Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Detecção de botnets baseada na análise de fluxos de rede utilizando estatística inversa

Daniele Adriana Goulart Lopes

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientador
Prof. Dr. João José Costa Gondim

Coorientador
Prof. Dr. Marcelo Antônio Marotta

Brasília
2022

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

GL864d

Goulart Lopes, Daniele Adriana
Detecção de botnets baseada na análise de fluxos de rede utilizando estatística inversa / Daniele Adriana Goulart Lopes; orientador João José Costa Gondim; co-orientador Marcelo Antônio Marotta. -- Brasília, 2022.
136 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2022.

1. botnet. 2. fluxo de rede . 3. detecção de anomalias. 4. estatística inversa . 5. seleção de atributos. I. Costa Gondim, João José, orient. II. Antônio Marotta, Marcelo , co orient. III. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Detecção de botnets baseada na análise de fluxos de rede utilizando estatística inversa

Daniele Adriana Goulart Lopes

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr. João José Costa Gondim (Orientador)
Presidente

Prof. Dr. Luis Paulo Faina Garcia
Membro Interno

Prof.a Dr.a Camila Ferreira Thé Pontes
Membro Externo

Prof. Dr. Lucas Bondan
Suplente

Prof. Dr. Marcelo Ladeira
Coordenador do Programa de Pós-graduação em Computação Aplicada

Brasília, 22 de Setembro de 2022

Dedicatória

Dedico este trabalho aos meus pais, por abdicarem de suas vidas em prol das realizações e da felicidade de seus filhos e por não medirem esforços para que eu chegasse até esta etapa da minha vida. É com muito amor que lhes dedico este trabalho.

Agradecimentos

Primeiramente, agradeço a Deus por me guiar, renovar minhas forças e por sempre permitir que eu alcance os meus objetivos.

Agradeço à minha família, em especial aos meus pais, Clóvis e Anita, por todas as orações, conselhos e por sempre acreditarem em mim, mais até do que eu mesma acredito. Aos meus irmãos, Elielson, Ju e Helaine, pela torcida e incentivo, principalmente nos momentos de desânimo e insegurança.

Agradeço imensamente ao meu orientador, o Prof. Dr. João Gondim e ao meu coorientador, o Prof. Dr. Marcelo Marotta, pelo profissionalismo, motivação e total apoio no desenvolvimento desta pesquisa, a qual não teria sido concluída com o mesmo êxito, sem o auxílio dos senhores. Também não poderia deixar de agradecer ao Prof. Dr. Marcelo Ladeira, por toda contribuição e orientação durante grande parte do desenvolvimento desta pesquisa.

Agradeço ao amigo Rafael Peccatiello, pelas incontáveis vezes que me ajudou, seja no mestrado ou no trabalho. Obrigada por toda paciência e amizade construída desde que cheguei a Brasília.

Por fim, agradeço à Universidade de Brasília, em especial ao Programa de Pós-Graduação em Computação Aplicada, por todo conhecimento ao longo desta trajetória e pela oportunidade de realizar este trabalho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Botnet é uma rede de computadores infectados, os quais são controlados remotamente por um *cybercriminal*, denominado *botmaster* e que tem como objetivo realizar ataques cibernéticos massivos, como DDoS, SPAM e roubo de informações. Os métodos tradicionais de detecção de *botnets*, normalmente baseados em assinatura, são incapazes de detectar *botnets* desconhecidas. A análise baseada em comportamento tem sido promissora para a detecção de tendências atuais de *botnets*, as quais estão em constante evolução. Considerando que um ataque de *botnet* à infraestrutura de TI do Centro de Coordenação de Operações Móvel (CCOp Mv) do Exército Brasileiro pode prejudicar o sucesso das operações, através do furto de informações sensíveis ou mesmo causando interrupção à sistemas críticos do CCOp Mv, esta dissertação propõe um mecanismo de detecção de *botnets* baseado na análise do comportamento de fluxos de rede. A técnica utilizada para detecção de *botnets* foi recentemente desenvolvida e é denominada *Energy-based Flow Classifier* (EFC). Essa técnica utiliza estatística inversa para detecção de anomalias e possui uma importante característica que é a sua fácil adaptação a novos domínios, o que pode ser promissor para detecção de *botnets* desconhecidas. Além disso, o EFC é um algoritmo considerado interpretável, permitindo analisar o modelo estatístico inferido. Com base nisso, propomos uma abordagem para seleção dos atributos mais informativos para a detecção de *botnets*, através da análise dos acoplamentos entre os pares de atributos calculados pelo EFC. Para avaliar a eficiência do modelo gerado, bem como avaliar os atributos selecionados pelo EFC, realizamos diversos experimentos, com três conjuntos de dados distintos. Os resultados obtidos foram comparados com diversos modelos gerados por algoritmos tradicionais de uma e de duas classes. Também fizemos experimentos com duas outras abordagens de seleção de atributos. Os resultados obtidos mostram que o EFC consegue manter resultados mais estáveis, independente do domínio, ao contrário dos demais algoritmos testados e principalmente, que o EFC pode ser empregado como uma técnica para seleção dos atributos mais relevantes.

Palavras-chave: *botnet*, fluxo de rede, detecção de anomalias, estatística inversa, seleção de atributos

Abstract

A botnet is a network of infected computers, which are remotely controlled by a cybercriminal, called botmaster, whose objective is to carry out massive cyberattacks, such as DDoS, SPAM, and information theft. Traditional botnet detection methods, usually signature-based, are unable to detect unknown botnets. Behavior-based analytics has held promise for detecting current botnet trends, which are constantly evolving. Considering that Botnet attacks on the IT infrastructure of the Brazilian Army's Mobile Operations Coordination Center (CCOp Mv) may harm the success of operations, through theft of sensitive information or even causing interruption to critical CCOp Mv systems, this dissertation proposes a botnet detection mechanism based on network flow behavior analysis. The main objective is to propose an additional layer of cyber protection to the CCOp Mv IT infrastructure. The technique used to detect botnets was recently developed and it is called Energy-based Flow Classifier (EFC). This technique uses inverse statistics for anomaly detection and has an important characteristic which is its easy adaptation to new domains, which can be promising for detecting unknown botnets. In addition, the EFC is considered an interpretable algorithm, allowing the analysis of the inferred statistical model. Based on this, we propose an approach for selecting the most informative features for botnet detection, by analyzing the couplings between the pairs of attributes calculated by the EFC. To evaluate the efficiency of the generated model, as well as to evaluate the features selected by the EFC, we carried out several experiments, with three different data sets. The results obtained were compared with several models generated by traditional one and two-class algorithms. We also experimented with two other feature selection approaches. The results obtained show that the EFC manages to maintain more stable results, regardless of the domain, unlike the other algorithms tested, and mainly, the EFC can be used as a technique for selecting the most relevant features.

Keywords: botnet, network flow, anomaly detection, inverse statistics, feature selection

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AD AdaBoost.

AM Aprendizado de Máquina.

ANN Artificial Neural Network.

APWG Anti-Phishing Working Group.

AUC Area Under the Curve.

CART Classification and Regression Trees.

CCOp Mv Centro de Coordenação de Operações Móvel.

CO Capacidades Operativas.

DDoS Negação de Serviço Distribuída.

DNS Domain Name System.

DoS Negação de Serviço.

DT Decision Tree.

EFC Energy-Based Flow Classifier.

Elenv Eliptic Envelop.

FPR False Positive Rate.

FTP File Transfer Protocol.

GLO Garantia da Lei e da Ordem.

GVA Garantia da Votação e Apuração.

HTTP Hypertext Transfer Protocol.

HTTPS Hyper Text Transfer Protocol Secure.

IDSs Sistemas de Detecção de Intrusão.

IForest Isolation Forest.

IoT Internet das Coisas.

IP Internet Protocol.

IRC Internet Relay Chat.

KNN k-Nearest Neighbors.

LOF Local Outlier Factor.

MGC Módulo de Gerenciamento das Comunicações.

MLP Multilayer Perceptron.

MTCA Módulo de Trabalho de Cooperação entre Agências.

MTCS Módulo de Trabalho de Células Segregadas.

MTEM Módulo de Trabalho de Estado-Maior.

NB Naive Bayes.

OCSVM One Class Support Vector Machine.

P2P Peer-to-Peer.

PC Posto de Comando.

PCA Principal Component Analysis.

PCAP Packet Capture.

PPC Pay-Per-Click.

RBF Radial Basis Function.

REPTree Reduced Error Pruning algorithm.

RF Random Forest.

SaaS Software as a Service.

SVC Support Vector Classifier.

SVM Support Vector Machine.

TCP Transmission Control Protocol.

TIC Tecnologia da Informação e Comunicações.

TPR True Positive Rate.

TTL Time To Live.

UNB University of New Brunswick.

USB Universal Serial Bus.

VOIP Voice Over Internet Protocol.

VPN Virtual Private Network.

WEB World Wide Web.

XSS Cross-Site Scripting.

Capítulo 1

Introdução

Neste Capítulo são apresentadas a contextualização e a definição do problema, a hipótese a ser investigada, a justificativa para o estudo de detecção de *botnets* baseada na análise de fluxo de rede, os objetivos geral e específicos, além das contribuições deste projeto de pesquisa.

1.1 Contextualização e Definição do Problema

O Centro de Coordenação de Operações Móvel (CCOp Mv) é um projeto do Exército Brasileiro que integra o Programa Proteger[1]. Seu objetivo é suprir a Força Terrestre de infraestrutura de Tecnologia da Informação e Comunicações (TIC) para apoiar as mais diversas operações de proteção da sociedade, como por exemplo as operações de Garantia da Lei e da Ordem (GLO) e de Garantia da Votação e Apuração (GVA) [2].

O CCOp Mv é constituído por Nós de Acesso e pelo Conjunto de Coordenação das Operações, o qual é composto por um Módulo de Gerenciamento das Comunicações (MGC), um Posto de Comando (PC), um Módulo de Trabalho de Estado-Maior (MTEM), um Módulo de Trabalho de Células Segregadas (MTCS) e um Módulo de Trabalho de Cooperação entre Agências (MTCA), conforme pode ser visualizado na Figura 1.1

O CCOp Mv implementa um Centro de Comando e Controle configurado para contribuir com a ampliação da capacidade de planejamento e coordenação da Força Terrestre e deve prover acesso aos sistemas táticos, estratégicos e críticos do Exército Brasileiro, bem como aos sistemas de órgãos públicos, como por exemplo Polícia Militar, Defesa Civil e Polícia Federal [2].

Dentre as diversas Capacidades Operativas (CO) que o CCOp Mv deve possuir, esta dissertação pretende contribuir com a CO30, relacionada à segurança das informações e comunicações, a qual visa fornecer proteção adequada, mantendo a integridade e a dispo-

[1] `http://www.epex.eb.mil.br/index.php/proteger`

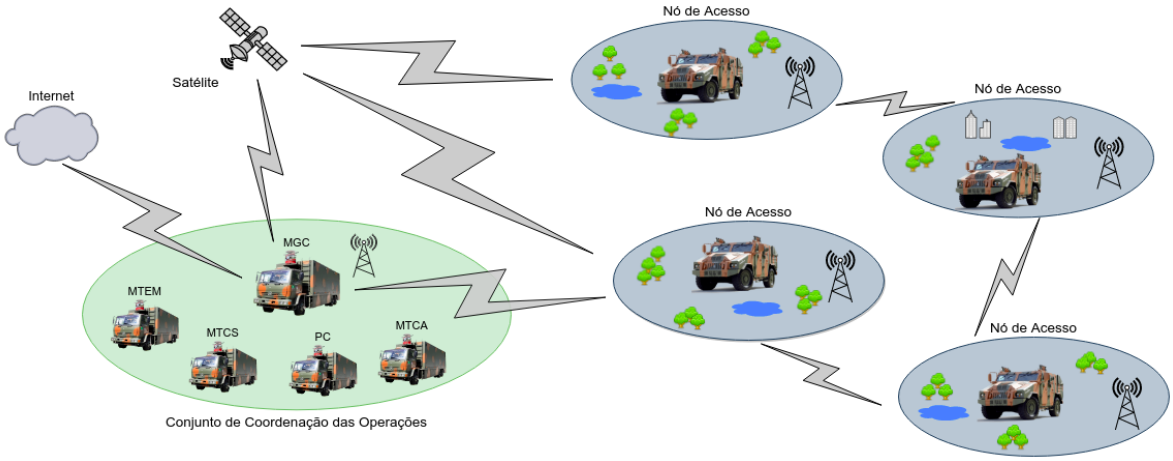


Figura 1.1: Composição do CCOp Mv.

nibilidade dos sistemas e das informações armazenadas, processadas ou transmitidas [2]. Pretende-se portanto, propor um modelo de classificação de fluxos de rede para detecção de *botnets*, as quais têm sido consideradas uma das principais ameaças à segurança entre todos os tipos de *malware* operando na Internet [3].

