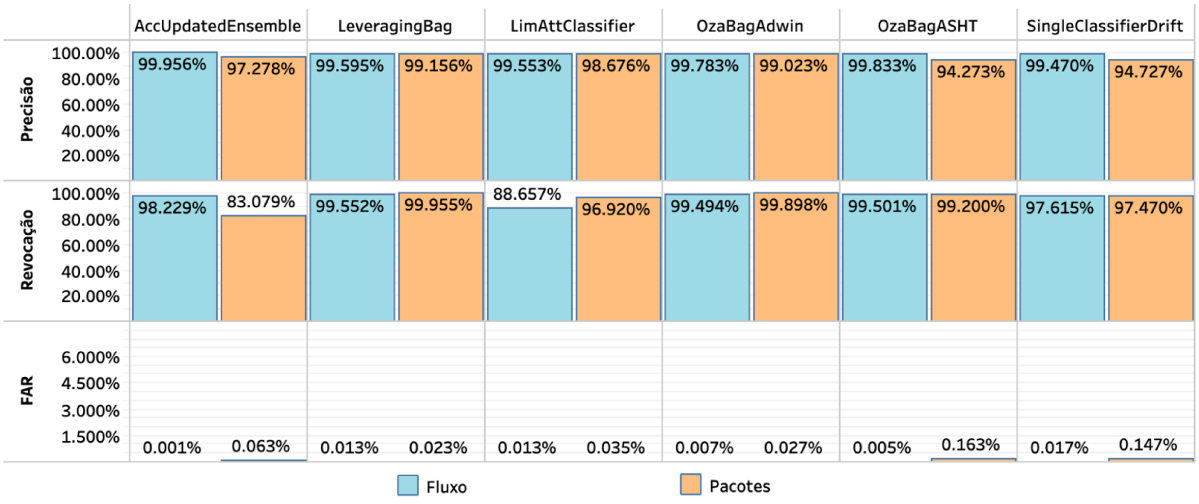


Figura 22 – Comparativo de desempenho preditivo dos algoritmos para classificação do Fluxo e dos Pacotes na base de terça-feira (Fonte: O autor (2021)).

A Figura 23 apresenta o resultados dos experimentos para a base de quarta-feira, a qual possui dois destaques. O primeiro é que, somente para o classificador *AccUpdateEnsemble*, o resultado de revocação de *FlData* foi substancialmente melhor que *PkData*, sendo que nos demais, todos os resultados foram semelhantes. O segundo é, que a medida FAR, foi ligeiramente menor em todos os classificadores para *FlData*. Fato que pode estar relacionado ao número elevando de instâncias que *PkData* possui frente a *FlData*.

A Figura 24 resume os resultados para a base de quinta-feira, analisando somente as medidas de precisão e FAR *PkData* atingiu resultados melhores que *FlData* em todos os classificadores. O mesmo ocorreu para a revocação, exceto para o classificador *AccUpdateEnsemble*. Sendo também importante destacar o classificador *LimAttClassifier*, o qual a revocação atingiu $35,321\%$ para *FlData*. Uma possível explicação para este resultado seria o elevando desbalanceamento da base de quinta-feira, na qual as poucas instâncias de ataques de *FlData* não foram suficientes para treinar um modelo de decisão que reconhecesse novos ataques.

Os resultados apresentados na Figura 25 referem-se a base de sexta-feira, nos quais as medidas de revocação e precisão foram semelhantes tanto para *FlData* quanto para *PkData*, ou em alguns casos um pouco melhor para *FlData*. Porém, observou-se um aumento significativo para o FAR e, como a base possui três tipos diferentes de ataques,

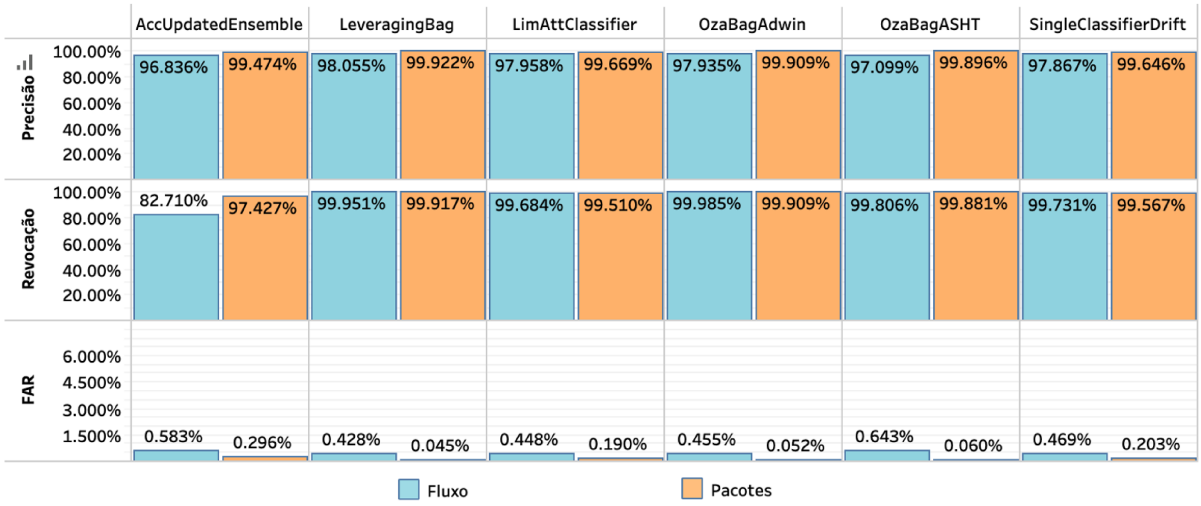


Figura 23 – Comparativo de desempenho dos algoritmos para classificação do Fluxo e dos Pacotes na base de quarta-feira (Fonte: O autor (2021)).

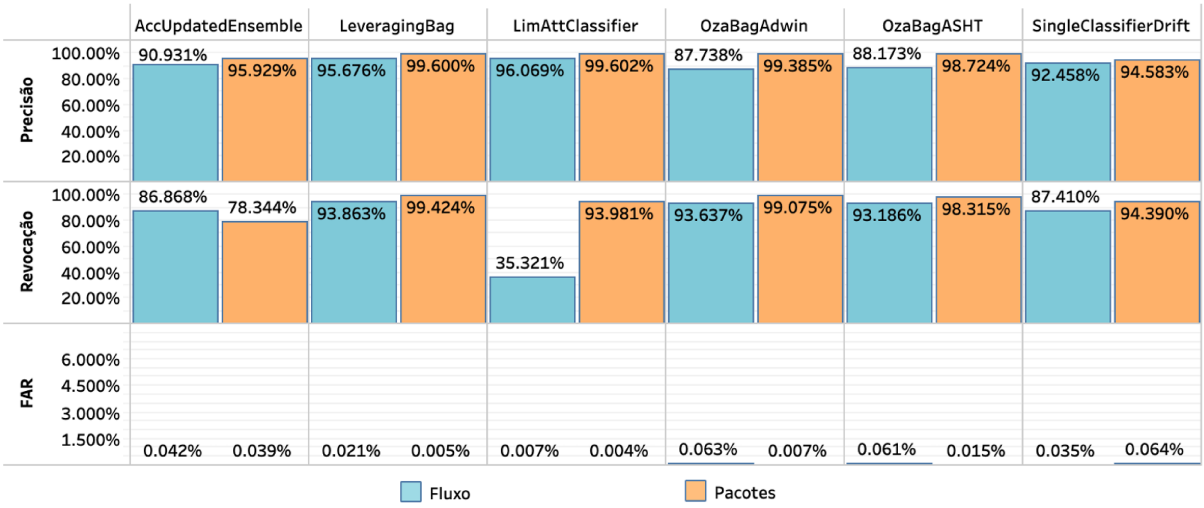


Figura 24 – Comparativo de desempenho preditivo dos algoritmos para classificação do Fluxo e dos Pacotes na base de quinta-feira (Fonte: O autor (2021)).

isso pode indicar que os classificadores precisaram de um tempo maior para iniciar a classificação correta de cada um dos ataques.

5.5.1.1 Comparação do Desempenho Preditivo dos Classificadores

No geral, os resultados indicaram grande semelhança no desempenho preditivo dos classificadores usando *PkData* e *FlData*. Assim, baseando-se nos resultados obtidos, verificou-se a possibilidade de equivalência no desempenho preditivo dos classificadores sob os dois conjuntos de dados. Assumindo como hipótese nula (H_0) não haver diferença significativa entre os modelos induzidos por *PkData* e *FlData*. Para isso, aplicou-se o teste de *Wilcoxon* para cada algoritmo de classificação e cada medida de avaliação, contrastando a medida de avaliação obtida com cada conjunto de dados. Em geral, ao utilizar um teste estatístico o objetivo é rejeitar a hipótese nula e provar que existe diferença entre

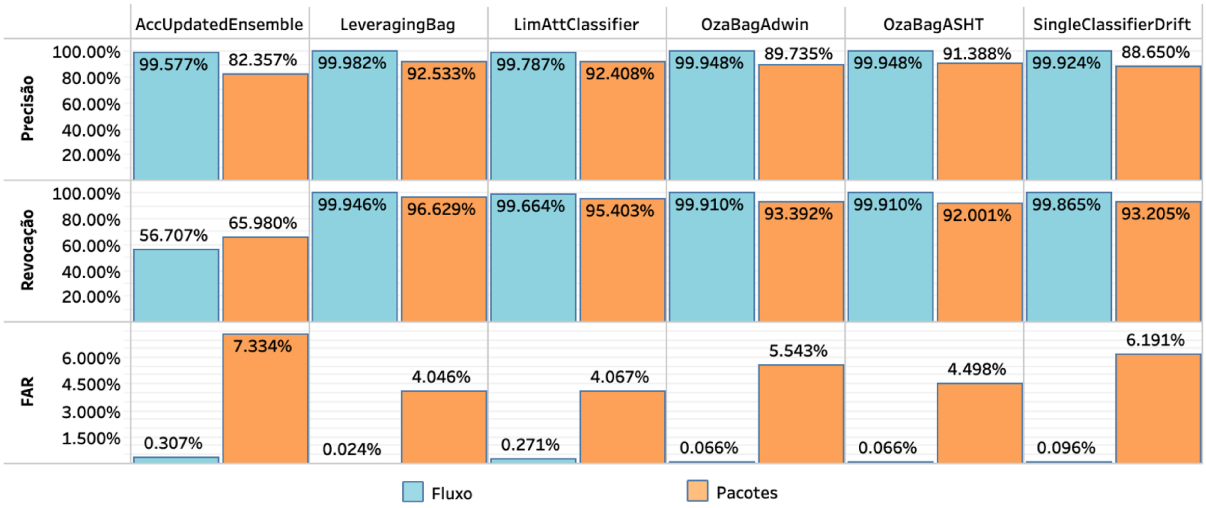


Figura 25 – Comparativo de desempenho preditivo dos algoritmos para classificação do Fluxo e dos Pacotes na base de sexta-feira (Fonte: O autor (2021)).

os resultados.

Após a aplicação do teste de *Wilcoxon*, verificou-se que a hipótese nula não foi rejeitada na maioria dos cenários, exceto para a medida FAR nos algoritmos *AccUpdatedEnsemble* e *OzaBagASHT*, considerando um nível de significância 5%. No entanto, não rejeitar a hipótese nula não implica necessariamente em aceitá-la, de modo que a classificação feita pelos algoritmos utilizando *PkData* e *FlData* podem ser consideradas equivalentes. Com isso em mente, avaliou-se as diferenças de desempenho preditivo usando intervalos de confiança para cada caso em que a hipótese nula não foi rejeitada. A Tabela 9 resume o intervalo de confiança de 95% para a diferença entre as bases *PkData* e *FlData* para a mediana das distribuições. Assim, suponha que o intervalo de confiança analisado contenha o zero e possua uma baixa amplitude; neste caso, existiriam evidências estatísticas de que a diferença entre *PkData* e *FlData* é muito pequena, ou seja, os algoritmos induzidos por ambas as bases são semelhantes. Observa-se que, para cada medida de avaliação, há pelo menos um classificador com desempenho estatístico semelhante para *PkData* e *FlData*. Por exemplo, o intervalo de confiança das diferenças entre *PkData* e *FlData* para a precisão do classificador *LeveragingBag* contém zero e possui uma baixa amplitude (com a confiança de 95% a diferença máxima é em torno de 6%). Este resultado indica que, em várias situações, o desempenho da detecção baseada em pacotes é semelhante à detecção baseada em fluxo, mostrando a viabilidade de construir IDSs baseados na análise individual dos pacotes.

5.5.1.2 Avaliação de Desempenho dos Classificadores para a Base de Pacotes

Diante dos resultados apresentados nas Figuras 22, 23, 24 e 25, *PkData* foi analisada com o intuito de encontrar a existência de um classificador com desempenho preditivo superior aos demais. Analisando as referidas medidas de desempenho preditivo, verificou-

Tabela 9 – Intervalo de confiança, por medida de avaliação aplicado aos classificadores analisados (Fonte: O autor (2021)).

