Detecção de *botnets* utilizando classificação de fluxos contínuos de dados

Guilherme Henrique Ribeiro

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

Uberlândia
2020

Guilherme Henrique Ribeiro

Detecção de *botnets* utilizando classificação 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: Rodrigo Sanches Miani
Coorientador: Elaine Ribeiro de Faria Paiva

Uberlândia
2020

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

R484 2020

Ribeiro, Guilherme Henrique, 1985-
Detecção de botnets utilizando classificação de fluxos contínuos de dados [recurso eletrônico] / Guilherme Henrique Ribeiro. - 2020.

Orientador: Rodrigo Sanches Miani.
Coorientadora: Elaine Ribeiro de Faria Paiva.
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.2021.31
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Miani, Rodrigo Sanches,1983-, (Orient.). II. Paiva, Elaine Ribeiro de Faria,1980-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. 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, 41/2020, PPGCO				
Data:	22 de dezembro de 2020	Hora de início:	14:00	Hora de encerramento:	16:55
Matrícula do Discente:	11812CCP015				
Nome do Discente	Guilherme Henrique Ribeiro				
Título do Trabalho:	Detecção de botnets utilizando classificação de fluxos contínuos de dados				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Sistemas de Computação				
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: Murillo Guimarães Carneiro - FACOM/UFU; Kelton Augusto Pontara - FATEC/UNESP; Elaine Ribeiro de Faria Paiva - FACOM/UFU (coorientadora) e Rodrigo Sanches Miani - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Kelton Augusto Pontara - Bauru/SP; Murillo Guimarães Carneiro, Elaine Ribeiro de Faria Paiva e Rodrigo Sanches Miani - Uberlândia/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Rodrigo Sanches Miani, 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 o senhor 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 **Elaine Ribeiro de Faria Paiva**, **Professor(a) do Magistério Superior**, em 23/12/2020, 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 24/12/2020, às 16:37, 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 **Murillo Guimarães Carneiro**, **Professor(a) do Magistério Superior**, em 29/12/2020, às 11:14, 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 **Kelton Augusto Pontara da Costa**, **Usuário Externo**, em 29/12/2020, às 20:01, 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 **2470027** e o código CRC **2B8C5321**.

Referência: Processo nº 23117.076418/2020-38 SEI nº 2470027

Agradecimentos

Agradeço a Deus, por ter iluminado meu caminho. Por ter me dado a força necessária para continuar e acreditar no quanto o conhecimento pode contribuir para o bem de todos.

Agradeço aos meus pais, Eli e Fátima, por terem me inspirado e me ajudado a encarar mais esse desafio. Vocês são meus exemplos de honestidade, bondade e perseverança.

Agradeço à minha irmã, Eloisa, por sempre estar perto, por me ajudar, aconselhar e querer o meu bem. Obrigado por me dar força para perseverar.

Agradeço a Camila, por me manter motivado, por todo carinho, compreensão e ajuda.

Agradeço ao meus gestores e colegas de trabalho por todo apoio e compreensão.

Agradeço ao professor Rodrigo Sanches Miani, meu orientador, e a professora Elaine Ribeiro de Faria Paiva, minha coorientadora, por todo apoio e paciência. Agradeço por todas as horas despendidas, todas as reuniões, dicas e conselhos. Sem vocês essa dissertação não seria possível.

"O que sabemos é uma gota; o que ignoramos é um oceano."
(Isaac Newton)

Resumo

O ano de 2016 marcou uma significativa mudança de paradigma associada ao comportamento das *botnets*. Ao infectar dispositivos computacionais não convencionais como câmeras e roteadores domésticos, o malware *Mirai* propiciou um aumento, não somente na abrangência, mas também na capacidade de ataques das *botnets*. Este fato enfatiza a importância de desenvolver novos métodos para detectar *botnets*. Um deles envolve o uso de algoritmos de mineração de fluxos contínuos para classificar tráfego malicioso em uma rede. Embora já existam algumas iniciativas que adotem essa abordagem para detectar *botnets*, vários problemas de pesquisa ainda estão abertos. Um ponto importante está relacionado ao alto custo e grande esforço dispendido pelos profissionais de segurança para obter dados rotulados. Sendo assim, o principal objetivo deste trabalho abrange a avaliação do uso de algoritmos de mineração de fluxos contínuos de dados para a detecção de *botnets* considerando requisitos mais próximos aos cenários reais, tais como: i) os fluxos de dados estão constantemente chegando, ii) novos ataques podem surgir e tais ataques não estão presentes no modelo de decisão, iii) poucos fluxos são rotulados e iv) a avaliação da qualidade do classificador deve ser feita atentando-se para o momento em que os fluxos chegam, em particular aqueles em que novos ataques chegam. Ao longo do trabalho, uma série de experimentos foi conduzido usando conjuntos de dados contendo tráfego real de diferentes tipos de *botnets*. Os resultados experimentais mostram o potencial da classificação de fluxos contínuos de dados para detecção de *botnets* e revelam que é possível minimizar a quantidade de instâncias rotuladas apresentadas ao classificador, mantendo um bom desempenho.

Palavras-chave: Segurança da informação. Sistemas de detecção de intrusão. *Botnets*. Classificação de fluxos contínuos de dados.

Abstract

The 2016 year has marked a significant paradigm shift associated with the behavior of *botnets*. By infecting unconventional computing devices such as home cameras and routers, the *Mirai* malware significantly impacted the scope and attack capacity of the *botnets*. This fact emphasizes the importance of developing new methods to detect *botnets*. One of them involves using data stream mining algorithms to classify malicious *botnet* traffic. Despite the existence of some initiatives that adopt this approach, several research problems remain open. An important research topic is related to the high cost and effort spent by security professionals to obtain labeled data. Therefore, the main objective of this dissertation covers the evaluation of stream mining algorithms for detecting *botnets* considering requirements closer to the real-world scenarios, such as i) data flows are continually arriving, ii) new *botnet* attacks may arise and such attacks might not be available to the decision model, iii) usually, few flows are labeled and iv) the evaluation of the classification should be done taking into account the moment when the flows arrive, in particular the ones in which new attacks arrive. Throughout the work, a series of experiments was conducted using datasets containing real traffic from different types of *botnets*. The experimental results show the potential of the stream mining approach for detection of *botnets* and reveal that it is possible to minimize the number of labeled instances presented to the classifier, maintaining a good performance.

Keywords: Information security. Intrusion detection systems. Botnets. Stream mining classification.

Lista de ilustrações

Figura 1 – Arquitetura de uma *botnet*. Adaptado de (KHATTAK; RAMAY; KHAYAM, 2014). 29
Figura 2 – Mineração de dados e sua relação com várias disciplinas. Adaptado de (TAN; KUMAR, 2006). 33
Figura 3 – O processo de KDD (*Knowledge Discovery in Databases*). Adaptado de (TAN; KUMAR, 2006). 33
Figura 4 – Representação da tarefa de Classificação. Adaptado de (TAN; KUMAR, 2006). 35
Figura 5 – Abordagem geral para construir um modelo de classificação. Adaptado de (TAN; KUMAR, 2006). 36
Figura 6 – Exemplo de um classificador baseado em conjunto (*ensemble*). Adaptado de (FARID et al., 2013) 39
Figura 7 – Visão geral do processo incremental de classificação de fluxos 58
Figura 8 – Etapa incremental opcional do Processo Iterativo para Classificação de Fluxos 59
Figura 9 – Desempenho do *Ozaboost* nos cenários 1, 2, 3, 4, 5 e 6 do CTU-13 . . . 75
Figura 10 – Desempenho do *Ozaboost* nos cenários 10,11,12 e 13 do CTU-13 76
Figura 11 – Desempenho *Ozaboost* por janela nos cenários 2 e 8 do CTU-13 79
Figura 12 – Distribuição de classes no cenário 2 do CTU-13. Escala do eixo Y limitada de 60 a 100% 81
Figura 13 – Distribuição de classes no cenário 8 do CTU-13. Escala do eixo Y limitada de 96 a 100% 81
Figura 14 – Aprendizado ativo - Aleatório relacionado ao cenário 2 do CTU-13 . . 85
Figura 15 – Aprendizado ativo - Aleatório relacionado ao cenário 8 do CTU-13 . . 86
Figura 16 – Aprendizado ativo - Incerteza Variável com Aleatorização relacionado ao cenário 2 do CTU-13 87
Figura 17 – Aprendizado ativo - Incerteza Variável relacionado ao cenário 8 do CTU-13 88

Figura 18 – Distribuição de classes para os cenários 1, 4, 14, 15, 19 e 20 do IoT-23 . 91
Figura 19 – Precisão x Revocação para cada classe do cenário 1 do IoT-23 92
Figura 20 – Precisão x Revocação para cada classe do cenário 4 do IoT-23 93
Figura 21 – Precisão x Revocação para cada classe do cenário 14 do IoT-23 94
Figura 22 – Precisão x Revocação para cada classe do cenário 15 do IoT-23 95
Figura 23 – Precisão x Revocação para cada classe do cenário 19 do IoT-23 96
Figura 24 – Precisão x Revocação para cada classe do cenário 20 do IoT-23 97

Lista de tabelas

Tabela 1 – Diferenças entre processamento de dados tradicionais e fluxos contínuos. Adaptado de (GAMA, 2010). 38
Tabela 2 – Trabalhos relacionados sobre detecção de *botnets* 47
Tabela 3 – Variáveis base do modelo proposto para parametrização nos cenários . 60
Tabela 4 – Variáveis do modelo proposto para cenário de comparação de classificadores 61
Tabela 5 – Variáveis do modelo proposto para cenário utilizando janelas 62
Tabela 6 – Variáveis do modelo proposto para cenário com aprendizado ativo . . . 64
Tabela 7 – Variáveis do modelo proposto para o cenário multi-classe 64
Tabela 8 – Quantidade de dados em cada cenário do CTU-13 68
Tabela 9 – Distribuição de classes em cada cenário do CTU-13 68
Tabela 10 – Quantidade de dados em cada cenário do IoT-23 69
Tabela 11 – Distribuição de classes nos cenários selecionados do IoT-23 69
Tabela 12 – Lista de atributos selecionados da base CTU-13 70
Tabela 13 – Lista de atributos selecionados da base IoT-23 71
Tabela 14 – Tabela consolidada com as características de cada experimento 72
Tabela 15 – Configuração para experimento de comparação de classificadores 73
Tabela 16 – Comparação de desempenho entre os algoritmos considerando média e *best n* obtidos em cada cenário selecionado do CTU-13 73
Tabela 17 – Desempenho para o *best n* - (COSTA et al., 2018) versus Ozaboost MOA 74
Tabela 18 – CTU-13 x Tipo de *Botnet* x Ozaboost 77
Tabela 19 – Configuração para experimento utilizando janelas 79
Tabela 20 – Configuração para experimento de aprendizado ativo 83
Tabela 21 – Relação de execuções para aprendizado ativo 83
Tabela 22 – Configuração para experimento multi-classe 90
Tabela 23 – Resultados consolidados para cenários IoT-23 90

Lista de siglas

ADWIN *Adaptive Sliding Window*

AUE *Accuracy Updated Ensemble*

C&C *Command & Control*

CTU *Czech Technical University*

DoS *Denial Of Service*

DDoS *Distributed Denial Of Service*

DNS *Domain Name System*

HB *Hoeffding Bound*

IDS *Intrusion Detection System*

IoT *Internet of Things*

ISOT *Information Security and Object Technology*

ISP *Internet Service Provider*

KDD *Knowledge Discovery in Databases*

MOA *Massive Online Analysis*

NIST *National Institute of Standards and Technology*

WEKA *Waikato Environment for Knowledge Analysis*

VFDT *Very Fast Decision Tree*

Sumário

1 INTRODUÇÃO 21
1.1 Motivação 23
1.2 Objetivos 24
1.3 Hipótese 25
1.4 Organização da Dissertação 25

2 FUNDAMENTAÇÃO TEÓRICA 27
2.1 Segurança de computadores 27
2.2 *Botnets* 29
2.2.1 Arquiteturas e modo de operação 30
2.2.2 Técnicas de detecção 31
2.3 Mineração de Dados 32
2.3.1 Pré-processamento dos dados 34
2.3.2 Classificação 35
2.3.3 Conjunto de Classificadores - *Ensemble* 36
2.4 Mineração de fluxos contínuos de dados 37
2.4.1 Classificação única incremental 39
2.4.2 Classificação baseada em conjuntos 40
2.4.3 *Prequential* 41
2.5 Atualização do modelo de decisão 41
2.5.1 Aprendizado ativo 41
2.5.2 Estratégias de aprendizado ativo 42
2.6 Medidas de Avaliação 44

3 TRABALHOS RELACIONADOS 45
3.1 Detecção de *botnets* 45
3.1.1 Detecção de *botnets* em um cenário tradicional (lote) 46
3.1.2 Detecção de *botnets* em fluxos contínuos de dados 49

3.2 Conjuntos de dados com amostras de *botnets* 52
3.2.1 CTU-13 53
3.2.2 IoT-23 54

4 PROPOSTA 55
4.1 Contextualização 55
4.2 Processo Iterativo para Classificação de Fluxos 56
4.2.1 Comparar o uso de um único classificador em contraste ao uso de conjuntos de classificadores 60
4.2.2 Analisar resultados da classificação utilizando janelas 62
4.2.3 Analisar desempenho de conjuntos de classificadores utilizando estratégias de aprendizado ativo 63
4.2.4 Analisar conjuntos de classificadores utilizando dados multi-classe . . . 64

5 EXPERIMENTOS E ANÁLISE DOS RESULTADOS 67
5.1 Conjuntos de dados, atributos e medidas de avaliação 67
5.1.1 Conjuntos de dados 67
5.1.2 Atributos 70
5.2 Experimentos e resultados 71
5.2.1 Comparação do uso de um único classificador em contraste ao uso de conjuntos de classificadores 72
5.2.2 Análise dos resultados da classificação utilizando janelas 78
5.2.3 Análise do desempenho de conjuntos de classificadores utilizando estratégias de aprendizado ativo 82
5.2.4 Análise de conjuntos de classificadores utilizando conjuntos de dados multi-classe 88

6 CONCLUSÃO 99
6.1 Principais Contribuições 101
6.2 Trabalhos Futuros 102
6.3 Contribuições em Produção Bibliográfica 102

REFERÊNCIAS 105

Capítulo 1

Introdução

A Internet se tornou tão popular que alcançou um nível de interatividade único na história, fornecendo inúmeras vantagens para a sociedade. Atualmente, é possível acessar, monitorar e controlar de forma remota diferentes tipos de recursos computacionais desde os domésticos até os industriais. Entretanto, a disseminação desenfreada de tecnologias inseguras por empresas e usuários, sem se preocupar com aspectos essenciais de segurança, criaram e expuseram falhas críticas que cibercriminosos podem explorar (PERAKOVIĆ; PERIŠA; CVITIĆ, 2015). Algumas questões sobre a infraestrutura da Internet podem ser levantadas: como exatamente ela funciona, contra quais ameaças deve ser protegida, e como uma proteção pode ser disponibilizada (BISHOP, 2003).

Neste cenário, surgem várias ameaças cibernéticas. Uma delas são as *botnets*, que consistem em redes formadas por máquinas comprometidas por *malwares*, os *bots* (SILVA et al., 2013). *Botnets* são utilizadas para uma variedade de atividades maliciosas como *fraud click*, *phishing*, *spamming*, entrega de *malware* e *DDoS (Distributed Denial Of Service)* (GAONKAR et al., 2020). Alguns exemplos de *botnets* utilizadas em ataques incluem *Mirai*, *Hajime*, *Aidra*, *Bashlite*, *Dofloo* e *Tsunami* (BEZERRA et al., 2018).

Uma das soluções mais comuns para detecção de *botnets* consiste em desenvolver sistemas para analisar o tráfego da rede e identificar componentes maliciosos. Tais sistemas são chamados de Sistemas de Detecção de Intrusão (*Intrusion Detection System - IDS*). A classificação do tráfego malicioso por IDS pode ser feita com o auxílio de bases de dados de ataques conhecidos ou usando técnicas para determinar comportamentos anômalos presentes nos pacotes de rede. O grande diferencial entre as duas técnicas reside no fato que a detecção de anomalia permite que novos ataques possam ser identificados pelo sistema. Essa motivação possibilitou que diferentes soluções para detecção de *botnet*s usando métodos baseados em identificação de comportamento anômalo fossem propostas ((SILVA et al., 2013), (GARCÍA; ZUNINO; CAMPO, 2014) e (RODRIGUEZ-GOMEZ; MACIA-FERNANDEZ; GARCIA-TEODORO, 2013)). Estas soluções se diferenciam em vários aspectos, tais como o foco em um protocolo específico de comunicação usado pelo servidor de comando e controle (C&C - *Command & Control*) e o *bot*, algoritmos usados

para classificar tráfego de rede normal e de *botnet*, e os conjuntos de dados utilizados.

De acordo com (GARCÍA; ZUNINO; CAMPO, 2014) a maioria dos métodos usados para detectar *botnets* à partir da identificação de comportamento anômalo do tráfego de rede, envolve o uso de algoritmos convencionais de aprendizado de máquina, tais como, *Support Vector Machine* e *Naive Bayes*. Tais algoritmos convencionais são baseados no aprendizado a partir de um conjunto de treinamento, para desenvolver um modelo de decisão. Esse modelo de decisão é então utilizado para classificar instâncias do conjunto de teste. Como o comportamento do tráfego de rede muda de forma contínua e novas *botnet*s estão surgindo (SPAMHAUS, 2019), o aprendizado de máquina tradicional baseado em lotes de dados estáticos (não baseados em fluxos) pode não ser adequado para desenvolver métodos de detecção de *botnet*s que sejam escaláveis e efetivos ((COSTA et al., 2018) e (KRAWCZYK et al., 2017)). Uma classe diferente de algoritmos de aprendizado de máquina, chamada Mineração de Fluxos Contínuos (*Stream Mining*), pode lidar com tais limitações dos algoritmos convencionais.

O principal objetivo dos algoritmos de mineração de fluxos contínuos é extrair conhecimento de itens contínuos de dados que chegam ao longo do tempo (KRAWCZYK et al., 2017). Algoritmos de fluxos contínuos são especialmente projetados para aprender de forma *online*, uma instância de cada vez. Usando esta ideia, os modelos de decisão criados com esta abordagem são atualizados quando novas instâncias se tornam disponíveis, sem a necessidade de retreinar o modelo do início. Como pacotes de rede chegando em um IDS podem ser vistos como fluxos de dados, aplicar mineração de fluxos contínuos ao problema de detecção de intrusão parece ser uma alternativa promissora. Faisal et al. (2014) e Viegas et al. (2019) propuseram sistemas de detecção de intrusão baseado em mineração de fluxo. Contudo, em ambos os trabalhos a detecção de *botnets* não é discutida. Esse assunto foi abordado de maneira específica mas ainda superficial nos trabalhos propostos por Costa et al. (2018) e Khanchi, Zincir-Heywood e Heywood (2018). Questões relevantes relacionadas a implementação de tais sistemas em cenários mais próximos de um ambiente real, como a entrega do rótulo para o classificador e a avaliação do modelo ao longo do fluxo, não foram devidamente abordadas em tais trabalhos.

Baseado no exposto anteriormente, o objetivo deste trabalho consiste em avaliar o uso de algoritmos de mineração de fluxos considerando requisitos mais próximos aos cenários reais do problema de detecção de *botnets*. Esta avaliação se dará por meio da execução de diferentes algoritmos de classificação utilizando conjuntos de dados com fluxos de *botnets*, e consequente comparação do desempenho obtido na detecção de tais fluxos.

1.1 Motivação

A análise de fluxos contínuos tem se tornado rapidamente uma área crucial nas pesquisas de mineração de dados, pois é crescente o número de aplicações que precisam deste tipo de processamento (BIFET G. HOLMES; GAVALDà, 2009). Exemplos de tais fluxos contínuos incluem tráfego de rede, registros de chamadas telefônicas, transações financeiras, dados de sensores e de monitoramento via vídeo, etc. (WANG et al., 2003). O paradigma de mineração de fluxos contínuos de dados vem sendo adotado cada vez mais na solução de problemas de segurança da informação, como por exemplo, no desenvolvimento de sistemas de detecção de intrusão ((COSTA et al., 2018), (FAISAL et al., 2014), (VIEGAS et al., 2019), (WANG et al., 2003), entre outros).

Em cenários com fluxos contínuos, os dados podem chegar rapidamente e a partir de múltiplas fontes de dados (GROSSI; TURINI, 2012). Isso impõe uma restrição de que os algoritmos de mineração de dados verifiquem os dados apenas uma vez. Logo, é desafiador ter um modelo de classificação dos dados eficiente e que mantenha o controle dos dados que já passaram no fluxo (GROSSI; TURINI, 2012). Portanto, gerenciar os recursos computacionais é um problema crítico para o processamento de fluxos contínuos de dados. A *Hoeffding Tree*, também conhecida como VFDT (*Very Fast Decision Tree*) é um dos algoritmos mais utilizados em mineração de fluxos contínuos devido ao seu bom desempenho no processamento dos fluxos. O trabalho proposto por Costa et al. (2018) mostra como tal algoritmo pode ser utilizado na detecção de *botnets*. Contudo, o paradigma de aprendizado de máquina em que vários classificadores são treinados para resolver o mesmo problema, conhecido como conjunto de classificadores ou *ensemble*, surge como uma estratégia interessante para melhorar o desempenho dos modelos de decisão. De acordo com Bifet G. Holmes e Gavaldà (2009), os conjuntos de classificadores, quando aplicados em fluxos contínuos de dados, tem várias vantagens sobre os classificadores únicos: são fáceis de escalar e paralelizar, se adaptam rapidamente a mudanças de comportamento e normalmente geram modelos com maior precisão.

Ao contrário dos algoritmos tradicionais de aprendizado de máquina, os quais são baseados em aprendizado a partir de conjuntos estáticos de dados (ou lotes), os algoritmos de mineração de fluxos contínuos são projetados para lidar com aprendizado incremental sem armazenar todos os dados, processando os dados como um fluxo contínuo e infinito (COSTA et al., 2018). Durante o aprendizado para classificar os dados de um fluxo contínuo, obter os rótulos verdadeiros das instâncias pode requerer um esforço maior e até um custo excessivo. O aprendizado ativo é uma sub-área do aprendizado de máquina, cujo objetivo é alcançar maior desempenho utilizando o menor número possível de instâncias rotuladas (SETTLES, 2010). Sendo assim, o custo de se obter dados rotulados é minimizando (ŽLIOBAITĖ et al., 2011).

Costuma-se avaliar o desempenho de algoritmos de classificação com base em medidas simplesmente calculadas ao fim do fluxo, tais como precisão, revocação, taxa de falsos

positivos, etc. Entretanto, considerando essa característica incremental de aprendizado dos algoritmos de mineração de fluxos contínuos, é importante que a avaliação também seja dada dessa forma, considerando uma medida calculada ao longo do fluxo. Exemplos de medidas que seguem essa abordagem incremental seriam: o tempo gasto (ou quantidade de instâncias requeridas) pelo modelo para detectar um novo ataque e a análise de momentos (ou janelas de tempo) onde o modelo atinge valores máximos/mínimos de suas métricas de desempenho.

Trabalhos como Costa et al. (2018) e Pektaş e Acarman (2018) consideram o problema de classificação de fluxos contínuos como uma tarefa de classificação binária, baseada em duas classes (ataque e normal). Contudo, o tráfego de rede associado a *botnets* pode ser agrupado em diferentes tipos de fluxos, como a comunicação dos *bots* com o C&C e os ataques que estão sendo disparados pelos *bots*. Essa diversidade de fluxos que compõe o tráfego precisa ser considerada durante a fase de treinamento e atualização do modelo de decisão, para tentar entender quais fluxos impactam no desempenho do modelo. Entretanto, um classificador que é capaz de diferenciar entre múltiplas classes é difícil de se conseguir, ao contrário de um classificador binário que é bem mais simples (GOMES et al., 2017). Um outro problema nesse sentido é a ausência de conjuntos de dados que fornecem os rótulos dos diferentes tipos de tráfego de uma *botnet*. Um trabalho que busca mitigar essa limitação é o conjunto de dados IoT-23 (PARMISANO; ERQUIAGA, 2020).

Portanto, a aplicação da mineração de fluxos contínuos de dados mostra-se uma alternativa promissora para a construção de novos modelos de detecção de *botnet*. Contudo, ela ainda precisa ser analisada sob uma ótica mais abrangente, seja utilizando diferentes tipos de algoritmos de classificação e conjuntos de dados e investigando outras estratégias para avaliação dos modelos, como o número de exemplos rotulados. Outros pontos relevantes envolvem (i) buscar a maximização do desempenho de classificação minimizando o número de instâncias rotuladas, (ii) entender o impacto da distribuição das classes dos conjuntos de dados nos resultados gerados pelos modelos, e (iii) utilizar métricas relacionadas ao aprendizado incremental para comparação dos resultados.

1.2 Objetivos

O objetivo principal deste trabalho é avaliar o uso de algoritmos de mineração de fluxos contínuos de dados para a detecção de *botnets* considerando requisitos mais próximos aos cenários reais. Ou seja, além de investigar o desempenho dos modelos de classificação de tráfego utilizando medidas tradicionais, como a taxa de acerto, o intuito desta dissertação é mostrar que técnicas de aprendizado ativo podem ser usadas para diminuir a quantidade de amostras rotuladas para atualizar o classificador, mantendo o mesmo desempenho preditivo. É importante notar que este é um requisito fundamental para a implantação de sistemas de detecção de ameaças em ambientes reais, dado que o conhecimento sobre

o tráfego de rede é limitado e a elaboração de rótulos é uma tarefa que demanda muito tempo.

Os objetivos específicos do trabalho podem ser organizados da seguinte forma:

- ❏ Comparar o desempenho entre classificadores únicos e conjuntos de classificadores (*ensemble*) para detecção de *botnets*;
- ❏ Examinar o desempenho dos algoritmos de classificação utilizando medidas calculadas ao longo do fluxo;
- ❏ Avaliar se estratégias de aprendizado ativo podem contribuir para diminuir o número de instâncias rotuladas para atualizar o modelo de decisão sem perder desempenho preditivo;
- ❏ Investigar o desempenho de classificadores considerando os diferentes tipos de tráfegos maliciosos de uma *botnet*.

1.3 Hipótese

As hipóteses deste trabalho são as seguintes:

1. Conjuntos de classificadores superam o desempenho dos classificadores únicos com relação a precisão, revocação e número de instâncias rotuladas necessárias para atualização do modelo.
2. O uso de medidas de desempenho calculadas ao longo do fluxo de dados, ao invés de uma única avaliação ao fim do fluxo, são mais adequadas para a avaliação do modelo de decisão pois fornecem mais detalhes.
3. O uso de aprendizado ativo com conjuntos de classificadores mantém alto desempenho enquanto diminui a necessidade de instâncias rotuladas para atualização do modelo.
4. Modelos de decisão treinados com vários classes permitem a identificação dos tipos de tráfego associados a uma *botnet* que impactam no desempenho do modelo.

1.4 Organização da Dissertação

A seguir é detalhada a organização desta dissertação. O Capítulo 2 trata da fundamentação teórica, abordando conceitos como segurança de computadores, mineração de dados, mineração de fluxos contínuos, atualização do modelo de decisão e medidas de avaliação. Já o Capítulo 3 apresenta os principais trabalhos relacionados ao problema de detecção de *botnets*, bem como os conjuntos de dados CTU-13 e IoT-23. O Capítulo 4

traz a descrição do Processo Iterativo para Classificação de Fluxos e também os cenários que foram definidos. No Capítulo 5 são apresentados os detalhes sobre os conjuntos de dados, os atributos e medidas de avaliação utilizadas, além dos experimentos, resultados e análises realizadas conforme os cenários definidos no Capítulo 4. Por fim, o Capítulo 6 conclui esta dissertação com um resumo do trabalho realizado, as principais contribuições e sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentados os conceitos fundamentais deste trabalho. Estes conceitos foram divididos em cinco seções: 2.1 *Segurança de computadores*, que aborda a definição deste termo e seus conceitos principais; 2.2 *Botnets*, que descreve a estrutura, funcionamento e tipos de *botnet*; 2.3 *Mineração de dados*, que detalha seus conceitos básicos, tarefa de classificação e os métodos *ensemble*; 2.4 *Mineração de fluxos contínuos*, com sua definição, características principais, modelos de classificação *single* e *ensemble*, e também os algoritmos utilizados neste trabalho; e 2.5 *Atualização do modelo de decisão*, que aborda as forma de aprendizado de máquina, o aprendizado ativo e suas estratégias.

2.1 Segurança de computadores

A abertura da Internet para uso comercial no início dos anos 90 aumentou a importância das políticas de segurança para transações remotas. Dados sensíveis (como senhas, informações das transações e dos próprios usuários, etc.) precisaram ser protegidos em trânsito, e os nós da Internet tiveram que ser protegidos contra acessos indesejáveis (GOLLMANN, 2010).

A infraestrutura de redes, roteadores, servidores de domínio (DNS - *Domain Name System*), e comutadores que ligam estes sistemas, não pode falhar, caso contrário os computadores não serão mais capazes de se comunicar de forma precisa ou confiável.

O termo "segurança de computadores" é definido pelo NIST (*National Institute of Standards and Technology*) (NIELES; DEMPSEY; PILLITTERI, 2017) como: "A proteção proporcionada a um sistema de informação automatizado para atingir os objetivos aplicáveis de preservar a integridade, disponibilidade, e confidencialidade dos recursos do sistema de informação." (STALLINGS, 2014). A segurança de computadores abrange conceitos e métodos para proteger recursos sensíveis dos sistemas, e começa partindo das políticas que regulam o acesso a recursos protegidos. O controle para impedir acessos não autorizados a sistemas e recursos é o principal serviço de segurança de forma a garantir confidencialidade, isto é, impedir a divulgação não autorizada de informação, e

integridade, isto é, impedir a modificação não autorizada de informação. Mecanismos de controle de acesso definem como um sujeito (processo) pode acessar um recurso (objeto). O controle de acesso pode ser dividido em três tarefas principais: (GOLLMANN, 2010)

- A configuração da política de segurança: diretores são autorizados a acessar certos recursos. Tradicionalmente, diretores estavam intimamente relacionados com usuários humanos, mas esta visão é muito restritiva para políticas de segurança contemporâneas baseadas em evidências.

- A autenticação da evidência fornecida com uma requisição de acesso. Tradicionalmente, o diretor que faz a requisição foi autenticado.

- A avaliação da requisição em relação a uma dada política. A literatura de controle de acesso se refere a essa tarefa como autorização (da requisição).

Os três conceitos que formam a base da segurança de computadores são detalhados a seguir (STALLINGS, 2014):

- **Confidencialidade**: assegurar que informação privada ou confidencial não esteja disponível ou seja divulgada a indivíduos não autorizados (confidencialidade de dados). Além disso, garantir que os indivíduos controlem ou influenciem quais informações relacionadas a eles podem ser coletadas e armazenadas, e por quem e para quem essas informações podem ser divulgadas (privacidade). Uma perda de confidencialidade é a divulgação não autorizada de informações.

- **Integridade**: garantir que os programas e informações são alterados apenas de forma especificada e autorizada (integridade de dados). Também deve garantir que um sistema execute sua função pretendida de maneira intacta, livre de manipulação deliberada, inadvertida ou não autorizada (integridade do sistema). Uma perda de integridade é a modificação não autorizada ou destruição de informações.

- **Disponibilidade**: assegurar que os sistemas funcionem prontamente e o serviço não seja negado a usuários autorizados. Uma perda de disponibilidade é a interrupção do acesso ou uso de informações ou de um sistema de informações.

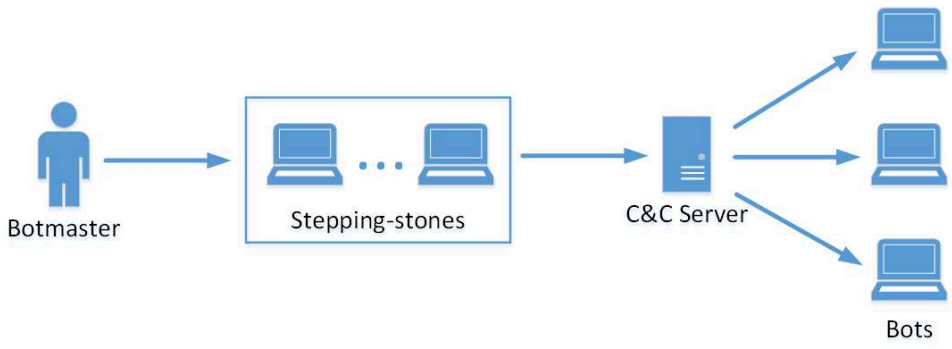
Alguns autores da área de segurança acreditam que conceitos adicionais são necessários para apresentar um quadro completo sobre o assunto. Os mais comumente mencionados são: *autenticidade*, que é a característica de ser genuíno, apto a ser verificado e confiável; e *responsabilização*, que está relacionado ao requisito de que as ações de uma entidade sejam rastreadas exclusivamente a essa entidade (STALLINGS, 2014).

2.2 *Botnets*

Uma *botnet* é um conjunto de *bots*, que são dispositivos interconectados via Internet, infectados por *malware*, e comandados por um criador, o *botmaster* (NOGUEIRA, 2016). Através de ataques coordenados, *botnets* são usadas em várias atividades ilegais; por exemplo, lançando ataques distribuídos de negação de serviços (*Distributed Denial Of Service - DDoS*), e obtendo lucro com exploração de informação de usuários normais (NOGUEIRA, 2016). A preparação de um ataque envolve recrutar, explorar e infectar os dispositivos alvo para que se tornem parte de uma *botnet* (MIRKOVIC; REIHER, 2004).