Uma *botnet* é uma rede formada por inúmeros dispositivos infectados por algum *malware*, os quais são denominados *bots* ou zumbis e que são controlados por um atacante, denominado *botmaster* [4]. O objetivo de uma *botnet* é realizar atividades maliciosas com base nas instruções fornecidas pelo *botmaster*. O principal componente de uma *botnet* é o servidor de Comando e Controle (C&C), por ser o meio pelo qual o *botmaster* controla e envia instruções aos *bots*, iniciando vários tipos de ataques cibernéticos, Negação de Serviço Distribuída (DDoS), SPAM, *phishing* e roubo de informações [5].

O potencial destrutivo das *botnets* têm aumentado exponencialmente com o avanço da tecnologia da Internet das Coisas (IoT) e à medida que aumenta o número de usuários e dispositivos conectados [6]. Em 2016 a *botnet* Mirai foi responsável por um dos maiores ataques de negação de serviço distribuído já registrado até hoje, estimado em 1,2 Tbps (terabits por segundo). Este ataque deixou fora do ar sites como Twitter, Netflix, CNN e vários outros pela Europa e Estados Unidos [7] [8]. Com a disponibilização do código-fonte da Mirai na Internet, muitos projetos variantes têm surgido. Em 2019, por exemplo, o número de variantes da *botnet* Mirai teve um crescimento de 57% em relação à 2018, ultrapassando 225.000 ocorrências [9].

Uma vez que os métodos tradicionais de detecção de *botnets* são baseados em assinaturas, estes se tornam eficientes para detectar tipos de *botnets* já conhecidos. Contudo, novos tipos de *botnets* ou de variantes de *botnets* conhecidas surgem cotidianamente e sua detecção é um grande desafio para os métodos tradicionais focados em assinaturas de ataques já conhecidos [4]. Além disso, as *botnets* evoluem constantemente, alterando sua

arquitetura e protocolos utilizados, com o intuito de evitar a detecção por sistemas de segurança. Adicionalmente, as *botnets* utilizam cada vez mais técnicas de criptografia e ofuscação [5]. Outro fator que aumenta o desafio da detecção é a dificuldade em diferenciar o tráfego normal da rede do tráfego contendo fluxo relacionado à atividade de uma *botnet*, uma vez que os protocolos utilizados pelas *botnets* são protocolos existentes, tais como *Hypertext Transfer Protocol (HTTP)*, *Peer-to-Peer (P2P)* e *Internet Relay Chat (IRC)*, tornando a caracterização do tráfego regular da rede uma tarefa não trivial [10].

À medida que as *botnets* têm progredido e se tornado mais complexas, várias estratégias de detecção de *botnet* têm sido propostas, principalmente utilizando métodos de Aprendizado de Máquina (AM) para análise de comportamento e detecção de anomalias [11]. Cada técnica proposta possui suas próprias vantagens e limitações no processo de detecção. Algumas técnicas são feitas particularmente para um protocolo específico, enquanto outras técnicas de detecção são dependentes de assinatura. Além disso, algumas técnicas não são capazes de detectar *botnets* que utilizam criptografia [12]. Sendo assim, faz-se necessária uma forma de detecção que seja independente dos protocolos de C&C e dos mecanismos de propagação utilizados e que não precise acessar dados do conteúdo do pacote, uma vez que estes podem estar criptografados [5].

No contexto de detecção de ataques por *botnets*, a maioria dos métodos diferenciam-se no tipo de análise realizada, sendo elas (*i*) análise profunda de pacotes ou (*ii*) análise de fluxos. Na primeira, os pacotes são individualmente analisados considerando seu cabeçalho e os dados sendo transportados (*payload*). Na segunda, um conjunto de pacotes são agrupados de acordo com características comuns presentes em seus cabeçalhos, sendo chamados de fluxos, os quais são avaliados de acordo com essas características e métricas estatísticas, como número de bytes e tempo de duração médio. A análise profunda de pacotes consome muito recurso computacional, uma vez que precisa processar todo o conteúdo do pacote, ao contrário da análise de fluxos, que só processa o cabeçalho dos pacotes, apresentando um processamento inferior a 10% do processamento gerado pela análise profunda de pacotes [13]. Além disso, a análise profunda de pacotes torna-se ineficiente se o tráfego de rede estiver criptografado, uma vez que o conteúdo dos pacotes não poderá ser lido. Já a análise baseada em fluxos pode detectar *botnets* que utilizam técnicas de criptografia ou ofuscação, como um túnel VPN, por exemplo, visto que esta técnica requer acesso apenas ao cabeçalho dos pacotes, os quais não são criptografados [5]. Dessa forma, a detecção por meio da análise de metadados de fluxos de *botnets* torna-se o foco a ser discutido nesse documento.

Técnicas de AM têm sido exploradas visando a detecção de *botnets* por meio da análise de metadados de fluxos de rede utilizando, muitas vezes, algoritmos convencionais de AM baseados em duas classes (binários) [14]. Estes algoritmos realizam o aprendizado a partir

de um conjunto de treinamento, contendo tráfego benigno e amostras de tráfego malicioso. A partir do conjunto de treinamento, um modelo é desenvolvido e é então utilizado para classificar instâncias do conjunto de teste. Como o comportamento do tráfego de rede muda e novas *botnets* estão surgindo continuamente [15], o aprendizado de máquina baseado nas duas classes pode não ser adequado para desenvolver métodos de detecção em *batch*, que sejam capazes de detectar *botnets* desconhecidas e que portanto, não estarão presentes no conjunto de treinamento. Além disso, não é fácil obter amostras de fluxos maliciosos que representem *malwares* recentes para compor o conjunto de treinamento [16]. Outra limitação dos algoritmos convencionais é que a maioria deles não consegue se adaptar bem a diferentes domínios, *i.e.,* após serem treinados em um conjunto de dados específico, não são facilmente generalizáveis para outros conjuntos de dados [17] [18]. Além disso, a maioria desses algoritmos gera modelos não interpretáveis, o que dificulta a análise e reajustes do modelo, caso necessário [19]. Dessa forma, é necessário o emprego de uma técnica de aprendizado de máquina para detecção de fluxos de *botnets*, capaz de generalizar seus resultados, utilizando apenas dados benignos, com boa adaptação a domínios e que gere modelos interpretáveis.

Pontes et al. [20] desenvolveu um algoritmo denominado *Energy-Based Flow Classifier (EFC)*, o qual foi inspirado no modelo inverso de *Potts* da mecânica estatística e adaptado para classificação de fluxos de rede. O EFC é um algoritmo *One Class*, que realiza a classificação utilizando apenas dados benignos para realizar o treinamento e não precisa conhecer o comportamento do tráfego malicioso para realizar a detecção de anomalias, contornando assim, o problema da dificuldade de se obter amostras maliciosas rotuladas [20]. Além disso, o EFC é um classificador intrinsecamente adaptável a diferentes domínios, uma vez que a inferência do modelo é baseada apenas em amostras benignas, sendo assim, não há necessidade de transformar os dados para adaptar o modelo ou fazer ajustes em um domínio diferente. Por fim, o EFC produz um modelo estatístico que pode ser analisado detalhadamente em relação aos valores dos parâmetros individuais, gerando portanto, um modelo interpretável [20].

O EFC foi utilizado em [20] para detecção de anomalias em geral, tais como ataques DDoS, *Port Scan*, *Ping Scan*, XSS, *Sql Injection* e *Brute Force*. Os resultados obtidos na detecção desses ataques, motivaram a escolha de utilização do EFC para detectar, especificamente, *botnets*, apesar de existirem outros algoritmos *One Class* na literatura. Além disso, uma vez que o EFC utiliza apenas dados benignos para realizar o treinamento, este algoritmo pode ser promissor para a detecção de tipos de *botnets* desconhecidos. Por fim, as redes militares possuem aplicações e serviços muito específicos, o que pode facilitar a rotulação do tráfego benigno, de forma a gerar um modelo de detecção confiável. Sendo assim, o objetivo deste trabalho consiste em avaliar o emprego do algoritmo *Energy-Based*

Flow Classifier (EFC) para detecção de *botnets* através da análise de metadados de fluxos de rede e ainda propor uma abordagem de seleção de atributos capaz de caracterizar o tráfego benigno, através da análise do modelo estatístico inferido. Para avaliar a eficiência do modelo, serão realizados testes de domínio cruzado, onde dois conjuntos de dados heterogêneos serão utilizados, sendo um conjunto de dados para treinamento do modelo e outro conjunto de dados para teste, com o objetivo de testar a adaptabilidade do modelo à diferentes domínios. Será realizada também a comparação de desempenho do EFC com diversos algoritmos tradicionais de uma e de duas classes, os quais serão descritos na Seção de Fundamentação Teórica. Por fim, utilizaremos um terceiro conjunto de dados para validar os atributos selecionados através da análise dos valores de acoplamento entre os pares de atributos calculados pelo EFC.

Dessa forma, esta pesquisa buscará evidências para a seguinte hipótese:

- O algoritmo EFC, o qual é baseado em estatística inversa, é capaz de detectar *botnets* através da análise de fluxos de rede, podendo ser uma solução para resolver o problema da adaptabilidade a diferentes domínios e ainda pode ser utilizado para seleção dos atributos que melhor caracterizam o tráfego benigno.

1.2 Justificativa

O recente crescimento da atividade de *botnet* no espaço cibernético atraiu de forma significativa a atenção da comunidade de pesquisa, principalmente pelo potencial destrutivo que as *botnets* possuem, sendo consideradas uma das ameaças mais danosas e complexas [21]. As *botnets* podem ser utilizadas, por exemplo, para interromper serviços através da coordenação de uma quantidade massiva de dispositivos infectados (*bots*), bem como podem auxiliar e executar ataques de roubo de informações. Segundo um estudo realizado pela Accenture, o prejuízo global estimado devido a crimes cibernéticos no período de 2018 a 2023 deve superar US$ 5 trilhões [22]. Para uma única empresa que é vítima desses ataques, o custo de ataques por *botnets* é de aproximadamente 400 mil dólares, em média [22].

Na tentativa de mitigar os ataques oriundos de *botnets*, diversos estudos científicos têm sido publicados, buscando maneiras de frear ou eliminar esta ameaça. Em resposta, os desenvolvedores e operadores de *botnets* têm se tornado mais agressivos e ofensivos, com o uso de criptografia em suas comunicações e maneiras de não evasão do seu ataque. Além disso, as *botnets* têm apresentado um alto nível de diversidade no que diz respeito a protocolos de comunicação, topologias e mecanismos de propagação, tornando a pesquisa contínua no campo pertinente [23].

No contexto do CCOp Mv, um ataque oriundo de uma *botnet* pode prejudicar o sucesso de uma operação, seja interrompendo sistemas críticos através de ataques DDoS ou mesmo com o roubo de informações sigilosas sobre determinada operação. Dessa forma, o presente estudo se justifica pelo impacto negativo que um ataque massivo oriundo de uma *botnet* pode causar às operações coordenadas pelo Exército Brasileiro através do projeto do CCOp Mv. Assim, a abordagem proposta nesta pesquisa visa contribuir com a segurança das informações e comunicações do projeto CCOp Mv, através da detecção antecipada da presença de tráfego relacionado à atividades de *botnets*.