Classificador	Precisão			Revocação			FAR		
	min	**max**	**valor-p**	**min**	**max**	**valor-p**	**min**	**max**	**valor-p**
AccUpdatedEnsemble	-0,1019740	0,0184010	0,3828000	-0,1543695	0,0042785	0,1484000	0,0002110	0,0371955	0,0390600
LeveragingBag	-0,0660490	0,0502165	0,4609000	-0,0331690	0,1731545	0,4609000	-0,0001355	0,0316980	0,1094000
LimAttClassifier	-0,0669265	0,2074705	0,4609000	-0,0294985	0,4206300	0,1484000	-0,0007680	0,0195405	0,2500000
OzaBagAdwin	-0,0611990	0,0689100	0,7422000	-0,0323050	0,0811050	0,5469000	-0,0002655	0,0291895	0,1484000
OzaBagASHT	-0,0856030	0,0387745	0,6406000	-0,0428115	0,0491950	0,5469000	0,0005600	0,0304545	0,0390600
SingleClassifierDrift	-0,1053305	0,0017330	0,0546900	-0,0461280	0,1108035	0,5469000	-0,0021600	0,0318025	0,1094000

se que o classificador *LeveragingBag* superou os demais classificadores em todos os cenários para todas as medidas de avaliação. A única exceção ocorreu nos dados da base de quinta-feira (Figura 24). Contudo, nesse caso, a diferença entre o classificador *LeveragingBag* e *LimAttClassifier* é insignificante. Considerando somente o classificador *LeveragingBag*, a precisão e a revocação atingiram resultados acima de 98% em todos os experimentos. Além disso, o FAR ficou abaixo de 5%, o que representa em termos práticos que menos de 5% dos pacotes normais foram classificados como ataques. Havendo uma única exceção, apresentada na Figura 25, onde a precisão girou em torno de 92%, a revocação 96% e o FAR de 4%.

A fim de confirmar a significância estatística dos resultados e se o algoritmo *LeveragingBag* realmente apresenta um desempenho preditivo melhor que os demais, um teste estatístico de significância foi realizado. Primeiro, foi aplicado o teste estatístico de *Friedman* com um nível de confiança de 95% ($\alpha = 0,05$) em cada um dos resultados das medidas de avaliação, tendo como hipótese nula (H_0) a não existência de diferenças significativa ($valor - p > \alpha$)) entre o desempenho preditivo dos seis classificadores para todos os grupos de *PkData*. Os resultados aplicados ao grupo indicaram valor-p para a revocação de $0,0000368$, de $0,0000685$ para a precisão e do FAR de $0,008180$, assim os resultados indicam que a hipótese nula foi rejeitada ($valor - p < \alpha$) para todas as medidas de avaliação.

Para o teste de *Friedman* é gerado um *ranking* dos algoritmos de acordo com o seu desempenho preditivo. Além disso, somente é possível identificar se H_0 foi rejeitada ou aceita, não sendo possível identificar qual algoritmo obteve um melhor desempenho preditivo frente ao outro. Para este caso, onde H_0 foi rejeitada, é necessário aplicar o teste *post-hoc* de *Nemenyi*, que identifica quais e o quanto os algoritmos são diferentes entre si utilizando a Diferença Crítica (DC) (DEMŠAR, 2006), dada pela Equação 5:

$$DC = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \tag{5}$$

onde q_α representa todos os valores de intervalos críticos da tabela $t - Student$ divididos por $\sqrt{2}$, k a quantidade de algoritmos analisados e N a quantidade de bases de dados (DEMŠAR, 2006).

A DC trabalha para identificar um limiar na comparação par-a-par entre os algoritmos, ela basicamente agrupa os que estão dentro da faixa de DC separando os que estão fora dessa área. As Figuras 26, 27 e 28 apresentam os resultados para o teste de *Nemenyi* para a revocação, precisão e FAR, respectivamente; sendo necessário à interpretação das figuras salientar que a medida superior indica a DC base ($DC \approx 3,77$) para todas as medidas de avaliação. Para a análise de diferenças, o eixo x numerado de 1 a 6 indica o *ranking* dos algoritmos conforme o seu desempenho preditivo médio no teste de *Friedman*. Quanto mais a esquerda, mais superior é o algoritmo. As linhas azuis indicam maior diferença e as verdes indicam a segunda maior diferença, elas também ligam os algoritmos que obtiveram um valor $DC > 3,77$ e, consequentemente, com diferenças estatísticas significativas.

Na Figura 26 estão listados os resultados do teste para a revocação, os mesmos indicam que o algoritmo *LeveragingBag* foi superior ao *AccUpdatedEnsemble* com uma DC de $4,75$ e o algoritmo *OzaBagAdwin* também foi superior ao *AccUpdatedEnsemble*, porém com uma DC de $4,00$.

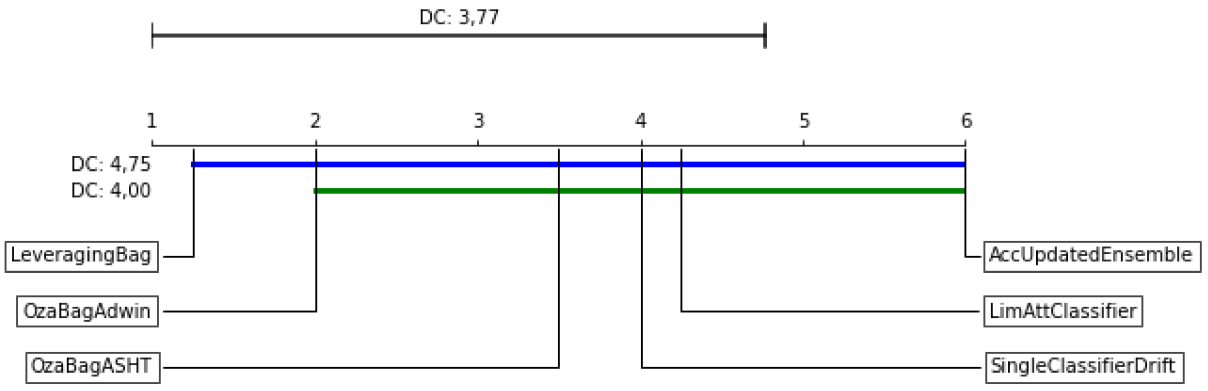


Figura 26 – Diagrama de Diferença Crítica (DC) demonstrando os resultados do teste *post-hoc* de *Nemenyi* para a medida de avaliação revocação para um $\alpha = 0,05$ (Fonte: O autor (2021)).

Os resultados obtidos para a precisão estão organizados na Figura 27 e, como pode ser visto, o algoritmo *LeveragingBag* foi superior ao *AccUpdatedEnsemble* com uma DC de $4,125$ e que o *SingleClassifierDrift* com uma DC de $3,875$.

Por fim, os resultados para o FAR estão organizados na Figura 28 e indicam que *LeveragingBag* novamente foi superior ao *AccuracyUpdatedEnsemble* e ao *SingleClassifierDrift* atingindo DC de $3,785$ para ambos.

Destacam-se, portanto, os algoritmos *LeveragingBag*, que obteve a primeira posição do *ranking* para todas as bases analisadas, o *LimAttClassifier*, que obteve duas segundas posições, o *OzaBagAdwin*, que obteve duas terceiras e uma segunda posição e, por fim, o *OzaBagASHT*, que obteve uma terceira posição.

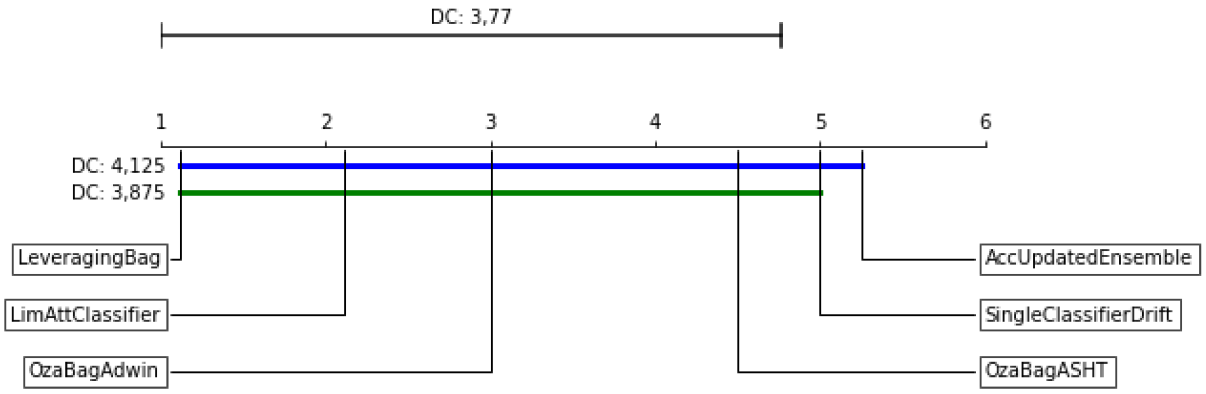


Figura 27 – Diagrama de Diferença Crítica (DC) demonstrando os resultados do teste *post-hoc* de *Nemenyi* para a medida de avaliação precisão para um $\alpha = 0,05$ (Fonte: O autor (2021)).

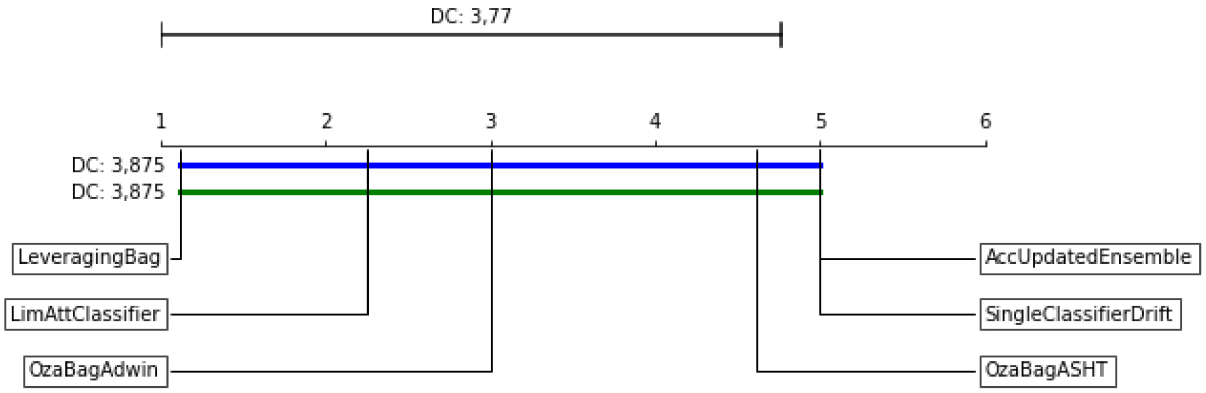


Figura 28 – Diagrama de Diferença Crítica (DC) demonstrando os resultados do teste *post-hoc* de *Nemenyi* para a medida de avaliação FAR para um $\alpha = 0,05$ (Fonte: O autor (2021)).

5.5.2 Atraso na Entrega dos Rótulos

Na intenção de analisar a hipótese $H2$, que visa identificar se o atraso dos rótulos verdadeiros das instâncias irão impactar no desempenho preditivo dos classificadores, foi utilizada novamente a ferramenta MOA. Para tanto, inicialmente foi empregado o avaliador chamado de sequenciamento preditivo com atraso, no qual a avaliação é feita a cada pacote após a apresentação do mesmo ao classificador. A diferença entre o avaliador de sequenciamento preditivo e o de sequenciamento preditivo com atraso é que este último permite que o usuário selecione o atraso na entrega dos rótulos, ou seja, após quantos *timestamps* será entregue o rótulo de uma instância.

Para esse experimento, foram selecionados os atrasos de 10.000, 50.000 e 100.000 instâncias. Essas janelas representam aproximadamente um atraso de $0,5$, 2 e 4 segundos respectivamente. Neste tipo de avaliação, há também a possibilidade de utilizar uma janela inicial de treinamento, onde um certo número de instâncias é utilizada para que o classificador faça um treinamento *offline*. Assim, foram utilizadas as 1.000 primeiras

instâncias para esse treinamento. Porém, como *PkData* inicialmente possuía somente instâncias da classe *Normal*, foi feita uma adaptação nas 1.000 primeiras instâncias incluindo uma quantidade de pacotes do tipo *Ataque* proporcionalmente ao total da base. Por exemplo, conforme a Tabela 5, a base *PkData* de terça-feira possui um total de 10.200.161 pacotes, onde $2,63\%$ deles são do tipo *Ataque*. Assim, dentre as 1.000 primeiras instâncias da base, foram inseridas 26 instâncias do tipo *Ataque*.

A base utilizada para o experimento foi *PkData* para os quatro dias da semana e os classificadores utilizados foram o *LeveragingBag*, *OzaBagAdwin*, *OzaBagASHT* e *LimAttClassifier*. A seleção dos classificadores justifica-se pelo desempenho preditivo que obtiveram no experimento para testar a hipótese $H1$. Diante disso, em todos os experimentos a seguir, as medidas de avaliação utilizadas foram a revocação, precisão e FAR.