A Figura 1 mostra a arquitetura típica de uma *botnet*. Em tal cenário, existem os elementos funcionais que são: o *botmaster*, entidade que tem a capacidade de gerenciar toda a estrutura de rede maliciosa; seus *bots*, que compreendem os dispositivos comprometidos e integrantes da *botnet*, mas que pode ser definido também como um programa de computador instalado em uma máquina a qual permite que um atacante (*botmaster*) execute comandos arbitrários no sistema infectado; a infraestrutura de comando e controle (C&C) (elemento crítico na arquitetura de uma *botnet* (CERON, 2010)) que permite que comandos sejam enviados às máquinas comprometidas; e um protocolo de comunicação que é usado para enviar mensagens para os *bots*, por exemplo, HTTP, IRC e P2P (CERON, 2010). Os *stepping-stones* mostrados na figura são máquinas comprometidas usadas para ocultar a relação do *botmaster* com o servidor C&C.

Figura 1 – Arquitetura de uma *botnet*. Adaptado de (KHATTAK; RAMAY; KHAYAM, 2014).



Cada máquina comprometida apresenta um conjunto de funcionalidades pré-programadas que podem ser invocadas remotamente pelo *botmaster*. Além disso, os *bots* possuem, na maioria dos casos, a opção de atualização remota. Com isso, novas funcionalidades podem ser incorporadas à rede sem a necessidade de renovação da estrutura ou do comprometimento de novas máquinas (CERON, 2010).

A atividade das *botnets* é responsável por grandes perdas financeiras de Provedores de Serviço da Internet (do inglês *Internet Service Provider* - ISP), empresas, governos e usuários. A motivo primário de uma *botnet* é para o criminoso controlador, grupo

de criminosos ou sindicato do crime organizado usar computadores sequestrados para atividades *online* fraudulentas (SILVA et al., 2013).

Assim que os *bots* estabelecem conexão com os servidores C&C, é possível desenvolver mecanismos para investigar as características desse tráfego de rede e identificar os componentes da *botnet* (SILVA et al., 2013). Técnicas de detecção baseadas em anomalias que utilizam algoritmos convencionais de aprendizado de máquina são comumente aplicadas para aprender tais padrões de tráfego de rede e detectar uma peça maliciosa deste tráfego.

Apesar da existência de diversas técnicas para detecção de *botnets*, (SILVA et al., 2013) elenca diversos fatores que dificultam a implementação de tais métodos:

- tráfego de *botnet* é similar ao tráfego normal e, em alguns casos, pode estar cifrado;
- *botnets* estão evoluindo rapidamente à medida que cada vez mais usuários falham em proteger suas máquinas. *Botmasters* também usam técnicas para evitar mecanismos de detecção, como hospedagem *fast-flux* (relacionado a mudar frequentemente o endereço IP associado ao DNS do servidor C&C) (NAZARIO; HOLZ, 2008);
- normalmente é necessário analisar uma grande quantidade de dados, o que é difícil de realizar em tempo real. Isto torna a detecção em redes de grande escala uma tarefa complexa.

2.2.1 Arquiteturas e modo de operação

Botnets podem ser classificadas conforme topologia. As principais topologias abordadas são:

- Centralizada - é similar ao modelo de rede clássico, o cliente e servidor. Nela todos os *bots* estabelecem seu canal de comunicação com um, ou poucos, pontos de conexão, que normalmente são os servidores de C&C (SILVA et al., 2013). *Botnets* que utilizam os protocolos de comunicação HTTP e IRC são típicos exemplos com topologia centralizada (KHATTAK; RAMAY; KHAYAM, 2014). Suas principais vantagens são rápido tempo de reação e boa coordenação. Entretanto, o principal problema das *botnets* HTTP e IRC é que nelas o servidor C&C sozinho é o ponto central de falha, e podem ser mais facilmente detectadas pelo mesmo motivo, os *bots* conectam todos em um mesmo ponto (KHATTAK; RAMAY; KHAYAM, 2014) e (BAILEY et al., 2009).
- Descentralizada - comumente é baseada no protocolo P2P. É bem mais difícil de desarticular pois descobrir muitos *bots* não necessariamente implica na perda da *botnet* inteira visto que não existe um único servidor C&C central para encontrar e desabilitar (BAILEY et al., 2009) e (SILVA et al., 2013). Uma das variações da topologia decentralizada seria a randômica, em que não há uma relação clara entre

mestre-escravo. Qualquer *bot* pode ser utilizado para emitir comandos para outros *bots* dentro da *botnet*.

- ❏ Híbrida - é uma combinação das topologias centralizada e decentralizada (SILVA et al., 2013) e (KHATTAK; RAMAY; KHAYAM, 2014). Como exemplo há o uso de protocolo P2P com *superpeers* (SILVA et al., 2013) (é um nó em uma rede P2P que opera tanto como um servidor para um conjunto de clientes, quanto como um igual em uma rede de *superpeers* (YANG; GARCIA-MOLINA, 2003)).

Atualmente existem duas classes principais de *botnets* que se diferenciam na forma como os *bots* se comunicam e recebem comandos do *botmaster*, são as tradicionais e as móveis (NOGUEIRA, 2016). Com relação a *botnets* móveis, ainda existe a vantagem de utilizar a rede de comunicação móvel (wifi, SMS, etc.).

2.2.2 Técnicas de detecção

Segundo Silva et al. (2013), a detecção de *botnets* deve ser uma das primeiras ações que devem ser tomadas para combater ameaças a segurança de rede. Dado o potencial das *botnets* em conduzir diferentes atividades maliciosas, as técnicas de detecção tem um importante papel neste processo. As técnicas de detecção pode ser classificadas em duas categorias principais: com base em configuração de *honeynets*, ou com uso de Sistemas de Detecção de Intrusão (do inglês *Intrusion Detection System* - IDS). A categoria de IDSs foi subdivida em sistemas baseados em assinatura e sistemas baseados em anomalia ((SILVA et al., 2013), (KHATTAK; RAMAY; KHAYAM, 2014) e (GARCÍA; ZUNINO; CAMPO, 2014)).

Honeynets são mais adequadas para coletar informações de *bots*, obter binários de *bots* e infiltrar uma *botnet*. Após coletar informações, é possível aprender e entender a tecnologia utilizada, e realizar uma análise completa das principais características da *botnet*. Entretanto, não conseguem capturar *bots* que não usem métodos de propagação e podem apenas reportar informações das máquinas infectadas que foram colocadas como armadilha (SILVA et al., 2013).

Por outro lado, IDSs baseados em assinatura utilizam as assinaturas dos *bots* atuais no sistema de detecção. A ideia básica é extrair informação dos pacotes do tráfego monitorado, marcar tais padrões e registrá-los em um banco de dados de conhecimento dos *bots* existentes. Entretanto, esta abordagem consegue detectar apenas *botnets* já conhecidas, não sendo funcional para *bots* desconhecidos. Além disso, a base de conhecimento deve sempre ser atualizada. Já a detecção baseada em anomalia pode ser considerada a principal área de pesquisa para técnicas de detecção. A ideia é considerar várias anomalias de tráfego de rede na detecção de *botnets* (SILVA et al., 2013). Tais anomalias podem ser definidas com o auxílio de diversas características como alta latência de rede, volume alto de tráfego, tráfego em portas incomuns e comportamento incomum do sistema (SILVA et

al., 2013). Tais características auxiliarão a criação de modelos de classificação de tráfego. Técnicas de aprendizado de máquina costumam ser empregadas para o desenvolvimento de tais modelos. A seções 2.3 e 2.4 fornecem mais detalhes sobre esse processo.

As técnicas de detecção baseadas em anomalia podem ainda ser subdividas em baseadas em *host* e baseadas em rede. Na abordagem baseada em *host* uma máquina individual (*host*) é monitorada para encontrar qualquer comportamento suspeito. Apesar de sua importância, esta abordagem não é facilmente escalável pois todas as máquinas da rede precisam ter a ferramenta de monitoramento instalada para ser efetiva (SILVA et al., 2013). A abordagem baseada em rede usa informações derivadas do tráfego de rede para detectar os elementos da *botnet* (KHATTAK; RAMAY; KHAYAM, 2014).

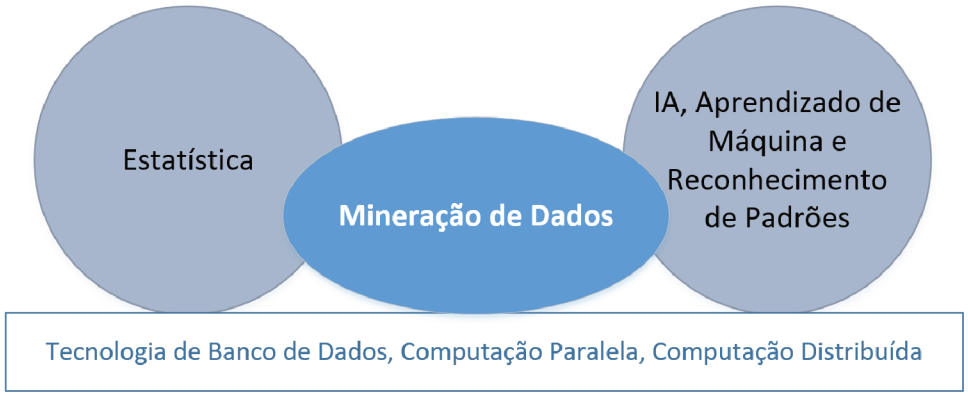
Entretanto, as novas gerações de *botnets* aproveitam dos avanços nas tecnologias e comunicação, aumentando sua velocidade de adaptação para protocolos de comunicação, políticas de segurança, e também seus alvos. Esses fatores criam vários desafios para prevenir e detectar a ameaça das *botnets*. *Botnets* são consideradas alvos em movimento, o que significa que todos os aspectos relacionados a *botnets* incluindo detecção, mitigação e resposta, estão mudando ao longo do tempo. Não existem técnicas de mitigação ou detecção que ofereçam uma solução permanente. De forma similar, diferentes tipos de *stakeholders*, por exemplo, empresas, governos, redes, e ISPs, empregam diferentes formas e objetivos para endereçar o problema das *botnets* (KARIM et al., 2014). Portanto, estudar novas formas de detectar e mitigar botnets assim como melhorar a compreensão acerca da aplicabilidade das técnicas já existentes são relevantes problemas de pesquisa.

2.3 Mineração de Dados

Mineração de dados é o processo para descoberta automática de informações úteis em grandes repositórios de dados (TAN; KUMAR, 2006). As técnicas de mineração de dados são implementadas para vasculhar grandes bancos de dados de forma a encontrar novos padrões e também possibilitar prever o resultado de futuras observações (TAN; KUMAR, 2006). Mineração de dados é baseada em conceitos de estatística (amostragem, estimativa e teste de hipóteses) e de inteligência artificial, reconhecimento de padrões e aprendizado de máquina (algoritmos de busca, técnicas de modelagem e teorias de aprendizado). A Figura 2 mostra o relacionamento da mineração de dados com as outras áreas (TAN; KUMAR, 2006).

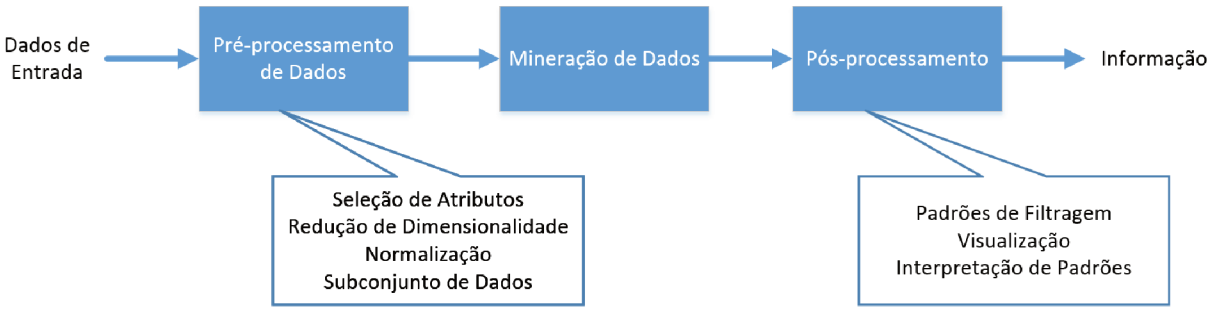
O principal desafio da mineração de dados é ter a habilidade de continuamente manter um modelo de decisão preciso mesmo em situações em que os dados são gerados de forma aleatória ou mudem de comportamento ao longo do tempo (GAMA, 2010). Esse é um cenário típico dos fluxos de *botnets*, pois o tráfego muda conforme as instruções lançadas pelo *botmaster*, por exemplo, na tentativa de ofuscar a comunicação entre os *bots* e o C&C na rede.

Figura 2 – Mineração de dados e sua relação com várias disciplinas. Adaptado de (TAN; KUMAR, 2006).



A mineração de dados está diretamente relacionada ao processo de descoberta de conhecimento em banco de dados, também conhecido como *Knowledge Discovery in Databases* – KDD. A Figura 3 ilustra o KDD. Ele consiste em uma série de passos de transformação, do pré-processamento dos dados até o pós-processamento dos resultados da mineração de dados (TAN; KUMAR, 2006). A fase de pré-processamento será apresentada na subseção 2.3.1.

Figura 3 – O processo de KDD (*Knowledge Discovery in Databases*). Adaptado de (TAN; KUMAR, 2006).



Ainda seguindo o processo definido pelo KDD, logo após o pré-processamento deve iniciar-se o processo de mineração de dados. As tarefas relacionadas à mineração de dados são geralmente divididas em duas categorias principais:

- Tarefas preditivas: o objetivo dessas tarefas é prever o valor de um atributo específico baseado nos valores de outros atributos. O atributo a ser previsto é normalmente conhecido como alvo, variável dependente ou classe, enquanto que os atributos utilizados para realizar a previsão são conhecidos como explicativos ou variáveis independentes.

- Tarefas descritivas: o objetivo é derivar padrões (correlações, tendências, *clusters*, trajetórias e anomalias) que sumarizem os relacionamentos subjacentes nos dados.

Tarefas descritivas de mineração de dados são frequentemente exploratórias por natureza e frequentemente exigem técnicas de pós-processamento para validar e explicar os resultados.

Considerando as tarefas preditivas, existem dois tipo de tarefas de modelo preditivo: (i) a classificação, que é utilizada para variáveis alvo discretas, e (ii) regressão, que é utilizada para variáveis alvo contínuas. A tarefa de classificação, abordada na subseção 2.3.2, é muito importante para detecção de *botnets* porque a partir de atributos específicos (do tráfego de rede, das máquinas, etc.) é possível prever a classe (ou classes) que esteja relacionada a um ataque.

Já o pós-processamento assegura que apenas resultados válidos e úteis sejam incorporados ao modelo de decisão. Exemplos de pós-processamento incluem filtragem, visualização e interpretação de padrões.

2.3.1 Pré-processamento dos dados

Dentro da etapa de pré-processamento dos dados, são abordados os passos que devem ser aplicados para tornar os dados mais adequados para a etapa seguinte. O objetivo é melhorar a análise em termos de tempo, custo e qualidade. O pré-processamento dos dados é uma área vasta que consiste em uma grande variedade de técnicas e estratégias que são inter-relacionadas. De forma geral, estas técnicas e estratégias podem ser divididas em duas categorias: (i) selecionar objetos de dados e atributos para a análise, ou (ii) criar/mudar os atributos.

Os principais tópicos citados por (TAN; KUMAR, 2006) são:

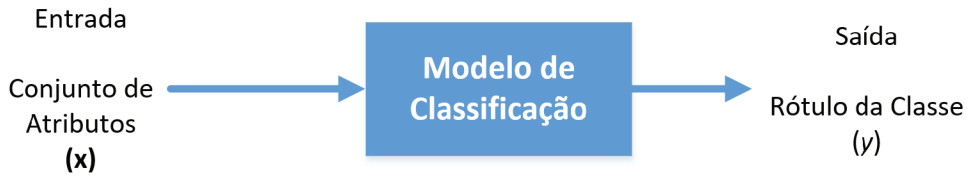
1. **Agregação**: combinar dois objetos ou mais em um único objeto;
2. **Amostragem**: selecionar um subconjunto de objetos de dados para análise;
3. **Redução de dimensionalidade**: conjuntos de dados podem ter uma grande quantidade de atributos, sendo interessante selecionar apenas os mais relevantes;
4. **Seleção de subconjuntos de atributos**: outra forma de reduzir a dimensionalidade, utilizando apenas um subconjunto dos atributos;
5. **Criação de atributos**: criar, a partir dos atributos originais, um novo conjunto de atributos que capturem as informações mais relevantes;
6. **Discretização e binarização**: transformar um atributo contínuo, que pode assumir qualquer valor, em um atributo categórico, ou discreto;
7. **Transformação de atributos**: aplicar uma transformação no valor de um atributo (como aplicar funções ou normalização).

No método deste trabalho são abordados os tópicos 4. e 5.

2.3.2 Classificação

Classificação é a tarefa de aprender uma função alvo f que mapeia cada conjunto de atributos x a uma classe de rótulos pré-definidos y (TAN; KUMAR, 2006). Conforme mostrado na Figura 4, um modelo de classificação é uma "caixa preta" que recebe um conjunto de atributos de um registro desconhecido e atribui automaticamente um rótulo (TAN; KUMAR, 2006).

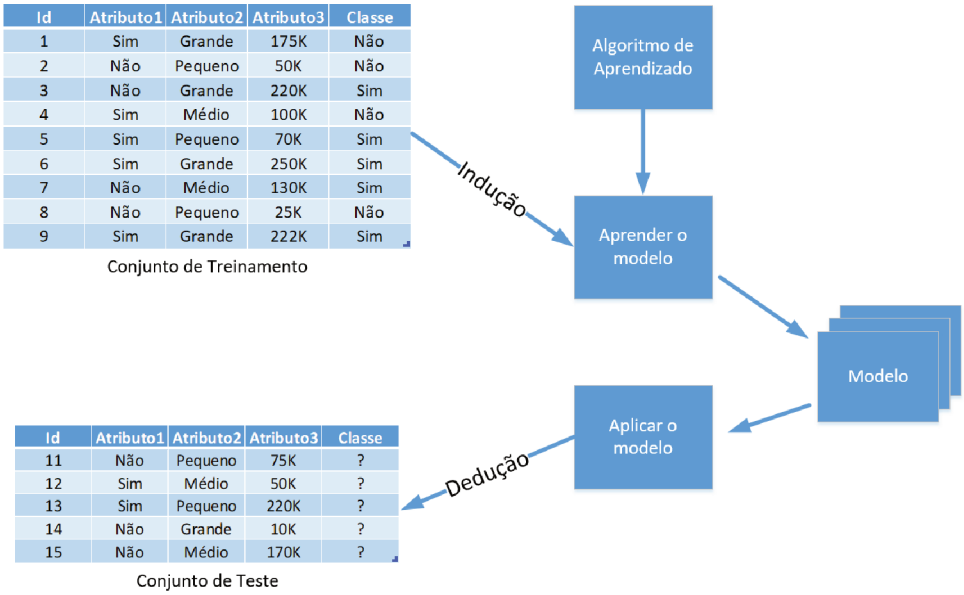
Figura 4 – Representação da tarefa de Classificação. Adaptado de (TAN; KUMAR, 2006).



Uma técnica de classificação é uma abordagem sistemática para construir modelos de classificação a partir de conjunto de dados de entrada (TAN; KUMAR, 2006). Exemplos de algoritmos de classificação incluem árvores de decisão, baseados em regras, redes neurais, *support vector machines* e *naive Bayes* (TAN; KUMAR, 2006). Cada técnica emprega um algoritmo de aprendizado para identificar um modelo que melhor se ajusta ao relacionamento entre o conjunto de atributos e o rótulo de classe do dado de entrada (TAN; KUMAR, 2006). O modelo gerado pelo algoritmo de aprendizado deve de forma conjunta se ajustar bem aos dados de entrada e prever corretamente os rótulos de classe de instâncias nunca antes vistos. Portanto, o objetivo principal do algoritmo de aprendizado é construir modelos com boa capacidade de generalização, isto é, modelos que retornem com precisão os rótulos de classes de instâncias anteriores que são desconhecidos (TAN; KUMAR, 2006). A Figura 5 mostra uma abordagem geral para resolver problemas de classificação. Primeiro, é fornecido um conjunto de treinamento (instâncias cujo rótulo de classe é conhecido). O conjunto de treinamento é utilizado para construir um modelo de classificação, que pode ser posteriormente aplicado em um conjunto de teste (instâncias cujo rótulo de classe não é conhecido).

Os erros cometidos por um modelo de classificação são geralmente divididos em dois tipos: erros de treinamento e erros de generalização. Erros de treinamento, também conhecidos como erro aparente, é número de erros de classificação (*misclassification*) cometidos em instâncias de treinamento, enquanto que erro de generalização é o erro esperado do modelo em instâncias anteriores não vistas. Um bom modelo de classificação não deve somente se ajustar bem aos dados de treinamento, mas também classificar com precisão as instâncias nunca antes vistas, ou seja, deve ter um erro de treinamento baixo e também um baixo erro de generalização. Isto é importante pois um modelo que se ajuste bem demais aos dados de treinamento pode ter um erro de generalização mais pobre que um modelo com alto valor para erros de treinamento. Esta situação é conhecida como *overfitting*.

Figura 5 – Abordagem geral para construir um modelo de classificação. Adaptado de (TAN; KUMAR, 2006).



A avaliação do desempenho de um modelo de classificação é baseada na contagem das instâncias de teste que foram previstas corretamente e incorretamente. Essas contagens são tabuladas em uma tabela conhecida como matriz de confusão.

2.3.3 Conjunto de Classificadores - *Ensemble*

A ideia de aprendizado por *ensemble* é construir um modelo de previsão por meio da combinação dos pontos fortes de uma coleção de modelos básicos mais simples (HASTIE; TIBSHIRANI; FRIEDMAN, 2013). As técnicas de classificação *single* preveem os rótulos de classe de instâncias desconhecidos usando um único classificador induzido a partir dos dados de treinamento. Já com *ensembles* geralmente ocorre um melhoramento da acurácia de classificação pelo fato de agregar as previsões de múltiplos classificadores (TAN; KUMAR, 2006). Um método *ensemble* constrói um conjunto de classificadores base a partir de dados de treinamento e realiza a classificação compondo as previsões feitas por cada classificador base. Alguns algoritmos utilizados neste trabalho (e citados mais adiante) utilizam duas técnicas específicas baseadas em *ensemble* que são *bagging* e *boosting*:

- *Bagging*: é uma técnica que segue uma distribuição de probabilidade uniforme e coleta repetidamente uma amostra (com substituição) de um conjunto de dados. Cada amostra tem o mesmo tamanho que os dados originais.

- *Boosting*: é um procedimento iterativo usado para alterar de forma adaptativa a distribuição de exemplos de treinamento para que os classificadores base se concentrem em exemplos que são difíceis de classificar. Ao contrário do *bagging*, o *boosting*

atribui um peso a cada exemplo de treinamento e pode alterar o peso de forma adaptativa no final de cada rodada.

2.4 Mineração de fluxos contínuos de dados

Nas últimas décadas, a pesquisa e a prática de Aprendizado de Máquina tem focado no aprendizado em lote (*batch*). No aprendizado em lote, ou tradicional, todos os dados estão disponíveis para treinamento do algoritmo, o qual gera um modelo de decisão após processar os dados eventualmente (ou na maioria dos vezes) múltiplas vezes (GAMA, 2010). Assim, é gerado um único modelo de aprendizado e este não é atualizado com a chegada de novos dados (SILVA, 2018). Em um configuração tradicional de aprendizado em lote, o aprendizado é realizado com base em um conjunto de dados finito em que sua distribuição não é conhecida, ainda que estacionária. Assim, apesar da complexidade de aprender o modelo, após completar o treinamento, não há necessidade de atualizá-lo (GOMES et al., 2017).

Algoritmos baseados nesta abordagem não são capazes de lidar com dados gerados continuamente em ambientes dinâmicos, pois neste tipo de ambiente, é necessário que o algoritmo assimile de forma incremental novos dados ao modelo de aprendizado. Outra característica importante diz respeito à mudança na distribuição dos dados nestes ambientes, pois em muitas aplicações do mundo real, o processo de geração de dados não é estacionário. Portanto, o algoritmo deve possuir mecanismos que possam detectar estas mudanças de forma rápida e eficiente, de modo a manter o modelo sempre atualizado (SILVA, 2018). Os algoritmos de aprendizado em lote não são capazes de analisar de forma eficiente uma quantidade de dados que cresça rapidamente, considerando as seguintes limitações: recursos computacionais, tempo, instâncias rotuladas, e mudança de conceito (isto é, mudanças na distribuição dos dados que ocorrem em um fluxo à medida que o tempo passa) ((KRAWCZYK et al., 2017) e (GOMES et al., 2017)).

Um fluxo contínuo de dados pode ser visto como um sequência de itens de dados que chegam à medida que o tempo passa (KRAWCZYK et al., 2017), ou como um processo estocástico em que eventos ocorrem continuamente e independentemente um do outro (GAMA, 2010). As diferenças mais relevantes que fazem que o processamento de fluxos contínuos seja diferente dos modelos relacionais convencionais são (GAMA, 2010):

- ❏ Os elementos de dados em um fluxo contínuo chegam de forma contínua;
- ❏ O sistema não possui controle sobre a ordem em que os elementos de dados chegam;
- ❏ Fluxos contínuos são potencialmente ilimitados em tamanho;
- ❏ Uma vez que um elemento de um fluxo contínuo foi processado ele é descartado ou arquivado.

Mineração de fluxos contínuos é um processo dinâmico usado para lidar com um descobrimento automático de padrões de dados. Algoritmos de fluxos contínuos de dados são projetados para aprender de forma *online*, um instância de cada vez. Isto representa um desafio para o aprendizado de máquina e também para mineração de dados. Na Tabela 1 estão resumidas as diferenças entre processamento de dados tradicional e em fluxos contínuos.

Tabela 1 – Diferenças entre processamento de dados tradicionais e fluxos contínuos. Adaptado de (GAMA, 2010).

	Tradicional	**Fluxo Contínuo**
Número de execuções	Múltiplo	Único
Tempo de processamento	Ilimitado	Restrito
Uso de memória	Ilimitado	Restrito
Tipo de resultado	Preciso	Aproximado
Distribuído	Não	Sim

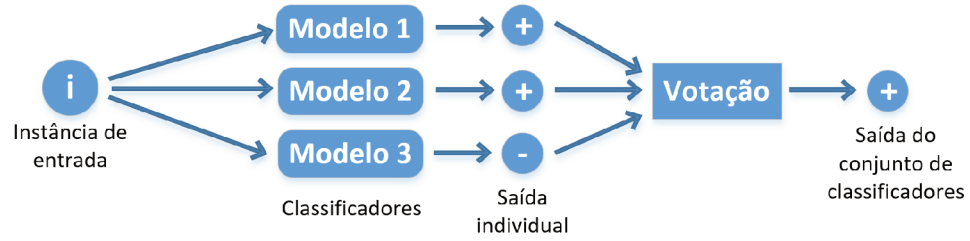
Vários cenários se beneficiam da análise de fluxos contínuos porque a geração automática de dados se tornou bem comum. Alguns exemplos são análise de redes, predição de dados financeiros, controle de tráfego de rede, processamento de sensores de medição, computação ubíqua, GPS, rastreamento de dispositivos móveis, mineração de logs de clique de usuários, análise de opinião (KRAWCZYK et al., 2017).

Uma das tarefas mais importantes para o aprendizado de máquina, referente a redes de computadores, é a classificação do tráfego. Classificação de fluxos contínuos de dados é uma variação da tarefa de classificação do aprendizado de máquina supervisionado baseado em lotes. O objetivo é prever o valor nominal a partir de um vetor de atributos. Alguns problemas precisam ser endereçados em cenários de fluxos contínuos. Primeiro, instâncias não estão prontamente disponíveis para o classificador pois fazem parte de um extenso conjunto de dados estático; ao invés disso, elas são fornecidas sequencialmente e rapidamente à medida que o tempo passa como um fluxo contínuo de dados. Segundo, devido ao ambiente não-estacionário, mudanças na distribuição dos dados precisam ser incorporadas no modelo de classificação, assim como os dados antigos precisam ser descartados (GOMES et al., 2017).

Existem duas abordagens principais para classificação de fluxos contínuos de dados (FARID et al., 2013): modelo de classificação único incremental (*single*) e modelo de classificação baseado em conjunto (*ensemble*). Técnicas de modelo de classificação único atualizam incrementalmente um único classificador com novos dados para acompanhar a evolução do fluxo contínuo de dados. Tais técnicas podem ter uma atuação ruim no ambiente de fluxos contínuos. Ao contrário, uma série de classificadores pode melhorar a composição do modelo para lidar de forma eficiente com mudança de conceito (vide Figura 6). Estes modelos baseados em conjunto (*ensemble*) podem ser mais eficientes se comparados a atualização de um único modelo. Eles também tem uma taxa de acurácia

para classificação maior que técnicas de classificação de modelo único (*single*) (FARID et al., 2013).

Figura 6 – Exemplo de um classificador baseado em conjunto (*ensemble*). Adaptado de (FARID et al., 2013)



A motivação primária deste trabalho para usar classificadores baseados em conjunto (*ensemble*) é sua abordagem atraente para construir classificadores para fluxos contínuos de dados, pois eles facilitam adaptação a mudanças na distribuição dos dados. *Online ensembles* aprendem separadamente cada instância de treinamento, ao invés de em lotes, e então as descarta. Fazendo isso, estas abordagens podem aprender os fluxos contínuos de dados em apenas uma passagem, sendo potencialmente mais rápidos e menos exigentes com relação a memória. Além disso, como as famílias de *botnets* podem apresentar características distintas e evoluir de forma diferente em um fluxo contínuo, os modelos treinados em porções diferentes dos fluxos e combinados em um conjunto de classificadores (*ensemble*) pode alcançar resultados melhores.

Os algoritmos de mineração de fluxos contínuos utilizados neste trabalho podem ser divididos em duas categorias de classificadores, *Single* e *Ensemble*. Classificadores *Single* incluem *Hoeffding Tree* ou *VFDT (Very Fast Decision Tree)* ((DOMINGOS, 2000) e (HULTEN; SPENCER; DOMINGOS, 2001)), e *Hoeffding Adaptive Tree* ((BIFET; GAVALDÀ, 2009)). Classificadores *Ensemble* incluem *Ozabag* ((OZA; RUSSEL, 2001)), *Ozaboost* ((OZA; RUSSEL, 2001)), e *Oza Bagging With Adwin* ((BIFET; GAVALDA, 2007) e (BIFET G. HOLMES; GAVALDà, 2009)).

2.4.1 Classificação única incremental

VFDT (Very Fast Decision Tree), ou *Hoeffding Tree*, é um tipo de árvore de decisão (DOMINGOS, 2000). Trata-se de um algoritmo de indução baseado em árvore de decisão que é incremental e independente de momento, capaz de aprender a partir de fluxos contínuos e massivos, assumindo que a distribuição quando os exemplos foram gerados não mude ao longo do tempo.*Hoeffding Trees* aproveitam do fato de que pequenas amostras podem frequentemente ser suficientes para escolher um atributo ótimo de divisão. A ideia é provada matematicamente pelo limite *Hoeffding* (*Hoeffding bound*), que quantifica o número de instâncias necessárias para estimar algumas estatísticas dentro de uma precisão prévia. Uma característica teoricamente atrativa das *Hoeffding Trees*, que não é comum

a outros algoritmos incrementais baseados em árvores de decisão, é a garantia de desempenho. Usando o *Hoeffding bound* é possível provar que o resultado é assintoticamente quase idêntico a de um algoritmo não-incremental usando infinitos exemplos (KIRKBY, 2020a).

A *Hoeffding Adaptive Tree* é uma versão melhorada da *Hoeffding Tree* porque adicionalmente lida com mudança de conceito (*concept drift*). Ela usa o método *ADWIN* para monitorar o desempenho dos ramos da árvore e trocá-los por novos com uma melhor acurácia (BIFET, 2020a), isto é, para reiniciar os membros *ensemble* quando sua acurácia preditiva degradar significantemente. *ADWIN* é um detector de mudança, no qual uma janela de exemplos de entrada cresce até que uma mudança no valor médio dentro da janela seja identificada (KRAWCZYK et al., 2017). Também não depende de parametrização, sendo que detecta automaticamente e continuamente a taxa de mudança nos fluxos contínuos, ao invés de usar suposições a priori, assim permitindo que o algoritmo reaja adaptativamente ao fluxo contínuo em processamento (BIFET; GAVALDÀ, 2009).

2.4.2 Classificação baseada em conjuntos

Em mineraçao de fluxos contínuos de dados, os métodos de *bagging* e *boosting* para conjunto de classificadores também foram adaptados. *Ozabag* e *Ozaboost* são exemplos de tais adaptações.

Com relação aos métodos *ensemble Ozabag* e *Ozaboost*, as técnicas *bagging* e *boosting onlines* e incrementais, são ambas versões *online* projetadas para corresponder aos algoritmos *batch* de *bagging* e *boosting*. São estratégias *ensemble* que usam o procedimento de ponderação de *AdaBoost* para dividir o peso total do exemplo em uma metade que é atribuída aos exemplos que foram corretamente classificados e em outra metade aos exemplos incorretamente classificados (KIRKBY, 2020b). O processo de amostragem de *Poisson* é usado para decidir a probabilidade aleatória de que um exemplo possa ser usado para treinamento e aproximar o algoritmo de reponderação. Este é o único momento em que o parâmetro muda de acordo com o peso do exemplo a medida que é passado em sequência por cada modelo (OZA; RUSSEL, 2001).