1.3 Objetivos

O objetivo deste trabalho consiste em avaliar o emprego do algoritmo *Energy-Based Flow Classifier* para detecção de *botnets* através da análise do fluxo de rede e ainda propor uma abordagem para seleção de atributos que seja capaz de caracterizar o tráfego de *botnets*, através da análise do modelo estatístico inferido pelo EFC.

1.3.1 Objetivos Específicos

- Comparar o desempenho do EFC com classificadores tradicionais de uma classe para detecção de *botnets*;
- Comparar o desempenho do EFC com classificadores tradicionais de duas classes para detecção de *botnets*;
- Testar a adaptabilidade dos modelos através da avaliação do desempenho do EFC e dos demais classificadores quando testados em um domínio diferente daquele onde foi realizado o treinamento, por meio do teste de domínio cruzado;
- Comparar o desempenho dos classificadores tradicionais de uma e de duas classes, utilizando os atributos selecionados pelo EFC;
- Comparar os resultados obtidos com a seleção de atributos do EFC com os resultados obtidos utilizando os atributos de um estudo considerado como referência;
- Comparar os resultados obtidos com a seleção de atributos do EFC com os resultados obtidos utilizando os atributos selecionados por outra abordagem.

1.4 Contribuições

Esta pesquisa apresenta as seguintes contribuições:

1. Proposta de utilização do algoritmo EFC para detecção de *botnets*, propondo uma abordagem promissora para a detecção de *botnets* desconhecidas, independente da arquitetura e dos protocolos utilizados pelas *botnets*.

2. Extensa análise comparativa do desempenho do EFC com o desempenho de diversos classificadores de uma e de duas classes para a detecção de *botnets*;

3. Proposta de uma nova abordagem para extração de atributos, por meio da exploração da capacidade do EFC de gerar modelos interpretáveis. Assim, por meio da análise dos valores de acoplamentos entre os pares de atributos calculados pelo EFC, propomos um conjunto de atributos que se mostraram relevantes para a detecção de *botnets*, independente do conjunto de dados utilizado, mantendo inclusive, a adaptabilidade dos modelos.

1.5 Estrutura do Documento

O restante deste documento está organizado da seguinte maneira. No Capítulo 2 é feita uma fundamentação teórica, visando oferecer a base necessária para a compreensão do assunto abordado e seus desafios. No Capítulo 3 são citados trabalhos similares ou que influenciaram de alguma forma esta proposta de dissertação. No Capítulo 4 são apresentados os principais conceitos para compreender o algoritmo EFC, os conjuntos de dados utilizados, a metodologia proposta para este projeto de pesquisa, bem como os métodos de avaliação empregados. No Capítulo 5 são apresentados os resultados obtidos nos dois Estudos de Caso propostos. Finalmente, no Capítulo 6 é apresentada a conclusão desta pesquisa e os direcionamentos para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo, é apresentada a fundamentação teórica com a finalidade de fornecer subsídios à compreensão dos próximos capítulos. Inicialmente, é apresentado um embasamento teórico sobre *botnets*, seu ciclo de vida e as diferentes arquiteturas de um canal de Comando e Controle (C&C). Também são apresentados os principais ataques oriundos de *botnets*, bem como são abordadas as principais técnicas de detecção existentes. Por fim, é apresentada uma visão geral do EFC e dos classificadores de uma e de duas classes utilizados para comparação dos resultados obtidos.

2.1 *Botnets*

Uma *botnet* pode ser definida como uma rede de máquinas infectadas por algum *malware* que permite à um atacante controlar remotamente os recursos computacionais dessa rede e assim, realizar atividades maliciosas, como ataques de negação de serviço, roubo de informações sensíveis, envio massivo de *e-mails* (SPAMs), entre outras [24].

Historicamente, as *botnets* se originaram do sistema de chat *Internet Relay Chat (IRC)* e foram projetadas com intenções benignas. O Eggdrop, publicado em 1993, foi o primeiro *bot* IRC conhecido e tinha como objetivo oferecer assistência administrativa ao chat IRC, interpretando comandos simples, recuperando informações sobre sistemas operacionais, logins, endereços de e-mail, etc [10]. A partir de 1998 começaram a aparecer os *bots* de IRC maliciosos, com o objetivo principal de atacar outros usuários de IRC ou até mesmo servidores inteiros. Pouco tempo depois, ataques de Negação de Serviço (DoS) e, em seguida, ataques de Negação de Serviço Distribuída (DDoS) foram implementados nesses *bots* [10].

Com o passar do tempo, as *botnets* se tornaram sofisticadas e robustas, passando a utilizar mecanismos complexos de comunicação, além de explorar outros protocolos disponíveis e de integrar novos e poderosos métodos de ataque. A geração atual de *bots*

pode lançar ataques grandes e coordenados e possui a capacidade de se espalhar por meio de redes de compartilhamento de arquivos, redes ponto a ponto (P2P), anexos de e-mail, sites WEB infectados e podem ainda, ser instalados previamente em *backdoors* [10].

2.1.1 Componentes de uma *Botnet*

Uma *botnet* consiste em quatro componentes principais: o *botmaster*, o servidor de Comando e Controle (C&C), os *bots* e a vítima, conforme pode ser observado na Figura 2.1

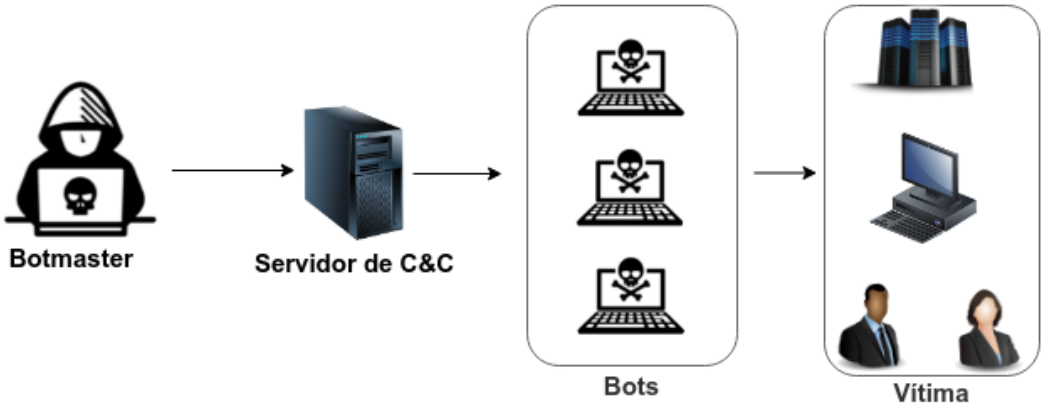


Figura 2.1: Componentes de uma Botnet.

- *bots*: são programas maliciosos (*malware*) instalados em um *host* vulnerável, capazes de realizar uma série de ações, normalmente ataques. Estes *malwares* podem ser instalados nas máquinas das vítimas de diversas maneiras, como através de vírus ou por meio de sites infectados [10]. Um *bot* pode compreender também os dispositivos comprometidos e integrantes da *botnet* [5].
- *botmaster*: é o indivíduo mal-intencionado que possui a capacidade de controlar os *bots* de modo que eles executem ações determinadas, sem o consentimento de seus respectivos donos [25]. Sendo assim, o *botmaster* é o mentor que instrui os *bots* e é o responsável por arquitetar estratégias para os mais variados tipos de ataques, além de ser o responsável por manter a comunicação com os *bots* por meio do servidor de Comando e Controle [5].
- Servidor de Comando e Controle (C&C): é o meio que atua como ponte entre o *botmaster* e a rede de *bots* [5]. Este é o principal componente no ambiente de *botnet*, uma vez que sem o servidor C&C, o *botmaster* não consegue controlar ou enviar instruções para os *bots* [10]. A estrutura deste servidor pode ser centralizada, por exemplo utilizando o protocolo IRC, ou descentralizada, fazendo uso de redes *Peer-to-Peer (P2P)* [5].

- Vítima: pode ser um sistema, uma pessoa ou uma rede e constituem o alvo do ataque executado. Existem muitos tipos de vítimas dependendo do objetivo principal do *botmaster*, como por exemplo: um usuário que recebe SPAM, alguém que teve informações confidenciais roubadas, uma empresa que perde milhões com um ataque DDoS, etc [25].

2.2 Ciclo de Vida de uma *Botnet*

O funcionamento de uma *botnet* pode ser compreendido por meio da análise do conjunto de fases funcionais durante a operação das *botnets*. Esse conjunto de fases é denominado ciclo de vida e sua compreensão é crucial, uma vez que as abordagens de detecção de *botnets* visam fases específicas do ciclo de vida [26]. O ciclo de vida básico de uma *botnet* pode ser compreendido em quatro fases [5], conforme ilustrado na Figura 2.2

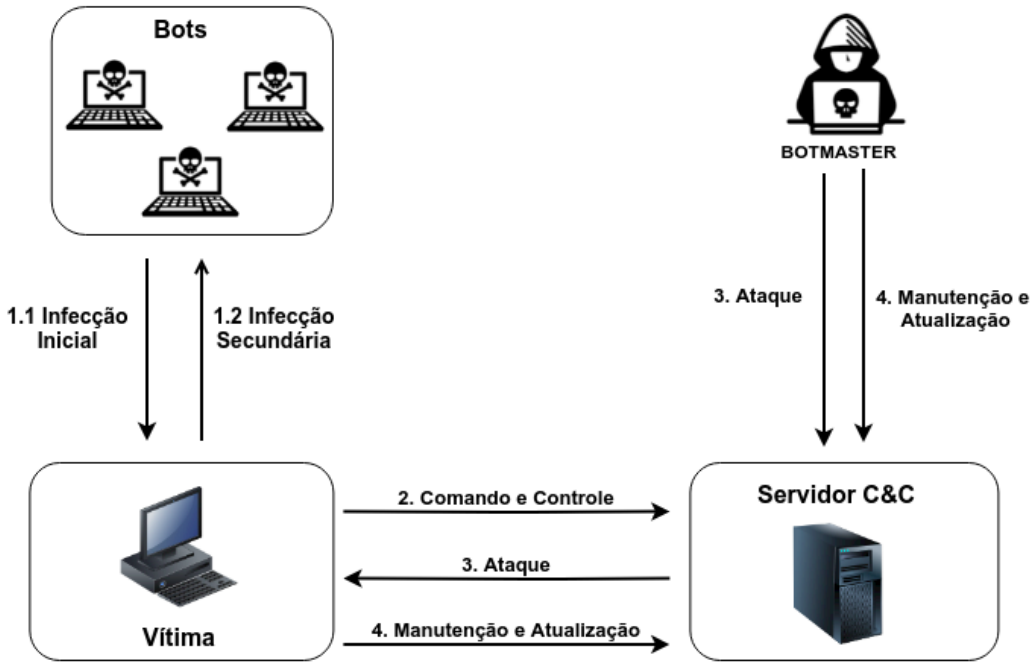


Figura 2.2: Ciclo de Vida de uma *Botnet*.

A primeira fase do ciclo de vida de uma *botnet* pode ser dividida em duas sub fases conhecidas como infecção inicial e infecção secundária. Durante a fase de infecção inicial, os computadores que possuem alguma vulnerabilidade são infectados por algum *malware*. A infecção inicial pode ser realizada de diferentes maneiras, como por meio do download de arquivos infectados anexados a mensagens de *e-mail*, por meio do download indesejado de *malwares* disponibilizados em sites maliciosos ou então através de discos removíveis infectados, etc [26]. A partir do momento em que a sub fase de infecção inicial é concluída

com sucesso, inicia-se a sub fase de infecção secundária, na qual o computador previamente infectado executa um programa que busca por códigos binários do *malware* de *bot* em um repositório externo. Esses binários podem ser baixados usando diversos protocolos, como por exemplo *File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP)* e *Hyper Text Transfer Protocol Secure (HTTPS)* [26]. Após o arquivo binário ter sido baixado e executado, a máquina passa a se comportar como um *bot* ou zumbi, podendo portanto, ser controlada pelo *botmaster* [5].