Nas Tabelas 10, 11, 12 e 13 são comparados os resultados dos classificadores usando diferentes atrasos com o *Baseline*, que foi o resultado atingido pelos mesmos classificadores utilizando *PkData* sem atrasos, conforme apresentado na Seção 5.5.1. Para cada um dos classificadores, foram feitas três rodadas de testes, com atrasos de 10.000, 50.000 e 100.000 instâncias, que estão indicadas na coluna *Atraso*. Para cada uma dessas rodadas de classificação, os valores das medida revocação, precisão e FAR foram comparados com a versão do *Baseline*. As diferenças percentuais entre o resultado dos classificadores comparado ao *Baseline* é calculado na coluna $Dif.\%$. Um valor positivo nesta coluna indica que o resultado do classificador com atraso foi maior que o *Baseline* enquanto que, um valor negativo, indica do inverso. Para as medidas de revocação e precisão, o valor da coluna $Dif.\%$ deve ser interpretada como "quanto maior melhor", já para a medida FAR deverá ser entendida como "quanto menor melhor". Um destaque na cor verde foi feito para facilitar a identificação dos valores que se sobressaíram ao *Baseline*.

A expectativa, com relação aos resultados desse experimento, é que o atraso na entrega dos rótulos impacta no desempenho preditivo do algoritmo e assim a hipótese $H2$ é confirmada. Assim, poderá ser verificado que as altas taxas de precisão e revocação mostradas na literatura, podem não ser tão boas, quando são consideradas condições mais próximas da realidade dos IDSs.

Os resultados tabulados para a base de terça-feira estão disponíveis na Tabela 10, eles indicam que tanto a precisão quanto a revocação foram impactadas com o atraso na entrega do rótulo, sendo que, essa última, foi a mais impactada. Por outro lado, a medida FAR foi beneficiada pelo atraso em alguns classificadores.

Numa analise mais detalhada, é possível identificar que a revocação sofreu um grande impacto sendo que, no pior caso, o resultado chegou a decair em torno de 58% para o classificador *LimAttClassifier* e com atraso de 100.000 rótulos, indicando um alto número de falsos negativos na classificação. Já a precisão, na maioria dos classificadores, sofreu menos impactos significativos, exceto no classificador *OzaBagASHT*, onde no pior caso houve uma piora na ordem de $13,30\%$ para o atraso de 100.000 rótulos. Para os atra-

sos menores, 10.000 e 50.000, a precisão foi menos impactada. A redução do FAR do *OzaBagASHT* em todos os atrasos foi uma exceção se comparado aos demais classificadores. No geral, a diferença média absoluta do FAR para todas as medidas de atrasos com o *Baseline*, foi de $0,005$ pontos percentuais com um desvio padrão de $0,01$.

Tabela 10 – Resultado dos Experimentos com o sequenciamento preditivo com atraso para a Base de terça-feira comparando com o *Baseline* (Fonte: O autor (2021)).

Classificador	Atraso	Revocação		Precisão		FAR	
Baseline		**99,9550%**	**Dif. %**	**99,1560%**	**Dif. %**	**0,0230%**	**Dif. %**
LeveragingBag	10.000	89,7765%	-10,1831%	97,2288%	-1,9437%	0,0279%	21,3426%
	50.000	72,2705%	-27,6969%	94,8351%	-4,3577%	0,0429%	86,6492%
	100.000	66,1043%	-33,8659%	95,1600%	-4,0300%	0,0367%	59,4362%
Baseline		**99,8980%**	**Dif. %**	**99,0230%**	**Dif. %**	**0,0270%**	**Dif. %**
OzaBagAdwin	10.000	89,5406%	-10,3680%	97,1883%	-1,8528%	0,0283%	4,6442%
	50.000	71,5366%	-28,3904%	95,0545%	-4,0076%	0,0406%	50,3484%
	100.000	64,7522%	-35,1817%	94,6624%	-4,4036%	0,0398%	47,4889%
Baseline		**99,2000%**	**Dif. %**	**94,2730%**	**Dif. %**	**0,1630%**	**Dif. %**
OzaBagASHT	10.000	85,5322%	-13,7780%	86,5837%	-8,1564%	0,1446%	-11,3181%
	50.000	64,5748%	-34,9044%	82,3748%	-12,6210%	0,1507%	-7,5490%
	100.000	58,2551%	-41,2751%	81,7366%	-13,2980%	0,1420%	-12,9017%
Baseline		**96,9200%**	**Dif. %**	**98,6760%**	**Dif. %**	**0,0350%**	**Dif. %**
LimAttClassifier	10.000	60,8655%	-37,2003%	88,1537%	-10,6635%	0,2213%	532,2857%
	50.000	43,6510%	-54,9618%	75,4663%	-23,5211%	0,3839%	996,9714%
	100.000	40,3961%	-58,3201%	74,0974%	-24,9084%	0,3821%	991,6286%

É essencial entender como os algoritmos de fluxos contínuos de dados se comportam ao longo do fluxo, e não somente analisar as medidas de avaliação consolidadas. Para suprir essa demanda, foram gerados gráficos que contemplam todas as medidas de avaliação em janelas de 100.000 instâncias, para uma avaliação mais criteriosa.

Os gráficos possuem um atraso de 10.000, 50.000 ou 100.000 pacotes. Em cada gráfico, o eixo X representa as janelas de avaliação, enquanto o eixo Y representa o percentual de cada uma das medidas utilizadas no presente trabalho. É necessário destacar dois pontos na análise do gráfico: o primeiro que, quando não foi possível executar o cálculo da medida de avaliação devido a inexistência de pacotes do tipo *Ataque*, a barra não será exibida; o segundo indica que, quando a medida avaliada é realmente zero, mesmo com pacotes do tipo *Ataque*, foi inserido um valor negativo na barra para melhor visualização do erro. Os gráficos foram gerados para todos os cenários e estão disponíveis no sítio eletrônico https://gilbertoolimpio.github.io/anexos.

Para o cenário dos IDSs a medida de revocação é essencial, pois indica a quantidade de pacotes do tipo ataque foram classificadas corretamente. A seguir, é apresentada a análise do gráfico disponível na Figura 29, que mostra o comportamento da revocação do *Baseline* (Figura 29a), para os atrasos de 10.000 (Figura 29b) e 100.000 instâncias (Figura

29c).

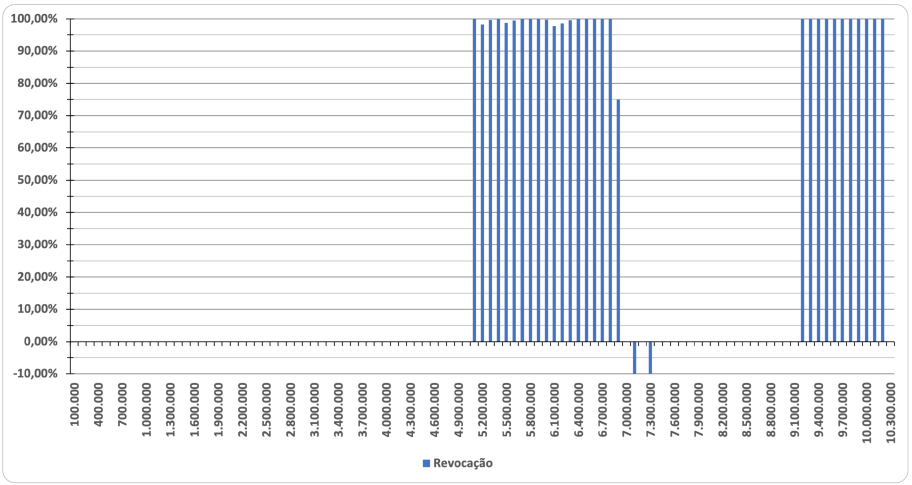
A distribuição dos ataques apresentado na Figura 17 indica que, até a janela de 5.000.000 pacotes, não houve instâncias do tipo *Ataque*. Já na janela de 5.100.000 pacotes, após a chegada dos primeiros ataques, identificou-se que a revocação do *Baseline* (Figura 29a) atingiu $99,96\%$ enquanto que, com o atraso em 10.000 instâncias atingiu $59,19\%$. Na análise com o atraso de 100.000, mostrou-se que o classificador *LeveragingBag* não identificou nenhuma instância de ataque, ou seja, 0% de revocação nesta janela. Já na janela de 6.100.000 pacotes com atraso de 100.000 instâncias, a revocação foi de $97,73\%$, obtendo o mesmo percentual do *baseline*. Isto mostra que, para este classificador, o atraso impacta negativamente no início da identificação dos ataques, como era esperado, pois ele necessita de um tempo maior para reconhecer as instâncias de contendo os ataques.

Os ataques voltam a ocorrer na janela de 9.200.000 pacotes e, como esperado, o classificador responde de forma satisfatória, pois o padrão de comportamento dos ataques já era conhecido. É importante notar que, o classificador com 100.000 instâncias de atraso, inicialmente não reconheceu 18.747 instâncias do tipo *Ataque* (revocação = 0%) na janela de 9.200.000 instâncias mas, na janela seguinte, obteve uma revocação de $95,83\%$. Isso mostra que, assim que o classificador obteve o rótulo das instâncias, ele aprendeu sobre o ataque e, consequentemente, a revocação melhorou. Porém, nas janelas subsequentes, o atraso causa um efeito não desejado, diminuindo a revocação sucessivamente, chegando a atingir o valor de $40,62\%$.

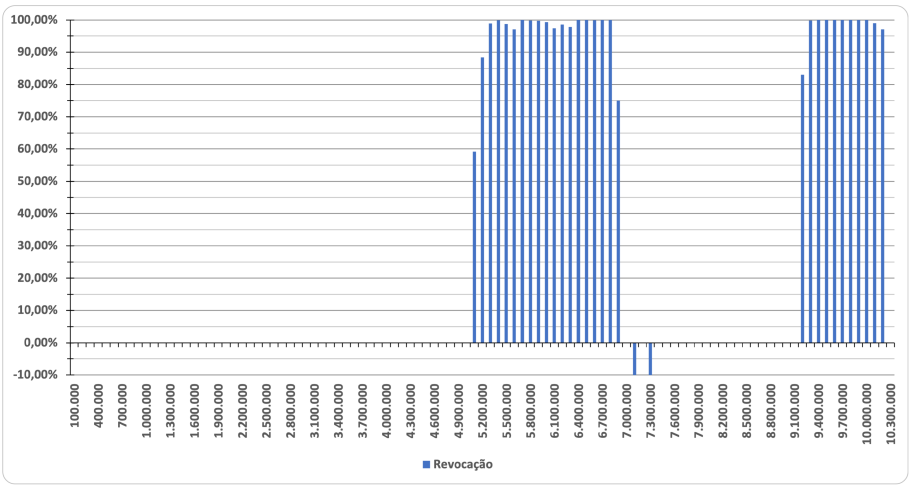
A Tabela 11 resume os resultados obtidos com *PkData* de quarta-feira. Para esse conjunto de dados, a precisão, revocação e FAR sofreram impacto para todos os classificadores e todos os atrasos. A única exceção ocorreu para o classificador *LeveragingBag* com o atraso de 50.000 instâncias, que obteve uma ligeira melhora no FAR. Pode-se perceber que quanto maior o atraso na entrega, maior a queda no desempenho preditivo dos classificadores.

Na Tabela 12 estão organizados os resultados para *PkData* de quinta-feira. As medidas de avaliação revocação e precisão, sofreram um impacto negativo significante quando houve atraso na entrega dos rótulos. O classificador *LimAttClassifier* foi um dos que teve maior piora no desempenho preditivo com o atraso dos rótulos, considerando a medida revocação e FAR. A base de quinta-feira foi a que sofreu maior impacto no desempenho da predição quando considerando cenários de atraso na entrega dos rótulos.

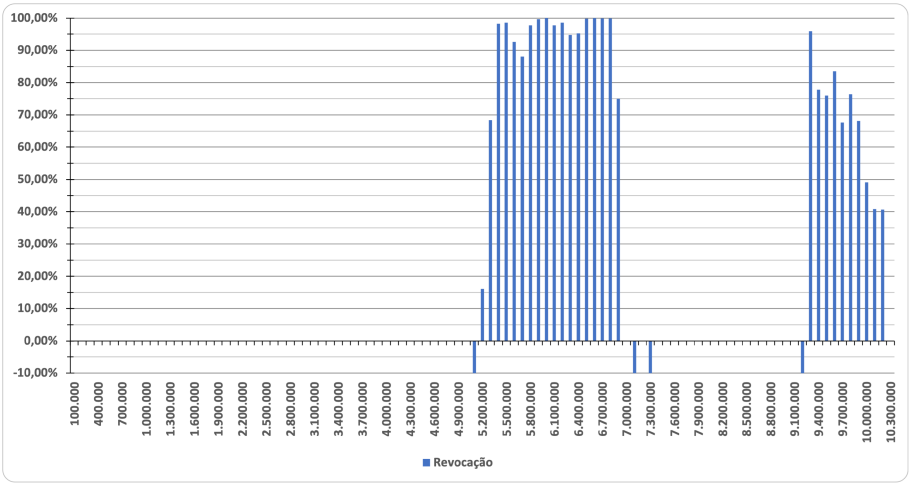
A Tabela 13 refere-se aos resultados dos classificadores para *PkData* de sexta-feira. Novamente, vemos resultados similares aos encontrados nas bases anteriores. A revocação é a medida mais afetada pelo atraso na entrega dos rótulos obtevre piora em todos os classificadores. Considerando a medida de avaliação FAR, na maioria dos classificadores, os resultados foram superiores ao *Baseline*, exceto para o classificador *LeveragingBag* com atraso de 100.000 instâncias e para o *LimAttClassifier* em todos os tamanhos de atraso. Na média, a melhora foi em torno de $2,48$ pontos percentuais com um desvio padrão de



(a) *Baseline*



(b) Atraso de 10.000 instâncias



(c) Atraso de 100.000 instâncias

Figura 29 – Resultados da revocação comparando o Baseline (a), com atrasos de 10.000 instâncias (b) e 100.000 instâncias (c) por janelas de 100.000 pacotes - *PkData* de terça-feira com o classificador *LeveragingBag* (Fonte: O autor (2021)).