Oza Bagging with Adwin é o método *bagging online* de (OZA; RUSSEL, 2001) com a adição do algoritmo ADWIN (*Adaptive Sliding Window*) como um detector de mudança e como um estimador para os pesos do método *boosting*. De acordo com (KRAWCZYK et al., 2017), o método ADWIN pode também ser integrado a um *ensemble bagging online*. Quando uma mudança é detectada, o pior classificador do *ensemble* é removido, e um novo é adicionado (BIFET, 2020b).

2.4.3 *Prequential*

O método *Prequential* consiste em apresentar uma instância, sem rótulo, ao classificador que está sendo avaliado e solicitar uma previsão. Depois disso, a previsão é armazenada e usada para calcular as métricas de desempenho desejadas. Quando todas as estatísticas de desempenho são atualizadas, o verdadeiro rótulo dessa instância é apresentado ao classificador, que é então capaz de aprender com ele (COSTA et al., 2018). Neste trabalho foi empregada uma implementação do MOA para *Prequential*, a tarefa *ALPrenquentialEvaluationTask*. Trata-se de uma tarefa de classificação para aprendizado ativo (vide seção 2.5.1) implementada para avaliação de validação cruzada prequencial de um classificador em um fluxo contínuo, testando e treinando com cada instância em sequência e fazendo validação cruzada ao mesmo tempo. Combinado a essa tarefa foi utilizado o *ALWindowClassificationPerformanceEvaluator* também implementado pelo MOA, que é um avaliador de classificação que atualiza os resultados da avaliação usando uma janela deslizante.

2.5 Atualização do modelo de decisão

A maioria dos algoritmos propõe que classificadores de aprendizado para fluxos contínuos sigam a abordagem de ter acesso completo e imediato as classes de rótulos para todas as instâncias processadas. Entretanto, em algumas aplicações, a suposição de todas as instâncias de aprendizado rotuladas pode ser não realística ou inviável. As classes rotuladas de instâncias recém-chegadas em fluxos contínuos de dados não estão disponíveis imediatamente (KRAWCZYK et al., 2017). Conseguir os rótulos das instâncias pelos especialistas tem um custo alto, exigindo esforço substancial. Portanto alguns pesquisadores consideram outros *frameworks* tais como i) aprendizado com rotulagem atrasada; ii) aprendizado semi-supervisionado onde os rótulos não estão disponíveis para todas as instâncias que chegam; iii) um *framework* não supervisionado ou aprendizado a partir de instâncias rotuladas iniciais.

2.5.1 Aprendizado ativo

Aprendizado ativo (do inglês, *active learning*) é uma sub-área do aprendizado de máquina. A ideia principal é a de que um algoritmo de aprendizado de máquina pode alcançar alta acurácia com menos instâncias rotuladas para treinamento, sendo bem aplicado em problemas em que dados sem rótulo podem ser abundantes e obtidos facilmente, mas rótulos são difíceis, demorados, ou caros de se obter (SETTLES, 2010).

Aprendizado ativo estuda como rotular seletivamente ao invés de requisitar por todos os rótulos verdadeiros dos dados (ŽLIOBAITĖ et al., 2011). Foi extensivamente estudado nos cenários *pool-based* e *online*. No cenário *pool-based* a decisão sobre quais instâncias

rotular é feita a partir de todos os dados históricos (ŽLIOBAITĖ et al., 2011). Já no cenário *online* a decisão deve ser tomada imediatamente para cada instância de entrada, pois não há re-acesso.

A principal diferença entre aprendizado ativo *online* e aprendizado ativo em fluxos contínuos está nas expectativas com relação às mudanças (ŽLIOBAITĖ et al., 2011). Aprendizado ativo *online* tipicamente estabelece um limite (como um limite de incerteza) e requisita o verdadeiro rótulo se o limite é excedido. Em fluxos contínuos o relacionamento entre os dados de entrada e os rótulos podem mudar, e estas mudanças podem acontecer em qualquer posição no espaço das instâncias. Assim, as estratégias de aprendizado ativo existentes podem nunca requerer instâncias em algumas regiões e assim podem nunca saber que mudanças estão acontecendo e então nunca se adaptar. Além disso, em fluxos contínuos não é possível manter o limite de decisão ou uma região de incerteza fixos, pois eventualmente o sistema pode parar de aprender e falhar em reagir às mudanças. Finalmente, o aprendizado ativo com fluxos contínuos deve preservar a distribuição dos dados que chegam na medida que mudanças podem ser detectadas assim que acontecem (ŽLIOBAITĖ et al., 2011).

2.5.2 Estratégias de aprendizado ativo

Žliobaitė et al. (2011) propuseram estratégias de aprendizado ativo especificamente para fluxos contínuos com objetivo de maximizar a predição de acurácia ao longo do tempo, enquanto mantém o custo de rotulagem fixo dentro de uma capacidade (*budget*) previamente alocada. Dados chegam em um fluxo contínuo e as predições precisam ser feitas em tempo real. Mudança de conceito é esperada, assim o aprendizado precisa ser adaptativo. O rótulo verdadeiro pode ser requisitado imediatamente ou nunca, como as instâncias são regularmente descartadas da memória. Estratégias de aprendizado ativo em fluxos contínuos devem ser capazes de:

- ❏ ter uma classificação precisa em situações estacionárias e não-estacionárias;
- ❏ balancear a capacidade (*budget*) de rotulagem ao longo do tempo;
- ❏ notar mudanças que possam acontecer em qualquer ponto do espaço de instâncias;
- ❏ preservar a distribuição dos dados de entrada para detectar mudanças.

As estratégias de aprendizado ativo abordadas neste trabalho são (ŽLIOBAITĖ et al., 2011):

1. **Estratégia Aleatória (*Random Strategy*)**: estabelece a base para as outras estratégias. A estratégia randômica é ingênua no sentido de que as instâncias de entrada são rotuladas de forma randômica ao invés de decidir de forma ativa qual

rótulo seria mais relevante. Para cada instância de entrada o rótulo verdadeiro é requisitado com uma probabilidade B, onde B é a capacidade (*budget*).

2. **Estratégia de Incerteza Fixa (*Fixed Uncertainty Strategy*)**: Amostragem por incerteza talvez seja a estratégia de aprendizado ativo mais simples e comum. A ideia é rotular as instâncias para as quais o classificador atual é o menos confiável. Em uma configuração *online*, corresponde a rotular as instâncias para as quais a certeza está abaixo de algum limite fixo (*threshold*).

3. **Estratégia de Incerteza Variável (*Variable Uncertainty Strategy*)**: Um dos desafios das estratégias baseadas em incerteza, considerando um cenário de fluxo contínuo de dados, é como distribuir o esforço de rotulagem ao longo do tempo. Se um limite fixo é utilizado, depois de algum tempo, um classificador esgotará sua capacidade (*budget*) ou atingirá o limite de certeza. Em ambos os casos, ele irá parar de aprender e, portanto, não conseguirá se adaptar às mudanças. Em vez de rotular as instâncias que são menos certas do que o limite, seria melhor rotular as instâncias menos certas dentro de um intervalo de tempo. Assim, um limite variável é introduzido, o qual se ajusta dependendo dos dados recebidos para se alinhar ao *budget*. Se um classificador se torna mais certo (situações estáveis), o limite se expande para ser capaz de capturar as instâncias mais incertas. Se uma mudança acontecer e de repente aparecerem muitos pedidos de rotulagem, então o limite é contraído para consultar as instâncias mais incertas primeiro.

4. **Estratégia de Incerteza com Aleatorização (*Uncertainty Strategy with Randomization*)**: As estratégias baseadas em incertezas sempre rotulam as instâncias que estão próximas do limite de decisão do classificador. Em fluxos contínuos de dados, as alterações podem acontecer em qualquer lugar no espaço das instâncias. Quando ocorre uma mudança de conceito nos rótulos, o classificador não o notará sem os rótulos verdadeiros. Para não perder o desvio de conceito, de tempos em tempos é necessário rotular as instâncias sobre as quais o classificador tem muita certeza. Para esse propósito, para cada instância, o limite de rotulagem é randomizado multiplicando por uma variável aleatória normalmente distribuída que segue $N(1, \delta)$. Assim, as instâncias que estão frequentemente mais perto do limite de decisão são rotuladas, mas ocasionalmente algumas instâncias distantes também são rotuladas. Em situações estacionárias, espera-se que esta estratégia tenha um desempenho pior do que a estratégia de incerteza, mas em situações de mudança, espera-se que ela se adapte mais rapidamente.

2.6 Medidas de Avaliação

As métricas usadas nessa dissertação são precisão e revocação. Também foi utilizada a média aritmética entre precisão e revocação. Essas métricas podem ser calculadas da seguinte forma:

- $precisão = \frac{TP}{TP+FP}$: ou valor predito positivo, representa a porcentagem de fluxos classificados como *botnets* que são de fato botnets.
- $revocação = \frac{TP}{TP+FN}$: também chamada de sensibilidade, taxa de acerto ou taxa positiva verdadeira, que é a porcentagem de fluxos de *botnet* classificados corretamente.

Dadas as métricas acima, considerar o seguinte: TP (do inglês *true positives*, verdadeiros positivos), FP (do inglês *false positives*, falsos positivos), e FN (do inglês *false negatives*, falsos negativos). A classe positiva está relacionada à botnet e a negativa ao *background*.

Capítulo 3

Trabalhos Relacionados

O objetivo deste capítulo é apresentar o estado da arte em detecção de *botnets* utilizando aprendizado de máquina tradicional e mineração de fluxos contínuos de dados, abordados respectivamente nas Seções 3.1.1 e 3.1.2. Além disso, a Seção 3.2 discute as limitações dos conjuntos de dados usados para validar métodos de detecção de intrusão e apresenta os dois conjuntos usados no decorrer do trabalho: CTU-13 (GARCÍA et al., 2014) e IoT-23 (PARMISANO; ERQUIAGA, 2020).

3.1 Detecção de *botnets*

García, Zunino e Campo (2014) apontam limitações nos trabalhos analisados, como o uso de conjuntos de dados privados e a adoção de terminologia confusa e não padronizada. Dentre as razões para a existência de tais problemas estão a rápida evolução das *botnets* e a dificuldade em obter dados reais de tráfego de rede. De forma mais detalhada, os principais problemas nos trabalhos anteriores que propuseram métodos de detecção de *botnets* são:

- A maioria dos métodos não pode ser reproduzida pois as bases de dados não são públicas e não há informação suficiente sobre os métodos. Faltam a descrição de cada passo dos algoritmos, os parâmetros utilizados, etc.;

- Algumas propostas filtram excessivamente os dados para produzir algoritmos que trabalhem melhor com uma base de dados específica. Isto está relacionado ao pré-processamento dos dados de entrada, como por exemplo, filtrar pacotes de um protocolo específico sem explicações mais aprofundadas;

- Algumas propostas não descrevem claramente seus experimentos, métricas utilizadas e os resultados alcançados.

- Métodos supervisionados podem detectar *botnets* conhecidas, mas novas *botnets* ainda são difíceis de se detectar. Os métodos devem se adaptar às mudanças de comportamento das *botnets* e deveriam ser mais flexíveis.

Com esses problemas em mente, foi realizada uma revisão bibliográfica em trabalhos relacionados à detecção de *botnets* publicados à partir de 2013, ano em que o estudo de García, Zunino e Campo (2014) foi publicado. A revisão da literatura foi organizada em duas seções. Primeiramente, a Seção 3.1.1 apresentará os trabalhos cuja abordagem segue as técnicas tradicionais de inteligência artificial e aprendizado de máquina, sendo baseados principalmente em aprendizado em lote, com geração dos modelos utilizando mineração de dados estacionários. Já a Seção 3.1.2 discutirá como o aprendizado de máquina é aplicado na classificação de dados usando a abordagem de mineração de fluxos contínuos de dados.

A Tabela 2 sumariza os trabalhos discutidos neste capítulo e suas características conforme escopo abordado nesta dissertação. A seguir uma breve descrição de cada um dos atributos presentes na tabela:

- *Botnet* - indica se a referência trata especificamente de ameaças do tipo *botnet*;
- Processamento - mostra se o modelo de processamento dos dados é feito em lote ou em fluxos contínuos;
- Conjunto de dados - exibe o conjunto de dados público utilizado para conduzir os experimentos;
- Avaliação ao longo do fluxo - denota se a avaliação do modelo de decisão é feita ao longo do fluxo usando diferentes janelas de tempo;
- Aprendizado ativo - indica se é empregado algum modelo de aprendizado ativo;
- Avaliação de instâncias rotuladas - mostra se o trabalho investiga o impacto do número de instâncias rotuladas no desempenho do modelo;
- Multi-classe - demonstra se a referência considera outros tipos de tráfego em uma classificação multi-classe;
- *Ensembles* - indica se o trabalho utiliza conjuntos de classificadores (*ensembles*);

3.1.1 Detecção de *botnets* em um cenário tradicional (lote)

As técnicas de aprendizado por lotes geram um modelo a partir de conjunto de treinamento, e o avalia utilizando um conjunto de teste. A característica principal da técnica de processamento em lote é a necessidade de previamente armazenar os dados para seu

Tabela 2 – Trabalhos relacionados sobre detecção de *botnets*

Referência	**Botnet**	**Processamento**	**Conjunto de dados**	**Avaliação ao longo do fluxo**	**Aprend. ativo**	**Avaliação instâncias rotuladas**	**Multi-classe**	***Ensemble***
(SINGH et al., 2014)	X	Lote	CAIDA					X
(HAMMERSCHMIDT et al., 2016)	X	Ambos	CTU-13	X		X		
(INDRE; LEMNARU, 2016)	X	Lote	KDD da DARPA			X	X	X
(WANG; PASCHALIDIS, 2017)	X	Lote	CTU-13 e CAIDA	X				
(CHEN; CHEN; TZENG, 2018)	X	Lote	PeerRush e CTU-13					
(DOSHI; APTHORPE; FEAMSTER, 2018)	X	Lote	Simulação	X				
(VIEGAS; SANTIN; OLIVEIRA, 2017)		Ambos	Próprio e KDD Cup 1999		X			X
(FAISAL et al., 2014)		Fluxo	KDD Cup 1999		X	X	X	X
(GARG; PEDDOJU; SARJE, 2016)	X	Fluxo	Zeus, Storm, Waledac, ISOT					
(COSTA et al., 2018)	X	Fluxo	CTU-13	X		X		X
(KHANCHI; ZINCIR-HEYWOOD; HEYWOOD, 2018)	X	Fluxo	CTU-13	X	X	X		
(CASSALES et al., 2019)		Fluxo	Kyoto 2006+	X		X		X
(VIEGAS et al., 2019)		Fluxo	MAWIFlow	X				X

processamento. Neste sentido, se o comportamento do ambiente muda o modelo atual é descartado, novos conjuntos de treinamento e teste são montados, e um novo modelo é gerado (VIEGAS; SANTIN; OLIVEIRA, 2017). Chen, Ranjan e Tan (2011) argumentam que no caso das soluções baseadas em aprendizado em lote o tamanho do conjunto de dados pode ser proibitivamente grande, de modo que os invasores podem contornar o método de detecção introduzindo novos atributos em uma taxa mais rápida do que a taxa de retreinamento. Além disso, durante a fase de classificação, as abordagens baseadas em aprendizado em lote simplesmente aplicam o modelo antigo ao novo conjunto de dados, enquanto ignoram os novos atributos assim que eles aparecem. Já no aprendizado em fluxos contínuos de dados, o modelo é atualizado com cada dado de entrada.

Singh et al. (2014) utilizaram ferramentas *open-source* como *Mahout*, *Hadoop* e *Hive* que fornecem uma implementação escalável de detecção a intrusão em *quasi-real-time* . A implementação foi feita utilizando uma abordagem baseada em aprendizado de máquina para detectar ataques de *botnets* P2P. Os módulos de classificação foram treinados com base nos dados capturados de ataques de *botnets* conhecidas como *Storm*, *Zeus*, *Waledac*, *Conficker* e *Kelihos-Hlux*. Apesar de terem obtido uma alta precisão (99,7%), não houve uma preocupação em minimizar a quantidade de instâncias rotuladas necessária, sendo 90% utilizado para treinamento e os 10% restantes para teste. Além disso utilizaram somente uma base de dados, e não compararam os resultados com outros trabalhos

relacionados.

O método de detecção baseado em anomalia proposto por Indre e Lemnaru (2016) obtém um *snapshot* do comportamento da rede. São utilizados *deep autoencoders* para detectar comportamento anômalo que possa surgir no tráfego de rede originado de dispositivos IoT comprometidos no ambiente. Baseado na capacidade e também na comunicação de rede normalmente produzida por dispositivos IoT, as dificuldades em capturar tráfego normal pode variar. Portanto, os autores hipotetizaram que a predição do comportamento do tráfego pode ser diretamente traduzida em várias métricas de desempenho para detecção de anomalias. Utilizaram apenas uma base de dados (KDD - DARPA), avaliando o desempenho de vários classificadores (incluindo conjuntos de classificadores) considerando tanto classificação binária quanto multi-classe. Interessante que, apesar de utilizarem uma grande quantidade de instâncias rotuladas no conjunto de treinamento inicial (cerca de 450.000), conseguiram reduzir esse número para aproximadamente 1.000 instâncias e manter alta taxa de detecção de ataques. Contudo, o teste foi realizado em um conjunto de dados antigo, desatualizado e sem amostras de *botnets*.

Hammerschmidt et al. (2016) criaram uma solução utilizando máquinas de estado finitas e atributos de fluxos contínuos de rede para detectar dispositivos infectados. Eles apresentaram métodos que avaliam a quantidade de informação no conjunto de treinamento e mostraram que a quantidade de dados necessários para aprendizado pode ser limitada de forma *online*. Os autores realizaram classificação binária dos fluxos e utilizaram três cenários da base CTU-13 (10, 11 e 12, por serem cenários com múltiplos *hosts* infectados ao mesmo tempo), e somente cinco dos atributos disponíveis nos fluxos. Não foi abordado o uso de conjuntos de classificadores, nem estratégias de aprendizado ativo para seleção das instâncias rotuladas. Contudo, com relação a avaliação do modelo ao longo do fluxo de dados e avaliação das instâncias rotuladas, os autores usam um critério chamado de *freshness* para identificar o impacto da redução de instâncias antigas no treinamento do modelo.

Wang e Paschalidis (2017) propuseram uma abordagem de duas fases para detecção de *botnets*: primeiro, detectar e coletar anomalias de rede com presença de *botnet* utilizando métricas estatísticas; segundo, identificar os *bots* pela análise dessas anomalias. Na primeira fase foram utilizados dados da base CTU-13, e na segunda dados de ataque originados da base CAIDA misturados com tráfego real *background* obtido na *University of Twente*. Utilizaram os primeiros vinte e cinco minutos dos conjuntos de dados do CTU-13 para treinamento, com janela de avaliação de cinco minutos, e janelas de detecção de dois segundos. Foi feita uma classificação binária das instâncias, e os resultados para cada cenário foram comparados com outros métodos, como por exemplo o *BotHunter*. Apesar disso, não ficou claro o critério de seleção de apenas cinco dos treze cenários do CTU-13, e nem o uso de apenas sete atributos dos fluxos. Outro ponto importante, é que em nenhum momento foi mencionada estratégia para reduzir a seleção das instâncias rotuladas.

A pesquisa de Chen, Chen e Tzeng (2018) propõe um sistema de detecção baseado em aprendizado de máquina para reconhecer tráfego de *botnets* P2P. A abordagem proposta pelos autores envolve o uso de atributos de fluxo da rede e redes neurais artificiais do tipo *feed-forward* para a classificação binária dos exemplos. Os autores usam os conjuntos de dados PeerRush e CTU-13 para validação da proposta. Apesar dos bons resultados em termos de acurácia, o número de instâncias utilizadas é baixo (somente 10 mil instâncias) comparado com a quantidade de exemplos existentes no CTU-13. Um outro problema envolve a fase de treinamento. Os autores usaram uma técnica de validação cruzada durante essa etapa. Em um cenário real, os fluxos chegam ao longo do tempo e ao usar tal método, é possível que o modelo tenha sido treinado com amostras que ainda não foram vistas, do ponto de vista das janelas temporais do conjunto de dados.

Doshi, Apthorpe e Feamster (2018) desenvolveram um *pipeline* baseado em aprendizado de máquina que realiza coleta de dados, extração de atributos e classificação binária para detecção de tráfego DDoS de dispositivos IoT. Os atributos tiram proveito de comportamentos de rede específico de IoT. Foi feita a comparação de uma variedade de classificadores para detecção de ataques, incluindo *random forest*, *K-nearest neighbors*, *support vector machines*, árvores de decisão e redes neurais. O treinamento do classificador foi realizado utilizando 85% de tráfego normal e de DoS combinados, e a precisão de classificação calculada nos 15% restantes do tráfego. Logo, não há preocupação em abordar estratégias de seleção de dados rotulados com o intuito de minimizar seu uso. Outro ponto importante é que, além de utilizarem dados simulados, não foi feita comparação com resultados de outros trabalhos.

Outros trabalhos também propuseram métodos de detecção de *botnets* ou sugeriram comparações entre diferentes algoritmos como: (STEVANOVIC; PEDERSEN, 2014), (JIANGUO et al., 2016), (HAMMERSCHMIDT et al., 2017), (ALAUTHAMAN et al., 2018), (PEKTAŞ; ACARMAN, 2018) e (SILVA et al., 2020). Contudo, nenhum deles abordou, com detalhes, pelo menos uma das características estudadas na dissertação e relacionadas na Tabela 2.

3.1.2 Detecção de *botnets* em fluxos contínuos de dados

Os algoritmos de mineração de dados tradicionais são desafiados por dois recursos característicos dos fluxos de dados: o fluxo infinito de dados e as mudanças de conceito. Como métodos que requerem múltiplas varreduras dos conjuntos de dados não podem lidar com fluxos de dados infinitos, vários algoritmos incrementais que refinam modelos incorporando continuamente novos dados do fluxo foram propostos durante os últimos anos (WANG et al., 2003).

Ao contrário das soluções que usam algoritmos de aprendizado de máquina tradicionais, a mineração de fluxos contínuos é capaz de aprender continuamente, sem a necessidade de armazenar todos os dados rotulados e usá-los para retreinar a solução quando

novos dados estiverem disponíveis, adaptando-se melhor as mudanças de comportamento. Da mesma forma, o modelo de predição é atualizado de forma incremental, diferente do aprendizado de máquina tradicional que processa dados passados, mesmo obsoletos, para obter um modelo. Além disso, os algoritmos de aprendizado de máquina para fluxos de dados consomem significativamente menos memória do que seus equivalentes em lote, o que torna possível implantá-los em diferentes componentes de rede (COSTA et al., 2018).

Faisal et al. (2014) implementaram uma arquitetura IDS específica para ameaças em *smart grids*. Eles estudaram o comportamento de um grupo de algoritmos em diferentes componentes desta arquitetura de IDS. Usando o desatualizado conjunto de dados *KDD Cup 1999*, avaliaram o desempenho e a viabilidade de usar algoritmos de mineração de fluxos para infraestruturas de medição avançadas. Com o uso de aprendizado ativo obtiveram bons resultados em termos de uso de memória, tempo gasto para classificação e uso reduzido de dados rotulados para treinamento e atualização do modelo. Contudo, o trabalho não lida com fluxos *botnets* ou investiga o desempenho do modelo ao longo do tempo.

Já Cassales et al. (2019) propõem uma arquitetura para detecção de ameaças IoT usando fluxos contínuos e recursos de computação em nuvem. Embora não tenham testado estratégias diferentes de aprendizado ativo, foi realizado treinamento *offline* do modelo de decisão com cerca de 10% da base de dados utilizando a técnica de rótulo parcial, ou seja, sem entregar todos os rótulos de todas as instâncias. Apesar de encontrarem grande variação nos resultados, concluíram que isso pode ser resolvido com melhoramentos na parametrização e seleção de atributos. Além disso, mostraram que é possível identificar rapidamente e adaptar os modelos de forma eficiente.

O objetivo desta dissertação é usar técnicas de mineração de fluxos contínuos no problema de detecção de *botnets* pois as soluções comumente utilizadas possuem dificuldade para se adaptar a chegada de novos dados. Para isso, é preciso estabelecer um relacionamento com resultados alcançados por trabalhos do estado da arte. Neste sentido, foi possível encontrar alguns trabalhos guiados pela mesma ideia. De acordo com (COSTA et al., 2018), muitos IDSs dependem de algoritmos de aprendizado de máquina para aprender comportamentos maliciosos e usar os padrões modelados nas tarefas de detecção. Assim, eles propõem detectar *botnets* de forma *online* analisando os fluxos de rede. Os autores avaliaram o desempenho para detectar fluxos de *botnet* com a VFDT (*Very Fast Decision Tree*) e *prequential learning*, utilizando como base de dados o CTU-13. Logo, o cenário inicial dessa dissertação, abordado no Capítulo 4, utiliza como referência base o trabalho de Costa et al. (2018). Esta decisão foi feita pois o trabalho proposto por Costa et al. (2018), além de seguir a maioria dos critérios abordados nessa dissertação, foi o único que realizou testes individuais em todos os cenários do CTU-13.

O trabalho de Viegas et al. (2019) proporcionou duas contribuições principais: o *MAWIFlow*, um novo conjunto binário com dados reais de tráfego de rede, e um novo IDS,

o *BigFlow*. Observaram que, passados alguns meses do treinamento inicial do modelo, as abordagens de aprendizado de máquina tradicional não conseguiam acompanhar mais as mudanças à medida que os fluxos do *MAWIFlow* chegavam, sendo que a precisão diminuía significativamente. Assim, eles construíram o *BigFlow*, que melhorou o desempenho na detecção de ataques. Entretanto, apesar de o *BigFlow* ser baseado em mineração de fluxos contínuos, ele não atualiza o modelo a cada instância recebida, e sim semanalmente. Isso é crítico, pois mudanças no tráfego poderiam não ser detectadas rapidamente, deixando os sistemas mais vulneráveis. Além disso, não fica claro se ameaças *botnets* foram avaliadas no decorrer dos experimentos.

Garg, Peddoju e Sarje (2016) propuseram um sistema de detecção com alta acurácia para *botnets* P2P baseadas em tráfego de *bots* para falha e comunicação. O sistema usa tráfego de rede como dois fluxos contínuos separados que são analisados em uma pequena janela de tempo. Várias bases de dados foram utilizadas (*Zeus, Storm, Waledac e ISOT*). Ao contrário de outros modelos, a detecção precoce é possível devido capacidade do sistema em detectar tráfego malicioso antes que os *bots* se juntem à *botnet*. Nenhum dado rotulado é necessário para treinar o modelo, dependendo da escolha de pequenos trechos de tráfego por um período de apenas 2 minutos para fins de análise. Isso minimiza necessidade de dados rotulados, e acima de tudo, mantém alta a taxas de detecção. Entretanto, não utilizaram outras abordagens, como aprendizado ativo, para minimizar seleção das instância rotuladas. Além disso, não há qualquer comparação dos resultados com outros trabalhos.

Khanchi, Zincir-Heywood e Heywood (2018) utilizaram um *framework* de aprendizado de máquina para aprender comportamentos de rede e se adaptar incrementalmente às mudanças no tráfego. Foram utilizados todos os conjuntos disponíveis na base de dados CTU-13. Os resultados demonstraram que a solução proposta, o *Stream Genetic Programming*, apresenta melhor desempenho que os algoritmos implementados no MOA (BIFET et al., 2010), podendo detectar a menor variação em uma classe *botnet* C&C a partir do momento que ela aparece no início de um fluxo. Uma vantagem é que relacionaram os parâmetros necessários para realização dos experimentos. As limitações incluem: os resultados não foram separados para cada conjunto de dados do CTU-13 e utilizaram apenas uma estratégia de aprendizado ativo para seleção das instâncias rotuladas.

Por fim, a análise da literatura mostra que as pesquisas sobre o desenvolvimento de sistemas de detecção de intrusão baseados em técnicas de aprendizado de máquina ainda é um tópico relevante da área. Contudo, muitos trabalhos ainda não lidam, especificamente, com tráfego de *botnets* e não discutem com profundidade a relação entre o desempenho do modelo e o número de instâncias rotuladas. Também foi possível notar que poucos trabalhos utilizam técnicas de aprendizado ativo e abordagens multi-classe.

3.2 Conjuntos de dados com amostras de *botnets*

O número de ameaças que os indivíduos e negócios estão enfrentando está subindo exponencialmente devido ao aumento na complexidade das redes e serviços. Para aliviar o impacto dessas ameaças, os pesquisadores propuseram várias soluções para detectá-las. No entanto, as ferramentas atuais muitas vezes não conseguem se adaptar a arquiteturas em constante mudança, ameaças associadas e ataques de dia zero (HINDY et al., 2020). Além disso, os conjuntos de dados atuais demonstram ausência de ameaças reais e incluem um grande número de ameaças obsoletas. Tais fatores podem limitar a eficiência das abordagens de IDS validados nesses conjuntos de dados (HINDY et al., 2020).

Uma grande parcela dos trabalhos que propõem métodos de detecção de *botnet* é realizada em ambientes simulados. Em tais soluções, os dados são obtidos usando um software de simulação ou um ambiente construído para os capturar pacotes de rede. Ou seja, o tráfego não é originado de dispositivos presentes em um ambiente real e operacional. Para mitigar essa limitação, métodos para gerar tráfego de rede mais próximos do real foram propostos. Um dos mais importantes foi proposto em (SHIRAVI et al., 2012) e envolve a construção de uma infraestrutura de rede e o uso de diferentes tipos de perfis para geração de tráfego normal e anômalo.

Considerando que algumas propostas ainda usam seus próprios conjuntos de dados para avaliar seus métodos, um novo problema que surge é a ausência de estudos comparativos entre os diferentes métodos de detecção de botnets existentes. O trabalho de García et al. (2014) elenca uma série de fatores que dificultam tal comparação: dificuldade para compartilhar o conjunto de dados recém-criado, e a ausência de conjuntos de dados robustos, de uma descrição adequada dos métodos e de uma metodologia de comparação.

Logo, um dos principais desafios para avançar no estado da arte de detecção de intrusão é a ausência de bons conjuntos de dados públicos. Gerar conjuntos de dados realísticos requer ativos de rede caros, geradores de tráfego especializados, e uma considerável preparação de projeto. Mesmo com avanços na virtualização ainda é desafiador criar e manter um ambiente representativo. Dada a velocidade de mudança no comportamento das ameaças, os conjuntos de dados estão se tornando obsoletos mais rapidamente, e alguns dos conjuntos de dados mais citados datam de mais de duas décadas. Conjuntos de dados antigos tem valor limitado: muitas vezes altamente filtrado e anonimizado, com distribuições de eventos não realistas e metodologia de projeto obscura (KENYON; DEKA; ELIZONDO, 2020). Adicionalmente, muitos conjuntos de dados dependem de simulação, proveniente de instituições acadêmicas ou governamentais.

Kenyon, Deka e Elizondo (2020) tentam classificar os conjuntos de dados públicos de intrusão mais utilizados, fornecendo referências a arquivos e literatura. Além disso, o trabalho ilustra o escopo e utilidade relativa, destacando a composição das ameaças, formatos, recursos especiais, e as limitações associadas. Foram identificadas também as melhores práticas no projeto de conjuntos de dados, descrevendo as potenciais armadi-

lhas no projeto de técnicas de detecção de ameaças com base em dados que podem ser inadequados ou comprometidos devido à cobertura irreal de ameaças.

Com relação a dados de botnets, Kenyon, Deka e Elizondo (2020) afirmam que faltam conjuntos de dados recentes com tráfego real de tais ameaças. Os autores citam apenas dois conjuntos com essas características: UNB IDS (ISCX *botnet*, ISCX Android *botnet*) e CTU-13. Já (HINDY et al., 2020) inclui outros como: CICIDS2018, CICIDS2017, *botnet* dataset, CTU-13 e Android *botnet*. Nas subseções 3.2.1 e 3.2.2 serão apresentados os conjuntos de dados usados nesta dissertação, CTU-13 e IoT-23, respectivamente.

3.2.1 CTU-13

O CTU-13 é um conjunto de dados de tráfego de *botnet* que foi desenvolvido pela Universidade CTU (*Czech Technical University*), República Tcheca, em 2011. García et al. (2014) criaram este novo conjunto de dados público com tráfego real de *botnets*. Este conjunto de dados foi projetado para cumprir com os seguintes objetivos:

- ter ataques reais de *botnets* e não simulações;
- ter tráfego desconhecido oriundo de uma grande rede;
- ter rótulos verdadeiros para treinamento e métodos de avaliação;
- incluir diferentes tipos de *botnets*;
- ter vários *bots* infectados ao mesmo tempo para capturar padrões de sincronização;
- ter arquivos NetFlow para proteger a privacidade dos usuários.

No conjunto de dados CTU-13, um cenário de *botnet* é uma infecção particular de máquinas virtuais usando um *malware* específico. Treze cenários foram criados, sendo que cada um deles foi projetado para representar o comportamento de algum *malware*. As principais características dos cenários e seus comportamento são: protocolo de comunicação utilizado (IRC, P2P ou HTTP), se enviam SPAM, realizam *Click-Fraud*, escaneiam portas, realizam ataques DDoS, utilizam técnicas *Fast-Flux*, ou se foram compilados de forma customizada. Os atributos do tráfego de rede em cada cenário abrangem tamanho dos pacotes, duração, número de pacotes, número de fluxos, número de *bots* e família dos *bots*.