O objetivo principal da primeira fase é maximizar o número de máquinas infectadas (*bots*) infectando outros dispositivos. A maioria dos códigos binários de *bot* possuem mecanismos embutidos para facilitar sua propagação para outros *hosts*. Esses mecanismos de propagação podem ser classificados como ativos, quando a *botnet* é capaz de localizar e infectar outros *hosts* sem qualquer intervenção do usuário ou passivos, quando a propagação do *malware* requer algum nível de intervenção do usuário, seja por do compartilhamento de um dispositivo USB infectado ou através de um clique em um link malicioso [27].

A segunda fase do ciclo de vida da *botnet* é a fase de comando e controle (C&C). Esta fase se refere a todas as interações realizadas entre o *botmaster* e os computadores comprometidos (*bots*). O *botmaster* se comunica com os *bots* por meio do servidor de comando e controle, enviando instruções ou atualizações de códigos para os *bots*, e recebendo deles um relatório atualizado com as vulnerabilidades presentes nesses dispositivos infectados [5]. Isso proporciona às *botnets* habilidades únicas para descobrir vulnerabilidades de dispositivos desconhecidos e evoluir de forma autônoma, além de dificultar a sua detecção [5]. O principal fato que diferencia *botnets* de outros tipos de *malware* é a existência dos canais de C&C para realizar a comunicação com os dispositivos infectados, sendo assim, os canais de C&C são a espinha dorsal de uma *botnet* [27].

A comunicação entre os *bots* e o canal de C&C é de particular importância para a comunidade de pesquisa e muitos artigos sobre *botnets* estão diretamente relacionados a essa questão. Essa comunicação abrange vários modos de operação [26]:

- tentativa de conexão inicial com o servidor C&C após a fase de infecção bem-sucedida;
- tentativa de conexão do *bot* após a reinicialização da máquina comprometida;
- tentativas de conexão periódicas para relatar o status da máquina infectada;
- tentativas de conexão iniciadas pelo servidor C&C para atualizar o código do *malware* ou propagar instruções para os *bots*.

A terceira fase do ciclo de vida de uma *botnet* é conhecida como fase de ataque. Uma vez que a quantidade de *bots* é grande o suficiente para lançar um ataque, o grupo de *bots*

passa a executar atividades maliciosas nas máquinas alvo, conforme instruído pelo *botmaster*, enviando os comandos necessários aos servidores C&C [28]. Atividades de ataque comuns incluem DDoS, spam, disseminação de *malware*, vazamento de informações, ataques de *phishing* e mineração de moeda virtual [29]. Nesta fase, os *bots* também podem implementar mecanismos de propagação, i.e., realizando a varredura de computadores vulneráveis ou distribuindo software malicioso [26].

A última fase do ciclo de vida de uma *botnet* é a fase de manutenção e atualização. A manutenção é necessária para que o *botmaster* consiga manter seu exército de zumbis (*bots*) [30]. Sendo assim, o objetivo do *botmaster* nessa fase é manter seus *bots* atualizados, instruindo-os a baixar binários atualizados periodicamente. Pode ser necessário atualizar códigos por vários motivos, incluindo evasão de técnicas de detecção, adição de novos recursos ou migração para outro C&C [10]. A abordagem proposta nesta dissertação se concentra na fase de comando e controle para realizar a detecção de *botnets*, uma vez que nesta fase há uma comunicação periódica entre os *bots* e o canal de C&C, permitindo que mecanismos sejam criados para identificar padrões nesta comunicação [5]. Além disso, na fase de infecção a propagação pode acontecer de diversas maneiras, o que dificulta bastante a detecção. Por outro lado, realizar a detecção na fase de ataque pode ser tarde demais, uma vez que o *botmaster* já terá atingido o seu objetivo nesta fase [5].

2.3 Arquitetura do Canal C&C

O canal de C&C é o elemento mais crítico de uma *botnet*, uma vez que é através dele que o *botmaster* gerencia a rede de *bots*. A arquitetura do canal de C&C determina a confiabilidade, a robustez e tempo de resposta de uma *botnet* [31]. As *botnets* podem ser categorizadas em três diferentes estruturas de acordo com a topologia do canal de C&C: arquitetura centralizada, arquitetura descentralizada e arquitetura híbrida [10].

2.3.1 Arquitetura Centralizada

Em uma arquitetura de *botnet* centralizada, o *botmaster* controla todos os *bots* através de um único servidor de comando e controle [27], conforme ilustra a Figura 2.3. Sendo assim, o servidor de C&C é o único meio para que o *botmaster* envie instruções e receba atualizações dos *bots*.

As principais vantagens dessa arquitetura são a facilidade de implementação e a baixa latência na comunicação entre o *botmaster* e os *bots* [26]. Além disso, essa arquitetura permite ao *botmaster* fácil monitoramento do status da *botnet*, através do fornecimento de informações importantes, como o número de *bots* ativos ou sua distribuição global

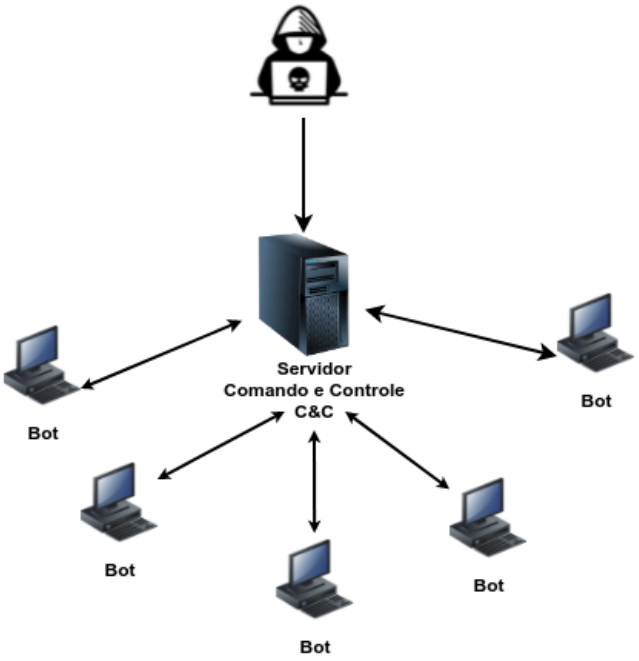


Figura 2.3: Arquitetura Centralizada.

[10]. Devido à sua simplicidade, arquiteturas centralizadas são amplamente utilizadas por muitas famílias de *botnets* [32].

No entanto, a principal desvantagem das *botnets* com arquitetura de rede centralizada é que elas apresentam um ponto único de falha. Sendo assim, uma vez que os servidores de C&C tenham sido identificados, é fácil bloquear este canal e impedir que o *botmaster* se comunique com os *bots*, interrompendo assim, o funcionamento de toda a *botnet* [32]. Os dois protocolos mais usados em uma arquitetura centralizada são o IRC e o HTTP [31].

a) *Botnet* baseada em IRC: O *Internet Relay Chat (IRC)* é um protocolo utilizado na Internet que permite aos usuários conversar através de mensagens de texto em tempo real. No caso de *botnets* que utilizam este protocolo, o *botmaster* cria canais de IRC no servidor de comando e controle (C&C) e faz com que as máquinas infectadas se conectem e aguardem os comandos para realizar uma atividade maliciosa [10]. Este protocolo é o canal de C&C de *botnet* mais popular e tem sido amplamente utilizado pelos *botmasters* [24] [33].

O protocolo IRC permite comunicação *unicast* privada entre dois membros e também comunicação *multicast* através de grupos, o que permite ao *botmaster* selecionar um grupo específico de *bots* para realizar um ataque [10]. Além disso, existem várias implementações de código aberto para servidores IRC, permitindo ao *botmaster* adaptar o protocolo para atender às suas necessidades e criar novas *botnets*. As *botnets* baseadas em IRC mais

famosas são: Spybot, Agobot, SDBot e GT Bot [31].

Apesar das vantagens apresentadas, o IRC possui sérias limitações, uma vez que em redes corporativas este protocolo geralmente não é liberado. Sendo assim, é fácil detectar e interromper a operação de uma *botnet* IRC, bastando para isso que o administrador da rede configure o *firewall* para bloquear o tráfego utilizando este protocolo [10].

b) *Botnet* baseada em HTTP: Devido às restrições de tráfego de IRC em redes corporativas, o *Hypertext Transfer Protocol (HTTP)* tornou-se popular como um mecanismo para implementar a comunicação C&C [24]. A principal vantagem sobre a comunicação IRC é que o tráfego HTTP é amplamente utilizado em muitas aplicações baseadas na web, assim a comunicação entre os *bots* e o *botmaster* se mistura ao tráfego HTTP regular, o que dificulta a detecção do tráfego de *botnet* [10]. *Botnets* que utilizam o protocolo HTTP como canal de C&C podem facilmente evadir os Sistemas de Detecção de Intrusão (IDSs) e contornar *firewalls* com técnicas de filtragem baseadas em portas [31]. No entanto, este protocolo ainda apresenta o problema de ser o ponto único de falha, já que também emprega uma arquitetura centralizada [10]. *Botnets* bem conhecidas que utilizam o protocolo HTTP são: Bobax, ClickBot, Rustock e a mais popular, Blackenergy [31].

2.3.2 Arquitetura Descentralizada

Com o intuito de tornar as *botnets* mais robustas, o *botmaster* pode utilizar uma estrutura de C&C descentralizada, através do emprego de protocolos *Peer-to-Peer (P2P)* como meio de comunicação dentro de uma *botnet* [27]. Nessa arquitetura os *bots* pertencentes à *botnet* P2P formam uma rede sobreposta, permitindo ao *botmaster* utilizar qualquer um dos *bots* para distribuir comandos a outros *bots* (*peers*) ou para coletar informações sobre eles [26]. Sendo assim, qualquer nó pode atuar tanto como cliente como servidor simultaneamente [23]. Para enviar comandos para todos os *bots* pertencentes à *botnet*, o *botmaster* precisa se conectar a apenas um dos *bots* [34]. As *botnets* P2P são implementadas utilizando alguns dos protocolos de transferência P2P existentes, tais como Waste, BitTorrent, Kademlia, Direct Connect, Gnutella e Overnet [27].

As *botnets* que possuem arquitetura descentralizada são mais difíceis de serem desarticuladas, visto que a descoberta de um ou vários *bots* não significa necessariamente a perda de toda a *botnet*, pois não há mais um único servidor de C&C a ser encontrado e desabilitado [10]. Assim, os *bots* restantes ainda podem ser capazes de se comunicar uns com os outros e com o *botmaster*, continuando a execução das atividades maliciosas propostas pelo *botmaster* [26]. Isso explica porque *botnets* P2P são geralmente mais resilientes contra defesas do que as *botnets* com arquitetura centralizada. Além disso, a arquitetura P2P oferece mais flexibilidade e robustez, especialmente quando o número de *bots* é grande [34].

No entanto, as *botnets* P2P apresentam uma grande desvantagem, pois não podem garantir alta confiabilidade e baixa latência na comunicação com o C&C, o que limita severamente a eficiência geral de orquestrar ataques coordenados em grande escala [26]. Além disso, possuem implementação e gerenciamento mais complexos em comparação com *botnets* centralizadas [21]. Storm, Waledac, ZeroAccess, Citadel, Kelihos e Conficker são alguns expemplos de *botnets* P2P populares [35]. A Figura 2.4 ilustra uma arquitetura P2P.