2, 37.

A principal conclusão deste experimento é que todos os algoritmos testados sofreram

Tabela 11 – Resultado dos Experimentos com o sequenciamento preditivo com atraso para a Base de quarta-feira comparado com o *Baseline* (Fonte: O autor (2021)).

Classificador	Atraso	Revocação		Precisão		FAR	
Baseline		**99,9510%**	**Dif. %**	**98,0550%**	**Dif. %**	**0,4280%**	**Dif. %**
LeveragingBag	10.000	97,2536%	-2,6987%	97,1165%	-0,9572%	0,4786%	11,8337%
	50.000	91,9884%	-7,9665%	97,3340%	-0,7353%	0,4176%	-2,4184%
	100.000	88,5744%	-11,3822%	96,9240%	-1,1534%	0,4660%	8,8670%
Baseline		**99,9850%**	**Dif. %**	**97,9350%**	**Dif. %**	**0,4550%**	**Dif. %**
OzaBagAdwin	10.000	97,9635%	-2,0218%	97,0099%	-0,9446%	0,5005%	10,0035%
	50.000	93,6659%	-6,3201%	96,7726%	-1,1869%	0,5178%	13,8005%
	100.000	90,8744%	-9,1120%	96,4982%	-1,4671%	0,5466%	20,1396%
Baseline		**99,8060%**	**Dif. %**	**97,0990%**	**Dif. %**	**0,6430%**	**Dif. %**
OzaBagASHT	10.000	91,1934%	-8,6294%	91,6438%	-5,6182%	1,3783%	114,3568%
	50.000	89,9009%	-9,9243%	91,5533%	-5,7114%	1,3748%	113,8172%
	100.000	86,9410%	-12,8900%	90,2691%	-7,0339%	1,5535%	141,6040%
Baseline		**99,5100%**	**Dif. %**	**99,6690%**	**Dif. %**	**0,1900%**	**Dif. %**
LimAttClassifier	10.000	86,1199%	-13,4561%	94,6907%	-4,9948%	1,0416%	448,2316%
	50.000	83,9430%	-15,6436%	93,3489%	-6,3411%	1,2902%	579,0579%
	100.000	80,0794%	-19,5263%	91,7229%	-7,9725%	1,5589%	720,4632%

Tabela 12 – Resultado dos Experimentos com o sequenciamento preditivo com atraso para a Base de quinta-feira comparado com o *Baseline* (Fonte: O autor (2021)).

Classificador	Atraso	Revocação		Precisão		FAR	
Baseline		**99,4240%**	**Dif. %**	**99,6000%**	**Dif. %**	**0,0050%**	**Dif. %**
LeveragingBag	10.000	65,7268%	-33,8924%	83,4323%	-16,2327%	0,0690%	1.280,3464%
	50.000	44,1237%	-55,6207%	63,9096%	-35,8337%	0,1318%	2.535,1991%
	10.0000	36,3889%	-63,4003%	59,7709%	-39,9891%	0,1295%	2.490,2170%
Baseline		**99,0750%**	**Dif. %**	**99,3850%**	**Dif. %**	**0,0070%**	**Dif. %**
OzaBagAdwin	10.000	63,9924%	-35,4101%	82,9396%	-16,5472%	0,0696%	894,3616%
	50.000	40,5449%	-59,0766%	61,4242%	-38,1957%	0,1346%	1.823,5349%
	10.0000	32,1306%	-67,5695%	57,1903%	-42,4558%	0,1272%	1.716,8827%
Baseline		**98,3150%**	**Dif. %**	**98,7240%**	**Dif. %**	**0,0150%**	**Dif. %**
OzaBagASHT	10.000	58,4780%	-40,5197%	79,8891%	-19,0784%	0,0778%	418,9590%
	50.000	33,5988%	-65,8254%	58,5018%	-40,7421%	0,1260%	740,1914%
	10.0000	22,1013%	-77,5199%	47,8805%	-51,5007%	0,1272%	748,1146%
Baseline		**93,9810%**	**Dif. %**	**99,6020%**	**Dif. %**	**0,0040%**	**Dif. %**
LimAttClassifier	10.000	60,4709%	-35,6563%	82,6100%	-17,0599%	0,1496%	3.640,2500%
	50.000	39,6593%	-57,8007%	68,7325%	-30,9928%	0,2120%	5.201,0000%
	10.0000	24,3250%	-74,1171%	56,1157%	-43,6600%	0,2236%	5.489,5000%

perda de desempenho preditivo quando há um atraso na entrega dos rótulos. Em especial, a medida que mais sofreu impacto foi a revocação, ou seja, houve uma diminuição do número de instâncias corretamente classificadas como ataque. Isso pode ter ocorrido porque o modelo não conseguiu se adequar às mudanças no comportamento dos dados, já que essa adequação só ocorre na presença de instâncias rotuladas. Por outro lado, a medida FAR, em vários cenários, foi beneficiada, indicando que houve uma diminuição do número de instâncias da classe normal incorretamente marcadas como ataque. No entanto, para o problema de detecção de intrusão, um aumento na taxa de revocação é um ponto importante a ser pensando já que ele expressa um tipo de erro que deve-se ser evitado, ou seja a não identificação de um ataque.

Tabela 13 – Resultado dos Experimentos com o sequenciamento preditivo com atraso para a Base de sexta-feira comparado com o *Baseline* (Fonte: O autor (2021)).

Classificador	Atraso	Revocação		Precisão		FAR	
Baseline		**96,6290%**	**Dif. %**	**92,5330%**	**Dif. %**	**4,0460%**	**Dif. %**
LeveragingBag	10.000	89,3674%	-7,5149%	90,4734%	-2,2258%	2,4104%	-40,4262%
	50.000	80,3601%	-16,8364%	84,8223%	-8,3329%	3,6832%	-8,9672%
	100.000	69,4233%	-28,1548%	77,6152%	-16,1216%	5,1286%	26,7569%
Baseline		**93,3920%**	**Dif. %**	**89,7350%**	**Dif. %**	**5,5430%**	**Dif. %**
OzaBagAdwin	10.000	88,6155%	-5,1145%	89,5743%	-0,1791%	2,6419%	-52,3378%
	50.000	79,6269%	-14,7390%	85,0308%	-5,2423%	3,5906%	-35,2225%
	100.000	68,6118%	-26,5335%	79,0866%	-11,8665%	4,6474%	-16,1581%
Baseline		**92,0010%**	**Dif. %**	**91,3880%**	**Dif. %**	**4,4980%**	**Dif. %**
OzaBagASHT	10.000	88,3468%	-3,9719%	92,5108%	1,2287%	1,8320%	-59,2715%
	50.000	79,0328%	-14,0957%	88,0325%	-3,6717%	2,7520%	-38,8165%
	100.000	68,1729%	-25,8999%	82,1553%	-10,1027%	3,7929%	-15,6761%
Baseline		**95,4030%**	**Dif. %**	**92,4080%**	**Dif. %**	**4,0670%**	**Dif. %**
LimAttClassifier	10.000	83,0001%	-13,0005%	84,1591%	-8,9266%	8,1090%	99,3841%
	50.000	77,2766%	-18,9998%	79,7746%	-13,6713%	10,1692%	150,0420%
	100.000	70,5608%	-26,0393%	75,2709%	-18,5450%	12,0324%	195,8542%

Por fim, considerando que vários trabalhos da literatura avaliam a qualidade dos IDSs supondo a entrega imediata dos rótulos, pode-se perceber que muito ainda precisa ser feito na tentativa de colocar esses sistemas em funcionamento no mundo real. Ainda, foi possível analisar o impacto gerado pelo atraso dos rótulos para os classificadores. Esse impacto consideravelmente negativo é um ponto de atenção importante para trabalhos futuros.

5.5.3 Análise da Aplicação da Estratégia de Aprendizado Ativo

Este experimento verificou a hipótese $H3$, que analisou se o método de aprendizado ativo permite manter o desempenho preditivo dos classificadores utilizando, para isso, um menor número de instâncias rotuladas para atualizá-los. O aprendizado ativo escolhe seletivamente as instâncias a serem rotuladas ao invés de solicitar o rótulo verdadeiro de todas as instâncias.

Novamente, o MOA foi utilizado para os experimentos. O método de aprendizado ativo utilizado foi o aleatório. Embora esse método seja ingênuo, devido a natureza aleatória da rotulagem, ele permite de uma forma rápida e simples avaliar como esse tipo de abordagem pode beneficiar o problema em questão. Nesse contexto, as instâncias que receberão os rótulos são selecionadas aleatoriamente e o percentual previsto de rótulos é selecionado previamente. Para esse experimento, os percentuais de rotulagem selecionados foram os seguintes: 1%, 3%, 5%, 7%, 10%, 30%, 50% e 70% do total de instâncias. A base utilizada no experimento foi *PkData* de quinta-feira e sexta-feira. A seleção dessas bases de dados justifica-se pela diversidade de ataques que constam nelas, além de que a base de quinta-feira ser a mais desbalanceada, enquanto que a de sexta-feira é a mais balanceada. Já o classificador utilizado no experimento foi o *OzaBagASHT* com os atributos padrões

que constam no MOA. A seleção deste algoritmo ocorreu devido a uma comparação entre algoritmos onde este obteve o melhor desempenho preditivo para o método de aprendizado ativo.

Os experimentos consistiram em executar a classificação com a base selecionada para todas as quantidades de percentuais de rotulagem citadas. As Tabela 14 e 15 mostram os resultados obtidos nos experimentos. As tabelas comparam os resultados do *Baseline* com aqueles obtidos com a técnica de aprendizado ativo. As colunas da tabela representam o seguinte: Tipo indica qual é o tipo de escolha de instância a serem rotuladas (*Baseline* ou usando aprendizado ativo); Percentual de Rotulagem indica o percentual fixo de rótulos utilizados pelo aprendizado ativo para rotular as instâncias; as colunas seguintes são as medidas de avaliação que serão utilizadas para a comparação dos resultados com o *Baseline*: revocação, precisão e FAR.

Os resultados na Tabela 14 referem-se a *PkData* de quinta-feira. Quando se analisa a revocação, é importante notar que, aparentemente, o sobre-ajuste do modelo ocorreu quando houve a entrega acima de 70% dos rótulos. Este comportamento observado na revocação mostra uma variação de $99,251\%$ a $99,162\%$ para a entrega de 30% e 50%, respectivamente, inclusive superando o *Baseline*.

Tabela 14 – Comparação entre os resultados do Aprendizado Ativo Aleatório com a estratégia de rotulagem aleatória com percentual fixo de rótulos e o *Baseline* para a base *PkData* de quinta-feira (Fonte: O autor (2021)).

Base *PkData* - quinta-feira Classificador *OzaBagASHT*				
Estratégia	**Percentual de Rotulagem**	**Revocação**	**Precisão**	**FAR**
Baseline	-	98,315%	98,724%	0,015%
Aprendizado Ativo Aleatório	1%	95,408%	93,207%	0,081%
	3%	97,285%	91,830%	0,101%
	5%	96,686%	93,805%	0,075%
	7%	97,195%	94,456%	0,067%
	10%	98,858%	94,210%	0,071%
	30%	**99,251%**	97,682%	0,027%
	50%	**99,162%**	98,465%	0,018%
	70%	98,768%	98,550%	0,017%

Já a Tabela 15 refere-se aos resultados obtidos pela base *PkData* de sexta-feira. Para essa base, o classificador novamente teve uma redução na revocação quando entregou ao menos 70% dos rótulos, apesar da diferença ser menor que na base de quinta-feira, ocorreu uma redução do desempenho preditivo dessa medida. Paralelamente, o FAR foi superior ao *Baseline*, para 30% e 70% de instâncias rotuladas.

Diante dos dados apresentados pode-se notar que, mesmo com apenas 10% dos rótulos,

Tabela 15 – Comparação entre os resultados do Aprendizado Ativo Aleatório com a estratégia de rotulagem aleatória com percentual fixo de rótulos e o *Baseline* para a base *PkData* de sexta-feira (Fonte: O autor (2021)).

Base *PkData* - sexta-feira Classificador *OzaBagASHT*				
Estratégia	**Percentual de Rotulagem**	**Revocação**	**Precisão**	**FAR**
Baseline	-	92,001%	91,388%	4,498%
Aprendizado Ativo Aleatório	1%	89,211%	90,749%	4,718%
	3%	89,768%	91,168%	4,511%
	5%	90,397%	91,171%	4,541%
	7%	90,718%	91,139%	4,575%
	10%	91,077%	91,129%	4,599%
	30%	91,523%	91,359%	**4,490%**
	50%	91,539%	91,251%	4,553%
	70%	91,518%	91,357%	**4,492%**

o resultado das quatro medidas de avaliação foi bastante satisfatório. Tanto a revocação, precisão e FAR obtiveram resultados semelhantes quando comparados ao *Baseline*. A queda no desempenho preditivo foi muito pequena ao diminuir o número de instâncias rotuladas, principalmente quanto ao FAR. Considerando as diferenças absolutas entre os experimentos e o *Baseline* a revocação teve uma piora média de $1,28$ enquanto que, para a precisão, a diferença média foi de apenas $0,22$ pontos percentuais.