A ferramenta utilizada para capturar o tráfego de rede foi a *tcpdump*. Após a captura dos pacotes, o conjunto de dados foi pré-processado e convertido em um formato comum para métodos de detecção. O formato escolhido foi o padrão de arquivo *NetFlow*, que é considerado padrão para representação de dados de rede. A conversão dos arquivos *pcap* para arquivos *NetFlow* foi feita utilizado a ferramenta Argus. A versão final do arquivos *NetFlow* é composta dos seguintes campos: Horário de Início, Horário de Fim, Duração,

Endereço IP de Origem, Porta de Origem, Direção, Endereço IP de Destino, Porta de Destino, Estado, SToS, Total de Pacotes e Total de Bytes.

A atribuição de rótulos verdadeiros é uma parte importante do processo de criação do conjunto de dados. Entretanto, é uma tarefa complexa e custosa. A estratégia de rotulagem utilizada atribui três rótulos diferentes: *background*, *botnet* e normal. A prioridade para atribuição dos rótulos é a seguinte (GARCÍA et al., 2014):

1. atribuir o rótulo *Background* para todo o tráfego, sem exceção;
2. atribuir o rótulo Normal ao tráfego que corresponda a certos filtros customizados (por exemplo, tráfego originado de máquinas dadas como benignas);
3. atribuir o rótulo *Botnet* para todo o tráfego originado ou destinado a qualquer endereço IP infectado já conhecido.

3.2.2 IoT-23

IoT-23 é um novo conjunto de dados (publicado em janeiro de 2020) com tráfego de rede de dispositivos *IoT* que foi capturado durante os anos de 2018 e 2019 no Stratosphere Laboratory, AIC group, FEL, CTU University, Czech Republic. Possui vinte e três cenários de captura de diferentes tráfegos de rede IoT (*Internet of Things*), sendo vinte capturas de *malware* executadas em Raspberry Pi's IoT infectados, e três capturas de tráfego de dispositivos IoT benignos. Em cada cenário malicioso foi executada uma amostragem específica do *malware*, o qual utiliza vários protocolos para realizar diferentes ações. O IoT-23 contém amostras de *botnets* que infectam, especificamente, dispositivos IoT. O objetivo foi oferecer um grande conjunto de dados reais e rotulados de infecções por *malware* e tráfego benigno de IoT para pesquisas envolvendo o desenvolvimento de algoritmos de aprendizado de máquina.

O fluxos foram obtidos com a ferramenta *Zeek Network Analyzer* utilizando o arquivo .pcap original com o tráfego de rede capturado. Este conjunto de dados fornece informações mais detalhadas dos rótulos, descrevendo a relação entre os fluxos maliciosos ou de possíveis atividades maliciosas. Os rótulos utilizados para detecção de fluxos maliciosos foram baseados em uma análise manual do tráfego de rede. Abrangem rótulo normal e para tráfego malicioso, ou de ataque, existem diferentes rótulos como *Attack*, *C&C*, *DDoS*, entre outros.

Considerando todos os cenários presentes no conjunto IoT-23, a quantidade de fluxos normais é de aproximadamente 31 milhões, e de 295 milhões para fluxos maliciosos. Dentre os fluxos maliciosos destacam-se os fluxos dos tipos *PortScan*, *DDoS* e *C&C*.

Capítulo 4

Proposta

Neste capítulo a proposta para avaliação do uso de algoritmos de mineração de fluxos contínuos para a detecção de *botnets* é apresentada. A Seção 4.1 fornece a contextualização e motivação da proposta. A Seção 4.2 apresenta a proposta, mostra o processo incremental de classificação de fluxos que será adotado no decorrer do trabalho e detalha cada um dos cenários associados às hipóteses enunciadas no Capítulo 1.

4.1 Contextualização

Conforme apresentado no Capítulo 1, o objetivo deste trabalho consiste em avaliar o desempenho dos algoritmos de mineração de fluxos contínuos de dados considerando fatores como: (i) minimizar o uso de instâncias rotuladas na atualização do modelo de decisão ao longo do fluxo de dados; (ii) avaliar o desempenho dos classificadores por meio de medidas extraídas de janelas de dados; e (iii) considerar a detecção de *botnets* como uma tarefa multi-classe.

No Capítulo 1 é mencionado que a maioria das abordagens de aprendizado de máquina para classificação de dados fazem uma suposição de que o conjunto de dados a ser processado é representativo, tendo acesso às instâncias do conjunto. Estes modelos de classificação em lote realizam o aprendizado a partir de um conjunto finito de treinamento para gerar um modelo de decisão, o qual é utilizado para classificar instâncias do conjunto de teste (GARCÍA; ZUNINO; CAMPO, 2014). Sendo assim, algoritmos baseados em aprendizado em lote possuem dificuldade em lidar com ambientes dinâmicos, em que os dados chegam continuamente. Em detecção de *botnets* é inviável e irreal supor que é possível ter acesso completo e a qualquer momento aos rótulos de todas as instâncias de um conjunto de dados. Não basta aprender uma vez, e também não é possível conhecer todos os tipos de ataque. Um exemplo é que as classes rotuladas de instâncias recém-chegadas em fluxos contínuos de dados não estão disponíveis imediatamente (KRAWCZYK et al., 2017). O custo de conseguir os rótulos das instâncias é alto. Além disso, o comportamento do tráfego de rede pode mudar de forma contínua. Logo, o aprendizado de máquina tradi-

cional baseado em lotes de dados estáticos (não baseados em fluxos) não parece adequado para desenvolver métodos de detecção de *botnets* que sejam escaláveis e efetivos ((COSTA et al., 2018) e (KRAWCZYK et al., 2017)).

Outras limitações de trabalhos anteriores envolvem a avaliação de desempenho somente ao final do fluxo de dados e não ao longo dele e a questão da classificação binária das instâncias em fluxos normais ou maliciosos, dado que o tráfego das *botnets* possui diversos componentes, como a comunicação do C&C com o *bot* e os ataques disparados pelos *bot*.

Assim, considerando os pontos críticos mencionados anteriormente, é preciso utilizar novas abordagens para melhorar o desempenho na detecção de *botnets*. Na Seção 2.4 é mencionado que ao contrário da maioria das propostas que se baseiam em classificadores únicos, o uso de conjuntos de classificadores na mineração de fluxos contínuos de dados é mais promissora pois se adaptam melhor às mudanças no comportamento dos dados. Além disso, dados rotulados são caros, sendo importante utilizar abordagens que diminuam a quantidade de amostras rotuladas apresentadas ao classificador. Logo, outra abordagem interessante detalhada na Seção 2.5.1 é o aprendizado ativo que abrange diferentes estratégias de seleção para minimizar a quantidade de instâncias rotuladas necessárias para treinar o modelo.

4.2 Processo Iterativo para Classificação de Fluxos

A proposta consiste em desenvolver um método para analisar o tráfego da rede e identificar fluxos de *botnets* usando algoritmos de mineração de fluxo contínuo de dados. O método desenvolvido foi baseado no trabalho de (COSTA et al., 2018) onde um processo incremental para detectar fluxos de *botnets* usando a *Hoeffding Tree* foi proposto. O detalhamento e as adequações feitas em tal método para suportar cada um dos cenários deste trabalho serão apresentadas a seguir.

Inicialmente, foi preciso elaborar uma forma de avaliar os fluxos de rede de uma maneira incremental. A Figura 7 ilustra o processo de classificação incremental de fluxos de redes usado no decorrer do trabalho. A seguir, cada um dos passos será detalhado.

Um conjunto específico de atributos é extraído de fluxos de tráfego de rede para gerar o conjunto de dados, sendo que um fluxo é um conjunto de pacotes que tem propriedades em comum, tais como os mesmos *sockets* de origem e destino. Foi adotada uma abordagem em fluxo devido às limitações dos sistemas de detecção de intrusão baseados em pacote e protocolo (como a complexidade e tempo gasto para realizar inspeção de conteúdo dos pacotes) e a prevalência de conjuntos de dados que utilizam este método (UMER; SHER; BI, 2017). Uma iteração começa quando a primeira instância disponível do conjunto de dados inicializa o algoritmo de mineração de fluxo selecionado. Depois disso, o algoritmo escolhido está pronto para realizar as previsões. Entretanto, é importante notar que a

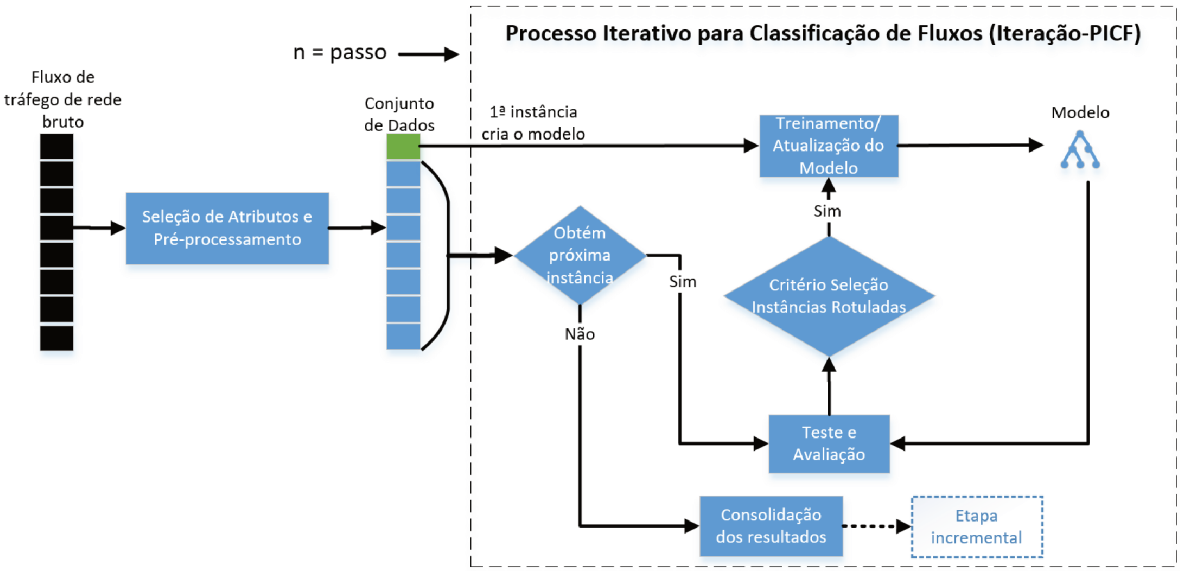
próxima previsão será relacionada à única classe aprendida. Em seguida, se uma nova instância estiver disponível, o algoritmo de mineração de fluxo tenta prever sua classe (*botnet* ou não). Isso é ilustrado pela etapa *Teste e Avaliação* na Figura 7. Essa previsão é usada para calcular as medidas de desempenho (por exemplo, precisão e revocação) de forma incremental. Se o rótulo verdadeiro dessa instância estiver disponível, a instância será usada para atualizar o modelo de decisão. Os rótulos verdadeiros geralmente são fornecidos por um especialista em segurança que pode fornecer *feedback* ao modelo. Este processo iterativo de fornecimento de rótulo continua enquanto o critério escolhido para liberação de instâncias rotuladas seja satisfeito. A ideia aqui é replicar um cenário em que os especialistas em segurança não têm conhecimento completo de seu cenário de ameaças. Uma vez que rotular instâncias é muito caro para um especialista, é essencial desenvolver uma estratégia para diminuir o número de instâncias rotuladas. Este processo foi baseado no trabalho de Costa et al. (2018), que aplicou a árvore VFDT na detecção de fluxos de *botnet* limitando a quantidade de instâncias rotuladas dadas ao classificador considerando um *max n* específico para cada conjunto de dados. As principais diferenças entre a proposta dessa dissertação e o do trabalho de Costa et al. (2018) são:

- ❏ além da VFDT, são utilizados outros algoritmos de classificação (Classificadores Únicos e Conjuntos de Classificadores);
- ❏ o critério de seleção das instâncias para treinamento e atualização do modelo não está limitado somente ao *max n* por conjunto de dados, mas são utilizadas também estratégias de aprendizado ativo;
- ❏ além de consolidar os resultados de forma cumulativa, também são utilizadas janelas;
- ❏ além de utilizar a base de dados CTU-13, também foi considerada a IoT-23.
- ❏ é realizada classificação binária e multi-classe dos dados;

O processo descrito anteriormente termina quando não houver instâncias restantes. Nesse caso, os resultados são consolidados na fase de avaliação. Conforme ilustrado na Figura 8, todo o processo pode ser repetido caso o número de instâncias tenha atingido o limite predefinido para o conjunto de dados. A seguir, são descritas cada uma das etapas da proposta do processo incremental de classificação de fluxos:

1. Seleção de Atributos e Pré-processamento: os pacotes de rede podem ser agrupados ou consolidados em fluxos. Cada instância é criada selecionando ou calculando atributos específicos desse fluxo de tráfego de rede bruto com base em sua relevância para a análise. O resultado é o conjunto de dados de linha de base, uma composição de todas as instâncias criadas.

Figura 7 – Visão geral do processo incremental de classificação de fluxos



2. Inicialização do modelo: esta etapa inicializa a *Hoeffding Tree*, que é o algoritmo de classificação base usado por todos as estratégias de classificação em nossa proposta.

3. Treinamento/Atualização do Modelo: com as instâncias rotuladas, o modelo de decisão precisa ser atualizado. Para classificadores únicos, o modelo é constantemente incrementado. Para conjuntos de classificadores, diferentes estratégias de classificação são usadas para atualizar os modelos que formam o conjunto.

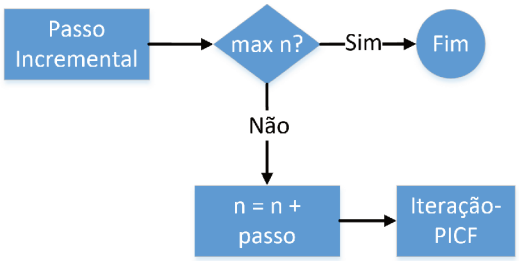
4. Teste e Avaliação: esta etapa visa prever o rótulo de cada instância de dados usando o modelo de decisão. As medidas de avaliação são calculadas de forma incremental sempre que uma nova instância é prevista.

5. Consolidação dos resultados: os resultados são calculados para todas as instâncias de dados em cada iteração. É proposto o uso de medidas de avaliação, como precisão (porcentagem de fluxos classificados como *botnets* que são de fato *botnets*) e revocação (porcentagem de fluxos de *botnet* classificados corretamente).

6. Etapa incremental: o número de instâncias a serem rotuladas pode ser opcionalmente incrementado em uma etapa fixa até que o valor máximo de n ($max\ n$) seja alcançado. Esta etapa é ilustrada na Figura 8.

Ao contrário de trabalhos que adotam a acurácia como principal medida de avaliação, a abordagem proposta nesta dissertação utiliza as métricas precisão e revocação. A principal razão para isso é o grande desbalanceamento entre a quantidade de instâncias *botnet* e *background*. Com precisão e revocação é possível identificar respectivamente o percentual de fluxos que foram classificados tal como previsto e a taxa de fluxos classificados corretamente. Também foi utilizada a média aritmética entre precisão e revocação. Outra

Figura 8 – Etapa incremental opcional do Processo Iterativo para Classificação de Fluxos



métrica que será usada para comparar o desempenho do classificador em determinados cenários é o *best n*. Ela representa o número mínimo de instâncias de *botnet* necessárias para treinar o modelo e atingir a média máxima entre revocação e precisão, considerando um limite de variação aceitável de 0,1 (10 %). Por exemplo, para cada passo de 100 em n, a média entre a revocação e a precisão é calculada e registrada. Na próxima etapa, a média atual é calculada e comparada com a anterior. Se a variação entre os dois valores (atual e anterior) for maior que 0,1, o *best n* é atualizado. O Algoritmo 1 abaixo detalha essas etapas.

Algoritmo 1 Cálculo do *best n*

```
Result: best n
mediaPrecisaoRevocacao = media(precisaoCompleta, revocacaoCompleta);
 difBestN = 1;
 variacaoAceitavel = 0,1;
 while n < maxN do
    dif = absoluto(mediaPrecisaoRevocacao - mediaPrecisaoRevocacaoN);
     if dif + variacaoAceitavel < difBestN then
        bestN = n;
         difBestN = dif;
    end
    n = n + passo;
end
```

Todos os algoritmos utilizados nesta proposta possuem implementações no MOA (BIFET et al., 2010), sendo que a versão utilizada foi a *moa-release-2019.05.0*. O MOA (*Massive Online Analysis*) é uma plataforma *open source* muito popular utilizada pela comunidade científica para pesquisas relacionadas a mineração de fluxos contínuos de dados (WAIKATO, 2018). Foi desenvolvido em Java e pode ser utilizado tanto via interface gráfica como linha de comando. O MOA fornece ferramentas para análise de fluxo de dados, que incluem: algoritmos de aprendizagem de fluxo de dados; conjuntos de classificadores; geradores de dados (*Random Tree Generator, SEA, AGR*); métodos de avaliação (*periodic holdout, test-then-train, prequential*); e estatísticas (*CPU time, RAM-hours, Kappa*). Essa plataforma compartilha muitas características com o WEKA (*Waikato Environment for Knowledge Analysis*), tal como permitir que os usuários estendam a plataforma herdando classes abstratas. Muitas vezes, os pesquisadores disponibilizam seu código-fonte

como uma extensão do MOA (GOMES et al., 2017). É possível criar novos algoritmos de mineração, novos geradores de fluxos contínuos de dados e até mesmos novas medidas de avaliação (WAIKATO, 2018). Assim, utilizando este recurso de extensão dentro do MOA, nesta proposta foi implementado uma nova estratégia de fornecimento de rótulos, a *ALLimitedInstances*, que é descrito em mais detalhes na Seção 4.2.1.

Considerando cada etapa descrita anteriormente, foram definidas variáveis ou parâmetros que auxiliam a definir e diferenciar cada um dos cenários da proposta. A Tabela 3 fornece uma descrição de cada variável. Os valores possíveis para estas variáveis dependem do cenário proposto conforme explicado nas Subseções 4.2.1, 4.2.2, 4.2.3 e 4.2.4. Além disso, para cada experimento presente no Capítulo 5 será fornecida a devida configuração do parâmetro em questão.

Tabela 3 – Variáveis base do modelo proposto para parametrização nos cenários

Variável	Nome	Descrição
D	Conjunto de Dados	Relacionado ao conjunto de dados utilizado para treinamento e teste do modelo de decisão e o formato dos fluxos (binário ou multi-classe).
P	Passo incremental	O número de instâncias rotuladas a ser liberadas pode ser incrementado por esse valor após cada execução até atingir o $max\ n$.
S	Estratégia de seleção de instâncias rotuladas	Estratégia utilizada para liberar ou selecionar instâncias rotuladas para treinamento e atualização do modelo de decisão.
C	Estratégia de classificação	Estratégia utilizada para classificação dos dados.
A	Algoritmo de classificação	Algoritmo de classificação utilizado pela estratégia de classificação.
T	Tamanho da janela	Valor definido para o tamanho da janela para consolidação dos resultados de cada execução. Trata-se de quantidade de instâncias pré-definida.

Nas subseções a seguir, cada um dos cenários da proposta será detalhado. É importante lembrar que o processo de classificação incremental de fluxos descrito na Figura 7 faz parte de todos os cenários. Além disso, as variáveis apresentadas na Tabela 3 fornecem a composição de cada um dos cenários e serão devidamente detalhadas por cenário.

4.2.1 Comparar o uso de um único classificador em contraste ao uso de conjuntos de classificadores

Neste primeiro cenário, o modelo descrito anteriormente foi utilizado para verificar a hipótese de que conjuntos de classificadores superam o desempenho dos classificadores únicos com relação a precisão, revocação e número de instâncias rotuladas necessárias para atualização do modelo. Neste cenário foi considerado um conjunto de dados rotulado com classe binária (ataque ou *background*). Na Tabela 4 são apresentados os valores possíveis para as variáveis do modelo proposto neste cenário.

Tabela 4 – Variáveis do modelo proposto para cenário de comparação de classificadores

Variável	**Valor possível**
D	Conjunto de dados com duas classes (binário).
P	Valor pré-definido.
S	*max n* pré-definido por conjunto de dados.
C	Classificador Único ou Conjunto de Classificadores.
A	*Hoeffding Tree.*
T	Valor corresponde ao tamanho do conjunto de dados.

O framework MOA (BIFET et al., 2010) foi estendido com a implementação de uma nova estratégia de seleção (S) para limitar o número de instâncias a serem rotuladas, chamada *ALLimitedInstances*. Nessa nova implementação, a apresentação do rótulo das instâncias de treinamento fica limitada até um número específico que pode ser configurado como entrada. Isso é importante porque o processo de atribuição de rótulos para um grande número de instâncias pode ser caro (FARID et al., 2013). Em vez de considerar abordagens convencionais baseadas em anomalias, onde uma parte substancial do rótulo é usada para criar um modelo estático, é mais realista considerar uma abordagem dinâmica em que o modelo está sendo treinado de acordo com a chegada dos fluxos. Nesse caso, a equipe de segurança não teria recursos para rotular um grande número de instâncias. Portanto, pode haver uma compensação entre o desempenho da detecção e o número de instâncias rotuladas.

A estratégia utilizada para avaliar o classificador é conhecida como *Prequential*, explicada na Seção 2.4.3 do Capítulo 2. Segundo Krawczyk et al. (2017) essa estratégia consiste em: (i) apresentar ao classificador uma instância sem o rótulo para treinamento; (ii) exigir uma previsão relacionada à instância; (iii) o resultado obtido é calculado usando as métricas; e (iv) o valor do rótulo original é apresentado para atualizar o modelo. Na abordagem deste trabalho, o número de instâncias rotuladas é limitado.

Os algoritmos utilizados neste cenário são: *Hoeffding Tree* ((DOMINGOS, 2000) e (HULTEN; SPENCER; DOMINGOS, 2001)), *Hoeffding Adaptive Tree* (BIFET; GAVALDÀ, 2009), *Ozabag* (OZA; RUSSEL, 2001), *Ozaboost* ((OZA; RUSSEL, 2001), e *Oza Bagging With Adwin* ((BIFET; GAVALDA, 2007) e (BIFET G. HOLMES; GAVALDà, 2009)). Todos estes algoritmos possuem implementação disponível no MOA. A *Hoeffding Tree* é um algoritmo de classificação utilizado nessa proposta tanto nas estratégias de classificador único quanto nas de conjunto de classificadores. Ela foi escolhida pois é um dos algoritmos de classificação mais utilizados em aprendizado de máquina para fluxos contínuos de dados (COSTA et al., 2018). Além disso, ao contrário das árvores de decisão tradicionais, a *Hoeffding Tree* pode aprender uma instância de cada vez, armazenando informação suficiente e tirando vantagem de uma propriedade estatística chamada *Hoeffding Bound* (HB). O modelo de predição obtido por este algoritmo é baseado em estatísticas sobre as instâncias em vez das próprias instâncias. Enfim, a *Hoeffding Tree* tem um desempenho competitivo em comparação com árvores de decisão moldadas para

treinamento em lote de conjuntos de dados convencionais (COSTA et al., 2018). Já a *Hoeffding Adaptive Tree* é uma implementação feita a partir da *Hoeffding Tree*, mas que consegue lidar com mudanças de comportamento ao longo do fluxo. Já os conjuntos de classificadores *Ozabag*, *Ozaboost*, e *Oza Bagging With Adwin* lidam bem com mudanças de comportamento, atualmente são os mais utilizados em pesquisas de mineração de fluxos contínuos, cuja configuração no MOA pode incluir árvores como algoritmos de classificação.

Uma particularidade deste cenário é que na etapa de *Teste e Avaliação* o conjunto de classificadores prevê a classe de uma nova instância usando a estratégia de voto majoritário, enquanto que os classificadores únicos só usam um classificador.

Além disso, o processo iterativo de fornecimento de rótulo continua até que o número de instâncias de *botnet* rotuladas seja menor que um valor inicial n (este valor pode ser iterado até $max\ n$ que foi pré-definido para o conjunto de dados). O rótulo das instâncias será apresentado até o atingir número n de instâncias especificas de uma classe ou não. Conforme ilustrado na Figura 8, todo o processo pode ser repetido se o valor n não atingiu o valor predefinido $max\ n$.

4.2.2 Analisar resultados da classificação utilizando janelas

Neste cenário o objetivo é verificar a hipótese de que o uso de medidas de desempenho, calculadas ao longo do fluxo de dados trazem mais detalhes sobre a a avaliação do modelo de decisão do que uma única avaliação ao fim do fluxo. A saída do cenário anterior 4.2.1 será o algoritmo com melhor desempenho, ou seja, que com menor número de instâncias de treinamento (*best n*) atingiu o valor máximo da média entre precisão e revocação. Esse algoritmo é considerado como entrada neste cenário. Na Tabela 5 são apresentados os valores possíveis para as variáveis do modelo proposto referente a este cenário.

Tabela 5 – Variáveis do modelo proposto para cenário utilizando janelas

Variável	**Valor possível**
D	Conjunto de dados com duas classes (binário).
P	Não se aplica.
S	*max n*. Corresponde ao *best n* do algoritmo com melhor desempenho obtido no cenário 4.2.1.
C	Algoritmo com melhor desempenho obtido cenário 4.2.1.
A	*Hoeffding Tree*.
T	Valor pré-definido.

Já neste cenário, a avaliação dos resultados se dá por janelas temporais T de tamanho pré-definido. As janelas temporais são porções dos dados de tamanho específico e que mantém a ordem de chegada ao classificador. A avaliação por janelas temporais auxilia a entender o comportamento do classificador de fluxo em momentos importantes para um IDS, por exemplo, entender o que acontece quando os primeiros fluxos de ataque são apresentados ao classificador. É esperada uma diminuição na taxa de acertos, dado que é

a primeira vez que esse tipo de tráfego aparece. Contudo, como a literatura não aborda devidamente esse assunto, não fica claro como os diferentes momentos presentes em um conjunto de dados impactam o sistema de detecção de *botnets*. Logo, faz parte desse cenário analisar as oscilações na precisão e revocação por janela considerando os casos com melhor (*best n*) por conjunto de dado do cenário 4.2.1. Sendo assim, foram mantidos os mesmos procedimentos do cenário anterior 4.2.1, com exceção de:

- ❏ será utilizado o algoritmo de melhor desempenho encontrado no cenário 4.2.1;
- ❏ as etapas de *Teste e Avaliação* e *Consolidação dos resultados* foram realizadas considerando uma janela de tamanho pré-definido;
- ❏ o fornecimento de rótulos (*Etapa incremental*) foi limitado ao valor do *best n* atingido pelo algoritmo com melhor desempenho

4.2.3 Analisar desempenho de conjuntos de classificadores utilizando estratégias de aprendizado ativo

Já neste cenário, o objetivo se baseia na hipótese de que o uso de aprendizado ativo com conjuntos de classificadores mantém alto desempenho enquanto diminui a necessidade de instâncias rotuladas para atualização do modelo. Seguindo a ideia de minimizar o uso de instâncias rotuladas, são aplicadas neste cenário as estratégias de aprendizado ativo, para escolha das instâncias de treinamento cujo rótulo verdadeiro será apresentado ao classificador para aprendizado. As estratégias são a Aleatória, de Incerteza Fixa, de Incerteza Variável e de Incerteza com Aleatorização explicadas na Seção 2.5.2. A estratégia Aleatória foi utilizada como base, pois é a mais simples, não tendo um critério definido para seleção das instâncias, selecionando-as ao longo do conjunto de forma aleatória. Já as outras três estratégias são baseadas em Incerteza, obedecendo critérios fixos, variáveis ou semi-aleatórios para seleção das instâncias.

Nos cenários anteriores a estratégia para liberação do rótulo das instâncias para treinamento do modelo era baseada somente em um limite sequencial que é o número máximo de instâncias pré-definido e específico de cada conjunto de dados de entrada. Quando este limite era atingido, os rótulos das instâncias simplesmente não eram mais liberados. Já neste cenário, o objetivo é verificar se usar aprendizado ativo pode alcançar bons resultados considerando diferentes tipos de tráfego, com a possibilidade do surgimento de novas *botnets*, e também utilizando um número menor de instâncias rotuladas para treinamento do modelo.

Na Tabela 6 são apresentados os valores possíveis para as variáveis do modelo proposto referente a este cenário. As particularidades deste cenário se resumem a:

- ❏ continuar com o algoritmo com melhor desempenho obtido no cenário 4.2.1;

- ❏ janela com tamanho pré-definido;
- ❏ ter como entrada conjuntos de dados binários;
- ❏ na etapa *Critério Liberação Instâncias Rotuladas* variar as estratégias de aprendizado ativo (Aleatória, e também as Baseadas em Incerteza - Fixa, Variável e Variável com Aleatorização) bem como seus parâmetros. A parametrização de cada uma das estratégias será apresentada no Capítulo 5.

Tabela 6 – Variáveis do modelo proposto para cenário com aprendizado ativo

Variável	**Valor possível**
D	Conjunto de dados com duas classes (binário).
P	Não se aplica.
S	Estratégias Aleatória, Incerteza Fixa, Incerteza Variável e Incerteza Variável com Aleatorização.
C	Classificador com melhor desempenho no cenário anterior.
A	*Hoeffding Tree*.
T	Valor pré-definido.

4.2.4 Analisar conjuntos de classificadores utilizando dados multi-classe

Tabela 7 – Variáveis do modelo proposto para o cenário multi-classe

Variável	**Valor possível**
D	Conjunto de dados com mais de duas classes (multi-classe).
P	Valor pré-definido.
S	*max n* pré-definido por conjunto de dados.
C	Classificador com melhor desempenho obtido cenário 4.2.1.
A	*Hoeffding Tree*.
T	Valor corresponde ao tamanho do conjunto de dados.

O objetivo principal deste cenário é verificar a hipótese de que modelos de decisão treinados com várias classes permitem a identificação de erros de classificação para os diferentes tipos de tráfego associado a uma *botnet*. A ideia é confirmar se o resultado de um modelo tem relação com o tipo e quantidade de tráfego, bem como com a quantidade inicial de instâncias rotuladas disponíveis para treinamento. O cenário segue uma ideia diferente dos demais, não considerando somente duas classes de tráfego (ataque e não-ataque), mas utilizando como entrada um conjunto de dados multi-classe (por exemplo, *botnet*, normal, *background*, ataque, C&C, etc.). Além disso, é utilizado o algoritmo com melhor desempenho obtido no cenário 4.2.1 e não há um passo incremental, são utilizadas janelas temporais ao longo do conjunto.

Na etapa 5 de *Consolidação dos resultados* os valores das medidas de avaliação foram separados para cada uma das classes em cada conjunto de dados utilizado. Isso foi feito

para conseguir identificar as classes específicas que impactam no desempenho dos classificadores, bem como os momentos (ou pontos) em que as oscilações ocorrem. Na Tabela 7 são apresentados os valores possíveis para as variáveis do modelo proposto referente a este cenário.

Capítulo 5

Experimentos e Análise dos Resultados

No presente capítulo são detalhados os experimentos realizados seguindo os métodos propostos no Capítulo 4. Conforme apresentado anteriormente, o objetivo principal deste trabalho é avaliar o uso de algoritmos de mineração de fluxos contínuos de dados para a detecção de *botnets* considerando requisitos mais próximos de cenários reais, como manter uma alta taxa de acerto utilizando o mínimo possível de instâncias rotuladas para treinamento do modelo.

Na Seção 5.1 são detalhados os conjuntos de dados CTU-13 e IoT-23 (utilizados como origem dos dados neste trabalho), bem como os atributos específicos de cada um deles, e também as métricas utilizadas para avaliar o desempenho dos experimentos. Já na Seção 5.2, cada um dos experimentos referentes as tarefas descritas nas Seções 4.2.1, 4.2.2, 4.2.4 e 4.2.3 serão apresentados e discutidos.

5.1 Conjuntos de dados, atributos e medidas de avaliação

A Subseção 5.1.1 apresenta as principais características dos conjuntos de dados usados durante o experimento enquanto que a Subseção 5.1.2 detalha o processo de seleção de atributos e pré-processamento dos conjuntos de dados.

5.1.1 Conjuntos de dados

Foram utilizados dois conjuntos de dados contendo tráfego de *botnet*, o CTU-13 (GARCIA MARTIN GRILL; ZUNINO, 2014) e o IoT-23 (PARMISANO; ERQUIAGA, 2020). Estes conjuntos foram escolhidos porque são públicos, apresentam tráfego real de *botnets* combinado com tráfego normal e *background* e estão disponíveis em formatos variados. Além disso, são recomendados e referenciados por um grande número de universidades, empresas e pesquisas. O CTU-13 possui treze cenários reais baseados em ataques de *botnet* que incluem diferentes tipos de protocolos de rede e *malware*, uma grande quantidade

de dados capturados e tráfego de dados rotulados. Cada cenário do CTU-13 possui um número diferente de instâncias pertencentes a cada classe do problema. Os rótulos de instância podem ser Normal (o tráfego corresponde a determinados filtros relacionados a computadores de rede conhecidos), *Botnet* (tráfego originado ou direcionado para endereços IP infectados conhecidos) e *Background* (tráfego entre os outros endereços IP de computadores que não são conhecidos nem infectados) (GARCIA MARTIN GRILL; ZUNINO, 2014). A maior parte do tráfego foi rotulada como *background*. Nos experimentos envolvendo essa base, os rótulos normal e *backgroud* não foram diferenciados. Sendo assim, apenas duas classes foram consideradas, *background* (BG) e *botnet*. Nas Tabelas 8 e 9 são detalhadas, respectivamente, a quantidade de dados e a distribuição de classes em cada cenário do CTU-13.