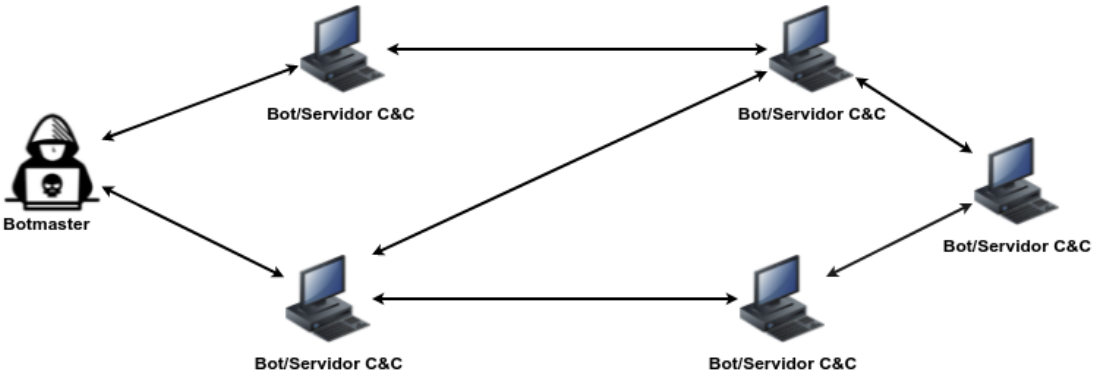


Figura 2.4: Arquitetura Descentralizada.

2.3.3 Arquitetura Híbrida

As arquiteturas híbridas combinam princípios das arquiteturas de C&C centralizadas e descentralizadas, com a finalidade de tentar obter a resiliência de *botnets* descentralizadas com a baixa latência de comunicação das *botnets* centralizadas [26]. Os *bots* pertencentes a uma *botnet* de arquitetura P2P híbrida são classificados em dois grupos distintos: o grupo de *bots* de servidores (proxy) e o grupo de *bots* de clientes.

Bots pertencentes ao grupo de servidores contém endereços IP roteáveis e podem se comportar tanto como clientes quanto como servidores, enquanto os *bots* clientes contêm endereços IP dinâmicos e não roteáveis. Os *bots* clientes se conectam periodicamente aos *bots* servidores para receber comandos e então implementar as atividades maliciosas, enquanto os *bots* servidores são responsáveis por fornecer a propagação das mensagens de C&C de e para o botmaster [36]. Os sistemas híbridos de C&C também podem ser localizados atrás de *firewalls* sem uma conexão global com a Internet.

Exemplos de *botnets* que utilizam uma topologia híbrida são a *botnet* Miner [37] e versões posteriores do *botnet* Zeus [38]. A versão 3 da *botnet* Sality também utiliza uma rede híbrida, na qual a parte P2P é responsável pela troca de comandos, enquanto servidores Web centralizados são utilizados para baixar os dados [39].

2.4 Ataques de *Botnets*

Independentemente da arquitetura utilizada, uma *botnet* é projetada com a finalidade de executar uma grande variedade de ataques, com alto poder destrutivo. Esta seção descreve alguns dos principais ataques executados por meio de uma *botnet*.

2.4.1 Negação de Serviço Distribuída (DDoS)

Os ataques de Negação de Serviço Distribuída (DDoS) são tentativas mal-intencionadas de interromper as operações normais de um servidor, serviço ou rede ao sobrecarregá-los com uma grande quantidade de tráfego, tornando os serviços providos pela vítima inacessíveis aos usuários legítimos [40]. Os ataques DDoS são uma das principais ameaças na Internet, uma vez que o número desses ataques tem aumentado exponencialmente ao longo dos anos [41]. As *botnets* são perfeitamente adequadas para lançar ataques DDoS, uma vez que o grande número de participantes (*bots*) de uma *botnet* amplifica o poder do ataque, aumentando seu potencial destrutivo [42].

Existem diferentes formas de realizar esse tipo de ataque, seja consumindo recursos essenciais para o funcionamento de determinado serviço (e.g., memória, processamento, espaço em disco e banda), ou através da exploração de vulnerabilidades existentes nas vítimas, como por exemplo, falhas no código da aplicação alvo, falhas no próprio sistema operacional utilizado ou até mesmo no protocolo de comunicação [42]. Exemplos de *botnets* usados para DDoS são Spybot, RBot e Agobot [43].

O principais alvos de um ataque DDoS são as instituições financeiras, sites de comércio eletrônico, as agências governamentais e o setor de saúde [44]. O impacto dos ataques DDoS pode variar desde pequenos inconvenientes para os usuários de um site, até sérias perdas financeiras para empresas que dependem da disponibilidade de sistemas online para fazer negócios [42]. Em fevereiro de 2000, o Yahoo sofreu um dos primeiros grandes ataques de inundação DDoS, o qual manteve os serviços da empresa fora da Internet por cerca de 2 horas, ocasionando uma perda significativa na receita de publicidade [45]. Somente em 2014, perdas monetárias de cerca de US$ 491 bilhões foram relatadas [46]. Durante o primeiro trimestre de 2018, cerca de 79 países foram afetados por ataques DDoS, sendo que o ataque mais longo teve a duração de aproximadamente 297 horas [47]. Em 2020, em meio à pandemia do COVID-19, um ataque de extorsão DDoS estimado em 2 TBps, em todo o mundo, visando o setor de finanças e indústrias de viagens foi relatado pela NetScout [48]. Prevê-se que os ataques DDoS cheguem a 15,4 milhões em 2023 [49].

A interrupção de serviços da rede pode potencialmente se transformar em guerra cibernética, caso o ataque seja motivado por um estado tentando interromper a infraestrutura cibernética de outro estado, como ficou evidente no caso do ataque cibernético contra a

Estônia [50]. Além disso, a capacidade de lançar ataques DDoS e tornar sites e serviços críticos indisponíveis também tem sido utilizada para extorsão cibernética, pois geralmente, as grandes empresas estão dispostas a pagar dinheiro de extorsão aos *botmasters* evitando a perda de vendas e de credibilidade [40]. Os ataques DDoS podem ser compreendidos como um campo de batalha de recursos entre os defensores e os atacantes, no qual quanto mais recursos, maior a chance de sucesso [40].

2.4.2 Propagação de SPAM em Massa

SPAM é qualquer tipo de comunicação digital indesejada e não solicitada, a qual é enviada em massa indiscriminadamente [51]. As *botnets* são utilizadas como um meio eficiente para a propagação de grandes quantidades de *SPAM* diariamente. Normalmente, o SPAM é enviado através de mensagens de *e-mail*, mas também pode ser distribuído por mensagens de texto, *Voice Over Internet Protocol (VOIP)* ou redes sociais [51]. No ano de 2021, o número total de usuários de *e-mail* em todo o mundo foi estimado em mais de 4,1 bilhões, e espera-se que ultrapasse 4,5 bilhões até o final de 2025 [52]. Isso implica que mais da metade da população mundial atualmente se comunica por *e-mail*, o que contribui para o aumento dos crimes cibernéticos relacionados a esse tipo de comunicação [53].

Normalmente, os *e-mails* de SPAM têm uma aparência atraente, contendo imagens ou textos tentadores para chamar a atenção dos usuários. Muitos deles são frequentemente utilizados como uma ferramenta de publicidade para entregar anúncios a um grande número de usuários-alvo, através da Internet [54]. Apesar de causar inconvenientes para os usuários, esse tipo de SPAM geralmente não é prejudicial. Porém, *e-mails* de SPAM se tornam uma ameaça à segurança cibernética quando utilizados como vetor de ataque para distribuir ou habilitar outras ameaças (e.g. ataque de *phishing* e distribuição de *malware*) [54].

Os atacantes costumam lançar campanhas de SPAM, aproveitando assuntos de grande vulto e de interesse global. As campanhas de SPAM com iscas relacionadas à COVID-19, por exemplo, foram muito exploradas, com o objetivo de tentar induzir os usuários a solicitar máscaras faciais de sites falsos, visando infectá-los com *malware* por meio de anexos maliciosos [9]. Três quartos dos anexos desses *e-mails* continham *infostealers*, i.e., um tipo de *malware* que rouba informações confidenciais, como senhas ou outras credenciais. À medida que a pandemia continuou e a vacina foi lançada, as campanhas passaram a incluir anúncios ou ofertas de acesso antecipado a vacinas, mediante o pagamento de caução ou taxa, além de campanhas de desinformação sobre as vacinas [9].

Exemplos populares de *botnets* que pertencem à esta classe são as *botnets* Necrus e Gamut, as quais de acordo com o relatório de ameaças do McAfee Labs foram responsáveis por 97% do tráfego global de *botnets* de SPAM em de março de 2018 [55]. Outras classes

incluem as *botnets* Rustock, Cutwail, Tempestade, Waledac, Bagle, Storm, Marina, entre outras [51].

2.4.3 *Phishing*

Em um ataque de *phishing*, o invasor envia um *e-mail* fazendo-se passar por uma organização ou uma pessoa legítima. O cibercriminoso utiliza técnicas de engenharia social para incentivar o destinatário a clicar em um link suspeito [56]. Este link pode baixar um aplicativo malicioso ou fornecer um formulário que solicita que o destinatário insira informações pessoais confidenciais (e.g. endereço de *e-mail*, nome de usuário, senha ou informações bancárias) [57]. Estas informações são então utilizadas pelo atacante em detrimento da vítima [58]. A lógica de denominar esse tipo de ataque com o termo *phishing* vem da ideia de um invasor utilizar uma "isca" para atrair a vítima e depois "pescar" as informações pessoais que deseja ter acesso [59]. Este é um dos principais vetores de ataque utilizados pelos hackers. Os ataques de *phishing* podem ter como alvo indivíduos, funcionários, corporações ou governos. Os invasores são motivados por muitos fatores, como obter benefícios financeiros, ganhar reputação na comunidade de criminosos cibernéticos, influenciar opiniões (e.g. campanha eleitoral), espalhar notícias falsas, etc [60].

De acordo com o Relatório de Tendências de Atividade de *Phishing* divulgado pelo APWG, foram observados 1.025.968 ataques de *phishing* somente no primeiro trimestre de 2022, sendo o pior trimestre para *phishing* que o APWG já observou, além de ser a primeira vez que a quantidade deste tipo de atividade ultrapassa um milhão em apenas um trimestre [61]. O relatório ainda revela que o setor financeiro foi a vítima mais frequente de *phishing*, com 23,6% de todos os ataques do primeiro trimestre. Além disso, os ataques contra provedores de *webmail* e *Software as a Service (SaaS)* continuam numerosos, correspondendo a 20,5% do total [61]. O relatório de Inteligência de Ameaças X-Force apontou que o *phishing* foi observado em 41% dos incidentes remediados por este time, emergindo como o principal vetor de infecção no ano de 2021 [62].

As *botnets* são utilizadas para ataques de *phishing* com o intuito de prover uma camada de segurança, protegendo os sites falsos que são criados para enganar as vítimas. Para isso, utiliza-se uma técnica chamada redes de fluxo rápido (*fast-flux*), que consiste em adquirir um grande número de *proxies* que redirecionam as solicitações dos usuários para o servidor de *phishing* [25]. Esses *proxies* mudam com muita frequência, utilizando entradas DNS com baixo *Time To Live (TTL)*, o que prolonga a vida útil dos sites de *phishing*, tornando difícil detectá-los e colocá-los offline. Para um proprietário de rede de fluxo rápido, é importante ter um grande número de *proxies* disponíveis e também é desejável que suas localizações sejam heterogêneas (i.e., redes diferentes). Assim, os *bots* em uma

botnet se encaixam perfeitamente no papel de *proxies* em uma rede *fast-flux* [25]. A *botnet* Storm, por exemplo, implementa esse mecanismo para ocultação de sites de *phishing* [63].