Aparentemente, a utilização de uma quantidade mínima de rótulos, manteve o desempenho preditivo do classificador bem próximo do *Baseline*. Dentre as possíveis causas, está o fato de que os classificadores podem ter sobre-ajustado o modelo. De fato, se for possível selecionar somente as instâncias que auxiliem na generalização do modelo, o resultado pode ter um desempenho preditivo satisfatório com bem menos instâncias rotuladas.

5.5.3.1 Aprendizado Ativo com Atraso na Entrega dos Rótulos

Um experimento secundário foi executado com a base *PkData* de sexta-feira. O objetivo é analisar o impacto da classificação do algoritmo *OzaBagASHT* quando forem combinadas, a estratégia de aprendizado ativo com a de atraso dos rótulos. Para sua execução no MOA, foi necessário que as instâncias selecionadas aleatoriamente pelo aprendizado ativo pudessem sofrer um atraso conforme um parâmetro selecionado.

Para o experimento, os atrasos selecionados foram de 10.000, 50.000 e 100.000 rótulos, assim se manteve a capacidade de comparação com os experimentos da Seção 5.5.2. Seguindo esse critério, o percentual de rotulagem das instâncias foi de 1%, 3%, 5%, 7%, 10%, 30%, 50% e 70% do total da base.

A seguir, são apresentados, na Tabela 16, os resultados obtidos pela associação dos parâmetros de atraso e percentual de rotulagem. O conteúdo das colunas possuem o seguinte significado:

- **classificador** descreve o tipo de classificador utilizado, onde *Baseline* é o resultado do experimento executado na Seção 5.5.1;
- **sequenciamento preditivo** com atraso mostra a estratégia utilizada nos experimentos da Seção 5.5.2 informando o atraso selecionado para rotular as instâncias;
- **percentual de rotulagem** indica o percentual de instâncias que serão rotuladas;
- **atraso** indica o atraso da entrega do rótulo ao classificador; e
- **revocação, precisão e FAR** indica os valores do desempenho preditivo das respectivas medidas para a comparação dos experimentos.

Para a interpretação dos resultados é necessário destacar que as linhas em cinza indicam os valores obtidos na Seção 2.3.3, exceto o *Baseline* que foi obtido na Seção 5.5.1.

Na intenção de analisar o comportamento da revocação e precisão, foi gerada a Figura 30, para uma melhor compreensão dos dados. As medidas plotadas consideram o cenário de atraso de 100.000 instâncias e diferentes percentuais de instâncias rotuladas.

Via de regra, ao se combinar o aprendizado ativo e o atraso, a precisão, a revocação e o FAR obtiveram um resultado inferior ao *Baseline*. Ainda pode-se analisar que, apesar do aprendizado ativo prover um resultado aceitável com uma quantidade mínima de rótulos, quando se insere atraso na entrega dos mesmos o desempenho preditivo decai consideravelmente, confirmando que o atraso dos rótulos necessita de uma atenção especial em trabalhos que utilizam algoritmos de aprendizado de máquina para a construção de IDSs. É importante destacar que, com apenas 1% das instâncias rotuladas e com atraso de 10.000 instâncias, o desempenho preditivo obtido é aceitável para os IDSs.

5.5.3.2 Detalhamento de Comportamento do Cenário Experimental

Diante dos experimentos analisados, principalmente aqueles que utilizaram o aprendizado ativo, é importante demonstrar o comportamento do classificador em um cenário experimental analisando não somente uma medida de avaliação única, mas sim o que ocorre ao longo do fluxo. Para exemplificar isso, a Figura 31 apresenta o comportamento da medida de revocação do classificador *OzaBagASHT* ao longo de todo o fluxo da base *PkData* de sexta-feira. Na Figura 31a, é apresentado o *Baseline*, onde todas as instâncias foram rotuladas e entregues ao classificador. Já a Figura 31b consiste no comportamento de atraso de 100.000 instâncias. Na Figura 31c, pode ser observado o comportamento do classificador quanto utilizada a técnica de aprendizado ativo e, por fim, a Figura 31d

Tabela 16 – Resultados da estratégia de aprendizado ativo com rotulagem aleatória acrescida de atraso na entrega dos rótulos (Fonte: O autor (2021)).

Estratégia	Percentual de Rotulagem	Atraso	Revocação	Precisão	FAR
Baseline	**-**	**-**	**92,0010%**	**91,3880%**	**4,4980%**
Prequential Delayed	-	10.000	88,3468%	92,5108%	1,8320%
		50.000	79,0328%	88,0325%	2,7520%
		100.000	68,1729%	82,1553%	3,7929%
Aprendizado Ativo Aleatório	**1%**	**-**	**89,2110%**	**90,7490%**	**4,7180%**
	1%	10.000	82,6317%	86,6561%	6,6045%
		50.000	77,7867%	82,8122%	8,3799%
		100.000	71,4906%	79,2571%	9,7116%
	3%	**-**	**89,7680%**	**91,1680%**	**4,5110%**
	3%	10.000	81,1459%	87,3792%	6,0835%
		50.000	76,6035%	83,9110%	7,6237%
		100.000	70,4469%	79,8353%	9,2356%
	5%	**-**	**90,3970%**	**91,1710%**	**4,5410%**
	5%	10.000	82,0979%	87,7782%	5,9332%
		50.000	77,5903%	83,4843%	7,9672%
		100.000	71,3789%	79,2523%	9,6992%
	7%	**-**	**90,7180%**	**91,1390%**	**4,5750%**
	7%	10.000	81,7574%	86,7509%	6,4811%
		50.000	77,1441%	83,2094%	8,0799%
		100.000	71,0983%	79,6180%	9,4472%
	10%	**-**	**91,0800%**	**91,1300%**	**4,6000%**
	10%	10.000	81,8600%	87,5194%	6,0591%
		50.000	77,2332%	83,1484%	8,1246%
		100.000	71,2179%	79,4815%	9,5428%
	30%	**-**	**91,5230%**	**91,3590%**	**4,4900%**
	30%	10.000	81,2204%	85,4227%	7,1941%
		50.000	76,6532%	81,9432%	8,7673%
		100.000	70,5426%	78,2138%	10,1990%
	50%	**-**	**91,5390%**	**91,2510%**	**4,5530%**
	50%	10.000	80,6090%	88,1045%	5,6491%
		50.000	76,0120%	83,4163%	7,8437%
		100.000	69,8046%	79,2502%	9,4865%
	70%	**-**	**91,5180%**	**91,3570%**	**4,4920%**
	70%	10.000	80,2400%	85,0629%	7,3135%
		50.000	75,5097%	81,6858%	8,7872%
		100.000	69,2582%	77,7635%	10,2795%

reflete o comportamento quando se adiciona o atraso de 100.000 instâncias ao aprendizado ativo.

A base de *PkData* de sexta-feira foi analisada detalhadamente na Seção 5.1.1, nela pode-se ver duas ondas de ataques. A primeira inicia-se na janela de 1.600.000 pacotes e

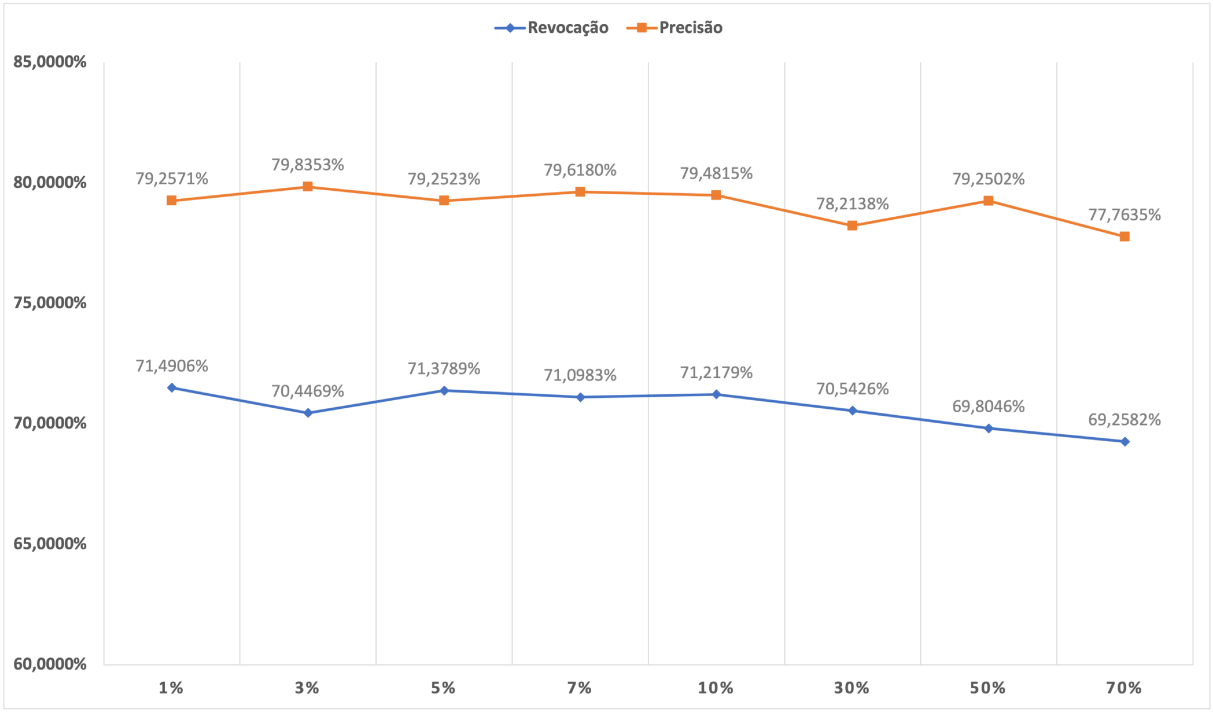


Figura 30 – Comportamento das medidas de revocação e precisão por percentual de rotulagem, com atrasos de 100.000 instâncias (Fonte: O autor (2021)).

indo até a janela de 2.000.000 de pacotes que indica ser um ataque *botnet* e uma segunda iniciando na janela de 2.400.000 pacotes e indo até 3.800.000 pacotes que contém ataques do tipo *DDoS* e escaneamento de portas.

Como pode ser visto na Figura 31a, na primeira onde de ataque, o classificador obteve uma taxa de perda abaixo dos 5% e logo após passou a classificar corretamente todos os pacotes de ataque. Quando o atraso de 100.000 instâncias foi aplicado (Figura 31b), o classificador não conseguiu identificar os pacotes maliciosos, atingindo 0% (valor negativo no gráfico) indicando que não foi possível identificar nenhum pacote malicioso. Já na janela seguinte o revocação ainda foi baixa, atingindo um valor acima de 5%.

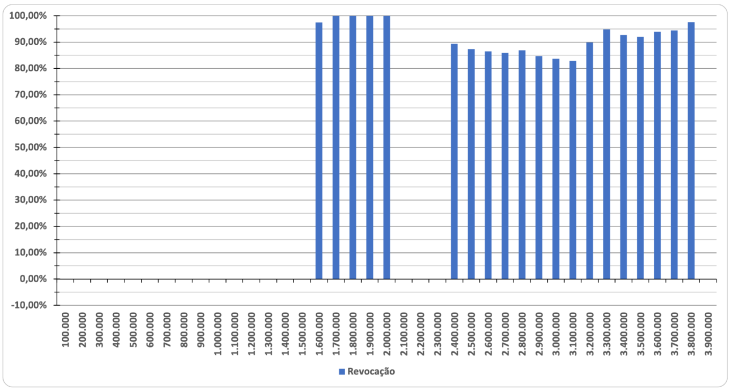
É importante notar que, quando se aplica a estratégia do aprendizado ativo (Figura 31c), a revocação atinge um pouco menos de 80% na primeira onda de ataque. Apesar do percentual razoável, é importante analisar que somente 10% das instâncias foram rotuladas. E mesmo com este percentual, o classificador respondeu de forma significativa ao conseguir classificar corretamente quase 80% das instâncias maliciosas na primeira onda e quase 90% na segunda onda ficando bem próximo do *Baseline*.

Por último, quando a estratégia de aprendizado ativo com atrasos é inserida (Figura 31d), houve um decaimento na detecção da primeira onda de ataques se comparado exclusivamente com o cenário do *Baseline*, além disso, na segunda onda de ataques, inicialmente o desempenho preditivo foi melhor, comparado ao cenário de atrasos.

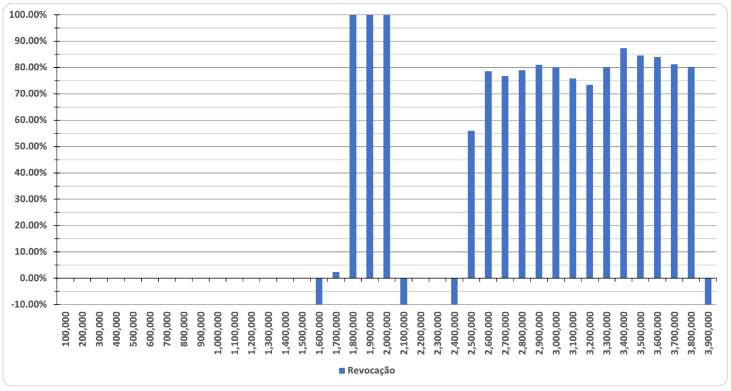
5.6 Limitações da pesquisa

O presente trabalho apresentou limitações na construção de um IDS adaptável e de alto desempenho. De fato, a proposta trouxe limitações quanto a diversidade dos ataques e protocolos de redes analisados, restringindo-se aos protocolos TCP e UDP. O método apresentado também não foi posto a prova em um ambiente real, onde a dinamicidade esperada poderia trazer resultados divergentes dos apresentados.