Tabela 8 – Quantidade de dados em cada cenário do CTU-13

Id	**Duração (h)**	**# Pacotes**	**# Fluxos**	**Tamanho**	**Tipo de *botnet***	**# *Bots***
1	6,15	71.971.482	2.824.637	52 GB	Neris	1
2	4,21	71.851.300	1.808.123	60 GB	Neris	1
3	66,85	167.730.395	4.710.639	121 GB	Rbot	1
4	4,21	62.089.135	1.121.077	53 GB	Rbot	1
5	11,63	4.481.167	129.833	37,6 GB	Virut	1
6	2,18	38.764.357	558.920	30 GB	Menti	1
7	0,38	7.467.139	144.078	5,8 GB	Sogou	1
8	19,5	155.207.799	2.954.231	123 GB	Murlo	1
9	5,18	115.415.321	2.753.885	94 GB	Neris	10
10	4,75	90.389.782	1.309.792	73 GB	Rbot	10
11	0,26	6.337.202	107.252	5,2 GB	Rbot	3
12	1,21	13.212.268	325.472	8,3 GB	NSIS.ay	3
13	16,36	50.888.256	1.925.150	34 GB	Virut	1

Tabela 9 – Distribuição de classes em cada cenário do CTU-13

Id	***Background***	***Botnet***	**Normal**
1	10.124.854 (95,40%)	94.972 (0,89%)	392.433 (3,69%)
2	6.071.419 (95,59%)	54.433 (0,85%)	225.336 (3,54%)
3	14.381.899 (94,60%)	75.891 (0,49%)	744.270 (4,89%)
4	3.895.469 (91,91%)	6466 (0,15%)	336.103 (7,93%)
5	416.267 (91,37%)	2129 (0,46%)	37.144 (8,15%)
6	2.031.967 (94,12%)	4927 (0,22%)	121.854 (5,64%)
7	425.611 (93,71%)	293 (0,06%)	28.270 (6,22%)
8	11.451.205 (95,47%)	12.063 (0,10%)	530.666 (4,42%)
9	6.881.228 (90,22%)	383.215 (5,02%)	362.594 (4,75%)
10	4.535.493 (87,54%)	323.441 (6,24%)	321.917 (6,21%)
11	119.933 (29,33%)	277.892 (67,97%)	11.010 (2,69%)
12	119.933 (29,33%)	277.892 (67,97%)	11.010 (2,69%)
13	1.218.140 (93,76%)	21.760 (1,67%)	59.190 (4,55%)

Conforme mencionado na Seção 3.2.2, o conjunto de dados IoT-23 possui vinte e três cenários de captura de tráfego em uma rede de dispositivos IoT. Em vinte destes cenários foram executados *malwares* simulando ataques, comunicação e atividades maliciosas (como varredura de portas) específicas de *botnets*. Os cenários foram selecionados com

base nos seguintes critérios: (i) número de classes presentes; (ii) relação da quantidade de fluxos maliciosos com a quantidade de fluxos benignos; e (iii) tipo de *botnet*. Na Tabela 10 é detalhada a quantidade de dados e na Tabela 11 a distribuição de classes em cada cenário selecionado no IoT-23.

Nota-se que os cenários selecionados de ambas as bases (CTU-13 e IoT-23) são extremamente desbalanceadas. Isso é comum nesse tipo de base pois esses conjuntos de dados espelham o tráfego de rede ocorrido em ambientes reais, onde a maioria do tráfego é normal e a minoria é ataque. Para manter a análise focada em requisitos mais próximos dos reais, não serão adotadas estratégias de mitigação de desbalanceamento.

Tabela 10 – Quantidade de dados em cada cenário do IoT-23

Id	**Nome**	**Duração (h)**	**# Pacotes**	**# Fluxos**	**Tamanho**	**Tipo de *botnet***
1	34-1	24	233.000,00	23.146,00	121 MB	Mirai
4	49-1	8	18.000.000,00	5.410.562,00	1,3 GB	Mirai
14	35-1	24	46.000.000,00	10.447.796,00	3,6 GB	Mirai
15	48-1	24	13.000.000,00	3.394.347,00	1,2 GB	Mirai
19	3-1	36	496.000,00	156.104,00	56 MB	Muhstik
20	1-1	112	1.686.000,00	1.008.749,00	140 MB	Hide and Seek

Tabela 11 – Distribuição de classes nos cenários selecionados do IoT-23

Id	**Rótulo**	**# Fluxos**	**%**
1	Benign	1,923	8,31%
	C&C	6,706	28,97%
	DDoS	14,394	62,19%
	PartOfAHorizontalPortScan	122	0,53%
4	Benign	3,665	0,07%
	C&C	1,922	0,04%
	C&C-FileDownload	1	0,0%
	PartOfAHorizontalPortScan	5,404,959	99,9%
14	Attack	3	0,0%
	Benign	8,262,389	79,08%
	C&C	81	0,0%
	C&C-FileDownload	12	0,0%
	DDoS	2,185,302	20,92%
15	Attack	2,752	0,08%
	Benign	3,734	0,11%
	C&C-HeartBeat-Attack	834	0,02%
	C&C-HeartBeat-FileDownload	11	0,0%
	C&C-PartOfAHorizontalPortScan	888	0,03%
	PartOfAHorizontalPortScan	3,386,119	99,76%
19	Attack	5,962	3,82%
	Benign	4,536	2,91%
	C&C	8	0,01%
	PartOfAHorizontalPortScan	145,597	93,27%
20	Benign	469,275	46,52%
	C&C	8	0,0%
	PartOfAHorizontalPortScan	539,465	53,48%

5.1.2 Atributos

A seleção de atributos e pré-processamento do conjunto de dados CTU-13 foi realizada conforme proposto por Costa et al. (2018). Isso foi feito com intuito de viabilizar as comparações realizadas no experimento da Seção 5.2.1. Primeiro, todos os fluxos CTU-13 foram classificados por seu horário de término. Este tempo foi calculado a partir da adição do horário de início com a duração do fluxo. Em seguida, um conjunto de atributos para um fluxo de rede foi calculado. Para facilitar a comparação com os dois métodos, foi definido utilizar os mesmos atributos descritos em (COSTA et al., 2018). A última etapa do pré-processamento foi a geração de um arquivo .arff (requerido pelo *framework* MOA) contendo os atributos e rótulos de cada cenário CTU-13. Na Tabela 12 estão listados os atributos selecionados para a base CTU-13.

Tabela 12 – Lista de atributos selecionados da base CTU-13

Atributo	Descrição
Duração	Duração total do fluxo em segundos
Protocolo	Protocolo da Transação. Por exemplo, TCP ou UDP
Porta de Origem	Porta de origem da conexão
Porta de Destino	Porta de destino da conexão
Direção	Direção do fluxo. Se refere à origem da conexão (de saída, de entrada, ambos ou desconhecido) e também combina informação relacionada ao estado da transaçãon (*NORMAL*, *RESET*, *TIME OUT*, etc)
Estado	Relacionado ao estado básico da transação. Por exemplo, *REQ|INT (requested|initial), ACC (accepted), EST|CON (established|connected), CLO (closed), TIM (timeout)*, etc.
sTos	Tipo de serviço da origem para o destino
dTos	Tipo de serviço do destino para a origem
Total de pacotes	Número de pacotes no fluxo
Total de *bytes*	Soma de *bytes* no fluxo
Bytes da Origem	Soma dos *bytes* da origem
Média de *bytes* por pacote	A quantidade média de *bytes* por pacote

Para o conjunto IoT-23 foi seguida a mesma estratégia de seleção e pré-processamento anterior, entretanto os atributos não são exatamente os mesmos que estão disponíveis no conjunto de dados CTU-13. A Tabela 13 tem a listagem dos atributos selecionados referente a base IoT-23. Conforme mencionado na Seção 3.2.2 do Capítulo 3, os fluxos no IoT-23 foram gerados a partir do tráfego de rede utilizando a ferramenta *Zeek*, o que implica em pequenas diferenças ao definir os atributos. Sendo assim, os atributos diferiram da seguinte forma:

- não existem os atributos: Duração, Direção, sTos, dTos, Total de pacotes e *Bytes* da Origem
- foi incluído o atributo “Total de pacotes da origem” para poder ter uma correspondência com o atributo “Total de pacotes” do CTU-13.

Tabela 13 – Lista de atributos selecionados da base IoT-23

Atributo	Descrição
Porta de Origem	Porta de origem da conexão
Porta de Destino	Porta de destino da conexão
Protocolo	Protocolo da Transação. Por exemplo, TCP ou UDP
Estado	Relacionado ao estado básico da transação. Por exemplo, *REQ|INT (requested|initial), ACC (accepted), EST|CON (established|connected), CLO (closed), TIM (timeout)*, etc.
Total de pacotes da origem	Número de pacotes vindos da origem
Média de *bytes* por pacote	A quantidade média de *bytes* por pacote
Total de pacotes	Número de pacotes no fluxo
Total de *bytes*	Soma de *bytes* no fluxo

5.2 Experimentos e resultados

Os resultados de cada experimento são apresentados e discutidos nesta seção. A Tabela 14 consolida as características de cada experimento, como por exemplo: as informações de parametrização, algoritmos, quantidade de classes, conjuntos de dados, etc. Em todos os experimentos é comum o uso da *ALPrenquentialEvaluationTask* como Tarefa de Avaliação do Classificador e o *ALWindowClassificationPerformanceEvaluator* como Avaliador de Desempenho, os quais foram apresentados na Seção 2.4.3 do Capítulo 2. Considere também que:

- experimentos 1 e 2 abrangem a Seção 5.2.1 *Comparação do uso de um único classificador em contraste ao uso de conjuntos de classificadores*;
- experimento 3 está relacionado com a Seção 5.2.2 *Análise dos resultados da classificação utilizando janelas*;
- experimento 4 está relacionado a Seção 5.2.3 *Análise do desempenho de conjuntos de classificadores utilizando estratégias de aprendizado ativo*
- experimento 5 é referente a Seção 5.2.4 *Análise de classificadores de conjuntos utilizando conjuntos de dados multi-classe*

A seguinte notação foi utilizada na Tabela 14: a coluna "#" indica o número do experimento; as colunas "Conjunto de Dados" e "Cenários" apontam o conjunto de dados usado ao longo do experimento e os respectivos cenários; "Alg. classif." indica o algoritmo de classificação; "Classif." indica se foi utilizado Classificador Único (abreviado para "Classif. Único") ou Conjunto de Classificadores (abreviado para "Conj. Classif."); a coluna "Janela" indica o tamanho definido para a janela temporal; e "Passo" o valor dado para o passo incremental.

Tabela 14 – Tabela consolidada com as características de cada experimento

#	Seção	Conjunto de Dados	Cenários	Alg. Classif.	Estr. Classif.	# classif.	Classif.	Janela	Passo
1	5.2.1	CTU-13	1 a 13 (exceto 7)	-	Classif. Único	1	Hoeffding, H. Adapt.	=tamanho conjunto	100
2	5.2.1	CTU-13	1 a 13 (exceto 7)	Hoeffding	Conj. Classif.	10	Ozabag, Ozabag c/ Adwin, Ozaboost	=tamanho conjunto	100
3	5.2.2	CTU-13	2 e 8	Hoeffding	Conj. Classif.	10	Ozaboost	1.000	-
4	5.2.3	CTU-13	2 e 8	Hoeffding	Conj. Classif.	10	Ozaboost	1.000	-
5	5.2.4	IoT-23	1, 4, 14, 15, 19 e 20	Hoeffding	Conj. Classif.	10	Ozaboost	=tamanho conjunto	100

5.2.1 Comparação do uso de um único classificador em contraste ao uso de conjuntos de classificadores

Tal como em todos os outros experimentos deste trabalho, neste foi utilizado como algoritmo de classificação, para ambas as estratégias de classificadores únicos e de conjunto, a *Hoeffding Tree*. O desempenho de classificadores únicos foi comparado com várias versões de conjuntos de classificadores considerando cada um dos cenários selecionados da base CTU-13 (detalhado na Seção 5.1.1). Seguindo a abordagem do trabalho de (COSTA et al., 2018), utilizado como base para a comparação, o número máximo de instâncias da classe *botnet* para treinamento e atualização do modelo de decisão ($max\ n$) foi definido como:

- ❏ 10.000, para os cenários 1, 2, 3, 4, 5, 6, 8, 10, 11 e 12 do CTU-13;
- ❏ 40.000, para o cenário 9 do CTU-13;
- ❏ 20.000, para o cenário 13 do CTU-13.

Observe que foi excluído apenas o cenário 7 da avaliação devido ao baixo número de fluxos de *botnet* (63). Mesmo com apenas 901 fluxos de *botnet*, o cenário 5 foi incluído para investigar uma situação potencialmente problemática para o conjunto de classificadores. O passo incremental também foi definido como no trabalho de Costa et al. (2018), iniciando com valor de 100 e depois incrementado em 100 (para cada execução) até atingir o valor limite de cada cenário ($max\ n$). Além disso, a janela temporal abrangia todo o conjunto de dados, ou seja, o tamanho da janela era igual a quantidade total de instâncias presente no conjunto. A configuração deste experimento, com valores para cada variável do modelo proposto, é apresentada na Figura 15.

O desempenho dos algoritmos em cada cenário é mostrado na Tabela 16. A coluna "média" refere-se à média entre a precisão e a revocação. Essa métrica foi usada para calcular o *best n*, tal como explicado na Seção 4.2 do Capítulo 4.

Tabela 15 – Configuração para experimento de comparação de classificadores

Variável	Valor
D	CTU-13 (binário).
P	100.
S	*max n*, sendo: 10.000, para os cenários 1, 2, 3, 4, 5, 6, 8, 10, 11 e 12 do CTU-13; 40.000, para o cenário 9 do CTU-13; 20.000, para o cenário 13 do CTU-13.
C	Classificadores únicos (*Hoeffding Tree* e *Hoeffding Adaptive Tree*) e Conjuntos de Classificadores (*Ozabag*, *Ozabag com Adwin* e *Ozaboost*).
A	*Hoeffding Tree*.
T	Valor corresponde ao tamanho de cada conjunto de dados.

Tabela 16 – Comparação de desempenho entre os algoritmos considerando média e *best n* obtidos em cada cenário selecionado do CTU-13

	Classificadores Únicos				*Conjuntos de Classificadores*					
	Hoeffding		**H. Adaptive**		**Ozabag**		**Ozabag c/ Adwin**		**Ozaboost**	
#	**média**	**best n**	**média**	**best n**	**média**	**best n**	**média**	**best n**	**média**	**best n**
1	93,0%	6200	87,5%	5300	90,7%	4400	94,6%	6700	96,8%	6400
2	78,7%	8700	70,1%	2700	85,9%	9900	66,3%	900	84,1%	8100
3	96,4%	1300	98,7%	1500	92,1%	1200	90,5%	1200	95,1%	100
4	96,2%	1900	95,2%	1700	95,9%	1700	97,5%	1700	89,9%	1500
5	89,2%	800	73,5%	700	77,0%	700	87,0%	800	87,3%	700
6	92,1%	200	89,3%	200	92,9%	200	91,3%	200	91,0%	200
8	94,7%	900	95,4%	1100	96,7%	2700	94,9%	1100	95,7%	900
9	88,7%	32900	89,2%	2000	88,4%	3600	85,4%	1700	92,7%	4800
10	98,9%	200	98,0%	200	98,8%	200	99,4%	200	97,7%	200
11	99,6%	100	99,8%	100	99,4%	100	99,7%	100	99,7%	100
12	56,1%	1600	56,8%	900	63,9%	1400	76,6%	1600	86,9%	1600
13	55,7%	19400	63,0%	10700	58,9%	16900	62,2%	6800	68,4%	1700

O algoritmo *Ozaboost* alcançou os melhores resultados com relação aos valores de precisão/revocação e *best n*. O valor de *best n* é o menor nos cenários 3, 4, 5, 8 e 13, e igual aos demais em 6, 10 e 11. Considerando o valor médio, o *Ozaboost* tem valores maiores nos cenários 1 , 9, 12 e 13. Também é importante mencionar que, conforme mostrado nos gráficos das Figuras 9 e 10, antes e depois de atingir o *best n*, o *Ozaboost* é o algoritmo que mostra menos flutuações nas medições de revocação e precisão. Com base nesses resultados, o algoritmo *Ozaboost* foi escolhido para ser comparado com a linha de base deste experimento. Os resultados obtidos corroboram a literatura, visto que conjuntos de classificadores são compostos por vários modelos cujas previsões individuais são combinadas em uma previsão final, frequentemente mais precisa e escalável do que em classificadores únicos (BIFET G. HOLMES; GAVALDà, 2009).

5.2.1.1 Comparação com a linha de base

O algoritmo *Ozaboost* foi selecionado e em seguida comparado com a linha de base deste experimento (COSTA et al., 2018). A Tabela 17 mostra a comparação das métricas coletadas no momento em que cada algoritmo atinge o *best n* para cada cenário CTU-13. As métricas usadas aqui são: a porcentagem de instâncias rotuladas apresentadas

ao classificador e os respectivos *best n* relacionados às classes *botnet* e *background* (BG), além dos valores de precisão e revocação, indicados respectivamente pelas colunas "Prec." e "Rev.". Destacados em negrito estão os valores mais altos obtidos ao comparar os resultados da linha de base com o do *Ozaboost*.

Tabela 17 – Desempenho para o *best n* - (COSTA et al., 2018) versus Ozaboost MOA

	Trabalho base (COSTA et al., 2018)				**MOA - Ozaboost com Hoeffding**			
#	**Botnet %**	**BG %**	**Prec.**	**Rev.**	**Botnet %**	**BG %**	**Prec.**	**Rev.**
1	16,3% (6700)	49,9% (1389655)	0,92	0,91	**15,3% (6400)**	**48,9% (1346359)**	**0,98**	**0,95**
2	41,1% (8600)	25,8% (461868)	0,67	0,69	**38,6% (8100)**	**23,3% (414288)**	**0,78**	**0,89**
3	31,7% (8500)	75,6% (3545024)	0,96	**0,96**	**0,3% (100)**	**37,8% (1726299)**	**0,99**	0,90
4	65,9% (1700)	90,1% (1007538)	0,96	**0,94**	**58,1% (1500)**	**87,7% (958761)**	**0,98**	0,81
5	-	-	-	-	**77,6% (700)**	**80,8% (100396)**	**0,89**	**0,85**
6	28,1% (1300)	30,1% (166930)	0,95	**0,95**	**4,3% (200)**	**9,0% (49212)**	**0,97**	0,84
8	31,0% (1900)	43,0% (1269715)	0,93	**0,94**	**14,6% (900)**	**19,5% (560680)**	**0,99**	0,92
9	15,2% (28200)	58,5% (1114439)	0,92	**0,92**	**2,5% (4800)**	**50,0% (1262782)**	**0,96**	0,88
10	0,4% (500)	40,5% (487595)	**0,99**	0,99	**0,1% (200)**	**40,3% (478600)**	0,95	**0,99**
11	2,4% (200)	86,6% (85846)	0,99	0,99	**1,2% (100)**	**86,1% (82974)**	**0,99**	**0,99**
12	78,4% (1700)	72,6% (234991)	0,66	0,42	**73,8% (1600)**	**68,5% (216237)**	**0,89**	**0,84**
13	49,7% (19900)	40,2% (758694)	0,635	0,45	**4,2% (1700)**	**4,0% (74129)**	**0,91**	**0,45**

Em todos os cenários, o *Ozaboost* precisou de um número menor de instâncias rotuladas de *botnet* para atingir os valores máximos de revocação e precisão. Os cenários 3, 6, 8, 9 e 13, por exemplo, mostram uma diminuição substancial nas instâncias rotuladas necessárias correspondendo a 98,82%, 84,61%, 52,63%, 82,97% e 91,45%, respectivamente.

Os valores de precisão e revocação também foram melhores usando o *Ozaboost*, sendo significativamente maiores do que a linha de base na maioria dos cenários. Valores mais altos de precisão e revocação foram alcançados utilizando menos instâncias rotuladas para treinamento e atualização do modelo de decisão. Nesse caso, considerando a precisão para os cenários 2, 12 e 13, houve aumento de 15,7%, 34,48% e 43,3%, respectivamente.

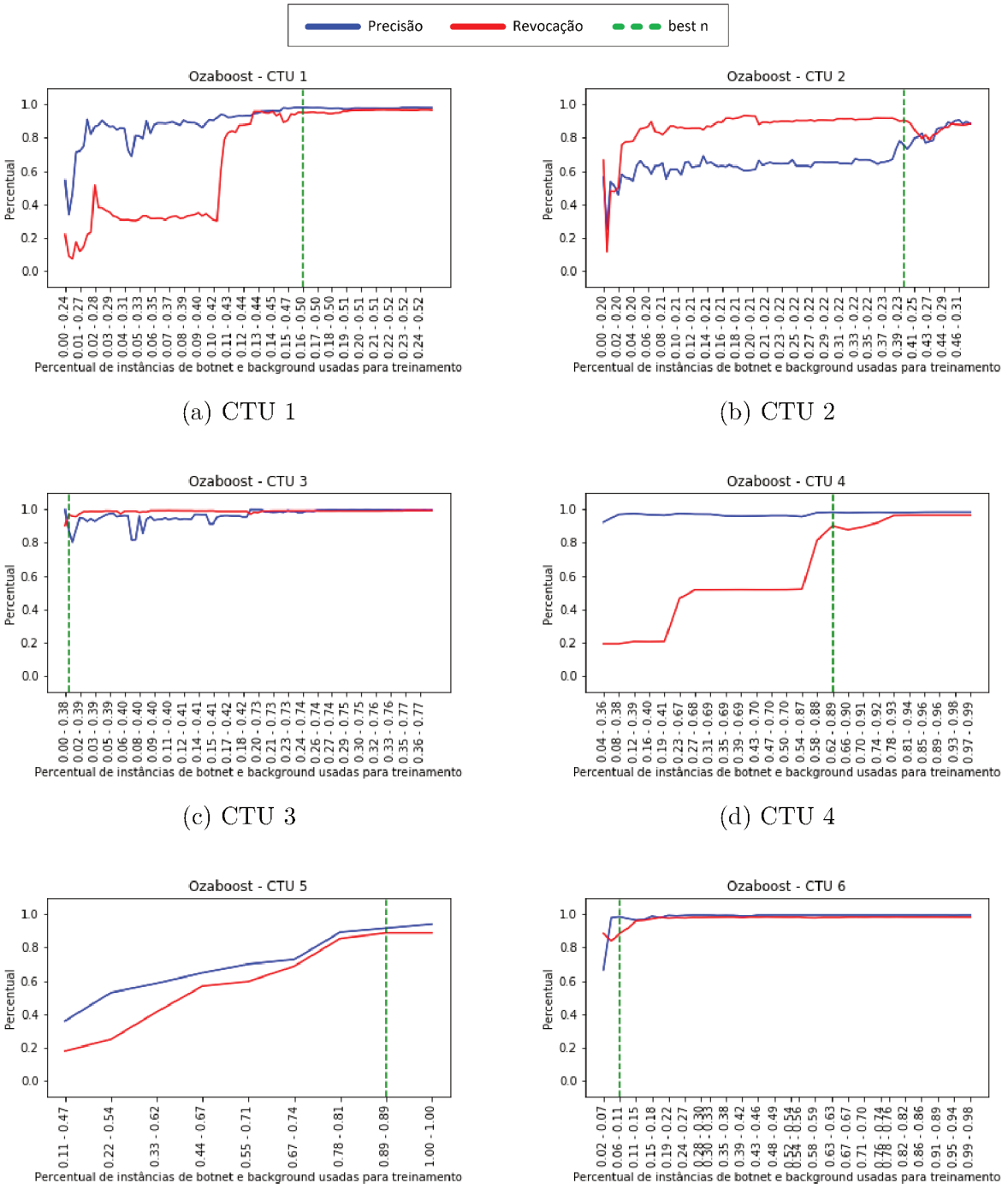
Apenas para o cenário 10 ocorreu uma diminuição de aproximadamente 4% na precisão do *Ozaboost*, mas com o uso apenas 0,1% das instâncias (200) de *botnet* rotuladas, contra 0,4% (500) da linha de base. Isso representa uma redução de 75% ou 400 instâncias rotuladas. Em relação aos valores de revocação, a linha de base apresenta melhor desempenho, por uma pequena diferença, em cinco cenários (3, 4, 6, 8 e 9). Isso pode ser parcialmente explicado pela diferença de instâncias rotuladas necessárias (*botnet* e *background*) nesses cenários. Por exemplo, o *Ozaboost* no cenário 3 usou apenas 0,3% das instâncias de *botnet* rotuladas contra 31,7% da linha de base. É uma redução de 98,82%, representando 8.400 instâncias de *botnet* e 1.818.725 instâncias *background*. Isso implica que o modelo gerado pela linha de base recebeu uma grande variedade de instâncias, aumentando, consequentemente, a revocação.

5.2.1.2 Desempenho do Ozaboost para cada cenário CTU-13

A seguir, desempenho do classificador *Ozaboost* para cada cenário do CTU-13 será analisado. Também será feita uma discussão sobre o impacto das características da *botnet*

no modelo de decisão e o número de instâncias rotuladas necessárias.

As Figuras 9 e 10 mostram os gráficos com o valor do *best n* e as métricas de precisão e revocação para cada cenário do CTU-13, exceto o cenário 7. O eixo X representa o percentual atingido para precisão e revocação, e o eixo Y o percentual de instâncias de *botnet* e *background* ao longo do conjunto que foram utilizadas para treinamento. É interessante ver que, em alguns cenários, o valor *best n* é obtido com uma pequena quantidade de instâncias de *botnet*/*background* rotuladas.



(a) CTU 1

(b) CTU 2

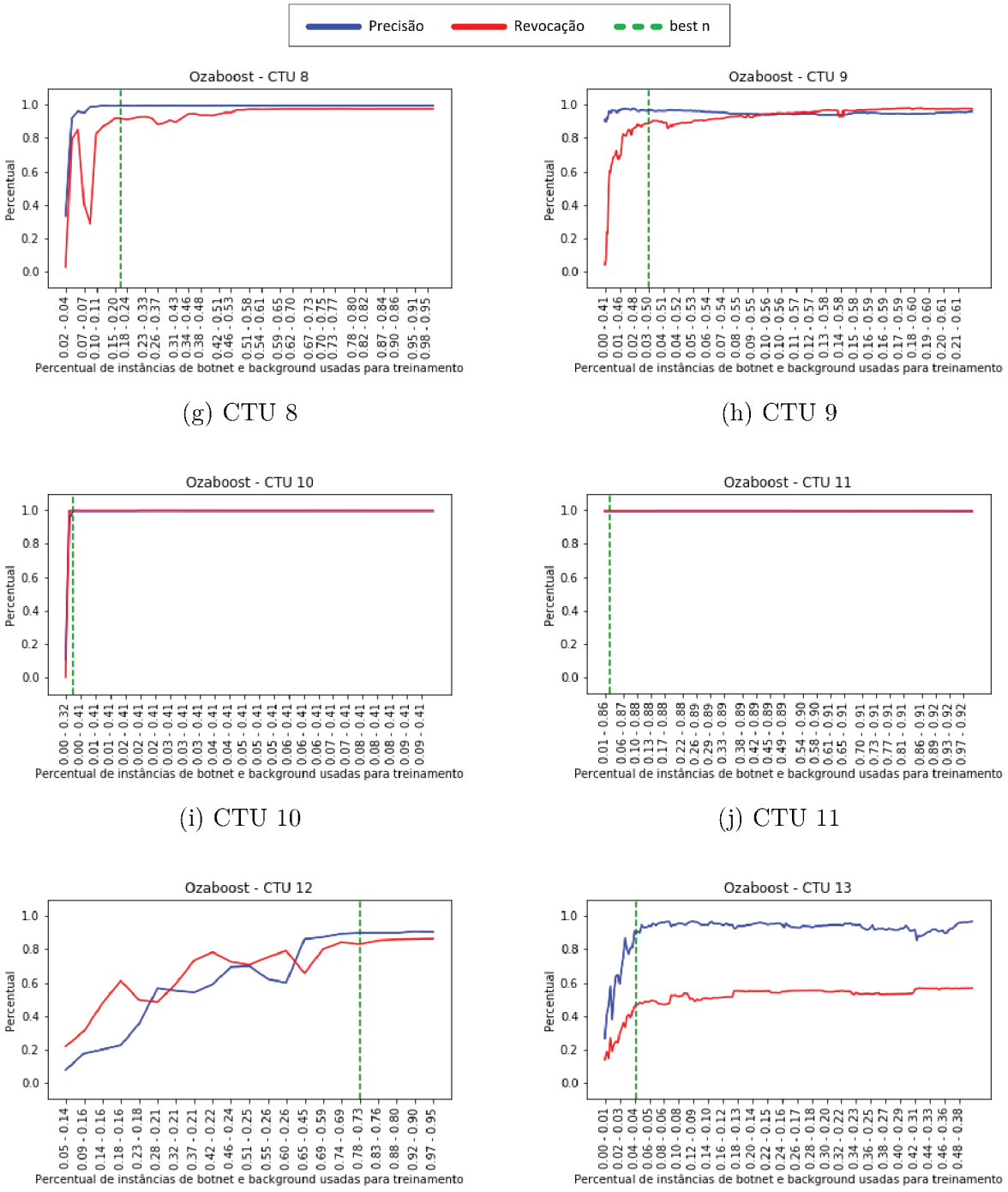
(c) CTU 3

(d) CTU 4

(e) CTU 5

(f) CTU 6

Figura 9 – Desempenho do *Ozaboost* nos cenários 1, 2, 3, 4, 5 e 6 do CTU-13



(g) CTU 8

(h) CTU 9

(i) CTU 10

(j) CTU 11

(l) CTU 12

(m) CTU 13

Figura 10 – Desempenho do *Ozaboost* nos cenários 10,11,12 e 13 do CTU-13

A Tabela 18 apresenta o desempenho do *Ozaboost* para cada tipo de *botnet*. É possível notar que quando a *botnet* usa o protocolo IRC, menos de 40% das instâncias rotuladas do *botnet* são necessárias para atingir *best n*. O cenário 4 é a única exceção aqui. Este cenário parece ser problemático porque tem poucos fluxos de *botnet* disponíveis. No entanto, com 58% das instâncias de *botnet* rotuladas, foi possível atingir um valor médio de precisão e revocação de 89%. Trabalhos anteriores, como de Pektaş e Acarman (2018) e Chen (2017), geralmente aplicam técnicas como validação 70/30 ou 80/20 (proporção

de treinamento/teste) e *10-fold-cross-validation* (treinamento/teste com razão de 90/10) as quais envolvem o uso de mais instâncias rotuladas para criar o modelo de decisão. Isso mostra o potencial das técnicas de mineração de fluxo combinadas a conjuntos de classificadores para criar métodos de detecção de *botnet* mais adequados ao problema.

Outra análise diz respeito ao protocolo HTTP. Considerando o cenário 5, apesar do bom valor para a média de precisão/revocação, foi necessário 77% das instâncias rotuladas de *botnet*. Por outro lado, o cenário 13 consumiu uma pequena porcentagem de instâncias rotuladas, mas não teve um bom desempenho como os outros. Seria importante investigar se esse comportamento também ocorre em outras *botnets* HTTP e se atributos específicos poderiam melhorar métricas de precisão e revocação (ACARALI et al., 2016). O cenário 12 (tráfego de *botnet* P2P) mostrou um comportamento semelhante. Outros experimentos devem ser realizados para verificar se tal resultado está relacionado ao baixo número de fluxos disponíveis ou às características específicas do protocolo P2P (ALAUTHAMAN et al., 2018).

Tabela 18 – CTU-13 x Tipo de *Botnet* x Ozaboost

CTU	**Família**	**Protocolo**	**Média**	**best n**	**Botnet %**	**BG %**
1	Neris	IRC	96,8%	6400	15,62	48,94
2	Neris	IRC	84,1%	8100	38,68	23,30
3	Rbot	IRC	95,1%	100	0,37	37,82
4	Rbot	IRC	89,9%	1500	58,13	87,78
5	Virut	HTTP	87,3%	700	77,69	80,85
6	Menti	IRC	91,0%	200	4,31	9,00
8	Murlo	IRC	95,7%	900	14,68	19,53
9	Neris	IRC	92,7%	4800	2,59	50,09
10	Rbot	IRC	97,7%	200	0,18	40,32
11	Rbot	IRC	99,7%	100	1,22	86,14
12	NSIS.ay	P2P	86,9%	1600	73,80	68,55
13	Virut	HTTP	68,4%	1700	4,24	4,00

5.2.2 Análise dos resultados da classificação utilizando janelas

Apesar do bom desempenho do *Ozaboost*, tanto em termos de identificação do tráfego *botnet* e uso de instâncias rotuladas, os gráficos nas Figuras 9 e 10 mostram apenas um resultado cumulativo para todo o fluxo. Considerando que, comumente na literatura de fluxos contínuos de dados, classificadores são avaliados ao longo do fluxo, usando janelas de dados, este experimento realizou esse tipo de avaliação. Portanto, o objetivo principal deste experimento é investigar o desempenho do modelo, em termos de precisão e revocação, considerando janelas de dados ao longo do fluxo.

No cenário 5.2.1 a conclusão é de que o algoritmo com melhor desempenho é o *Ozaboost*. Levando isso em consideração, neste experimento todas as avaliações realizadas utilizaram o conjunto de classificadores *Ozaboost* e agora com janelas temporais de tamanho 1.000 instâncias. Este tamanho foi definido após testes com janelas de tamanhos variados (100, 500, 1.000, 2.000, 5.000 e 10.000) sendo que com 1.000 instâncias os resultados obtidos por janela apresentaram maior correspondência com a distribuição das classes ao longo do conjunto de dados. Por conta disso, não há mais um passo incremental, e o limite usado para liberação do rótulo verdadeiro para avaliação do modelo de decisão passa a ser o *best n* atingido pelo *Ozaboost* no conjunto de dados.