2.4.4 Coleta/Reconhecimento de Informações

Essa classe de *botnets* é usada para extrair, diariamente, grandes quantidades de informações da Internet. Elas também aparecem nas operações de espionagem. Um exemplo popular de uma *botnet* que pertence a esta classe é a *botnet* Mirai, descoberta em agosto de 2016, a qual tinha o objetivo de escanear a Internet em busca de endereços IP de dispositivos IoT vulneráveis [8], para então infectá-los e utilizá-los como parte da *botnet*. A *botnet* Satori é uma das variantes mais temidas da *botnet* Mirai, sendo descoberta em maio de 2018. A Satori apresenta operação semelhante ao Mirai, porém possui como foco a mineração de informações relativas a infraestruturas vulneráveis de gerenciamento remoto de criptomoedas, com o objetivo de infiltrar posteriormente as carteiras dos usuários para roubar suas criptomoedas [64].

2.4.5 Roubo de Identidade

O roubo de identidade, ou *identity theft*, é o ato pelo qual uma pessoa tenta se passar por outra, atribuindo-se uma falsa identidade, com o objetivo de obter vantagens indevidas. Esta classe de *botnets* está envolvida no roubo de grandes quantidades de informações privadas de usuários, geralmente para fins fraudulentos, tais como: detalhes de cartão de crédito, informações de registro de saúde, nomes de usuário e senhas, entre outras formas de informações confidenciais [65].

A combinação de várias funcionalidades de *botnets* pode, muitas vezes, ser utilizada para realizar este tipo de ataque. Neste caso, uma mensagem falsa enviada por *e-mail* (*e-mails* de *phishing*), exige que a possível vítima acesse determinado site e confirme seus dados privados. Esses *e-mails* podem ser gerados e enviados por *botnets* utilizando mecanismos de *SPAM* [65]. Em uma etapa adicional, as *botnets* também podem configurar vários sites falsos fingindo ser sites de negócios oficiais para coletar informações das vítimas.

Exemplos populares de *botnets* que pertencem a essa classe incluem a *botnet* Zeus, a qual comprometeu mais de 74.000 contas FTP em sites de empresas como o *Bank of America*, NASA, Oracle, Cisco e Amazon, roubando informações bancárias confidenciais por meio do registro do pressionamento de teclas e da captura de formulários HTTP, e também a *botnet* Bredolab, a qual foi desenvolvida em 2009 para extrair senhas de contas bancárias e outras informações confidenciais de computadores infectados [66]. Outras *botnets* desta classe incluem as *botnets* Torpig, Alureon e Mariposa.

2.4.6 Fraude de Cliques

Com o crescimento das tecnologias *Web* e visando atrair novos clientes, as empresas de publicidade têm investido fortemente nos anúncios online, em detrimento dos anúncios em jornais convencionais ou televisão. A publicidade online é a maior fonte de receita para grandes empresas como *Google, Yahoo e Facebook*, as quais são consideradas redes de publicidade, atuando como intermediárias entre os anunciantes e os editores de conteúdo [67].

Normalmente, os anúncios disponibilizados por uma rede de publicidade são cobrados em um modelo denominado *Pay-Per-Click (PPC)*, no qual para cada clique em um anúncio que leva ao site de um anunciante, este paga à rede de publicidade, que, por sua vez, paga uma parte do valor recebido ao editor. A fraude de cliques é um tipo de crime que abusa do modelo PPC, induzindo, por engano, os usuários a clicar em anúncios online ou a visitar determinados sites, com o objetivo de aumentar as receitas dos editores ou impactar negativamente o orçamento dos anunciantes [67].

As *botnets* permitem simular o comportamento de milhões de usuários legítimos sendo, portanto, ideais para esse tipo de ataque, no qual os computadores comprometidos (*bots*) são instruídos para visitar diferentes sites e para clicar em seus anúncios sem o conhecimento dos proprietários desses computadores, gerando assim, muitos cliques a partir de endereços IP diferentes. Um exemplo popular de uma *botnet* pertencente a esta classe é a *botnet* Chameleon, que acumulou uma receita mensal de mais de US$ 6 milhões para os proprietários de *botnets* após uma infecção de mais de 120.000 máquinas Windows [66]. Outros exemplos de *botnets* que realizam este tipo de ataque são as *botnets* Clickbot, TDL-4, Fiesta e 7c.

2.5 Técnicas de Detecção de *Botnets*

Dado o poder potencial das *botnets* para conduzir diferentes atividades maliciosas, as técnicas de detecção desempenham um papel importante neste processo. Nos últimos anos, diferentes arquiteturas e técnicas têm sido propostas por pesquisadores com o objetivo de detectar e rastrear *botnets*, entretanto, a detecção de *botnets* através da análise do tráfego de rede possui os seguintes desafios: (*i*) como as *botnets* utilizam protocolos existentes, o tráfego malicioso se assemelha ao tráfego normal, (*ii*) o volume de tráfego de *botnets* geralmente é baixo, (*iii*) o número de máquinas infectadas pode ser muito pequeno, e (*iv*) o tráfego de rede pode conter comunicação criptografada [68].

As técnicas de detecção de *botnets* podem ser classificadas em dois tipos de abordagens: a abordagem baseada na configuração de *honeypots* e a abordagem baseada em Sistemas

de Detecção de Intrusão (IDSs) [69]. Todas as técnicas de detecção de *botnets* serão descritas e resumidas nesta seção.

2.5.1 Análise Baseada em *Honeypot*

Honeypots são sistemas computacionais configurados com vulnerabilidades intencionais e possuem o objetivo de atrair um atacante para que o mesmo invada estes sistemas e a partir daí tenha todos os seus passos monitorados [31]. Esta técnica é muito eficaz para coletar informações sobre *botnets*. Após a coleta de informações, é possível conhecer e entender a tecnologia e as técnicas utilizadas pelo atacante e assim, realizar uma análise completa das principais características daquela *botnet*[10]. *Honeypots* também podem ser utilizados para obter os binários de *bots*, para descobrir informações sobre o canal de C&C, para entender as motivações do atacante e também para identificar brechas de segurança desconhecidas que permitiram que os *bots* entrassem na rede [69].

No entanto, existem algumas desvantagens para os sistemas baseados em *honeypots*, incluindo escalabilidade limitada e a necessidade de interação com atividades maliciosas, o que pode possibilitar aos invasores assumir o controle do sistema infectado e comprometer máquinas ou sistemas fora do *honeypot* [21]. Além disso, os *bots* podem evitar a detecção do *honeypot* lançando ataques que reconhecem esse tipo de sistema. Portanto, os *honeypots* por si só não são necessariamente capazes de detectar *botnets* [31].

2.5.2 Sistemas de Detecção de Intrusão (IDS)

Um IDS é um aplicativo de software ou hardware utilizado para monitorar os serviços de um sistema, buscando identificar atividades maliciosas ou violações de políticas e relatar essas violações ao site de gerenciamento da rede [21]. O IDS para detecção de *botnet* pode ser classificado em duas técnicas: IDS baseado em assinaturas e IDS baseado em anomalias [28] [69] [70].

2.5.2.1 Sistema de Detecção de Intrusão Baseado em Assinatura

Esta abordagem é baseada no reconhecimento de padrões característicos do tráfego malicioso, também conhecidos como “assinaturas” [71]. A detecção baseada em assinatura realiza a análise do tráfego em nível de pacote utilizando inspeção profunda de pacotes para reconhecer assinaturas de *payloads* relacionados a *botnets*. Esta abordagem cobre todas as três fases do ciclo de vida de uma *botnet* e é capaz de detectar *botnets* conhecidas com alta precisão [26].

Entretanto, a principal desvantagem das abordagens baseadas em assinaturas é que elas são capazes de detectar apenas *botnets* conhecidas [26] [31]. Isso significa que ataques

de *bots* do tipo *zero-day*, bem como *bots* com assinaturas ligeiramente diferentes de *bots* conhecidos, podem não ser detectados [69]. Além disso, o uso eficiente dessa abordagem exige a atualização da base de conhecimento de assinaturas constantemente, o que aumenta o custo da detecção [31]. Por fim, esta abordagem é passível de várias técnicas de evasão que podem alterar a assinatura do tráfego de *botnet*, como técnicas de ofuscação de código, por exemplo [26].

2.5.2.2 Sistema de Detecção de Intrusão Baseado em Anomalias

A detecção de intrusão baseada em anomalias é um campo de pesquisa importante na detecção de *botnets* [23]. Esta abordagem é baseada no comportamento do *host* ou nas anormalidades do tráfego de rede, como a alta latência, alto volume de tráfego repentino, tráfego em portas incomuns e comportamentos incomuns dos sistemas. Sendo assim, o objetivo é detectar desvios no comportamento benigno ou semelhanças com o comportamento de *botnets* [23]. Em contraste com as abordagens baseadas em assinatura, a detecção de anomalias é geralmente capaz de detectar novas formas de atividades maliciosas, além de ser mais resistente às técnicas existentes de resiliência de *botnets* [26]. As abordagens de detecção baseadas em anomalias são divididas em categorias baseadas em *host* e categorias baseadas em rede [21].

Detecção de anomalias baseada em *host*

Na abordagem baseada em *host*, o monitoramento e o processo de análise são feitos em cada computador individualmente, com o intuito de detectar qualquer atividade maliciosa, através do monitoramento de processos do sistema, acesso às rotinas em nível de *kernel*, chamadas de sistema, sobrecarga de processamento e acesso a arquivos suspeitos [10].

Apesar de ser uma abordagem importante para minimizar a disseminação de *malwares*, a análise baseada em *hosts* não é escalável e ainda, é limitada apenas a *bots* dentro dos *hosts* monitorados. Além disso, para cobrir uma área mais ampla da rede, cada *host* individualmente deve ser equipado com ferramentas de monitoramento sofisticadas, o que torna o monitoramento uma tarefa complexa e cara [10] [28].

Detecção de anomalias baseada em rede

Essas técnicas de detecção monitoram o tráfego da rede para identificar anomalias no tráfego, as quais podem indicar a existência de instâncias maliciosas na rede. Podem ser classificadas em metodologias de monitoramento ativo e monitoramento passivo [10]. No monitoramento ativo, pacotes criados para teste são injetados na rede monitorada com o objetivo de estimular a rede a responder. As respostas são então capturadas e analisadas buscando qualquer evidência relacionada a atividades maliciosas [28]. Uma desvantagem dessa abordagem é que ela gera tráfego de rede adicional e como interfere diretamente na rede, pode ser detectada pelo *botmaster* [10].

Por outro lado, no monitoramento passivo, a ideia é observar os dados do tráfego de rede e procurar por comunicações suspeitas que possam estar relacionadas a *bots* ou a servidores de C&C, considerando todas as fases do ciclo de vida das *botnets* [26]. Essa abordagem não interfere diretamente na operação da *botnet*, uma vez que realiza a identificação de tráfego malicioso com base apenas na observação de anomalias no tráfego, o que a torna indetectável pelo atacante. A maioria das abordagens utilizadas para detecção de *botnet* são passivas [26]. Sendo assim, a abordagem proposta nesta dissertação se enquadra no monitoramento passivo do tráfego de rede.

O tráfego de rede pode ser analisado em nível de pacote ou em nível de fluxo. Apesar da análise em nível de pacote fornecer mais informações sobre os vetores de ataque, uma vez que inspeciona a *payload* do pacote, essa abordagem é limitada quando o tráfego de rede está criptografado e além disso, consome muito recurso computacional. Por outro lado, a análise de anomalias baseada em fluxos de rede possui a vantagem de não necessitar acessar o conteúdo dos pacotes, o que a torna efetiva em redes com tráfego criptografado [5].

A detecção baseada em anomalias pode ser realizada utilizando diferentes algoritmos, que variam entre abordagens estatísticas, técnicas de AM, análise de grafo, etc. O AM têm sido um método bastante promissor para detecção de padrões de tráfego relacionados a *botnets*. O pressuposto básico por trás dessa abordagem é que os *bots* produzem padrões de tráfego distintos durante a comunicação com o *botmaster* e que esses padrões podem ser detectados empregando algoritmos de AM [11]. Além disso, fornece a capacidade de reconhecer os padrões de tráfego malicioso sem um conhecimento prévio sobre suas características [26]. As próximas seções apresentam uma descrição dos algoritmos de AM utilizados nesta dissertação.