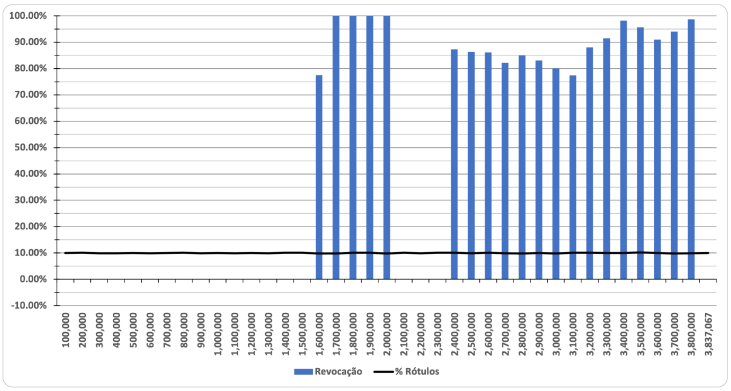
Outro fato limitante foram as janelas selecionadas para medir o atraso dos pacotes, é importante salientar que os atrasos consistiram em um espaço de tempo de no máximo 4 segundos e, apesar de ser uma janela temporal mínima, impactou consideravelmente o desempenho preditivo. Por fim, o método de aprendizado ativo utilizado restringiu-se somente a uma seleção aleatória dos rótulos para rotulagem e, em geral, existem diversos outros métodos que utilizam técnicas menos ingênuas mas que não foram propostas no escopo deste trabalho.



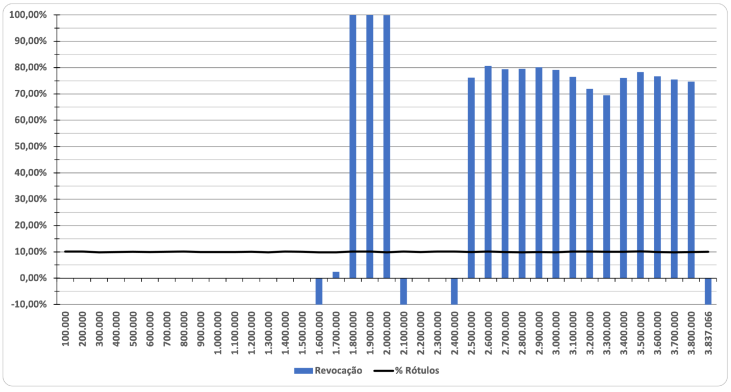
(a) *Baseline*



(b) Atraso de 100.000 instâncias



(c) Aprendizado ativo



(d) Aprendizado ativo com atraso

Figura 31 – Resultados da revocação comparando o *Baseline* (a), com atrasos de 100.000 instâncias (b), utilizando o aprendizado ativo (c) e com o aprendizado ativo mais atraso. As janelas de 100.000 pacotes - *PkData* de sexta-feira com o classificador *OzaBagASHT* (Fonte: O autor (2021)).

Capítulo 6

Conclusão

Diante das inúmeras abordagens disponíveis na literatura para a detecção de intrusão em redes de computadores, este trabalho se propôs preliminarmente a comparar a classificação de pacotes individuais e do fluxo na intenção de analisar se ambos produziriam resultados semelhantes. Para isso o desempenho preditivo de seis classificadores de fluxo contínuo de dados foram analisados usando diferentes medidas de avaliação. Além disso, com a finalidade de analisar o comportamento dos classificadores de fluxo contínuo de dados em um ambiente próximo da realidade, os rótulos das instâncias foram atrasados para a atualização dos modelos de classificação utilizados, a fim de avaliar o impacto no resultado da classificação. Por fim, a técnica de aprendizado ativo foi empregada na intenção de verificar se é possível rotular apenas um subconjunto de instâncias e manter o desempenho preditivo do classificador.

As principais conclusões obtidas foram:

- os diferentes algoritmos de classificação de fluxos contínuo de dados utilizados neste trabalho, apresentaram um desempenho preditivo semelhante, tanto para os pacotes quanto para o fluxo. Para a constatação da igualdade de desempenho preditivo, foram utilizados testes estatísticos e a hipótese nula, que indica a igualdade entre os resultados não foi rejeitada. Além disso, um teste de significância desse primeiro experimento concluiu que as diferenças estatísticas entre os algoritmos utilizados não são significativas, permitindo concluir que nenhum algoritmos se sobressaiu sobre os demais;

- a entrega atrasada dos rótulos para atualização do modelo de decisão, conforme esperado, causou uma queda no desempenho preditivo dos classificadores testados. Isso mostra que são necessárias importantes discussões sobre algoritmos de classificação de fluxos contínuos de dados para adaptá-los a ambientes reais dos IDSs. Um desses cenários é que, de fato, os rótulos das instâncias não são entregues imediatamente aos modelos;

- algoritmos simples de escolha aleatória das instâncias a serem rotuladas pelo especialista conseguiram manter o desempenho preditivo dos modelos de decisão requisitando apenas 10% dos rótulos. Este fato indica que o uso da técnica de aprendizado ativo mostrou-se como uma opção interessante para o problema e que estratégias refinadas podem trazer resultados ainda melhores;

- O impacto da entrega de somente uma parte dos rótulos é bem menor do que atrasar os mesmos. Por exemplo, se em cada janela de dados somente 10% de instâncias fossem rotuladas, medidas de avaliação como a revocação se deteriorariam em torno de 1%, o que seria suficiente para manter as medidas de desempenho preditivo adequadas para IDSs. Mas, caso ocorresse um atraso na entrega de instâncias ao classificador na ordem de 100.000 instâncias, haveria uma redução relativa de 25% na revocação, o que seria uma redução substancial, e;

6.1 Principais Contribuições

As principais contribuições deste trabalho são listadas a seguir:

1. Análise experimental do impacto no desempenho predito dos modelos quando ocorrem atrasos em diferentes janelas de tempo, esse impacto deve ser considerado na construção de IDSs adaptáveis e de alto desempenho;

2. Análise experimental das diferenças entre o uso de dados dos pacotes individuais e do fluxo na detecção de intrusão, estabelecendo contrapontos aos trabalhos recentes da literatura que tem se focado apenas no uso dos dados dos fluxos;

3. Análise experimental do comportamento de classificadores para fluxos contínuos de dados em diferentes cenários onde os rótulos não são entregues imediatamente ao classificador, aproximando do que poderá ocorrer em um ambiente real dos IDSs; e

4. Análise experimental sobre o uso de aprendizado ativo, com diferentes condições de disponibilidade de rótulos, demonstrando que essa técnica poderia ser utilizado como uma das maneiras de se lidar com a escassez de rótulos em cenários reais.

6.2 Trabalhos Futuros

Inicialmente, seria importante analisar outras bases de dados contendo uma variação na quantidade de tipos de ataques, além de estender a quantidade de algoritmos de fluxos contínuos de dados. Com isso seria possível analisar o comportamento desses novos classificadores para diferentes tipos de ataques e, para quais deles, os algoritmos usados apresentam os melhores resultados. A partir disso, adicionar outras medidas de

avaliação para analisar a concordância entre os modelos de classificação baseados em pacotes individuais ou em fluxos. Também é aconselhável que a complementação com novos algoritmos de classificação, contenha outras abordagens além das árvores de decisão, onde uma sugestão é experimentar o uso de algoritmos de detecção de novidades, que permitam identificar o surgimento de novas classes de forma não-supervisionada.

É necessário o desenvolvimento de técnicas para identificar a quantidade de pacotes maliciosos que estão associados a um fluxo malicioso e sua distribuição temporal. Isso ajudaria a entender, por exemplo, em quanto tempo um IDS baseado em análise de pacotes individuais pode identificar ataques em comparação com um baseado em fluxo. Além disso, a comparação entre as duas abordagens (inspeção individual dos pacotes e fluxos) foi feita usando as métricas convencionais da literatura. Contudo, seria importante desenvolver medidas que levem em consideração a relação entre o número e a ordem de um pacote malicioso dentro de um fluxo. Por exemplo, seria necessário que apenas um único pacote de uma conexão fosse classificado como malicioso para que todos os demais também o fossem. Sendo assim, os resultados podem ser ainda melhores do que os apresentados neste trabalho, uma vez que, as medidas que fazem a respectiva comparação sejam corretamente implementadas e utilizadas.

É importante analisar outras técnicas de aprendizado ativo, a utilizada no presente trabalho é uma técnica inocente, onde as instâncias são selecionadas aleatoriamente para serem rotuladas. Seria interessante melhorar esta abordagem utilizando técnicas baseadas em amostragem por incertezas fixas ou variáveis ou, ainda, a estratégia de incerteza aleatória. Novas técnicas poderiam comprovar que ocorrem sobre-ajuste dos modelos de decisão porém, uma análise mais detalhada seria interessante, a fim de criar estratégias que evitariam tal comportamento.

Por fim, sugere-se a implementação dos classificadores de fluxos contínuos de dados de forma distribuída e em comutadores de rede, por exemplo usando a linguagem P4, permitindo efetuar a classificação em tempo real.

6.3 Contribuições em Produção Bibliográfica

O artigo "*Intrusion Detection over Network Packets using Data Stream Classification Algorithms*" (OLIMPIO et al., 2021) apresentou uma investigação onde utilizou algoritmos de fluxos contínuos de dados para determinar que o desempenho preditivo da inspeção individual de pacotes é semelhante ao exame de fluxos em várias situações e que apenas um subconjunto dos cabeçalhos dos pacotes é suficiente para a classificação, satisfazendo a necessidade para um IDS. Ele foi aceito na conferência ICTAI 2021 (*33th IEEE International Conference on Tools with Artificial Intelligence*).

Referências

AGGARWAL, C. C. **Data Mining**. Springer International Publishing, 2015. Disponível em: <https://doi.org/10.1007%2F978-3-319-14142-8>.

AGGARWAL, C. C. et al. A framework for clustering evolving data streams. In: **Proceedings 2003 VLDB Conference**. Elsevier, 2003. p. 81–92. Disponível em: <https://doi.org/10.1016%2Fb978-012722442-8%2F50016-1>.

AKOSA, J. Predictive accuracy: A misleading performance measure for highly imbalanced data. In: **Proceedings of the SAS Global Forum**. [s.n.], 2017. v. 12. Disponível em: <https://support.sas.com/resources/papers/proceedings17/0942-2017.pdf>.

AL-UTAIBI, K. A.; EL-ALFY, E.-S. M. Intrusion detection taxonomy and data preprocessing mechanisms. **Journal of Intelligent & Fuzzy Systems**, IOS Press, v. 34, n. 3, p. 1369–1383, 2018. Disponível em: <https://doi.org/10.3233/JIFS-169432>.

Ariyaluran Habeeb, R. A. et al. Real-time big data processing for anomaly detection: A survey. **International Journal of Information Management**, v. 45, p. 289–307, 2019. ISSN 0268-4012. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0268401218301658>.

ATLI, B. G. et al. Anomaly-based intrusion detection using extreme learning machine and aggregation of network traffic statistics in probability space. **Cognitive Computation**, Springer, v. 10, n. 5, p. 848–863, 2018. Disponível em: <https://doi.org/10.1007/s12559-018-9564-y>.

BARBARA, D.; WU, N.; JAJODIA, S. Detecting novel network intrusions using bayes estimators. In: SIAM. **Proceedings of the 2001 SIAM International Conference on Data Mining**. 2001. p. 1–17. Disponível em: <https://doi.org/10.1137/1.9781611972719.28>.

BHOSALE, S. **Network intrusion detection**. 2018. Disponível em: <https://www.kaggle.com/sampadab17/network-intrusion-detection>.

BHUYAN, M. H.; BHATTACHARYYA, D. K.; KALITA, J. K. Network traffic anomaly detection techniques and systems. In: **Computer Communications and Networks**. Springer International Publishing, 2017. p. 115–169. Disponível em: <https://doi.org/10.1007%2F978-3-319-65188-0_4>.

BIFET, A. et al. Accurate ensembles for data streams: Combining restricted hoeffding trees using stacking. In: SUGIYAMA, M.; YANG, Q. (Ed.). **Proceedings of 2nd Asian Conference on Machine Learning**. Tokyo, Japan: PMLR, 2010. (Proceedings of Machine Learning Research, v. 13), p. 225–240. Disponível em: <https://proceedings.mlr.press/v13/bifet10a.html>.

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

BIFET, A.; HOLMES, G.; PFAHRINGER, B. Leveraging bagging for evolving data streams. In: BALCÁZAR, J. L. et al. (Ed.). **Machine Learning and Knowledge Discovery in Databases**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 135–150. ISBN 978-3-642-15880-3. Disponível em: <https://doi.org/10.1007/978-3-642-15880-3_15>.

BIFET, A. et al. Improving adaptive bagging methods for evolving data streams. In: SPRINGER. **Asian conference on machine learning**. 2009. p. 23–37. Disponível em: <https://doi.org/10.1007/978-3-642-05224-8_4>.