Outro ponto importante neste experimento é que foi mantido como base dos dados o CTU-13, mas considerados somente os cenários 2 e 8, cujos valores de *best n* são respectivamente 8.100 e 900. Estes cenários foram selecionados com base nos seguintes fatores:

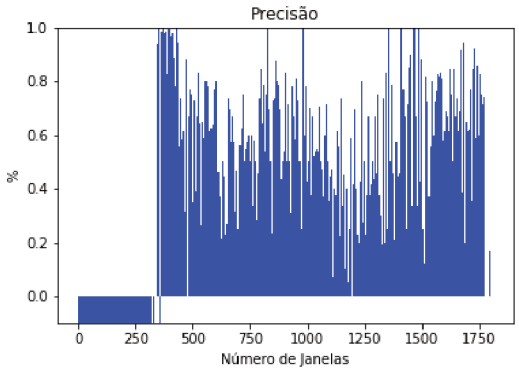
- ❏ no experimento de Seção 5.2.1, ambos possuíam número máximo de instâncias de *botnet* para treinamento igual a 10.000;
- ❏ são de tipos de *botnet* diferentes (*Neris* para cenário 2, e *Murlo* para o 8);
- ❏ apresentam baixo percentual da classe *botnet* no conjunto (cenário 2 com 0,85%, e 8 com 0,10%);
- ❏ o cenário 2 foi um dos cenários em que todas as medidas de avaliação foram melhores que o trabalho base utilizado na Seção 5.2.1;
- ❏ o cenário 8 apresentou redução considerável no número de instâncias rotuladas necessárias e manteve altos valores de precisão e revocação, conforme visto na Seção 5.2.1.

Os valores de configuração para este experimento estão na Tabela 19. A Figura 11 ilustra os resultados das medidas de precisão e revocação por janela. Tais gráficos foram gerados a partir do cálculo individual das medidas em uma janela de tamanho 1.000. O eixo X representa a quantidade de janelas de tamanho 1.000 ao longo do conjunto. O eixo Y representa o percentual alcançado em precisão ou revocação, dependendo do gráfico. Os valores no eixo Y que estão abaixo de zero (0.0) correspondem as janelas em

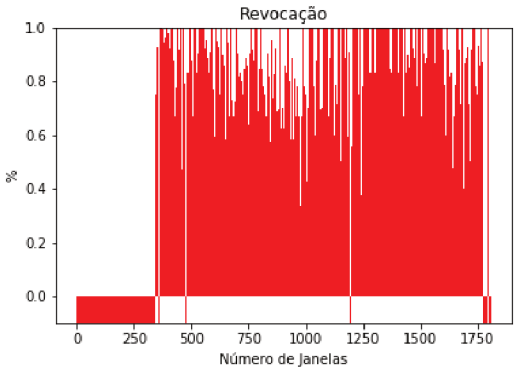
Tabela 19 – Configuração para experimento utilizando janelas

Variável	Valor
D	CTU-13 (binário).
P	Não se aplica, execução única por conjunto.
S	*max* n de 8.100 para o cenário 2 e 900 para o cenério 8. Corresponde ao valor *best* n obtido com o *Ozaboost* para cada cenário.
C	Conjunto de Classificadores *Ozaboost*.
A	*Hoeffding Tree*.
T	1.000.

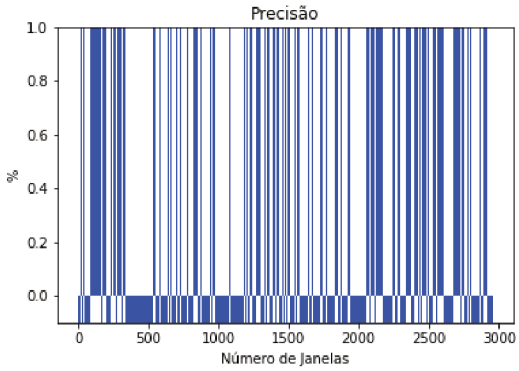
que o modelo não conseguiu calcular os valores de precisão ou revocação para a classe de ataque, pois não há presença de instâncias dessa classe. Tais janelas correspondem a momentos do fluxo onde não existem instâncias de ataque o que impede o cálculo de tais medidas, conforme definido na Seção 2.6. No experimento da Seção 5.2.1 os valores de precisão e revocação eram acumulados para cada execução, sendo calculados com base em uma janela cujo tamanho era igual ao tamanho total do conjunto. Já nos gráficos deste experimento, temos valores separados para janelas com tamanho menor e pré-definido, permitindo analisar o comportamento do classificador ao longo do fluxo de dados.



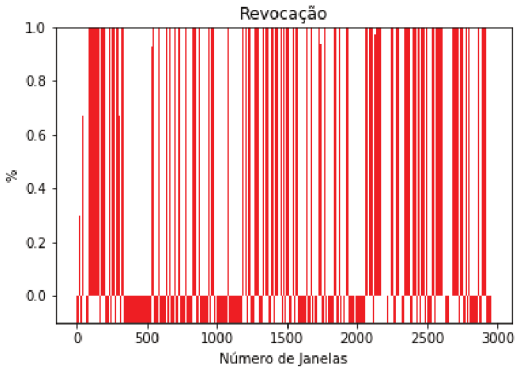
(a) CTU 2 - *best* n - Precisão



(b) CTU 2 - *best* n - Revocação



(c) CTU 8 - *best* n - Precisão



(d) CTU 8 - *best* n - Revocação

Figura 11 – Desempenho *Ozaboost* por janela nos cenários 2 e 8 do CTU-13

No caso do cenário 2, nota-se que do início do conjunto até aproximadamente 300.000 instâncias, não foram obtidos valores de precisão ou revocação por janela, aparecendo no

gráfico como linhas abaixo do valor zero (sendo que cada linha representa uma janela temporal de 1.000 instâncias ou fluxos). O modelo de decisão não conseguiu calcular os valores para precisão ou revocação pois não existia instância de ataque nas janelas. Pela distribuição de classes deste cenário, mostrada na Figura 12, é possível constatar que no início do conjunto há uma maioria absoluta de instâncias da classe *background*, o que torna o modelo de decisão bem mais aderente a esta classe. Como a existência de instâncias com classe *botnet* nesta parte do conjunto é praticamente inexistente, quaisquer variações recebidas pelo modelo ocasionariam em erro, até que o mesmo fosse treinado com instâncias suficientes de *botnet* para garantir maior taxa de acerto (o que começa a acontecer a partir de 300.000 instâncias).

A partir do ponto mencionado no parágrafo anterior, os fluxos de ataque chegam e os valores de precisão e revocação são calculados. É possível notar um bom desempenho do classificador nas primeiras janelas após o surgimento dos ataques. Contudo, em alguns pontos a precisão foi baixa e até chegou a 0. A revocação, no entanto, se comportou melhor. Isso indica que o modelo conseguiu detectar a maioria dos fluxos maliciosos mas ao custo de falsos positivos. Em geral, é preferível que o um modelo de detecção de *botnets* exiba justamente esse comportamento com valores de revocação maiores do que de precisão (WANG; PASCHALIDIS, 2017).

Ao comparar o presente resultado com o desempenho do cenário anterior é possível notar que em nenhum momento a precisão ou revocação atingiram valor zero (Figura 9). Esse comportamento ocorre, pois, no experimento anterior da Seção 5.2.1 os valores eram acumulados e neste experimento existe uma separação por janela, permitindo investigar o que ocorre em cada janela definida. Considerando que neste experimento as instâncias do cenário 2 são liberadas para treinamento e atualização do modelo até completar 8.100 instâncias de *botnet*, o gráfico de distribuição de classes na Figura 12 mostra que o conjunto de dados possui uma quantidade relevante de instâncias desse tipo dentro do limite de treinamento, e esta classe é presente, mesmo que em menor quantidade, ao longo de todo o conjunto. Isso contribuiu para a geração de um modelo de decisão razoavelmente consistente, que consegue detectar os fluxos maliciosos (alta revocação) dentro das janelas definidas.

Em suma, a avaliação do desempenho do classificador no CTU-2 utilizando medidas calculadas ao longo do fluxo, revelou diversos momentos onde a precisão foi baixa e até chegou a 0. Isso mostra que o valor de $0,78$ (Tabela 17), obtido com a avaliação ao fim de todo o fluxo de dados oculta as situações onde o classificador teve um desempenho ruim dificultando a proposta de melhorias nos modelos de decisão.

O desempenho do cenário 8 no experimento anterior (vide Tabela 17 e Figura 9) mostra altos valores de precisão $(0,99)$ e revocação $(0,92)$ ao custo de um baixo número de instâncias rotuladas. Portanto, a ideia aqui é avaliar se existe alguma janela onde o desempenho do modelo não será satisfatório. A Figura 11 mostra os valores de precisão

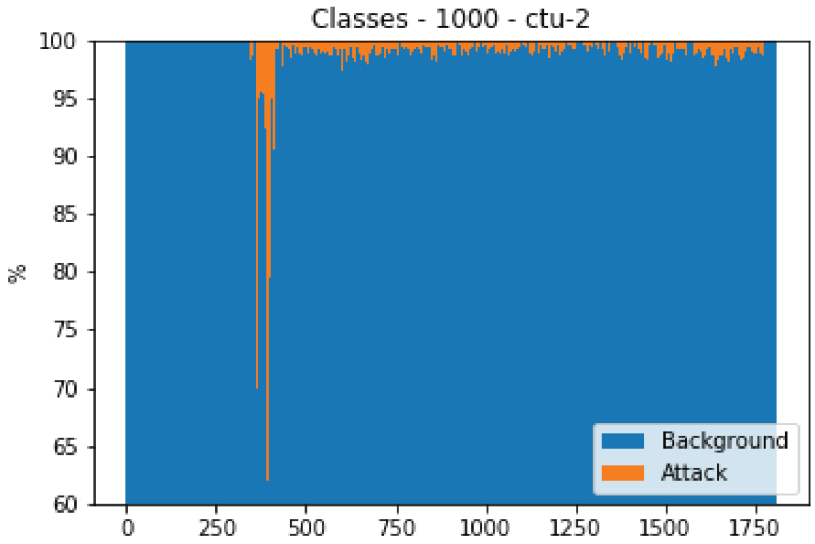


Figura 12 – Distribuição de classes no cenário 2 do CTU-13. Escala do eixo Y limitada de 60 a 100%

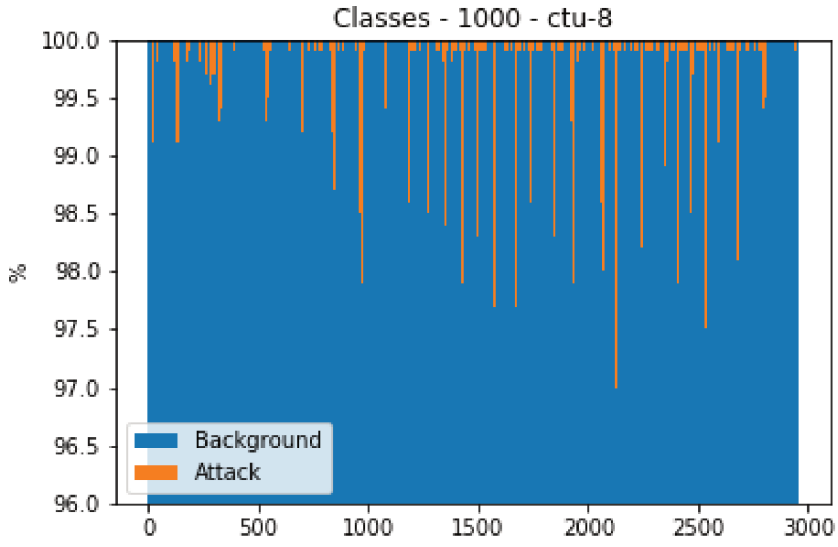


Figura 13 – Distribuição de classes no cenário 8 do CTU-13. Escala do eixo Y limitada de 96 a 100%

e revocação em cada uma das janelas. É possível notar que praticamente em todas as janelas onde existem fluxos *botnet* os valores de precisão foram de 100%. Comportamento semelhante ocorreu com os valores de revocação. A exceção pode ser vista no início das janelas onde os valores de revocação começam por volta de 30% e 70% antes de subirem para 100%. Isso mostra que as instâncias rotuladas apresentadas ao classificador foram suficientes para detectar corretamente as instâncias *botnet* ao decorrer do fluxo de dados.

Os resultados obtidos no presente experimento mostram que é necessário analisar os

modelos de detecção de intrusão baseados em aprendizado de máquina usando outras medidas. Considerar somente os valores de precisão e revocação calculados para todo o conjunto de teste pode levar a conclusões equivocadas sobre a efetividade do modelo. Além disso, a análise dos resultados da classificação utilizando janelas mostrou a existência de oscilações nas medidas ao longo do conjunto (em particular no cenário 2), diferente do que foi mostrado nos resultados cumulativos da Seção 5.2.1 e também em outros trabalhos relacionados.

Por fim, os resultados indicam a importância de investigar o modelo de decisão gerado para entender as causas associadas a diminuição das medidas. Somente uma possibilidade foi parcialmente explorada aqui, que é análise da distribuição do tipo de tráfego dentro de cada janela. A investigação detalhada do conteúdo dos fluxos em janelas com baixos valores de precisão e revocação pode fornecer importantes *insights* acerca do desempenho do classificador.

5.2.3 Análise do desempenho de conjuntos de classificadores utilizando estratégias de aprendizado ativo

Os resultados apresentados na Seção 5.2.2 ressaltam o impacto do fornecimento de rótulos no desempenho de modelos de aprendizado incremental. Além disso, em cenários reais seria difícil prever o valor do *best n*, e ainda que a princípio possa ser determinado utilizando uma amostra dos dados, a medida que o fluxo contínuo de dados evolui novas instâncias rotuladas poderiam ser necessárias para atualizar o modelo. Portanto, o objetivo deste grupo de experimentos é avaliar o uso de diferentes estratégias de aprendizado ativo na escolha de um conjunto de instâncias a serem rotuladas e usada na atualização de modelos de detecção de *botnets*. A análise das estratégias será feita de modo a minimizar o número de instâncias para treinamento e atualização do modelo de decisão e manter os valores de precisão e revocação em um patamar aceitável, considerando como base de comparação os resultados dos experimentos anteriores.

Foram utilizadas duas estratégias de aprendizado ativo. A primeira, discutida na Subseção 5.2.3.1, é a Aleatória (*ALRandom*), que de forma aleatória seleciona as instancias a serem rotuladas. A segunda, discutida na Subseção 5.2.3.2, é a Baseado em Incerteza (*ALUncertainty*) e suas variações: Incerteza Fixa (*FixedUncertainty*), Incerteza Variável (*VarUncertainty*) e Incerteza Variável com Aleatorização (*RandVarUncertainty*). As execuções deste experimento seguiram algumas definições comuns. São elas:

- ❏ tamanho da janela padrão igual a 1.000 instâncias, seguindo mesma abordagem e justificativa do experimento na Seção 5.2.2;
- ❏ conjunto de classificadores *Ozaboost* com a *Hoeffding Tree*;
- ❏ execuções realizadas somente para os cenários 2 e 8 do CTU-13;

❑ sempre zerado o número inicial de instâncias rotuladas para atualizar o modelo;

Os valores de configuração para cada uma das variáveis do modelo proposto para este experimento estão na Tabela 20. Já a Tabela 21 mostra os parâmetros específicos de cada execução. A coluna "#" indica o número de controle da execução. Na coluna "Classificador" tem qual o classificador de aprendizado ativo utilizado. A coluna "Seção" indica a seção deste trabalho correspondente ao classificador. A coluna "Estratégia" indica qual estratégia de aprendizado ativo foi utilizada (não aplicável para o classificador Aleatório). A coluna "Passo" indica o valor do parâmetro *passo* para variação da *capacidade* (apenas para estratégias Variável e Variável com Aleatorização). Por fim a coluna "Limite" tem o valor fixado para o parâmetro *limite* (apenas para a estratégia Fixa). Foram gerados quatro gráficos para cada resultado: precisão por janela, revocação por janela, número absoluto de instâncias requisitadas e número relativo de instâncias requisitadas.

Tabela 20 – Configuração para experimento de aprendizado ativo

Variável	**Valor possível**
D	CTU-13 (Binário)
P	Não se aplica
S	Estratégias Aleatória, Incerteza Fixa, Incerteza Variável e Incerteza Variável com Aleatorização
C	Conjunto de classificadores *Ozaboost*
A	*Hoeffding Tree*
T	1.000

Tabela 21 – Relação de execuções para aprendizado ativo

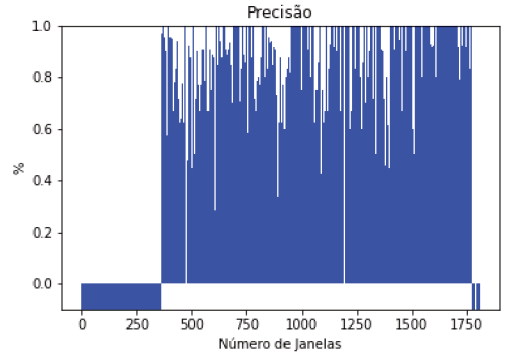
#	**Classificador**	**Seção**	**Estratégia**	**Passo**	**Limite**
1	Aleatório	5.2.3.1	Não se aplica	Não se aplica	Não se aplica
2	Incerteza	5.2.3.2	Incerteza Fixa	Não se aplica	0,5
3	Incerteza	5.2.3.2	Incerteza Fixa	Não se aplica	0,7
4	Incerteza	5.2.3.2	Incerteza Fixa	Não se aplica	0,9
5	Incerteza	5.2.3.2	Incerteza Variável	0,01	Não se aplica
6	Incerteza	5.2.3.2	Incerteza Variável	0,1	Não se aplica
7	Incerteza	5.2.3.2	Incerteza Variável	0,5	Não se aplica
8	Incerteza	5.2.3.2	Incerteza Variável com Aleatorização	0,01	Não se aplica
9	Incerteza	5.2.3.2	Incerteza Variável com Aleatorização	0,1	Não se aplica
10	Incerteza	5.2.3.2	Incerteza Variável com Aleatorização	0,5	Não se aplica

5.2.3.1 Seleção aleatória

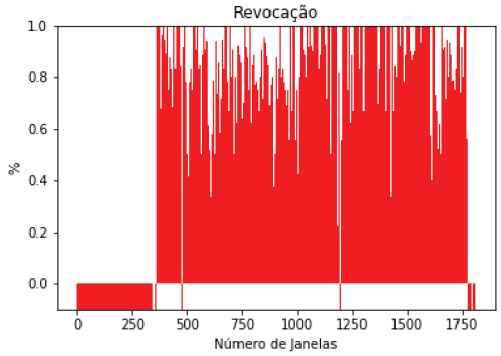
A execução com estratégia de seleção aleatória é a mais simples pois não possui um critério definido para seleção das instâncias para treinamento e atualização do modelo, as instâncias simplesmente são selecionadas ao longo do conjunto de forma aleatória. Por esse motivo ela foi utilizada como base para as demais execuções do experimento com aprendizado ativo. Conforme explicado na Seção 2.5.2, a estratégia de seleção Aleatória (no MOA chamada de *ALRandom*) possui o parâmetro *capacidade* que está relacionado ao percentual de instâncias que o classificador pode selecionar ao longo do tempo para treinar e atualizar o modelo. A *capacidade* foi definida sempre como 0,1 (ou seja, 10%). O algoritmo base de aprendizado utilizado é o *Ozaboost* com a *Hoeffding Tree* e o tamanho da janela temporal é de 1.000 instâncias. Os cenários executados foram 2 e 8 do CTU-13.

A Figura 14 mostra quatro gráficos com os resultados obtidos para o cenário 2. Todos os gráficos possuem no eixo X o número de janelas ao logo do conjunto. Na parte de cima da Figura estão os gráficos de precisão por janela à esquerda (linhas azuis) e o gráfico de revocação por janela à direita (linhas vermelhas). Nestes gráficos de precisão e revocação, o eixo Y indica o percentual atingido. Novamente, quando o modelo não consegue calcular um valor, este é representado como uma linha abaixo de zero (0.0). Na parte de baixo da Figura estão os gráficos relacionados a quantidade instâncias selecionadas por janela, sendo o da esquerda o de número absoluto (eixo Y sendo o número de instâncias), e o da direito o de número relativo (eixo Y representa o percentual de instâncias).

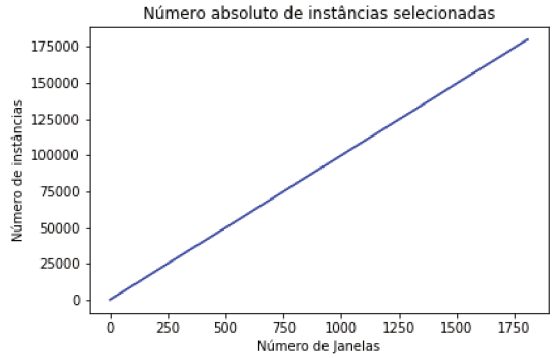
O resultado do cenário 2 (vide Figura 14) mostra que, de forma análoga ao resultado discutido no cenário anterior, no início do conjunto não existiam instâncias de ataque o que impossibilitou o cálculo das medidas. Após a chegada dos fluxos de ataque, o modelo se ajusta aos rótulos recebidos e consegue detectar os fluxos maliciosos. Uma comparação entre as Figuras 11 e 14 mostra um aumento nos valores de precisão e valores tão bons quanto na revocação. Esse resultado é notável pois tal melhoria está associada a uma diminuição drástica na quantidade de instâncias rotuladas disponíveis ao classificador. Enquanto que nos experimentos usando a estratégia *best n* o classificador utilizou aproximadamente 23% de instâncias rotuladas (Tabela 17), no aprendizado ativo esse mesmo número foi de 10%. Em termos absolutos, isso representa uma diminuição de aproximadamente 241 mil fluxos. Uma possível explicação para esse resultado pode ser feita a partir do estudo do número de instâncias rotuladas durante o experimento. A Figura 14 mostra que o modelo solicitou o rótulo de instâncias em todas as janelas do conjunto, mantendo o limite da *capacidade* e variando entre 7% e 13% por janela. Ou seja, solicitar instâncias rotuladas em todas as janelas, apesar de não ser o ideal em um cenário real, aumentou os valores de precisão e revocação.



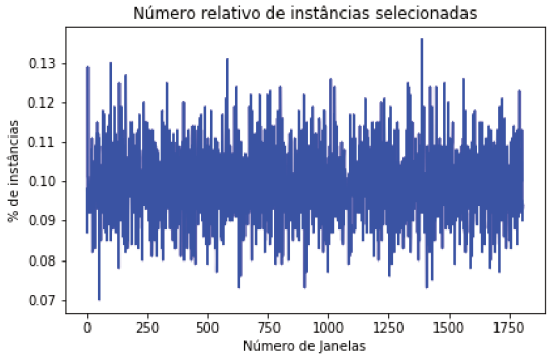
(a) Precisão por janela



(b) Revocação por janela



(c) Número absoluto de instâncias



(d) Número relativo de instâncias

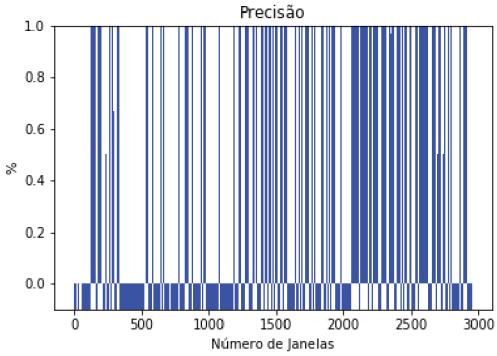
Figura 14 – Aprendizado ativo - Aleatório relacionado ao cenário 2 do CTU-13

A Figura 15 apresenta os quatro gráficos com os resultados obtidos para o cenário 8. Os gráficos representam as mesmas medidas tal como já foi explicado para os gráficos do cenário 2. Assim como na Figura 11 que mostra a execução por janela usando *best n*, é possível notar que praticamente em todas as janelas onde existem fluxos *botnet* os valores de precisão foram de 100%. Comportamento semelhante ocorreu com os valores de revocação. Ainda comparando as duas figuras é possível notar um aumento na quantidade de janelas com 100% de precisão e revocação, especialmente no último quarto do experimento. Novamente, a estratégia utilizada pelo Aleatório que foi entregar instâncias rotuladas em todas as janelas pode ter contribuído com a melhoria dos resultados.

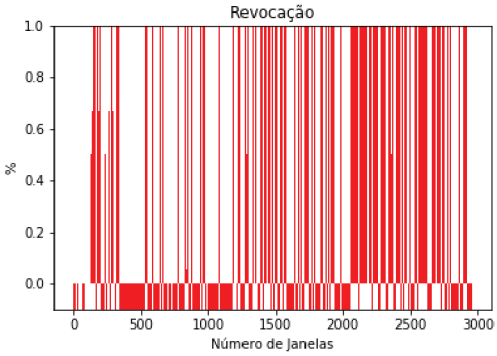
Um outro ponto que deve ser considerado é a quantidade de instâncias rotuladas usadas em ambos os cenários: *best n* e Aleatório. Enquanto que com o *best n* a quantidade de instâncias rotuladas correspondeu a aproximadamente 19% (Tabela 17) de todo o conjunto, com o aprendizado ativo esse número é de 10%. Em números absolutos isso representa uma diminuição de 265 mil instâncias rotuladas.

5.2.3.2 Seleção baseada em incerteza

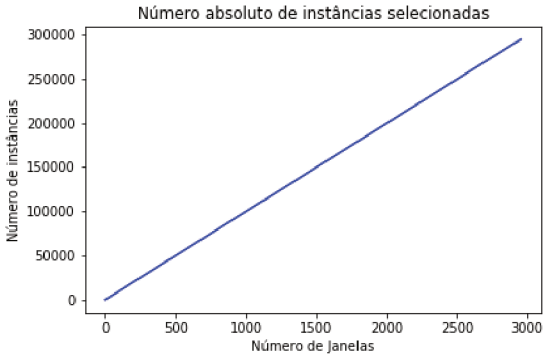
Conforme mencionado na Seção 2.5.2 do Capítulo 2, a classificação de aprendizado ativo baseada em Incerteza possui as seguintes estratégias: Incerteza Fixa, Incerteza



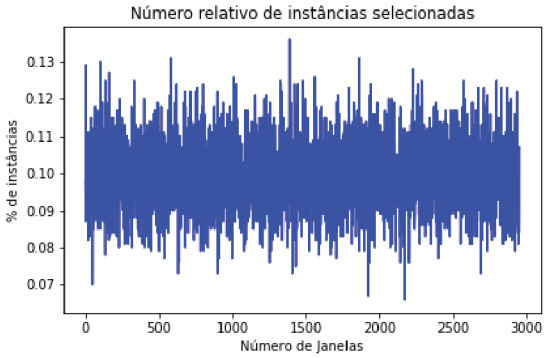
(a) Precisão por janela



(b) Revocação por janela



(c) Número absoluto de instâncias



(d) Número relativo de instâncias

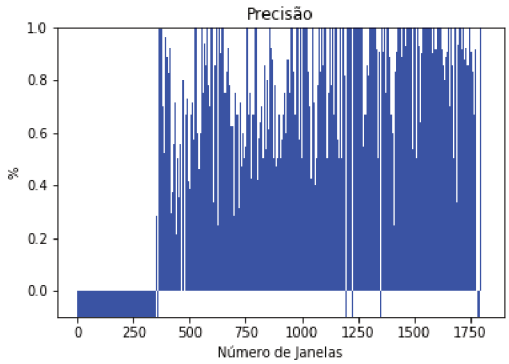
Figura 15 – Aprendizado ativo - Aleatório relacionado ao cenário 8 do CTU-13

Variável e Incerteza Variável com Aleatorização. Foram realizadas execuções utilizando cada umas destas estratégias e como conjuntos de dados de entrada os cenários 2 e 8 do CTU-13 (vide Tabela 21). A seguir será discutido o melhor resultado encontrado para cada um dos cenários 2 e 8, considerando precisão, revocação e quantidade de instâncias requisitadas dentro das janelas ao longo do conjunto.

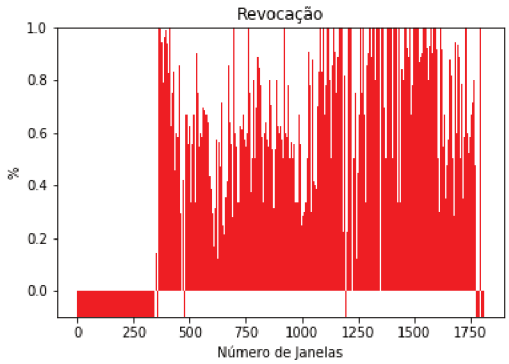
O melhor resultado obtido para o cenário 2 corresponde a execução número 9 da Tabela 21, em que foram utilizados o classificador Baseado em Incerteza, a estratégia Variável com Aleatorização e *passo* igual a 0,1. Os quatro gráficos gerados para este resultado estão na Figura 16.

Novamente, ao comparar o desempenho dos valores de precisão e revocação da estratégia baseada em incerteza com o *best n* (Figura 11) é possível notar um aumento da precisão e diminuição da revocação. Tal diminuição pode estar relacionada a quantidade e a forma de seleção de instâncias rotuladas entregues ao classificador. Aqui, um número constante de 10% de instâncias por janela foi selecionado. Isso pode ter influenciado diretamente no tipo de fluxo selecionado, impactando na revocação do modelo. Considerando o resultado obtido para o cenário 2 com a abordagem Aleatória (14), percebe-se que são piores os valores de precisão e revocação obtidos com o classificador de Incerteza. Novamente, a forma com que os rótulos foram selecionados pelo classificador pode explicar o

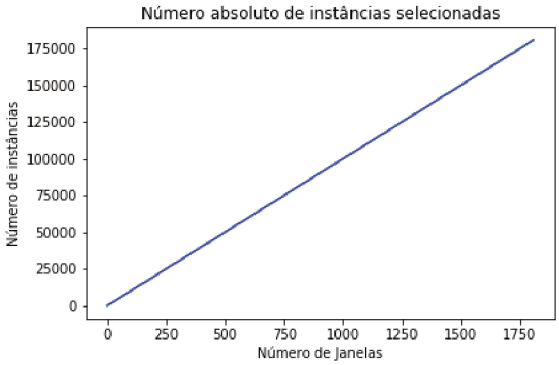
resultado. É possível que o modelo tenha passado por um cenário de *overfitting* já que não houve variação na forma com que os rótulos foram selecionados.



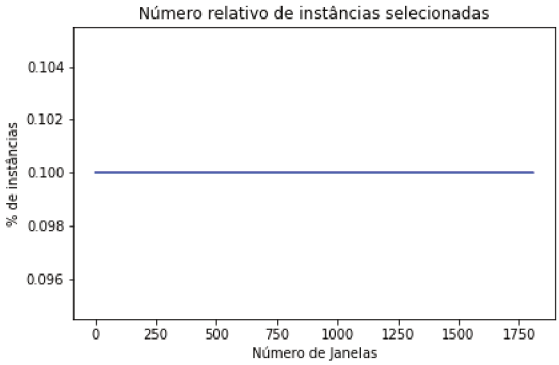
(a) Precisão por janela



(b) Revocação por janela



(c) Número absoluto de instâncias



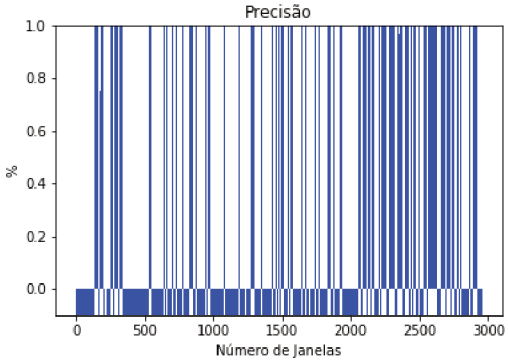
(d) Número relativo de instâncias

Figura 16 – Aprendizado ativo - Incerteza Variável com Aleatorização relacionado ao cenário 2 do CTU-13

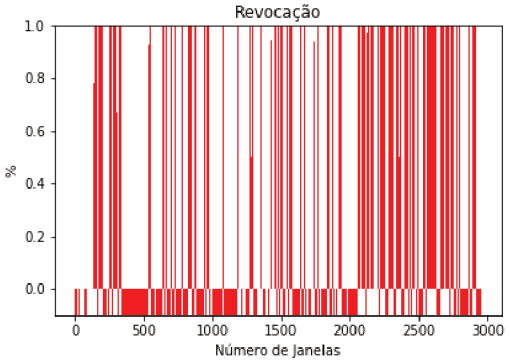
No cenário 8, o melhor resultado com o classificador de Incerteza foi obtido com a estratégia Variável e um *passo* igual a 0,1. Essa execução corresponde a número 6 na Tabela 21. Os gráficos gerados encontram-se na Figura 17. O percentual de seleção de instâncias se manteve próximo a 10%, atingindo em uma janela um percentual aproximado de 50%, mas depois, a partir da metade do conjunto, oscilava principalmente entre 8% e 12%. A própria distribuição do conjunto de dados, com fluxos *botnet* espalhados por todas as janelas pode explicar esse comportamento.