2.6 Algoritmo *Energy-based Flow Classifier*

O *Energy-Based Flow Classifier (EFC)* é um novo método de classificação desenvolvido no contexto de sistemas de detecção de intrusão de rede, sendo inspirado em um modelo teórico da física estatística, o modelo de Potts, o qual faz uma descrição matemática de um conjunto de spins interagindo em uma malha cristalina [20]. Utilizando essa teoria, o classificador de fluxos EFC infere um modelo estatístico representativo da classe benigna na etapa de treinamento do modelo e utiliza as características aprendidas para realizar, na etapa de teste, a classificação dos fluxos em benignos ou maliciosos [20]. Sendo assim, a ideia principal do EFC é obter um modelo estatístico a partir de amostras de fluxos benignos e inferir valores de acoplamentos e campos locais que caracterizam esse tipo de tráfego para, posteriormente, realizar uma classificação baseada em energias [20].

O acoplamento mede a probabilidade das instâncias de todos os possíveis pares de atributos ocorrerem no conjunto de amostras benignas utilizado para o treinamento do modelo, enquanto o campo local indica a probabilidade das instâncias de cada um dos atributos do fluxo ocorrerem no conjunto de treinamento [20]. A energia de um dado fluxo pode ser calculada conforme a equação 2.1, sendo representada pela soma negativa dos acoplamentos e dos campos locais associados aos seus atributos, tendo como base os parâmetros do modelo estatístico inferido:

$$H(a_{k1}...a_{kN}) = -\sum_{i,j|i<j} e_{ij}(a_{ki}, a_{kj}) - \sum_{i} h_i(a_{ki}) \tag{2.1}$$

onde $e_{ij}(a_{ki}, a_{kj})$ é o conjunto de todos os possíveis valores de acoplamentos entre dois atributos i e j, e $h_i(a_{ki})$ é o conjunto de todos os possíveis campos locais associados a um atributo i [20]. Dessa forma, se um fluxo possuir atributos comuns no conjunto de fluxos benignos que foi utilizado para inferir o modelo, esse fluxo terá baixa energia [20]. Assim, é possível classificar amostras de fluxos como benignas ou maliciosas com base em um dado limiar de energia. A classificação é realizada partindo do princípio de que as amostras com energia abaixo do limiar são benignas e as amostras com energia maior ou igual ao limiar são maliciosas [20]. Na Seção 4.1 serão apresentados maiores detalhes sobre o algoritmo EFC. Além disso, detalhes teóricos sobre a inferência do modelo estatístico podem ser consultados em Pontes et al. [20].

2.7 Algoritmos Utilizados Para Comparação dos Resultados

Esta seção apresenta uma visão geral, de alto nível, dos algoritmos tradicionais de aprendizado de máquina, binários e *One Class*, os quais foram utilizados no presente estudo visando comparar os resultados obtidos com o desempenho do EFC. Nas fontes citadas encontra-se uma explicação mais detalhada desses algoritmos.

2.7.1 Classificadores Binários

Na classificação binária, o conjunto de dados rotulado pode ser dividido em duas classes diferentes de acordo com suas características. Os algoritmos tradicionais de aprendizado de máquina mais utilizados nos trabalhos relacionados foram utilizados nos experimentos e são descritos a seguir.

2.7.1.1 *Naive Bayes* (NB)

O *Naive Bayes (NB)* é um classificador estatístico baseado no Teorema de *Bayes*. Este classificador assume que o efeito do valor de uma variável de entrada (x) em uma determinada classe (c) não depende dos valores de outras variáveis de entrada, i.e., são independentes. Esta suposição é conhecida como independência condicional da classe [34]. Apesar da suposição "inocente", os classificadores NB são considerados um dos mais eficientes e efetivos algoritmos da área de AM [72]. O que diferencia os tipos de algoritmos NB é o modo como a função de probabilidade condicional (verossimilhança) é estimada, i.e., qual a distribuição utilizada. Normalmente, as distribuições mais utilizadas são a Multinomial, a Gaussiana e a Bernoulli [73].

Neste trabalho, foi utilizada a implementação *Gaussian NB* do *scikit-learn*[1], versão 1.1.1. No *Gaussian NB*, assume-se que a verossimilhança dos atributos seja Gaussiana:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i\mu_y)^2}{2\sigma_y^2}\right) \tag{2.2}$$

onde $P(x_i|y)$ é a probabilidade de se observar o atributo x_i, dada a classe y. O desvio padrão σ_y e a média μ_y são estimados utilizando máxima verossimilhança [74].

2.7.1.2 *K-Nearest Neighbors* (KNN)

O *k-Nearest Neighbors (KNN)* é um algoritmo que pode ser utilizado para tarefas de classificação, regressão e agrupamento [74]. No presente estudo, utilizaremos o KNN para classificar os fluxos de rede em duas categorias: tráfego de botnet e tráfego benigno. Nesse contexto, o KNN considera cada amostra do conjunto de treinamento como um ponto no espaço n-dimensional $\Re^n$, onde n é igual ao número de atributos. O objetivo do algoritmo é encontrar os k pontos no conjunto de treinamento que estão mais próximos do ponto que estamos tentando classificar [75]. Geralmente, o método da votação é o mais utilizado na tarefa de classificação, i.e., o rótulo da categoria que mais aparece nas k amostras é selecionado como resultado da previsão. Assim, durante a fase de treinamento, o classificador aprende os pontos a serem usados durante a tarefa de classificação. Esses pontos são a representação de cada amostra do conjunto de treinamento no espaço n-dimensional [75].

Ao utilizar o algoritmo KNN para uma tarefa de classificação, três parâmetros principais devem ser observados: o valor de k, que corresponde ao número de vizinhos a considerar, o peso de cada voto e a métrica utilizada para calcular a distância entre dois pontos [75]. Diferentes métricas podem ser aplicadas para calcular essa distância (e.g.,

[1] https://scikit-learn.org/stable/modules/naive$_b$*ayes.html*$gaussian - naive - bayes$

distância euclidiana, distância de Manhattan, entre outros [76]), sendo que a distância euclidiana é a métrica de distância mais usada em KNN, embora outras métricas possam ser mais adequadas para conjuntos de dados específicos [77].

Neste trabalho, foi utilizada a implementação do KNN do *scikit-learn* [74] versão 1.1.1. Foram utilizados os parâmetros padrão dessa implementação, i.e., número de vizinhos k = 5, função peso = uniforme (i.e., todos os pontos em cada vizinhança possuem o mesmo peso), *algorithm* = auto (i.e., infere automaticamente com base na entrada o melhor algoritmo para computar os vizinhos mais próximos de um nodo - *BallTree, KDTree* ou *brute force*), tamanho das folhas = 30 e métrica *Minkowski* para computar as distâncias, com parâmetro p = 2, que equivale à métrica euclidiana.

2.7.1.3 *Decision Tree* (DT)

As *Decision Tree (DT)* são algoritmos de aprendizado supervisionado que podem ser utilizados tanto em tarefas de classificação quanto de regressão [74]. Este algoritmo utiliza a estratégia de dividir para conquistar, na qual um problema complexo é decomposto em subproblemas mais simples esperando que a solução final, obtida por meio dessa estratégia, se assemelhe com a solução esperada para o problema em questão [78]. Para definir os testes e divisões, as DT trabalham com uma abordagem denominada *splitting* onde, de maneira recursiva, os dados são divididos em regiões menores de acordo com as classes, buscando também minimizar a entropia, ou seja, a aleatoriedade das decisões tomadas [78].

Uma DT possui três tipos de nós [79]:

- Nó raiz: O nó inicial da árvore, por onde começa o processo de divisão;
- Nós de decisão: Os nós que contêm cada uma das decisões feitas pelo algoritmo. Definem através de comparações qual o caminho a ser percorrido pela árvore;
- Nós folha: Os nós presentes no final da árvore. Representam o resultado final de uma combinação de decisões ou eventos (e.g., classificação).

As DT criam modelos que permitem classificar uma determinada amostra criando um conjunto de regras de decisão que são extraídas dos atributos das amostras dos conjuntos de treinamento. Cada nó na árvore pode ser visto como um nó de decisão *if-then-else*. O teste condicional utilizado é feito sobre o conjunto possível de atributos e sua respectiva faixa de valores. Assim, para classificar uma determinada amostra, começa-se a testar o valor do atributo do nó raiz e percorre-se a árvore seguindo as ramificações correspondentes ao valor desse atributo. Este processo é então repetido para a sub-árvore que inicia no novo nó, até finalmente atingir um nó folha. O resultado da classificação é dado pela

classificação das amostras do conjunto de treinamento que pertencem ao mesmo nó folha [75].

Neste trabalho, utilizamos a implementação do DT do *scikit-learn* [57] versão 1.1.1 com configuração padrão. O critério utilizado para a construção da árvore é a função *gini impurity* e a estratégia para bifurcação de nodos é a melhor (não aleatória). Os demais parâmetros da configuração padrão podem ser consultados na documentação do *scikit-learn*, disponível no sítio https://scikit-learn.org.

2.7.1.4 *Multi-layer Perceptron* (MLP)

O algoritmo *Multilayer Perceptron (MLP)* é um tipo de rede neural artificial, a qual pode ser definida como uma estrutura composta por elementos de processamento simples, adaptativos, densamente interconectados (chamados neurônios artificiais ou nós) e que são capazes de realizar cálculos massivamente paralelos [80]. Os neurônios artificiais são unidades de processamento de informação fundamentais para a operação de uma rede neural e podem ser descritos por um conjunto de X_n entradas, as quais são multiplicadas por um determinado peso W_n, (i.e., pesos sinápticos) e, em seguida, os resultados são somados e comparados a um limiar ou função de ativação [81].

O modelo MLP típico inclui três camadas que são a camada de entrada, a camada oculta e a camada de saída, sendo que cada camada é composta pelos neurônios artificiais [82]. Os dados são alimentados na camada de entrada, onde essas informações serão pesadas e processadas, de forma a determinar o resultado que será repassado para as próxima camadas. Pode haver uma ou mais camadas ocultas e não lineares, as quais fornecem diferentes níveis de abstração. As previsões são feitas na camada de saída, que também é chamada de camada visível [83]. O aprendizado de uma rede neural é realizado por meio de processos iterativos de ajustes aplicados aos pesos sinápticos e o aprendizado só ocorre quando a rede neural atinge uma solução generalizada para um determinado problema [81].

Neste trabalho, foi utilizado o classificador *Multi-Layer Perceptron* (MLP) do *scikit-learn* [74] versão 1.1.1 com configuração padrão. O classificador MLP padrão possui apenas uma camada oculta com 100 neurônios, a função de ativação dos neurônios é a *rectified linear unit function* (f (x) = max(0, x)), a função de otimização dos pesos é a 'adam' (i.e., otimizador baseado em gradiente estocástico), parâmetro *alpha* = 0.0001 e taxa de aprendizagem constante = 0.001. Maiores detalhes podem ser obtidos na documentação do *scikit-learn*, disponível no sítio: https://scikit-learn.org.

2.7.1.5 *Support Vector Machine* (SVM)

Support Vector Machine (SVM) são um conjunto de métodos de aprendizado supervisionado utilizados para classificação, regressão e detecção de *outliers* [74]. A ideia básica do algoritmo SVM consiste em encontrar, por meio de um processo de otimização, um hiperplano de solução única tal que a margem entre esse hiperplano e os pontos de duas classes, se linearmente separáveis, seja máximo, resultando em um hiperplano com garantias de boa generalização [84].

Porém, como uma vasta gama de problemas não apresenta uma separação linear entre as classes, i.e., não podem ser separadas por um hiperplano, o SVM utiliza funções kernel para fazer o mapeamento dos dados do espaço de entrada para um espaço de alta dimensionalidade, no qual as classes passam a ser linearmente separáveis [85]. Essa estratégia é conhecida na literatura como *kernel trick*. Alguns exemplos de kernel são Linear, Polinomial e *Radial Basis Function (RBF)*.