_____. New ensemble methods for evolving data streams. In: **Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2009. (KDD '09), p. 139–148. ISBN 9781605584959. Disponível em: <https://doi.org/10.1145/1557019.1557041>.

_____. Moa: Massive online analysis, a framework for stream classification and clustering. PMLR, Cumberland Lodge, Windsor, UK, v. 11, p. 44–50, 01–03 Sep 2010. Disponível em: <https://proceedings.mlr.press/v11/bifet10a.html>.

BORTOLAMEOTTI, R. et al. Headprint: Detecting anomalous communications through header-based application fingerprinting. In: **Proceedings of the 35th Annual ACM Symposium on Applied Computing**. New York, NY, USA: Association for Computing Machinery, 2020. (SAC '20), p. 1696–1705. ISBN 9781450368667. Disponível em: <https://doi.org/10.1145/3341105.3373862>.

BREIMAN, L. Bagging predictors. **Machine learning**, Springer, v. 24, n. 2, p. 123–140, 1996. Disponível em: <https://doi.org/10.1007/BF00058655>.

_____. Random forests. **Machine learning**, Springer, v. 45, n. 1, p. 5–32, 2001. Disponível em: <https://doi.org/10.1023/A:1010933404324>.

BRZEZIŃSKI, D. Mining data streams with concept drift. **Cs Put Pozna**, v. 89, 2010.

BUCZAK, A. L.; GUVEN, E. A survey of data mining and machine learning methods for cyber security intrusion detection. **IEEE Communications Surveys Tutorials**, v. 18, n. 2, p. 1153–1176, 2016. Disponível em: <https://doi.org/10.1109/COMST.2015.2494502>.

CANO, A.; KRAWCZYK, B. Evolving rule-based classifiers with genetic programming on gpus for drifting data streams. **Pattern Recognition**, Elsevier, v. 87, p. 248–268, 2019. Disponível em: <https://doi.org/10.1016/j.patcog.2018.10.024>.

_____. Kappa updated ensemble for drifting data stream mining. **Machine Learning**, Springer, v. 109, n. 1, p. 175–218, 2020. Disponível em: <https://doi.org/10.1007/s10994-019-05840-z>.

CARELA-ESPAÑOL, V. et al. A streaming flow-based technique for traffic classification applied to 12+ 1 years of internet traffic. **Telecommunication Systems**, Springer, v. 63, n. 2, p. 191–204, 2016. Disponível em: <https://doi.org/10.1007/s11235-015-0114-6>.

CASSALES, G. W. et al. Idsa-iot: An intrusion detection system architecture for iot networks. In: **2019 IEEE Symposium on Computers and Communications (ISCC)**. [s.n.], 2019. p. 1–7. Disponível em: <https://doi.org/10.1109/ISCC47284.2019.8969609>.

CHENG, Z. et al. Development of deep packet inspection system for network traffic analysis and intrusion detection. In: **2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)**. [s.n.], 2020. p. 877–881. Disponível em: <https://doi.org/10.1109/TCSET49122.2020.235562>.

CHOWDHURY, G. G. Natural language processing. **Annual review of information science and technology**, Wiley Online Library, v. 37, n. 1, p. 51–89, 2003. Disponível em: <https://doi.org/10.1002/aris.1440370103>.

CHURCHER, A. et al. An experimental analysis of attack classification using machine learning in iot networks. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 21, n. 2, p. 446, 2021. Disponível em: <https://doi.org/10.3390/s21020446>.

COMER, D. E. **Computer networks and internets**. [S.l.: s.n.], 2017.

COSTA, V. G. T. da et al. Online detection of botnets on network flows using stream mining. In: SBC. **Anais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**. 2018. Disponível em: <https://doi.org/10.5753/sbrc.2018.2418>.

CREECH, G.; HU, J. A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. **IEEE Transactions on Computers**, Institute of Electrical and Electronics Engineers (IEEE), v. 63, n. 4, p. 807–819, apr 2014. Disponível em: <https://doi.org/10.1109%2Ftc.2013.13>.

DARPA. **1998 DARPA Intrusion Detection Evaluation Dataset**. 1998. Disponível em: <https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset>.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **The Journal of Machine Learning Research**, JMLR. org, v. 7, p. 1–30, 2006. Disponível em: <https://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf>.

DHANABAL, L.; SHANTHARAJAH, S. A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. **International journal of advanced research in computer and communication engineering**, v. 4, n. 6, p. 446–452, 2015. Disponível em: <https://doi.org/10.17148/IJARCCE.2015.4696>.

DIETTERICH, T. G. Ensemble methods in machine learning. In: SPRINGER. **International workshop on multiple classifier systems**. 2000. p. 1–15. Disponível em: <https://doi.org/10.1007/3-540-45014-9_1>.

DOMINGOS, P.; HULTEN, G. Mining high-speed data streams. In: **Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2000. (KDD '00), p. 71–80. ISBN 1581132336. Disponível em: <https://doi.org/10.1145/347090.347107>.

FAISAL, M. A. et al. Securing advanced metering infrastructure using intrusion detection system with data stream mining. In: SPRINGER. **Pacific-Asia Workshop on Intelligence and Security Informatics**. 2012. p. 96–111. Disponível em: <https://doi.org/10.1007/978-3-642-30428-6_8>.

_____. Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study. **IEEE Systems Journal**, v. 9, n. 1, p. 31–44, 2015. Disponível em: <http://doi.org/10.1109/JSYST.2013.2294120>.

FALL, K. R.; STEVENS, W. R. **TCP/IP illustrated, volume 1: The protocols**. [S.l.]: addison-Wesley, 2011.

FARIA, E. R. de; CARVALHO, A. C. P. de L. F.; GAMA, J. MINAS: multiclass learning algorithm for novelty detection in data streams. **Data Mining and Knowledge Discovery**, Springer Science and Business Media LLC, v. 30, n. 3, p. 640–680, aug 2015. Disponível em: <https://doi.org/10.1007%2Fs10618-015-0433-y>.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI Magazine**, v. 17, n. 3, p. 37, Mar. 1996. Disponível em: <https://ojs.aaai.org/index.php/aimagazine/article/view/1230>.

_____. The KDD process for extracting useful knowledge from volumes of data. **Communications of the ACM**, Association for Computing Machinery (ACM), v. 39, n. 11, p. 27–34, nov 1996. Disponível em: <https://doi.org/10.1145%2F240455.240464>.

FENG, C.; LI, T.; CHANA, D. Multi-level anomaly detection in industrial control systems via package signatures and lstm networks. In: **2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)**. [s.n.], 2017. p. 261–272. Disponível em: <https://doi.org/10.1109/DSN.2017.34>.

FONTUGNE, R. et al. Mawilab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In: **Proceedings of the 6th International COnference**. New York, NY, USA: Association for Computing Machinery, 2010. (Co-NEXT '10). ISBN 9781450304481. Disponível em: <https://doi.org/10.1145/1921168.1921179>.

GABER, M. M.; ZASLAVSKY, A.; KRISHNASWAMY, S. A survey of classification methods in data streams. In: **Data streams**. Springer, 2007. p. 39–59. Disponível em: <https://doi.org/10.1007/978-0-387-47534-9_3>.

GAMA, J. et al. Learning with drift detection. In: SPRINGER. **Brazilian symposium on artificial intelligence**. 2004. p. 286–295. Disponível em: <https://doi.org/10.1007/978-3-540-28645-5_29>.

GAMA, J. a. et al. A survey on concept drift adaptation. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 46, n. 4, mar 2014. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/2523813>.

GARCIA, S. et al. An empirical comparison of botnet detection methods. **computers & security**, Elsevier, v. 45, p. 100–123, 2014. Disponível em: <https://doi.org/10.1016/j.cose.2014.05.011>.

GARCÍA-TEODORO, P. et al. Anomaly-based network intrusion detection: Techniques, systems and challenges. **Computers & Security**, Elsevier BV, v. 28, n. 1-2, p. 18–28, feb 2009. Disponível em: <https://doi.org/10.1016%2Fj.cose.2008.08.003>.

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

GOMES, J. B. et al. Mining recurring concepts in a dynamic feature space. **IEEE Transactions on Neural Networks and Learning Systems**, v. 25, n. 1, p. 95–110, 2014. Disponível em: <https://doi.org/10.1109/TNNLS.2013.2271915>.

GROUP, M. W. et al. Traffic archive. **http://tracer.csl.sony.co.jp/mawi/**.

GU, J.; LU, S. An effective intrusion detection approach using svm with naïve bayes feature embedding. **Computers & Security**, Elsevier, v. 103, p. 102158, 2021. Disponível em: <https://doi.org/10.1016/j.cose.2020.102158>.

Habibi Lashkari., A. et al. Characterization of tor traffic using time based features. In: INSTICC. **Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP,**. SciTePress, 2017. p. 253–262. ISBN 978-989-758-209-7. ISSN 2184-4356. Disponível em: <https://doi.org/10.5220/0006105602530262>.

HAIDER, A. et al. A real-time sequential deep extreme learning machine cybersecurity intrusion detection system. TECH SCIENCE PRESS, 2021. Disponível em: <http://dx.doi.org/10.32604/cmc.2020.013910>.

HAMID, Y.; MUTHUKUMARASAMY, S.; RANGANATHAN, B. Ids using machine learning -current state of art and future directions. **British Journal of Applied Science and Technology**, v. 15, p. 1–22, 03 2016. Disponível em: <http://dx.doi.org/10.9734/BJAST/2016/23668>.

HAN, J.; PEI, J.; KAMBER, M. **Data mining: concepts and techniques**. [S.l.]: Elsevier, 2011.

HANSEN, L.; SALAMON, P. Neural network ensembles. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 12, n. 10, p. 993–1001, 1990. Disponível em: <https://doi.org/10.1109/34.58871>.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 1997. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>.

HOEFFDING, W. Probability inequalities for sums of bounded random variables. In: **The collected works of Wassily Hoeffding**. Springer, 1994. p. 409–426. Disponível em: <https://doi.org/10.1007/978-1-4612-0865-5_26>.

HSU, Y.-F. et al. Toward an online network intrusion detection system based on ensemble learning. In: **2019 IEEE 12th International Conference on Cloud Computing (CLOUD)**. [s.n.], 2019. p. 174–178. Disponível em: <https://doi.org/10.1109/CLOUD.2019.00037>.

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

JI, S.-Y. et al. A multi-level intrusion detection method for abnormal network behaviors. **Journal of Network and Computer Applications**, v. 62, p. 9–17, 2016. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804515002970>.

KANIMOZHI, V.; JACOB, T. P. Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset cse-cic-ids2018 using cloud computing. In: **2019 International Conference on Communication and Signal Processing (ICCSP)**. [s.n.], 2019. p. 0033–0036. Disponível em: <https://doi.org/10.1109/ICCSP.2019.8698029>.

KANTARDZIC, M. **Data Mining**. John Wiley & Sons, Inc., 2011. Disponível em: <https://doi.org/10.1002%2F9781118029145>.

KAVITHA, S.; MAHESWARI, N. U. et al. Network anomaly detection for nsl-kdd dataset using deep learning. **Information Technology in Industry**, v. 9, n. 2, p. 821–827, 2021.

KHRAISAT, A. et al. Survey of intrusion detection systems: techniques, datasets and challenges. **Cybersecurity**, Springer Science and Business Media LLC, v. 2, n. 1, jul 2019. Disponível em: <https://doi.org/10.1186%2Fs42400-019-0038-7>.

KIM, A.; PARK, M.; LEE, D. H. Ai-ids: Application of deep learning to real-time web intrusion detection. **IEEE Access**, v. 8, p. 70245–70261, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2020.2986882>.

KNOWLEDGE Discovery from Data Streams. Chapman and Hall/CRC, 2010. 21–26 p. Disponível em: <https://doi.org/10.1201%2Febk1439826119-7>.

KOLIAS, C. et al. Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. **IEEE Communications Surveys & Tutorials**, Institute of Electrical and Electronics Engineers (IEEE), v. 18, n. 1, p. 184–208, 2016. Disponível em: <https://doi.org/10.1109%2Fcomst.2015.2402161>.

KORONIOTIS, N. et al. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. **Future Generation Computer Systems**, Elsevier BV, v. 100, p. 779–796, nov 2019. Disponível em: <https://doi.org/10.1016%2Fj.future.2019.05.041>.

KOZIK, R.; PAWLICKI, M.; CHORAŚ, M. A new method of hybrid time window embedding with transformer-based traffic data classification in iot-networked environment. **Pattern Analysis and Applications**, Springer, p. 1–9, 2021. Disponível em: <https://doi.org/10.17762/itii.v9i2.419>.

KRAWCZYK, B. et al. Ensemble learning for data stream analysis: A survey. **Information Fusion**, v. 37, p. 132–156, 2017. ISSN 1566-2535. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1566253516302329>.

KUMARI, V. V.; VARMA, P. R. K. A semi-supervised intrusion detection system using active learning SVM and fuzzy c-means clustering. In: **2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)**. IEEE, 2017. Disponível em: <https://doi.org/10.1109%2Fi-smac.2017.8058397>.

KURNIABUDI, K. et al. Important features of cicids-2017 dataset for anomaly detection in high dimension and imbalanced class dataset. **Indonesian Journal of Electrical Engineering and Informatics (IJEEI)**, v. 9, n. 2, p. 498–511, 2021.