Com relação à precisão e revocação, os resultados foram ligeiramente piores comparando com os cenários do *best n* e da estratégia Aleatória. É possível notar que o número de janelas onde a precisão e revocação atinge valores máximos é menor na abordagem de Incerteza. Apesar disso, é importante relembrar que a quantidade de instâncias rotuladas selecionadas pelo classificador ainda é muito menor do que a estratégia *best n*, evidenciado o potencial do aprendizado ativo para a detecção de *botnets*.

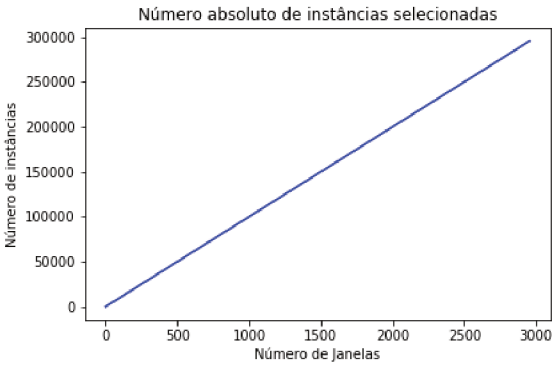
Apesar de utilizar estratégias mais robustas para seleção de rótulos, como a baseada em Incerteza, os resultados de precisão e revocação não melhoraram. Foi possível constatar



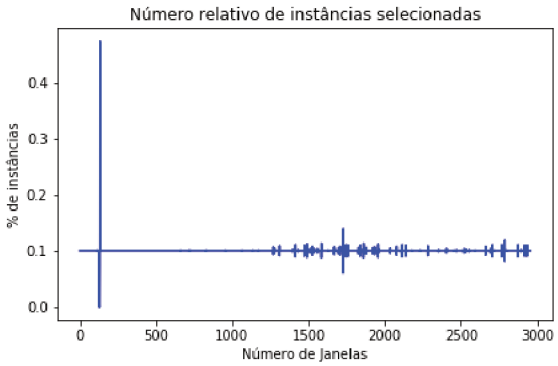
(a) Precisão por janela



(b) Revocação por janela



(c) Número absoluto de instâncias



(d) Número relativo de instâncias

Figura 17 – Aprendizado ativo - Incerteza Variável relacionado ao cenário 8 do CTU-13

que o modelo rotulou menos instâncias, ou seja, ele estava com alta confiança naquilo que foi classificado e por isso não precisava pedir rotulo, ao contrário da estratégia Aleatória que sempre pedia rótulos ao longo do fluxo. O modelo pode ter aprendido muito com tráfego normal, atingindo um alto nível de confiança, e nunca mais pediria outros rótulos. Possíveis soluções para isso seriam:

- melhorar o modelo via configuração, mudança de algoritmos, etc.;
- tentar esquecer partes do modelo a medida que uma quantidade de instância chega, ou quando atingir um indicador específico;
- rever os atributos selecionados para compor os fluxos.

5.2.4 Análise de conjuntos de classificadores utilizando conjuntos de dados multi-classe

Até o momento foram utilizados apenas os cenários do conjunto de dados CTU-13, considerando a tarefa de classificação como binária, que abrange as classes *botnet* e *background*. Contudo, em ataques *botnet* existem diferentes tipos de tráfegos de interesse

como a comunicação do bot com o C&C, a varredura de portas *PortScan* feita pelo *bot* na tentativa de encontrar novos alvos e os ataques disparados pelo *bot* como *DDoS*. Agrupar todos esses tráfegos em somente uma classe "Ataque" é uma abordagem comum na literatura. O objetivo deste grupo de experimentos é, justamente, separar esses tipos de tráfego e avaliar o desempenho dos classificadores nos diferentes tipos de tráfego maliciosos. Novamente, foi adotada a abordagem de aprendizado incremental usando classificação de fluxos contínuos de dados.

Nos experimentos anteriores foi constatado que o conjunto de classificadores *Ozaboost* tem o melhor desempenho dentre todos os que foram analisados. Logo, o *Ozaboost* foi mantido como conjunto de classificadores padrão em todas as execuções deste experimento.

O recém-criado conjunto de dados IoT-23 passa a ser a base de dados neste experimento, mantendo-se as classes originais sem qualquer tipo de agrupamento ou restrição. Conforme discutido anteriormente, a ideia aqui é investigar como os algoritmos de aprendizado se comportam diante dos diversos tipos de tráfegos associados a um ataque *botnet*, tais como: C&C, *PortScan*, *FileDownload*, *DDOS*, etc. Logo, o CTU-13 não continuou a ser utilizado por não ter este tipo de separação das classes, tendo apenas tráfego identificado como *botnet*, *background* ou normal em todos os cenários.

Neste conjunto de dados o número máximo de instâncias para treinamento (*max n*) não considerou somente a classe *botnet*, mas todas as classes de cada cenário. No caso do IoT-23, o *max n* foi definido para cada cenário com base nos seguintes fatores: (i) quantidade total de fluxos do cenário; (ii) distribuição das classes ao longo do conjunto; (iii) relação entre a quantidade de fluxos rotulados como benignos e os fluxos relacionados a atividade de *botnet*. Sendo assim, segue abaixo o *max n* correspondente a cada cenário.

- ❏ 4.000 instâncias quaisquer, para cenário 1;
- ❏ 50.000 instâncias quaisquer, para os cenários 4 e 14;
- ❏ 1.500 instâncias quaisquer, para o cenário 15;
- ❏ 3.000 instâncias quaisquer, para os cenários 19 e 20.

A estratégia de avaliação dos classificadores é mantida como o *ALPrenquentialEvaluationTask*, e conforme experimento da Seção 5.2.1, a janela também continua abrangendo o tamanho total do conjunto de dados. Além disso, o limite para liberação dos rótulos segue a mesma estratégia implementada com o *ALLimitedInstances*, tendo um número específico (*max n*) para cada cenário utilizado do IoT-23 (vide Seção 5.1.1). Diferente do experimento 5.2.1 em que havia um limite de instâncias específico para a classe *botnet* para cada conjunto de dados, neste experimento o limite é separado por conjunto mas abrange qualquer uma das classes presentes. A Tabela 22 apresenta os valores definidos para a configuração deste experimento.

Tabela 22 – Configuração para experimento multi-classe

Variável	**Valor**
D	IoT-23 (multi-classe)
P	100
S	*max n* de 4.000 para cenário 1; 50.000 para os cenários 4 e 14; 1.500 para o cenário 15; e 3.000 para os cenários 19 e 20.
C	Conjunto de Classificadores *Ozaboost*
A	*Hoeffding Tree*
T	Valor corresponde ao tamanho de cada conjunto de dados

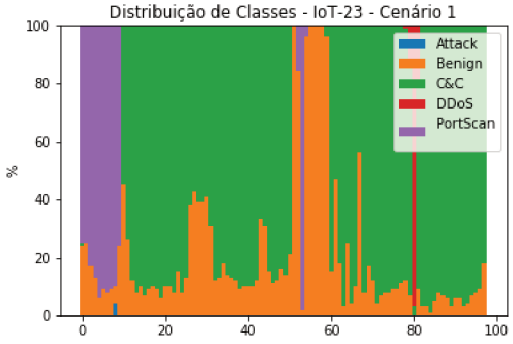
Os resultados consolidados das métricas *best n*, precisão e revocação para cada classe encontrada nos cenários selecionados do IoT-23 podem ser verificados na Tabela 23. Quando uma classe não existe em um dado cenário na célula é apresentado um hífen ("-"). Nos casos em que a classe existe no cenário mas não foram obtidos valores de precisão e revocação, ou os valores ficaram zerados, na célula aparecem valores zerados ("0 / 0 / 0"). A classe *Malicious* está presente somente no cenário 4. A classe *DDoS* apesar de estar presente nos cenários 1 e 14 teve valor zerado para as métricas. Fluxos benignos (classe *Benign*) permeiam todos os cenários.

Tabela 23 – Resultados consolidados para cenários IoT-23

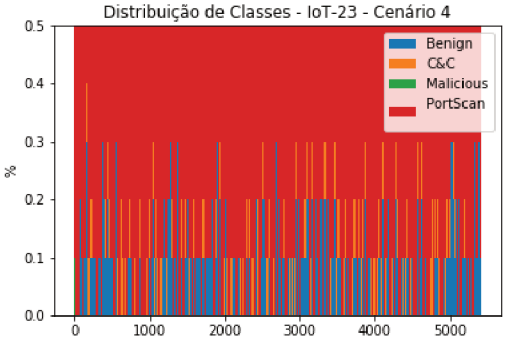
	best n / **Precisão / Revocação**		
Classe	**1**	**4**	**14**
Attack	1.100 / 0,37 / 0,75	-	3.100 / 0 / 0,33
Benign	400 / 0,98 / 0,93	28.600 / 0,91 / 0,95	29.800 / 0,99 / 0,99
C&C	1.200 / 0,98 / 0,99	30.500 / 0,93 / 0,94	16.300 / 0,84 / 0,43
DDoS	0 / 0 / 0	-	0 / 0 / 0
Malicious	-	17.200 / 0,7 / 0,77	-
PortScan	1.200 / 0,99 / 0,87	400 / 0,99 / 0,99	-

	best n / **Precisão / Revocação**		
Classe	**15**	**19**	**20**
Attack	1.900 / 0,002 / 100	1.300 / 0,97 / 0,99	-
Benign	0 / 0 / 0	1.300 / 0,99 / 0,94	300 / 0,99 / 0,96
C&C	7.400 / 0,33 / 0,001	0 / 0 / 0	0 / 0 / 0
DDoS	-	-	-
Malicious	-	-	-
PortScan	2.200 / 0,99 / 0,99	300 / 0,99 / 0,99	300 / 0,96 / 0,99

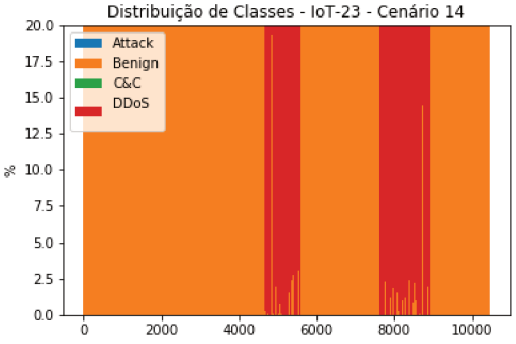
Os gráficos com valores de precisão, revocação e *best n* para cada classe presente no cenário 1 estão na Figura 19. Neste cenário, a classe detectada com menor *best n* (cerca de 4% do conjunto) e maiores valores de precisão e revocação (respectivamente 98% e 93%) é a *Benign*. Em segundo lugar está a C&C com um *best n* de 12%, precisão 98% e revocação 99%. É possível notar que os fluxos da classe *Attack*, mesmo com um *best n* de 1.100 que é próximo ao da classe C&C (1.200), atingem valores bem mais baixos de precisão e revocação (37% e 75% respectivamente). Interessante notar que para a classe *DDoS* os valores das métricas ficaram zerados. Isso porque esta classe apresenta poucas ocorrências ao longo de todo o conjunto, não sendo suficientes para treinar o modelo de



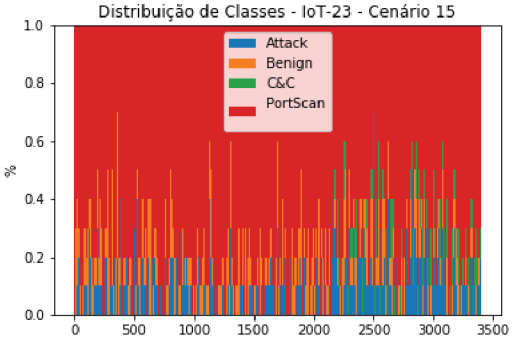
(a) Cenário 1. Agrupamento igual a 100 instâncias.



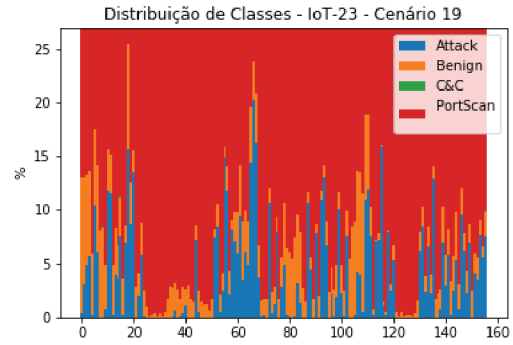
(b) Cenário 4. Eixo Y limitado de 0,0 a 0,5%.



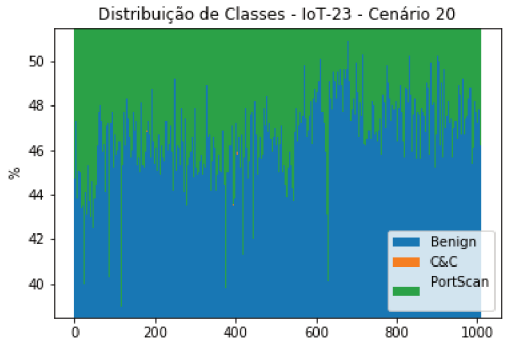
(c) Cenário 14. Eixo Y limitado de 0 a 20%.



(d) Cenário 15. Eixo Y limitado de 0 a 1%.



(e) Cenário 19. Eixo Y limitado de 0 a 25%.

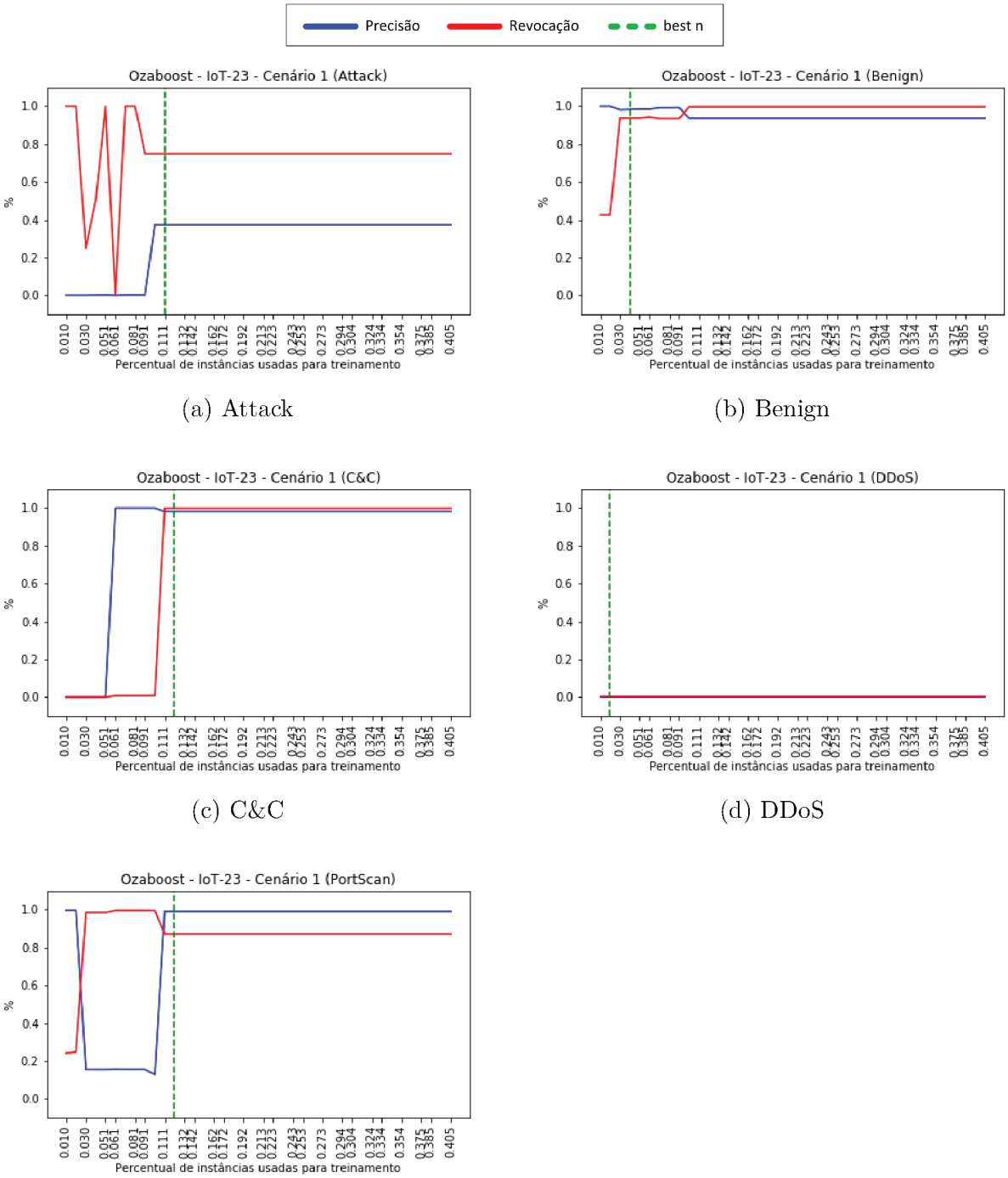


(f) Cenário 20. Eixo Y limitado de 0 a 60%.

Figura 18 – Distribuição de classes para os cenários 1, 4, 14, 15, 19 e 20 do IoT-23

tal forma que consiga detectá-las (vide gráfico de distribuição de classes do cenário 1 na Figura 18).

Os resultados obtidos para o cenário 4 podem ser vistos na Figura 20. Neste cenário, a classe *PortScan* está presente de forma massiva em todo o conjunto, representando quase 99% de todos os fluxos. As demais classes dividem os 1% restantes (vide Figura 18).



(a) Attack

(b) Benign

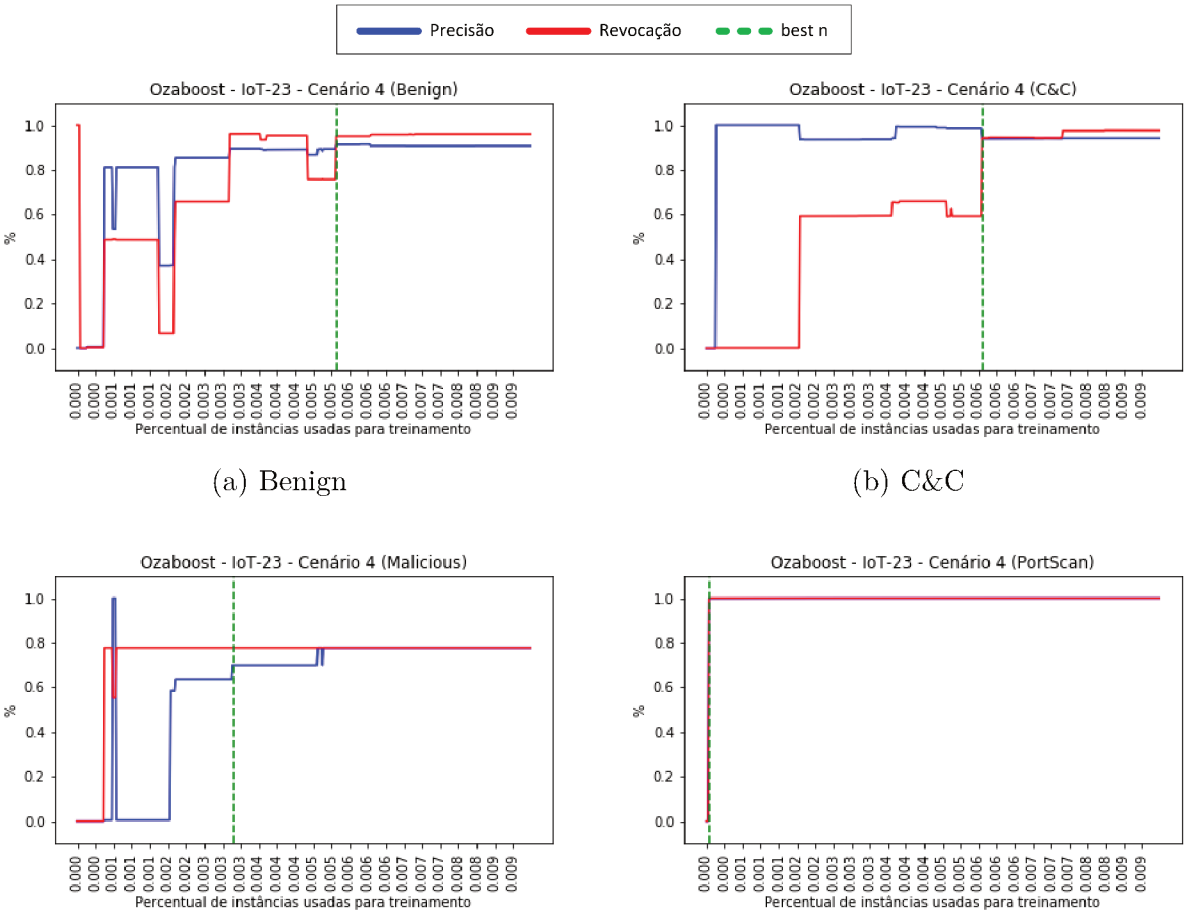
(c) C&C

(d) DDoS

(e) PortScan

Figura 19 – Precisão x Revocação para cada classe do cenário 1 do IoT-23

Sendo assim, a classe *PortScan* é detectada com maior desempenho, atingindo *best n* com apenas 400 instâncias, e valores de precisão e revocação próximos a 99%. As classes *Benign* e *C&C* apesar de serem bem menos presentes ao longo do conjunto, atingiram *best n* entre 0,5% e 0,6%, além de valores acima de 91% para precisão e revocação. Vale ressaltar que este é o único cenário selecionado que possui a classe *Malicious*, a qual apresentou um *best n* de 0,3% (menor que das classes C&C e *Bening*), mas com valores de 70% para precisão e de 77% para revocação.



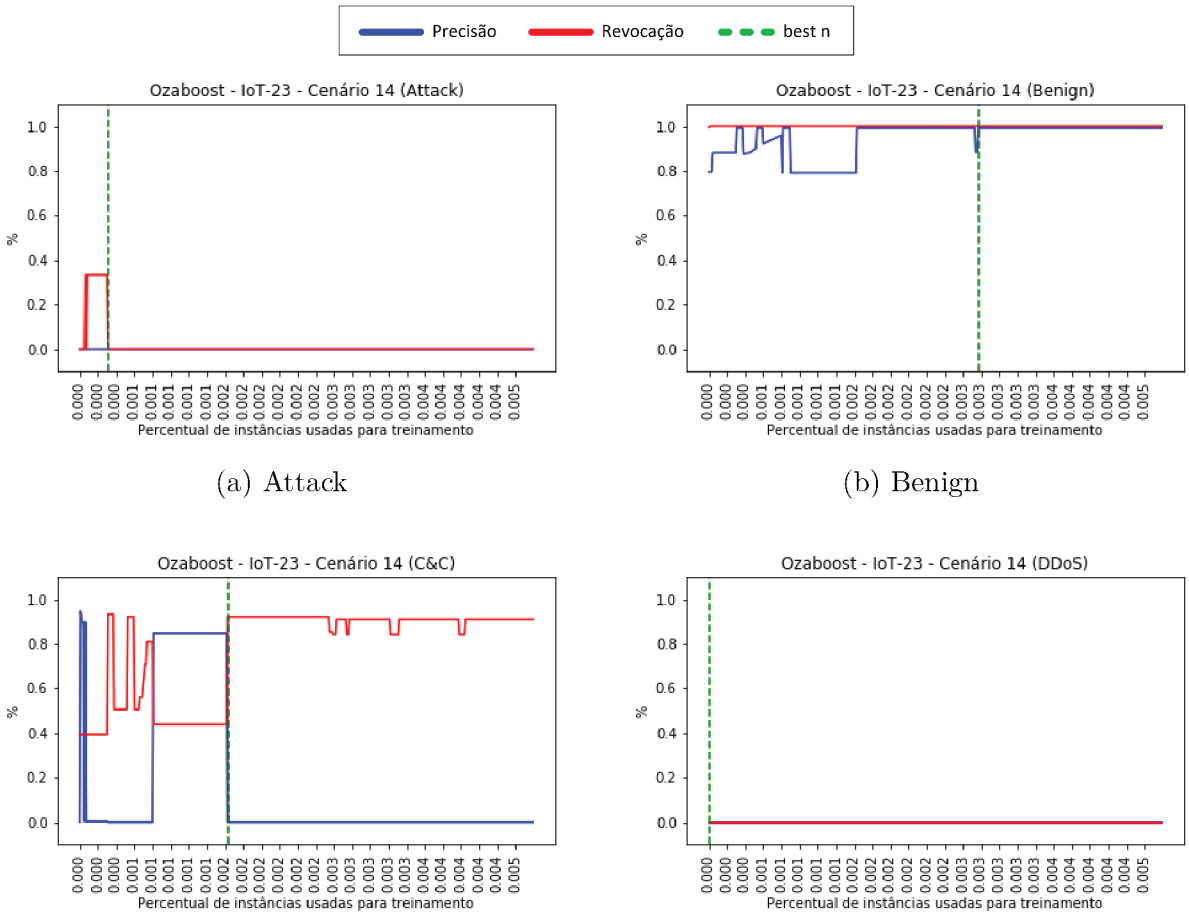
(a) Benign (b) C&C

(c) Malicious (d) PortScan

Figura 20 – Precisão x Revocação para cada classe do cenário 4 do IoT-23

No cenário 14 as classes que tiveram pior desempenho em sua detecção foram *Attack* e *DDoS* (vide gráficos na Figura 21). É fato que a distribuição de classes ao longo deste cenário é desigual, com presença marcante para as classes *Bening* e *DDoS* (vide gráfico de distribuição do cenário 14 na Figura 18). Entretanto a classe DDoS teve os valores das métricas todos zerados. Apesar de ter um limite de 50.000 instâncias, ou seja, 0,5% do conjunto para treinamento e atualização do modelo de decisão, não foram utilizadas instâncias da classe *DDoS* para treinamento pois estas só aparecem a partir de aproximadamente 45% do conjunto. Logo, o modelo de decisão nunca foi treinado com instâncias da classe DDoS. O melhor caso é da classe *Benign*, cujo valor *best n* foi de 0.3%, além de precisão e revocação com valores próximos as 99%.

Pela Figura 18 é possível constatar que o cenário 15 apresenta uma distribuição de classes similar a do cenário 4. A classe *PortScan* corresponde a maioria absoluta dos fluxos ao longo de todo o conjunto. Os gráficos com precisão, revocação e *best n* para cada classe do cenário 15 estão na Figura 22. Da mesma forma que no cenário 4, a classe *PortScan* atinge valores altos de precisão e revocação (aproximadamente 99% para ambos) utilizando cerca de 3.100 instancias para treinamento (menos de 0,001% do conjunto). A



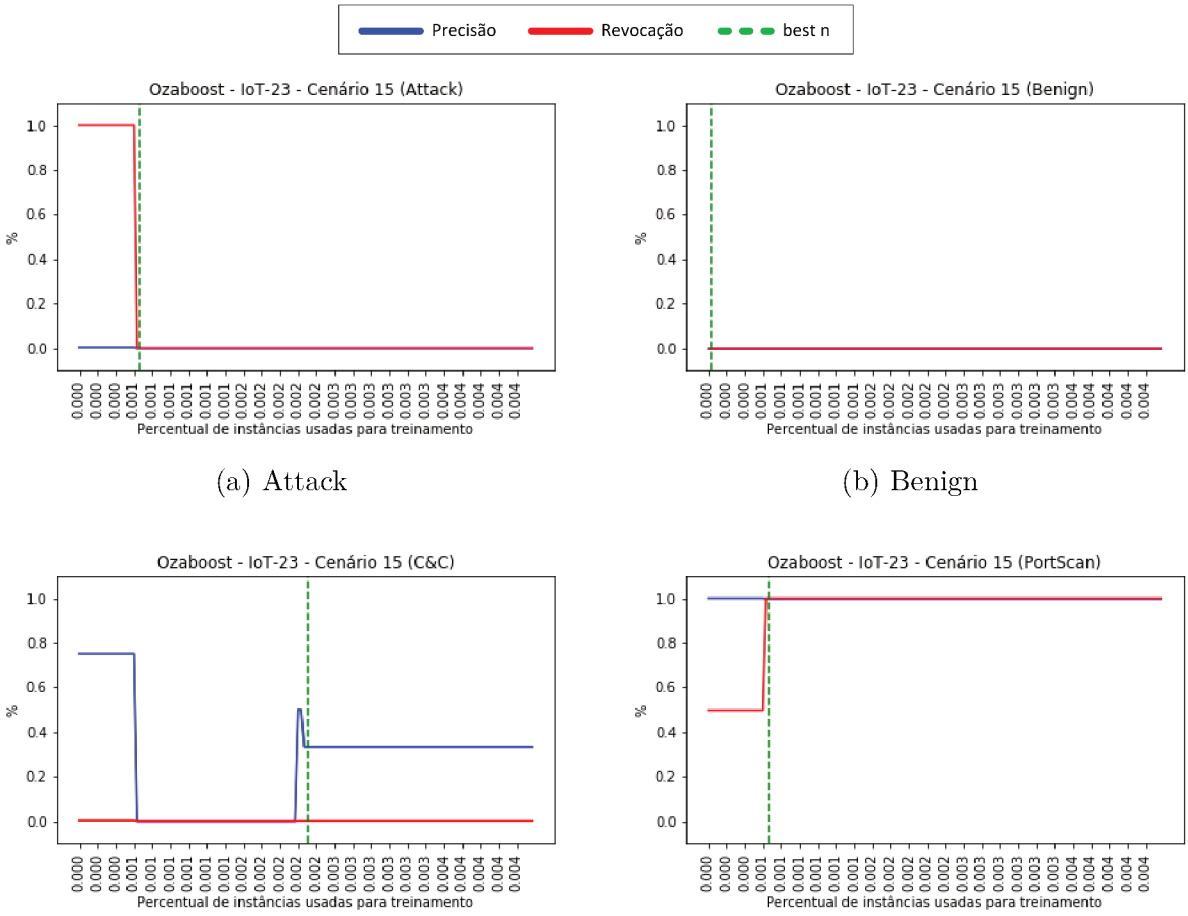
(a) Attack (b) Benign

(c) C&C (d) DDoS

Figura 21 – Precisão x Revocação para cada classe do cenário 14 do IoT-23

classe C&C obteve um *best n* de 7.400 instâncias, mas apresentou oscilação na precisão atingindo valor de 33% e tendo somente valores zerados para revocação. As demais classes, *Attack* e *Benign*, tiveram somente valores zerados para as métricas. Nota-se que a distribuição das classes ao longo do conjunto novamente impactou o desempenho de detecção. Considerando que o limite para treinamento e atualização do modelo de decisão era de 1.500 instâncias, as classes que até este limite não tinham instâncias suficientes, acabaram obtendo um resultado com valores baixos ou zerados de precisão e revocação.

Com exceção da C&C, as classes do cenário 19 aparecem em grande quantidade ao longo de todo o conjunto, sendo que a *PortScan* continua sendo a que possui maioria absoluta dos fluxos (vide Figura 18). Os resultados de detecção por classe podem ser vistos na Figura 23. Por possuir uma quantidade baixíssima de instâncias ao longo do conjunto (8 fluxos), a classe C&C apresentou o pior desempenho, tendo valores zerados para precisão e revocação. A classe *PortScan* novamente teve melhor resultado, com *best n* de 0,3% das instâncias do conjunto, e valores de precisão e revocação de aproximadamente 99%. As classes de *Attack* e *Bening* também obtiveram bons resultados, com mesmo *best n* de 1.300 instâncias (ou 0,9%) e valores de precisão e revocação acima de 94%.

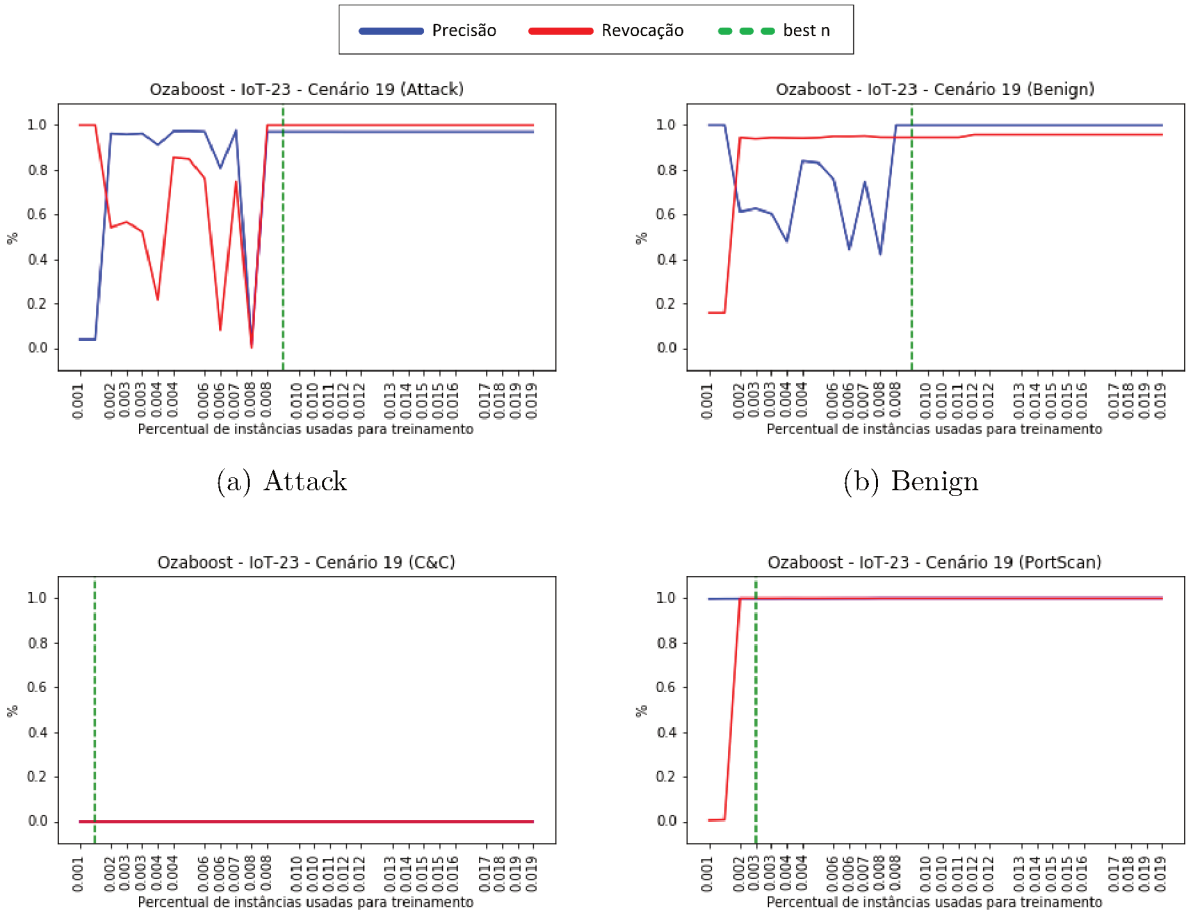


(a) Attack (b) Benign

(c) C&C (d) PortScan

Figura 22 – Precisão x Revocação para cada classe do cenário 15 do IoT-23

Conforme mostrado na Figura 18, o gráfico de distribuição do cenário 20 possui três classes sendo que a maioria absoluta das instâncias fica dividida entre *Bening* e *PortScan*, restando poucas instâncias da classe C&C ao longo do cenário. Na Figura 24 estão os resultados para o cenário 20. Os gráficos das classes *Bening* e *PortScan* mostraram valores altos para precisão e revocação (acima de 96%) com *best n* igual a 300 instâncias. Já para classe C&C não foi possível calcular nenhum valor, visto a existência de uma quantidade mínima de instâncias ao longo do conjunto.



(a) Attack

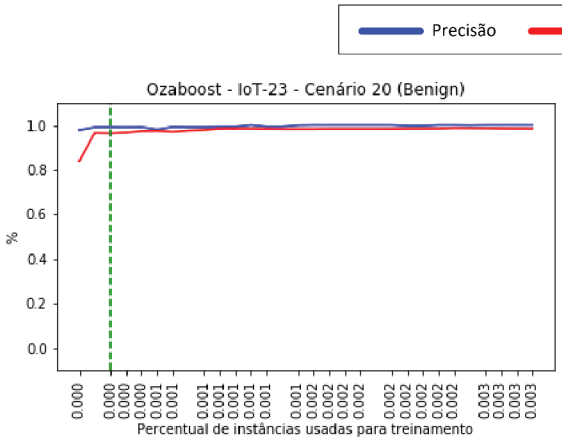
(b) Benign

(c) C&C

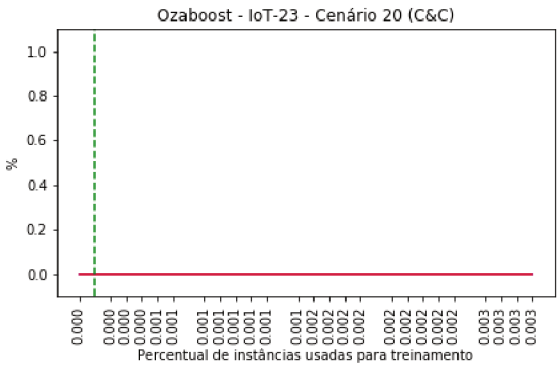
(d) PortScan

Figura 23 – Precisão x Revocação para cada classe do cenário 19 do IoT-23

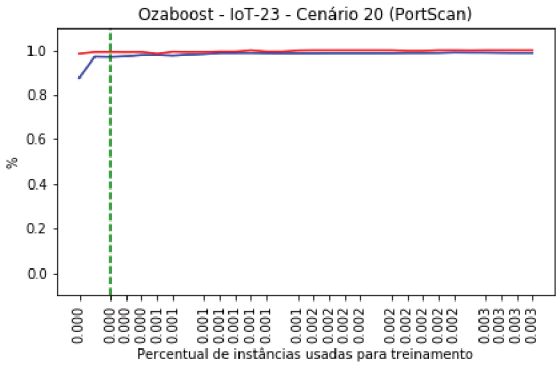
Com o treinamento do modelo considerando várias classes separadamente, é possível identificar uma relação entre o tipo e a quantidade de tráfego distribuído ao longo do conjunto, com o desempenho alcançado. Trabalhar com resultados cumulativos e limite máximo sequencial para seleção das instâncias rotuladas talvez não seja a melhor abordagem. Conforme resultados obtidos neste experimento, o desempenho sempre estará atrelado a quantidade de rótulos de determinados tipos de tráfego que o modelo recebeu até atingir o limite *max n*. Classes que possuem poucas instâncias dentro desse limite, ou estão fora dele, não terão um resultado satisfatório. Seria importante aplicar estratégias de seleção de rótulos mais inteligentes, tal como as estratégias de aprendizado ativo, para poder selecionar instâncias relevantes ao longo de todo o conjunto.



(a) Benign



(b) C&C



(c) PortScan

Figura 24 – Precisão x Revocação para cada classe do cenário 20 do IoT-23

Capítulo 6

Conclusão

Neste trabalho, foi apresentada uma avaliação de desempenho para algoritmos de mineração de fluxo contínuo referente ao problema de detecção de *botnets* considerando elementos mais próximos da realidade de segurança da informação. O Capítulo 2 abordou desde conceitos fundamentais relacionados à segurança da informação, passando pela arquitetura e funcionamento das *botnets*, até mineração de fluxos contínuos e aprendizado ativo. Já o Capítulo 3 focou na revisão da literatura correlata, dando visão do estado da arte na área de detecção de *botnets*, além dos aspectos importantes das *botnets* e também da mineração de dados que foram pouco explorados, e poderiam contribuir para melhorar o desempenho dos sistemas de detecção à intrusão.

O Capítulo 4 detalhou a proposta elaborada durante a fase de pesquisa e desenvolvida neste trabalho. A proposta foi concebida a partir de uma visão genérica e mais abrangente, a qual depois deu origem a cenários específicos relacionados às hipóteses levantadas. O detalhamento dos processos gerais da proposta (que abrangem seleção, classificação, treinamento, teste e avaliação de dados) resultou em quatro cenários, que em ordem, mostram uma sequência evolutiva e complementar da análise. No primeiro cenário ocorreu avaliação considerando um trabalho base da literatura, conjuntos com classe binária e um limite fixo pré-estabelecidos para liberação dos rótulos. Já o segundo cenário continuou com classes binárias assumindo o melhor algoritmo do cenário anterior, mas agora com avaliação contínua utilizando janelas temporais em bases selecionadas. No terceiro foi introduzida a técnica de aprendizado ativo e diferentes estratégias de seleção de instâncias rotuladas para atualizar o classificador. O quarto, e último cenário, mudou a base origem dos dados e passou a considerá-la multi-classe.

O Capítulo 5 apresentou os detalhes e resultados dos quatro experimentos realizados. Cada um destes experimentos correspondia a um cenário da proposta. Na avaliação realizada foram considerados elementos que aproximam ao máximo de cenários reais, tais como: (i) uso fluxos com dados reais de tráfego de *botnets*; (ii) execução de algoritmos de aprendizado incremental próprios para fluxos contínuos; e (iii) limitação da quantidade de amostras rotuladas apresentadas ao classificador. De forma consolidada, as principais

características dos experimentos realizados abrangem:

- ❏ uso de duas estratégias de classificação, classificadores únicos e conjuntos de classificadores;
- ❏ as medidas de avaliação foram precisão, revocação, *best n* e taxa de requisição de instâncias rotuladas;
- ❏ uso das bases de dados CTU-13 e IoT-23 com tráfego real de ataques de *botnets*;
- ❏ avaliação consolidada do conjunto de dados e também por janelas temporais, ao longo do fluxo;
- ❏ avaliação de diferentes estratégias para liberação dos rótulos verdadeiros;
- ❏ classificação binária e multi-classe.

O primeiro experimento utilizou o CTU-13 como conjunto de dados de entrada, foi baseado em classificação binária (ataque e *background*) e consolidou os resultados considerando todo o tamanho do conjunto. Foi realizada a comparação do desempenho entre classificadores únicos (*Hoeffding Tree* e *Hoeffding Adaptive Tree*) e conjuntos de classificadores (*Ozaboost*, *Ozabag* e *Ozabag com Adwin*) que utilizam a *Hoeffding Tree* como algoritmo base para aprendizado. O trabalho de Costa et al. (2018) foi utilizado como referência para avaliar o desempenho de classificadores únicos. Os resultados indicaram que o conjunto de classificadores *Ozaboost* obteve os melhores valores de precisão/revocação e *best n* na maioria dos casos comparados, corroborando a Hipótese 1. Além disso, considerando conjuntos de classificadores e alguns atributos das *botnets* (como o protocolo utilizado pelo C&C), estes requerem menos instâncias rotuladas enquanto mantém alto desempenho.

Apesar de o *Ozaboost* apresentar o melhor desempenho, foi constatado que existem oscilações na precisão e revocação ao longo dos fluxos. Logo, o segundo experimento teve como objetivo avaliar o desempenho do modelo gerado pelo *Ozaboost* em janelas temporais. Com base nos resultados obtidos no primeiro experimento, foram considerados desta vez apenas os cenários 2 e 8 do CTU-13 neste experimento. Os valores de precisão e revocação encontrados por janela mostraram algumas divergências com os valores calculados para todo o conjunto de teste. O cenário 2, por exemplo, mostrou várias janelas onde a precisão foi baixa, chegando até 0. Isso indica que o valor obtido originalmente, $0,78$, deve ser cuidadosamente analisado pelos envolvidos no desenvolvimento e operação do sistema. Tais resultados comprovam a Hipótese 2.

Como os resultados por janela mostram diversos momentos em que o modelo atinge um baixo desempenho, contrariando os resultados acumulados, um terceiro conjunto de experimentos foi proposto para investigar as diferentes maneiras de entregar as instâncias

rotuladas para o modelo. Foram utilizadas duas estratégias de aprendizado ativo: o Aleatório, e o Baseado em Incerteza, sendo que este último possui diferentes estratégias de seleção de instâncias. O classificador Aleatório obteve resultados melhores de precisão e revocação do que os obtidos com as estratégias do classificador Baseado em Incerteza. Isso ocorreu pois o classificador Aleatório manteve seleção de instâncias em aproximadamente 10% em todas as janelas ao longo do conjunto. Com relação ao classificador Baseado em Incerteza, o melhor resultado obtido para o cenário 2 foi com estratégia Variável com Aleatorização, e para o cenário 8 com a estratégia Variável. É importante ressaltar que ao comparar as estratégias de aprendizado ativo com o *best* n, notou-se que foi possível diminuir drasticamente a quantidade de instâncias rotuladas e manter o desempenho (precisão e revocação), comprovando a Hipótese 3.

Por fim, ao longo da revisão da literatura correlata foi constatado que a maioria dos trabalhos agrupam todo o tráfego de *botnet* em somente duas classes (ataque e normal), mesmo sabendo que existem diferentes tipos de tráfego em tal ameaça. Assim, no quarto experimento, com o intuito de avaliar o desempenho dos classificadores nos diferentes tipos de tráfego, foi utilizada uma outra base, o IoT-23. Para cada um dos cenários selecionados foram avaliadas as medidas precisão, revocação e *best* n considerando as múltiplas classes existentes, sem quaisquer tipos de agrupamento ou restrição. Novamente, a distribuição das classes ao longo dos conjuntos e a quantidade de instâncias rotuladas inicialmente foram determinantes no desempenho obtido para cada uma das classes dos cenários. Esses fatores são determinantes para a criação de modelos de decisão que identifiquem diferentes tipos de tráfego associados a uma *botnet*. Estes resultados comprovam a Hipótese 4.

6.1 Principais Contribuições

A principal contribuição deste trabalho é fornecer uma avaliação sobre a aplicação de algoritmos de mineração de fluxos contínuos no problema de detecção de *botnet*. Elas podem ser detalhadas da seguinte forma:

1. conjuntos de classificadores superam, na grande maioria dos casos, o desempenho de classificadores únicos, tanto em precisão e revocação quanto no número de instâncias rotuladas, sendo *Ozaboost* o algoritmo com melhor desempenho dentre os avaliados;

2. os resultados obtidos em janelas individuais de tempo revelam oscilações no desempenho do classificador que não são detectadas nos resultados cumulativos. O uso de gráficos de precisão e revocação por janela com a devida indicação dos valores que não podem ser calculados pelo classificador permitiram a devida visualização dos dados;

3. o tipo e a quantidade de tráfego por janela, assim como a quantidade de instâncias rotuladas utilizadas inicialmente para treinamento e atualização do modelo impactam diretamente no desempenho do mesmo;

4. as estratégias de aprendizado ativo se mostram promissoras para a diminuição da quantidade necessária de instâncias rotuladas, mantendo o bom desempenho na detecção dos fluxos maliciosos;

5. a avaliação multi-classe mostrou quais classes impactam no desempenho do modelo, e sua relação com a distribuição das classes ao longo do conjunto.

6.2 Trabalhos Futuros

A partir dos experimentos realizados e resultados alcançados neste trabalho, surgiram várias possibilidades de trabalhos futuros:

- ❏ avaliar o impacto da seleção de atributos para classificação de tráfego de *botnet*, propondo atributos específicos para protocolos como HTTP e P2P;

- ❏ estender a comparação do algoritmo de mineração de fluxo para outras abordagens, como *Accuracy Updated Ensemble* (AUE) (BRZEZIŃSKI; STEFANOWSKI, 2011), *Active Classifier* (ŽLIOBAITĖ et al., 2011), *Leveraging Bagging* (BIFET; PFAHRINGER, 2010), *Limited Attribute Classifier* (BIFET et al., 2010), *Oza Bagging With Adaptive Size Hoeffding Tree* (BIFET G. HOLMES; GAVALDà, 2009) e *Single Classifier Drift* ((GAMA P. MEDAS; RODRIGUES, 2004) e (BAENA-GARCIA et al., 2006));

- ❏ ampliar a discussão sobre como esses modelos deveriam ser implementados em ambientes reais. Métricas de desempenho como o consumo de memória e tempo de atualização do classificador poderiam ser exploradas;

- ❏ investigar técnicas para melhorar os modelos de classificação criados com o auxílio do aprendizado ativo. Uma das abordagens poderia ser utilizar a estratégia Baseada em Incerteza com *SelSampling*;

- ❏ ampliação dos experimentos para outros cenários do CTU e também em outros conjuntos de dados como o ISOT-IoT (SAAD et al., 2011).

6.3 Contribuições em Produção Bibliográfica

O artigo "*A Comparison of Stream Mining Algorithms on Botnet Detection*" (RIBEIRO; PAIVA; MIANI, 2020) que apresenta a proposta para classificação incremental

de tráfego *botnet* e compara o desempenho de conjunto de classificadores (*ensemble*) e classificadores únicos foi publicado e apresentado na conferência ARES 2020 (*15th International Conference on Availability, Reliability and Security*).

Referências

ACARALI, D. et al. Survey of approaches and features for the identification of HTTP-based botnet traffic. **Journal of Network and Computer Applications**, v. 76, p. 1–15, 2016. ISSN 10958592. Disponível em: <https://doi.org/10.1016/j.jnca.2016.10.007>.

ALAUTHAMAN, M. et al. A P2P Botnet detection scheme based on decision tree and adaptive multilayer neural networks. **Neural Computing and Applications**, Springer London, v. 29, n. 11, p. 991–1004, 2018. ISSN 09410643. Disponível em: <https://doi.org/10.1007/s00521-016-2564-5>.

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

BAILEY, M. et al. A Survey of Botnet Technology and Defenses. **2009 Cybersecurity Applications Technology Conference for Homeland Security**, p. 299–304, 2009. Disponível em: <https://doi.org/10.1109/CATCH.2009.40>.

BEZERRA, V. H. et al. Providing iot host-based datasets for intrusion detection research. In: **Anais Principais do XVIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais**. SBC, 2018. p. 15–28. Disponível em: <https://sol.sbc.org.br/index.php/sbseg/article/view/4240>.

BIFET, A. **Hoeffding Adaptive Tree**. 2020. Disponível em: <https://github.com/Waikato/moa/blob/cd7c5c6f16b320db4573ceaf5cf219a3c24e9d3e/moa/src/main/java/moa/classifiers/trees/HoeffdingAdaptiveTree.java>.

_____. **OzaBagAdwin**. 2020. Disponível em: <https://github.com/Waikato/moa/blob/cd7c5c6f16b320db4573ceaf5cf219a3c24e9d3e/moa/src/main/java/moa/classifiers/meta/OzaBagAdwin.java>.

BIFET, A. et al. Accurate ensembles for data streams: combining restricted Hoeffding trees using stacking. In: **Proceedings of 2nd Asian Conference on Machine Learning**. [S.l.: s.n.], 2010. p. 225–240.

BIFET, A.; GAVALDA, R. Learning from time-changing data with adaptive windowing. In: SIAM. **Proceedings of the 2007 SIAM international conference on data mining**. 2007. p. 443–448. Disponível em: <https://doi.org/10.1137/1.9781611972771.42>.

BIFET, A.; GAVALDÀ, R. Adaptive learning from evolving data streams. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 5772 LCNS, p. 249–260, 2009. ISSN 03029743. Disponível em: <https://doi.org/10.1007/978-3-642-03915-7_22>.

BIFET, A. et al. MOA: massive online analysis. **Journal of Machine Learning Research**, v. 11, p. 1601–1604, 2010. Disponível em: <http://portal.acm.org/citation.cfm?id=1859903>.

BIFET, G. H. A.; PFAHRINGER, B. Leveraging bagging for evolving data streams. **Proc. ECML-PKDD Part I**, p. 135–150, 2010. Disponível em: <https://doi.org/10.1007/978-3-642-15880-3_15>.

BIFET G. HOLMES, B. P. R. K. A.; GAVALDà, R. New ensemble methods for evolving data streams. **Proc. 15th ACM SIGKDD Int. Conf. KDD**, p. 139–148, 2009. Disponível em: <https://doi.org/10.1145/1557019.1557041>.

BISHOP, M. What is computer security? **IEEE Security and Privacy**, v. 1, n. 1, p. 67–69, 2003. ISSN 15407993. Disponível em: <https://doi.org/10.1109/MSECP.2003.1176998>.

BRZEZIŃSKI, D.; STEFANOWSKI, J. Accuracy updated ensemble for data streams with concept drift. In: SPRINGER. **International conference on hybrid artificial intelligence systems**. 2011. p. 155–163. Disponível em: <https://doi.org/10.1007/978-3-642-21222-2_19>.

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

CERON, J. **Arquitetura distribuída e automatizada para mitigação de botnet baseada em análise dinâmica de malwares**. Dissertação (Mestrado) — UFRGS, 2010. Disponível em: <http://www.lume.ufrgs.br/handle/10183/70238>.

CHEN, F.; RANJAN, S.; TAN, P. N. Detecting bots via incremental LS-SVM learning with dynamic feature adaptation. **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, p. 386–394, 2011. Disponível em: <https://doi.org/10.1145/2020408.2020471>.

CHEN, S. C.; CHEN, Y. R.; TZENG, W. G. Effective Botnet Detection Through Neural Networks on Convolutional Features. **Proceedings - 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018**, IEEE, p. 372–378, 2018. Disponível em: <https://doi.org/10.1109/TrustCom/BigDataSE.2018.00062>.

CHEN, W. R. Exploring a Service-Based Normal Behaviour Profiling System for Botnet Detection. p. 947–952, 2017. Disponível em: <https://doi.org/10.23919/INM.2017.7987417>.

COSTA, V. G. T. da et al. Online detection of Botnets on Network Flows using Stream Mining. **Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)**, SBC, Porto Alegre, RS, Brasil, 2018. ISSN 2177-9384. Disponível em: <https://sol.sbc.org.br/index.php/sbrc/article/view/2418>.

DOMINGOS, P. Mining High-Speed Data Streams. **Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '00**, p. 71–80, 2000. Disponível em: <https://doi.org/10.1145/347090.347107>.

DOSHI, R.; APTHORPE, N.; FEAMSTER, N. Machine learning DDoS detection for consumer internet of things devices. **Proceedings - 2018 IEEE Symposium on Security and Privacy Workshops, SPW 2018**, n. Ml, p. 29–35, 2018. Disponível em: <https://doi.org/10.1109/SPW.2018.00013>.

FAISAL, M. A. et al. Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study. **IEEE Systems journal**, IEEE, v. 9, n. 1, p. 31–44, 2014. Disponível em: <https://doi.org/10.1109/JSYST.2013.2294120>.

FARID, D. M. et al. An adaptive ensemble classifier for mining concept drifting data streams. **Expert Systems with Applications**, Elsevier Ltd, v. 40, n. 15, p. 5895–5906, 2013. ISSN 09574174. Disponível em: <https://doi.org/10.1016/j.eswa.2013.05.001>.

GAMA, J. **Knowledge discovery from data streams**. CRC Press, 2010. 1–235 p. ISBN 9781439826126. Disponível em: <https://doi.org/10.1201/EBK1439826119-c1>.

GAMA P. MEDAS, G. C. J.; RODRIGUES, P. Learning with drift detection. **Proc. SBIA**, p. 286–295, 2004. Disponível em: <https://doi.org/10.1007/978-3-540-28645-5_29>.

GAONKAR, S. et al. A survey on botnet detection techniques. In: IEEE. **2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)**. 2020. p. 1–6. Disponível em: <https://doi.org/10.1109/ic-ETITE47903.2020.Id-70>.

GARCIA MARTIN GRILL, H. S. S.; ZUNINO, A. An empirical comparison of botnet detection methods. **Computers and Security Journal, Elsevier**, v. 45, p. 100–123, 2014. Disponível em: <https://doi.org/10.1016/j.cose.2014.05.011>.

GARCÍA, S. et al. An empirical comparison of botnet detection methods. **Computers and Security**, v. 45, p. 100–123, 2014. ISSN 01674048. Disponível em: <https://doi.org/10.1016/j.cose.2014.05.011>.

GARCÍA, S.; ZUNINO, A.; CAMPO, M. Survey on network-based botnet detection methods. **Security and Communication Networks**, Wiley Online Library, v. 7, n. 5, p. 878–903, 2014. Disponível em: <https://doi.org/10.1002/sec.800>.

GARG, S.; PEDDOJU, S. K.; SARJE, A. K. Scalable P2P bot detection system based on network data stream. **Peer-to-Peer Networking and Applications**, Peer-to-Peer Networking and Applications, v. 9, n. 6, p. 1209–1225, 2016. ISSN 19366450. Disponível em: <https://doi.org/10.1007/s12083-016-0440-9>.

GOLLMANN, D. Computer security. **Wiley Interdisciplinary Reviews: Computational Statistics**, v. 2, n. 5, p. 544–554, 2010. ISSN 19395108. Disponível em: <https://doi.org/10.1002/wics.106>.

GOMES, H. M. et al. A survey on ensemble learning for data stream classification. **ACM Computing Surveys**, v. 50, n. 2, 2017. ISSN 15577341. Disponível em: <https://doi.org/10.1145/3054925>.

GROSSI, V.; TURINI, F. Stream mining: A novel architecture for ensemble-based classification. **Knowledge and Information Systems**, v. 30, n. 2, p. 247–281, 2012. ISSN 02191377. Disponível em: <https://doi.org/10.1007/s10115-011-0378-4>.

HAMMERSCHMIDT, C. et al. Efficient Learning of Communication Profiles from IP Flow Records. **Proceedings - Conference on Local Computer Networks, LCN**, p. 559–562, 2016. Disponível em: <https://doi.org/10.1109/LCN.2016.92>.

_____. Behavioral clustering of non-stationary IP flow record data. **2016 12th International Conference on Network and Service Management, CNSM 2016 and Workshops, 3rd International Workshop on Management of SDN and NFV, ManSDN/NFV 2016, and International Workshop on Green ICT and Smart Networking, GISN 2016**, p. 297–301, 2017. Disponível em: <https://doi.org/10.1109/CNSM.2016.7818436>.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. The Elements of Statistical Learning Data Mining, Inference, and Prediction. 2013. Disponível em: <10.1007/BF02985802>.

HINDY, H. et al. A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems. **IEEE Access**, v. 8, p. 104650–104675, 2020. ISSN 21693536. Disponível em: <https://doi.org/10.1109/ACCESS.2020.3000179>.

HULTEN, G.; SPENCER, L.; DOMINGOS, P. Mining time-changing data streams. **Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, v. 18, p. 97–106, 2001. Disponível em: <https://doi.org/10.1145/502512.502529>.

INDRE, I.; LEMNARU, C. Detection and prevention system against cyber attacks and botnet malware for information systems and internet of things. In: IEEE. **2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP)**. 2016. p. 175–182. Disponível em: <https://doi.org/10.1109/ICCP.2016.7737142>.

JIANGUO, J. et al. Botnet detection method analysis on the effect of feature extraction. In: **2016 IEEE Trustcom/BigDataSE/ISPA**. [s.n.], 2016. p. 1882–1888. Disponível em: <https://doi.org/10.1109/TrustCom.2016.0288>.

KARIM, A. et al. Botnet detection techniques: review, future trends, and issues. **Journal of Zhejiang University-SCIENCE C (Computers & Electronics)**, 2014. Disponível em: <https://doi.org/10.1631/jzus.C1300242>.

KENYON, A.; DEKA, L.; ELIZONDO, D. Are Public Intrusion Datasets Fit for Purpose Characterising the State of the Art in Intrusion Event Datasets. **Computers and Security**, Elsevier Ltd, v. 99, p. 102022, 2020. ISSN 0167-4048. Disponível em: <https://doi.org/10.1016/j.cose.2020.102022>.

KHANCHI, S.; ZINCIR-HEYWOOD, N.; HEYWOOD, M. Streaming Botnet traffic analysis using bio-inspired active learning. **IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS**

2018, IEEE, p. 1–6, 2018. Disponível em: <https://doi.org/10.1109/NOMS.2018.8406293>.

KHATTAK, S.; RAMAY, N. R.; KHAYAM, S. A. L. I. A Taxonomy of Botnets: Features , Detection and Defense. **IEEE Communications Surveys and Tutorials (2014)**, V, p. 1–48, 2014. Disponível em: <https://doi.org/10.1109/SURV.2013.091213.00134>.

KIRKBY, R. **Hoeffding Tree**. 2020. Disponível em: <https://github.com/Waikato/moa/blob/master/moa/src/main/java/moa/classifiers/trees/HoeffdingTree.java>.

_____. **Ozaboost**. 2020. Disponível em: <https://github.com/Waikato/moa/blob/master/moa/src/main/java/moa/classifiers/meta/OzaBoost.java>.

KRAWCZYK, B. et al. Ensemble learning for data stream analysis: A survey. **Information Fusion**, v. 37, p. 132–156, 2017. ISSN 15662535. Disponível em: <https://doi.org/10.1016/j.inffus.2017.02.004>.

MIRKOVIC, J.; REIHER, P. A taxonomy of DDoS attack and DDoS defense mechanisms. **SIGCOMM Comput. Commun. Rev.**, v. 34, n. 2, p. 39–53, 2004. ISSN 01464833. Disponível em: <https://doi.org/10.1145/997150.997156>.

NAZARIO, J.; HOLZ, T. As the net churns: Fast-flux botnet observations. In: IEEE. **2008 3rd International Conference on Malicious and Unwanted Software (MALWARE)**. 2008. p. 24–31. Disponível em: <https://doi.org/10.1109/MALWARE.2008.4690854>.

NIELES, M.; DEMPSEY, K.; PILLITTERI, V. Y. NIST Special Publication 800-12 Revision 1 - An introduction to information security. **NIST Special Publication**, 2017. Disponível em: <https://doi.org/10.6028/NIST.SP.800-12r1>.

NOGUEIRA, M. Anticipating moves to prevent botnet generated ddos flooding attacks. **arXiv preprint arXiv:1611.09983**, 2016.

OZA, N. C.; RUSSEL, S. Online bagging and boosting. **Artificial Intelligence and Statistics 2001**, Artificial Intelligence and Statistics 2001, p. 105–112, 2001. Disponível em: <https://doi.org/10.1109/ICSMC.2005.1571498>.

PARMISANO, S. G. A.; ERQUIAGA, M. J. **Stratosphere Laboratory. A labeled dataset with malicious and benign IoT network traffic.** 2020. Disponível em: <https://www.stratosphereips.org/datasets-iot23>.

PEKTAŞ, A.; ACARMAN, T. Botnet detection based on network flow summary and deep learning. **International Journal of Network Management**, Wiley Online Library, v. 28, n. 6, p. e2039, 2018. Disponível em: <https://doi.org/10.1002/nem.2039>.

PERAKOVIĆ, D.; PERIŠA, M.; CVITIĆ, I. Analysis of the IoT impact on volume of DDoS attacks. **XXXIII Simpozijum o novim tehnologijama u poštanskom i telekomunikacionom saobraćaju, PosTel 2015, Beograd**, 2015.

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>.

RODRIGUEZ-GOMEZ, R. A.; MACIA-FERNANDEZ, G.; GARCIA-TEODORO, P. Survey and taxonomy of botnet research through life-cycle. **ACM Computing Surveys**, v. 45, n. 4, 2013. ISSN 03600300. Disponível em: <https://doi.org/10.1145/2501654.2501659>.

SAAD, S. et al. Detecting P2P botnets through network behavior analysis and machine learning. **2011 9th Annual International Conference on Privacy, Security and Trust, PST 2011**, IEEE, p. 174–180, 2011. Disponível em: <https://doi.org/10.1109/PST.2011.5971980>.

SETTLES, B. Active Learning Literature Survey. **Computer Sciences Technical Report 1648**, v. 65, n. 5, p. 854–856, 2010. ISSN 0167577X. Disponível em: <https://doi.org/10.1016/j.matlet.2010.11.072>.

SHIRAVI, A. et al. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. **Computers and Security**, Elsevier Ltd, v. 31, n. 3, p. 357–374, 2012. ISSN 01674048. Disponível em: <https://doi.org/10.1016/j.cose.2011.12.012>.

SILVA, L. F. B. et al. Study on machine learning techniques for botnet detection. **IEEE Latin America Transactions**, IEEE, v. 18, n. 05, p. 881–888, 2020. Disponível em: <https://doi.org/10.1109/TLA.2020.9082916>.

SILVA, S. S. et al. Botnets: A survey. **Computer Networks**, v. 57, n. 2, p. 378–403, 2013. ISSN 13891286. Disponível em: <https://doi.org/10.1016/j.comnet.2012.07.021>.

SILVA, T. P. da. **Abordagem Fuzzy para Detecção de Novidade em Fluxo Contínuo de Dados**. Dissertação (Mestrado) — UFSCar, 2018. Disponível em: <https://repositorio.ufscar.br/handle/ufscar/10544>.

SINGH, K. et al. Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests. **Information Sciences**, Elsevier Inc., v. 278, p. 488–497, 2014. ISSN 00200255. Disponível em: <https://doi.org/10.1016/j.ins.2014.03.066>.

SPAMHAUS. **Botnet Threat Report 2019**. [S.l.], 2019. Accessed: 24 Jun 2020. Disponível em: <http://www.spamhaus.org>.

STALLINGS, W. **Computer Security Third Edition**. [S.l.: s.n.], 2014. ISBN 9780133773927.

STEVANOVIC, M.; PEDERSEN, J. M. An efficient flow-based botnet detection using supervised machine learning. **2014 International Conference on Computing, Networking and Communications, ICNC 2014**, p. 797–801, 2014. Disponível em: <https://doi.org/10.1109/ICCNC.2014.6785439>.

TAN, M. S. P.-N.; KUMAR, V. **Introduction to Data Mining, (First Edition)**. [S.l.]: Pearson, 2006. ISBN 0321321367.

UMER, M. F.; SHER, M.; BI, Y. Flow-based intrusion detection: Techniques and challenges. **Computers & Security**, Elsevier, v. 70, p. 238–254, 2017. Disponível em: <https://doi.org/10.1016/j.cose.2017.05.009>.

VIEGAS, E. et al. BigFlow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. **Future Generation Computer Systems**, Elsevier B.V., v. 93, p. 473–485, 2019. ISSN 0167739X. Disponível em: <https://doi.org/10.1016/j.future.2018.09.051>.

VIEGAS, E. K.; SANTIN, A. O.; OLIVEIRA, L. S. Toward a reliable anomaly-based intrusion detection in real-world environments. **Computer Networks**, Elsevier, v. 127, p. 200–216, 2017. Disponível em: <https://doi.org/10.1016/j.comnet.2017.08.013>.

WAIKATO, T. U. of. **MOA - Machine Learning for Streams**. 2018. Disponível em: <https://moa.cms.waikato.ac.nz/>.

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**. [s.n.], 2003. p. 226–235. Disponível em: <https://doi.org/10.1145/956750.956778>.

WANG, J.; PASCHALIDIS, I. C. Botnet Detection Based on Anomaly and Community Detection. **IEEE Transactions on Control of Network Systems**, v. 4, n. 2, p. 392–404, 2017. ISSN 23255870. Disponível em: <https://doi.org/10.1109/TCNS.2016.2532804>.

YANG, B. B.; GARCIA-MOLINA, H. Designing a super-peer network. In: IEEE. **Proceedings 19th international conference on data engineering (Cat. No. 03CH37405)**. 2003. p. 49–60. Disponível em: <https://doi.org/10.1109/ICDE.2003.1260781>.

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>.