KUROSE, J. F.; ROSS, K. W. **Computer networking: a top-down approach**. [S.l.]: Addison Wesley, 2011.

LAROCHELLE, H. et al. Exploring strategies for training deep neural networks. **Journal of machine learning research**, v. 10, n. 1, 2009. Disponível em: <https://www.jmlr.org/papers/volume10/larochelle09a/larochelle09a.pdf>.

LEE, W.; STOLFO, S. J. A framework for constructing features and models for intrusion detection systems. **ACM Transactions on Information and System Security**, Association for Computing Machinery (ACM), v. 3, n. 4, p. 227–261, nov 2000. Disponível em: <https://doi.org/10.1145%2F382912.382914>.

LEMAIRE, V.; SALPERWYCK, C.; BONDU, A. A survey on supervised classification on data streams. In: ZIMÁNYI, E.; KUTSCHE, R.-D. (Ed.). **Business Intelligence: 4th European Summer School, eBISS 2014, Berlin, Germany, July 6-11, 2014, Tutorial Lectures**. Cham: Springer International Publishing, 2015. p. 88–125. ISBN 978-3-319-17551-5. Disponível em: <https://doi.org/10.1007/978-3-319-17551-5_4>.

LI, Y.; GUO, L. An active learning based TCM-KNN algorithm for supervised network intrusion detection. **Computers & Security**, Elsevier BV, v. 26, n. 7-8, p. 459–467, dec 2007. Disponível em: <https://doi.org/10.1016%2Fj.cose.2007.10.002>.

LI, Y. et al. An efficient intrusion detection system based on support vector machines and gradually feature removal method. **Expert Systems with Applications**, Elsevier BV, v. 39, n. 1, p. 424–430, jan 2012. Disponível em: <https://doi.org/10.1016%2Fj.eswa.2011.07.032>.

LIAO, H.-J. et al. Intrusion detection system: A comprehensive review. **Journal of Network and Computer Applications**, Elsevier BV, v. 36, n. 1, p. 16–24, jan 2013. Disponível em: <https://doi.org/10.1016%2Fj.jnca.2012.09.004>.

LOH, W.-Y. Fifty years of classification and regression trees. **International Statistical Review**, v. 82, n. 3, p. 329–348, 2014. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/insr.12016>.

MAHONEY, M. V.; CHAN, P. K. **PHAD: Packet header anomaly detection for identifying hostile network traffic**. [S.l.], 2001. Disponível em: <http://hdl.handle.net/11141/94>.

Data mining and knowledge discovery handbook. Springer-Verlag, 2005. Disponível em: <https://doi.org/10.1007%2Fb107408>.

MASUD, M. et al. Classification and novel class detection in concept-drifting data streams under time constraints. **IEEE Transactions on Knowledge and Data Engineering**, v. 23, n. 6, p. 859–874, 2011. Disponível em: <https://doi.org/10.1109/TKDE.2010.61>.

MASUD, M. M. et al. Addressing concept-evolution in concept-drifting data streams. In: **2010 IEEE International Conference on Data Mining**. [s.n.], 2010. p. 929–934. Disponível em: <https://doi.org/10.1109/ICDM.2010.160>.

MIN, E. et al. Tr-ids: Anomaly-based intrusion detection through text-convolutional neural network and random forest. **Security and Communication Networks**, Hindawi, v. 2018, 2018. Disponível em: <https://doi.org/10.1155/2018/4943509>.

MOHAMAD, S.; SAYED-MOUCHAWEH, M.; BOUCHACHIA, A. Active learning for classifying data streams with unknown number of classes. **Neural Networks**, v. 98, p. 1–15, 2018. ISSN 0893-6080. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0893608017302435>.

MORALES, G. D. F. et al. Iot big data stream mining. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 2119–2120. ISBN 9781450342322. Disponível em: <https://doi.org/10.1145/2939672.2945385>.

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

MOUSTAFA, N.; SLAY, J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: **2015 Military Communications and Information Systems Conference (MilCIS)**. IEEE, 2015. Disponível em: <https://doi.org/10.1109%2Fmilcis.2015.7348942>.

NAKAMURA, E. T.; GEUS, P. L. de. **Segurança de redes em ambientes cooperativos**. [S.l.]: Novatec Editora, 2007.

NEWMAN, M. **Networks**. [S.l.]: Oxford university press, 2018.

NOORBEHBAHANI, F. 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, p. e3002, 2017. Disponível em: <https://doi.org/10.1002/dac.3002>.

OLIMPIO, G. et al. Intrusion detection over network packets using data stream classification algorithms. In: **2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)**. IEEE, 2021. (ICTAI '21). Disponível em: <https://doi.org/10.1109/ICTAI52525.2021.00157>.

OUSSOUS, A. et al. Big data technologies: A survey. **Journal of King Saud University - Computer and Information Sciences**, Elsevier BV, v. 30, n. 4, p. 431–448, oct 2018. Disponível em: <https://doi.org/10.1016%2Fj.jksuci.2017.06.001>.

OZA, N. C.; RUSSELL, S. Experimental comparisons of online and batch versions of bagging and boosting. In: **Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2001. (KDD '01), p. 359–364. ISBN 158113391X. Disponível em: <https://doi.org/10.1145/502512.502565>.

PALOALTO. **Palo Alto**. 2019. Disponível em: <https://docs.paloaltonetworks.com/cortex/cortex-xdr/cortex-xdr-xql-language-reference/get-started-with-xql/datasets.html>.

PAPAMARTZIVANOS, D.; MáRMOL, F. G.; KAMBOURAKIS, G. Introducing deep learning self-adaptive misuse network intrusion detection systems. **IEEE Access**, v. 7, p. 13546–13560, 2019. Disponível em: <https://doi.org/10.1109/ACCESS.2019.2893871>.

PARTHASARATHY, S.; KUNDUR, D. Bloom filter based intrusion detection for smart grid scada. In: **2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)**. [s.n.], 2012. p. 1–6. Disponível em: <https://doi.org/10.1109/CCECE.2012.6334816>.

PAXSON, V. Bro: A system for detecting network intruders in real-time. **Computer networks**, Elsevier, v. 31, n. 23-24, p. 2435–2463, 1999. Disponível em: <https://doi.org/10.1016/S1389-1286(99)00112-7>.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011. Disponível em: <https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf>.

POSTEL, J. et al. **RFC 768: User datagram protocol**. [S.l.]: August, 1980.

QUINLAN, J. R. Induction of decision trees. **Machine learning**, Springer, v. 1, n. 1, p. 81–106, 1986. Disponível em: <https://doi.org/10.1007/BF00116251>.

RIBEIRO, G. H.; PAIVA, E. R. de F.; MIANI, R. S. A comparison of stream mining algorithms on botnet detection. In: **Proceedings of the 15th International Conference on Availability, Reliability and Security**. New York, NY, USA: Association for Computing Machinery, 2020. (ARES '20). ISBN 9781450388337. Disponível em: <https://doi.org/10.1145/3407023.3407053>.

RING, M. et al. Flow-based network traffic generation using generative adversarial networks. **Computers & Security**, v. 82, p. 156–172, 2019. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404818308393>.

_____. A survey of network-based intrusion detection data sets. **Computers & Security**, v. 86, p. 147–167, 2019. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S016740481930118X>.

ROESCH, M. et al. Snort: Lightweight intrusion detection for networks. In: **Lisa**. [s.n.], 1999. v. 99, n. 1, p. 229–238. Disponível em: <https://www.usenix.org/legacy/publications/library/proceedings/lisa99/full_papers/roesch/roesch.pdf>.

ROKACH, L. Ensemble-based classifiers. **Artificial Intelligence Review**, Springer Science and Business Media LLC, v. 33, n. 1-2, p. 1–39, nov 2009. Disponível em: <https://doi.org/10.1007%2Fs10462-009-9124-7>.

SARKER, I. H. et al. Intrudtree: a machine learning based cyber security intrusion detection model. **Symmetry**, Multidisciplinary Digital Publishing Institute, v. 12, n. 5, p. 754, 2020. Disponível em: <https://doi.org/10.3390/sym12050754>.

SETTLES, B. **Active Learning Literature Survey**. [S.l.], 2009. Disponível em: <http://digital.library.wisc.edu/1793/60660>.

_____. From theories to queries: Active learning in practice. In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. **Active Learning and Experimental Design workshop In conjunction with AISTATS 2010**. 2011. p. 1–18. Disponível em: <https://proceedings.mlr.press/v16/settles11a>.

SHARAFALDIN, I.; LASHKARI, A. H.; GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: **Proceedings of the 4th International Conference on Information Systems Security and Privacy**. SCITEPRESS - Science and Technology Publications, 2018. Disponível em: <https://doi.org/10.5220%2F0006639801080116>.

SHIRAVI, A. et al. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. **computers & security**, Elsevier, v. 31, n. 3, p. 357–374, 2012. Disponível em: <https://doi.org/10.1016/j.cose.2011.12.012>.

SOMMER, R.; PAXSON, V. Outside the closed world: On using machine learning for network intrusion detection. In: **2010 IEEE Symposium on Security and Privacy**. IEEE, 2010. Disponível em: <https://doi.org/10.1109%2Fsp.2010.25>.

SONG, J. 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**. New York, NY, USA: Association for Computing Machinery, 2011. (BADGERS '11), p. 29–36. ISBN 9781450307680. Disponível em: <https://doi.org/10.1145/1978672.1978676>.

SOUZA, V. M. et al. Classification of evolving data streams with infinitely delayed labels. In: **2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)**. [s.n.], 2015. p. 214–219. Disponível em: <http://doi.org/10.1109/ICMLA.2015.174>.

TANENBAUM, A. S.; WETHERALL, D. et al. **Computer networks**. [S.l.]: Harlow, Essex: Pearson„ 2014. ISBN 9780130384881.

TAVALLAEE, M. et al. A detailed analysis of the KDD CUP 99 data set. In: **2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications**. IEEE, 2009. Disponível em: <https://doi.org/10.1109%2Fcisda.2009.5356528>.

TSYMBAL, A. The problem of concept drift: definitions and related work. **Computer Science Department, Trinity College Dublin**, Citeseer, v. 106, n. 2, p. 58, 2004. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.9085&rep=rep1&type=pdf>.

UHM, Y.; PAK, W. Service-aware two-level partitioning for machine learning-based network intrusion detection with high performance and high scalability. **IEEE Access**, v. 9, p. 6608–6622, 2021. Disponível em: <https://doi.org/10.1109/ACCESS.2020.3048900>.

VIEGAS, E. et al. Bigflow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. **Future Generation Computer Systems**, v. 93, p. 473–485, 2019. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X18307635>.

VIEGAS, E.; SANTIN, A. O.; JR, V. A. Machine learning intrusion detection in big data era: A multi-objective approach for longer model lifespans. **IEEE Transactions on Network Science and Engineering**, Institute of Electrical and Electronics Engineers (IEEE), v. 8, n. 1, p. 366–376, jan 2021. Disponível em: <https://doi.org/10.1109%2Ftnse.2020.3038618>.

VIEGAS, E. K.; SANTIN, A. O.; OLIVEIRA, L. S. Toward a reliable anomaly-based intrusion detection in real-world environments. **Computer Networks**, v. 127, p. 200–216, 2017. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128617303225>.

VIEGAS, E. K. et al. Facing the unknown: A stream learning intrusion detection system for reliable model updates. In: **AINA**. [s.n.], 2020. p. 898–909. Disponível em: <https://doi.org/10.1007/978-3-030-44041-1_78>.

WAITZMAN, C. **Computer Networks: A Systems Approach**. Elsevier BV, 2001. v. 24. 1116–1117 p. Disponível em: <https://doi.org/10.1016%2Fs0140-3664%2800%2900345-5>.

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

YANG, K. et al. Active learning for wireless iot intrusion detection. **IEEE Wireless Communications**, v. 25, n. 6, p. 19–25, 2018. Disponível em: <https://doi.org/10.1109/MWC.2017.1800079>.

YIN, C. et al. A deep learning approach for intrusion detection using recurrent neural networks. **IEEE Access**, Institute of Electrical and Electronics Engineers (IEEE), v. 5, p. 21954–21961, 2017. Disponível em: <https://doi.org/10.1109%2Faccess.2017.2762418>.

ZARPELãO, B. B. et al. A survey of intrusion detection in internet of things. **Journal of Network and Computer Applications**, v. 84, p. 25–37, 2017. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804517300802>.

ZHANG, S.; ZHANG, C.; YANG, Q. Data preparation for data mining. **Applied Artificial Intelligence**, Taylor & Francis, v. 17, n. 5-6, p. 375–381, 2003. Disponível em: <https://doi.org/10.1080/713827180>.

ZHONG, Y. et al. Helad: A novel network anomaly detection model based on heterogeneous ensemble learning. **Computer Networks**, v. 169, p. 107049, 2020. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128619304086>.

ZHU, X. et al. Active learning from data streams. In: **Seventh IEEE International Conference on Data Mining (ICDM 2007)**. [s.n.], 2007. p. 757–762. Disponível em: <https://doi.org/10.1109/ICDM.2007.101>.

ŽLIOBAITĖ, I. et al. Active learning with evolving streaming data. In: SPRINGER. **Joint European Conference on Machine Learning and Knowledge Discovery in Databases**. 2011. p. 597–612. Disponível em: <https://doi.org/10.1007/978-3-642-23808-6_39>.