Neste trabalho, foi utilizada a implementação do *Support Vector Classifier (SVC)* do *scikit-learn* [74] versão 1.1.1 com a configuração padrão. Os parâmetros padrão desse algoritmo são: parâmetro de regularização C=1.0, kernel=RBF, parâmetro *gamma = scale*, tolerância=1e-3 (critério de parada), *shrinking heuristic=True* e função de decisão=*one-vs-rest*. Maiores detalhes podem ser obtidos na documentação do *scikit-learn*, disponível no sítio: https://scikit-learn.org.

2.7.1.6 *Random Forest* (RF)

O algoritmo *Random Forest (RF)* é um método *ensemble*, que utiliza um conjunto de Árvores de Decisão para realizar a predição. O RF funciona com base em um número n, que determina quantas árvores serão criadas (estimadores), e para cada árvore n é selecionado, de maneira aleatória, um subconjunto de p atributos que esta árvore utilizará, considerando um número t total de atributos [86]. Assim, apenas um subconjunto aleatório de atributos estará disponível em cada nó, evitando que as árvores se tornem correlacionadas. Este procedimento é muito eficiente, pois uma vez que as árvores são criadas com um número reduzido de atributos, se existir um atributo muito mais forte do que os outros, ele não influenciará tanto no resultado, já que as árvores geradas serão bem diferentes umas das outras, apresentando um valor de variância mais alto entre elas [87].

O resultado da previsão se dá por meio de um sistema de votação. Assim, os dados a serem avaliados são passados por todas as árvores da floresta e cada uma irá predizer uma classe. Uma vez que o RF pode ser utilizado para tarefas de classificação e regressão, a previsão no caso da classificação é baseada na maioria dos votos dos valores previstos

pelas árvores da floresta e, no caso da regressão, o resultado é a média dos resultados das referidas árvores [88].

Neste trabalho, foi utilizada a implementação do RF do *scikit-learn* [74] versão 1.1.1 com configuração padrão. Assim, o número de árvores utilizadas para a construção da floresta foi 100 e foi utilizado *bootstrap* de atributos para construção das árvores. Os demais parâmetros da configuração padrão podem ser consultados na documentação do *scikit-learn*, disponível no sítio https://scikit-learn.org.

2.7.1.7 *AdaBoost* (AD)

O *AdaBoost (AD)*, abreviação de *Adaptive Boosting*, é um algoritmo iterativo, onde a ideia principal é treinar diferentes classificadores (classificadores fracos) a partir de um conjunto de treinamento e, em seguida, integrar esses classificadores fracos para construir um classificador mais forte [89]. A abordagem iterativa visa atribuir pesos aos classificadores de acordo com seu desempenho. Assim, as amostras classificadas incorretamente recebem maior peso, para que na próxima iteração essas amostras possam ser corrigidas. Isso permite que o próximo classificador se concentre na classificação dos casos mais difíceis ou raros, diminuindo o erro de treinamento [90]. Ao final do processo, um modelo robusto é gerado com base nos passos anteriores dos classificadores fracos [89].

Neste trabalho, foi utilizada a implementação do *AdaBoost* do *scikit-learn* [74] versão 1.1.1 com a configuração padrão. Na configuração padrão, as Árvores de Decisão são utilizadas como estimador base, o número de estimadores = 50 e a taxa de aprendizagem é = 1. Além disso, o algoritmo de *boosting* utilizado é o SAMME.R, uma nova versão do SAMME [91] com convergência mais rápida, alcançando um erro de teste menor, com menos iterações de reforço.

2.7.2 Classificadores *One Class*

A classificação *One Class*, também conhecida como classificação unária, tenta distinguir as amostras alvo de todas as outras amostras possíveis, aprendendo a partir de um conjunto de treinamento que contém apenas as amostras da classe alvo [92]. Dependendo do campo de aplicação e da área de pesquisa, os problemas *One Class* também são chamados de detecção de valores discrepantes, detecção de novidades, detecção de anomalias e detecção de extrapolação [93]. Essencialmente, todos eles visam descrever a classe alvo e detectar as amostras que não pertencem a ela [92]. Uma das principais vantagens dos classificadores *One Class* é que estes não requerem que os dados sejam rotulados, uma vez que são treinados apenas com a classe alvo [94].

Uma vez que o EFC é um algoritmo que realiza a classificação baseada no aprendizado a partir de uma única classe, também realizamos experimentos com os algoritmos *One Class* disponíveis no *scikit-learn*, para comparar os resultados obtidos com o desempenho do EFC. A seguir apresentamos uma descrição dos classificadores *One Class* utilizados nesta pesquisa.

2.7.2.1 *One Class* SVM (OCSVM)

O algoritmo *One Class Support Vector Machine (OCSVM)* é uma aplicação especial do algoritmo SVM para problemas de uma classe. O OCSVM funciona construindo um limite de decisão, de modo que quaisquer dados que estejam fora do limite sejam considerados discrepantes. O OCSVM representa as amostras como pontos em um espaço de n dimensões, de forma análoga ao SVM. Durante o treinamento, o OCSVM tenta encontrar a menor hiperesfera na qual esteja incluído o máximo de elementos da amostra e o mínimo de espaço vazio. Então, após o treinamento, para checar se uma instância é pertencente à classe treinada, o OCSVM verifica se o ponto que representa a amostra está dentro na hiperesfera [95]. Este algoritmo é bastante utilizado para detecção de anomalias [94].

Neste trabalho, foi utilizada a implementação do OCSVM do *scikit-learn* [74] versão 1.1.1 com a configuração padrão. Os parâmetros padrão desse algoritmo são: kernel=RBF, parâmetro *gamma=scale*, tolerância=1e-3 (critério de parada), *shrinking heuristic=True*, max_iter=-1 e nu=0.5. Maiores detalhes podem ser obtidos na documentação do *scikit-learn*, disponível no sítio: https://scikit-learn.org

2.7.2.2 *Isolation Forest* (iForest)

O algoritmo *Isolation Forest (IForest)* utiliza um conjunto de árvores aleatórias e independentes chamadas de floresta aleatória. O termo isolamento neste contexto significa separar uma instância das demais instâncias e, uma vez que os valores discrepantes são poucos e diferentes, espera-se que eles sejam mais propensos ao isolamento [96]. Dessa forma, o Iforest funciona particionando recursivamente os dados até que todas as instâncias sejam isoladas. O particionamento aleatório produz caminhos notavelmente mais curtos para anomalias. Portanto, quando uma floresta de árvores aleatórias produz coletivamente comprimentos de caminho mais curtos para amostras específicas, é altamente provável que essas amostras sejam anomalias [74].

As partições são criadas selecionando aleatoriamente um atributo e, em seguida, selecionando um valor de divisão entre o valor mínimo e máximo desse atributo selecionado. Considerando que uma estrutura em árvore pode representar particionamento recursivo, o número de partições necessárias para isolar uma instância é o comprimento do cami-

nho do nó raiz até o nó final [74]. Geralmente mais partições são necessárias para isolar instâncias normais, enquanto o oposto é verdadeiro para anomalias [96].

Neste trabalho, foi utilizada a implementação do *Isolation Forest (IForest)* do *scikit-learn* [74] versão 1.1.1 com a configuração padrão. Os parâmetros padrão desse algoritmo são: número de árvores utilizadas para a construção da floresta=100, max_samples=min(256, n_samples), *contamination*=auto, *max_features*=1.0. Maiores detalhes podem ser obtidos na documentação do *scikit-learn*, disponível no sítio: https://scikit-learn.org

2.7.2.3 *Local Outlier Factor* (LOF)

O algoritmo *Local Outlier Factor (LOF)* é um método clássico de detecção de valores discrepantes de alta precisão baseado em densidade. Cada ponto de dados possui um fator de *outlier* (lof), que depende da densidade dos k vizinhos mais próximos, de acordo com a qual pode-se determinar se este ponto é um *outlier* [97]. Esse fator de *outlier* está relacionado à densidade localmente alcançável dos pontos de dados. Dessa forma, quanto menor a densidade localmente alcançável de um ponto, maior será a probabilidade de que este ponto seja um valor discrepante. Por outro lado, quanto maior a densidade localmente alcançável, menor será a probabilidade de que este ponto seja um *outlier* [74]. Em outras palavras, o fator de *outlier* é determinado por quão isolado o ponto de dados de teste está, em relação aos seus vizinhos [98].

Neste trabalho, foi utilizada a implementação do *Local Outlier Factor (LOF)* do *scikit-learn* [74] versão 1.1.1 com a configuração padrão. Os parâmetros padrão desse algoritmo são: número de vizinhos=20, *algorithm*=auto (i.e., decide o algoritmo mais apropriado com base nos valores passados pelo método *fit*), *leaf_size*=30, *metric=minkowski*, p=2 (i.e., utiliza distância euclidiana), *contamination*=auto. Maiores detalhes podem ser obtidos na documentação do *scikit-learn*, disponível no sítio: https://scikit-learn.org

2.7.2.4 *Elliptic Envelop* (Elenv)

O algoritmo *Eliptic Envelop (Elenv)* parte do princípio de que os dados regulares vêm de uma distribuição bem conhecida, geralmente gaussiana. A partir desta consideração, o Elenv ajusta uma estimativa de covariância robusta aos dados delineando uma elipse aos pontos de dados centrais, para que a maioria das instâncias de dados caibam nela. Dessa forma, as instâncias de dados que ficarem fora dessa elipse serão consideradas anomalias ou *outliers*.

Neste trabalho, foi utilizada a implementação do *Eliptic Envelop (Elenv)* do *scikit-learn* [74] versão 1.1.1 com a configuração padrão. Os parâmetros padrão desse algoritmo são: *store_precision=True* (i.e., armazena a precisão estimada), *assume_centered=False* (i.e., a localização robusta e a covariância são calculadas diretamente com o algoritmo

FastMCD, sem tratamento adicional), *support_fraction=None* (i.e., o valor mínimo para este parâmetro será definido pela fórmula: $[n\_sample + n\_features + 1]/2$, considerando um intervalo de 0 a 1), *contamination*=0.1 (i.e., a proporção de *outliers* no conjunto de dados) e *random_state=None*. Maiores detalhes podem ser obtidos na documentação do *scikit-learn*, disponível no sítio: https://scikit-learn.org

2.8 Considerações Finais

Neste capítulo, foram abordados conceitos importantes para o entendimento deste trabalho, tais como: o ciclo de vida das *botnets*, as arquiteturas comumente utilizadas, os principais tipos de ataques e as principais formas de detecção. Além disso, foi apresentada uma descrição do algoritmo EFC, o qual é a base para a proposta de solução dessa pesquisa. Por fim, todos os algoritmos binários e *One Class* que serão implementados para comparação com o EFC foram descritos.

Muitas abordagens utilizando o aprendizado de máquina para detecção de *botnets* através da análise de fluxos de rede têm sido propostas. Algumas técnicas são projetadas especificamente para determinados protocolos, enquanto outras tentam ser mais genéricas, envolvendo vários protocolos e arquiteturas [10]. No capítulo seguinte serão abordados os principais trabalhos sobre o tema, apresentando as vantagens e limitações de cada um.

Capítulo 3

Trabalhos Relacionados

Neste capítulo são apresentadas algumas abordagens que exploram o problema de detecção de *botnets* através da análise de fluxos de rede utilizando técnicas de aprendizado de máquina, além de identificar as limitações encontradas em tais abordagens às quais esta pesquisa investiga. A seguir serão abordadas as principais contribuições dos trabalhos relacionados a este tema de pesquisa e que serviram de motivação para a proposta desta dissertação.

A detecção de *botnets* tem sido um tópico de pesquisa amplamente discutido em muitos estudos relacionados à segurança de redes. Porém, apesar das inúmeras técnicas existentes na literatura, García et al. [99] aponta algumas limitações na maioria das abordagens propostas, tais como:

- A maioria dos métodos não podem ser reproduzidos pois os conjuntos de dados não são públicos;
- Algumas propostas alteram excessivamente os dados para produzir algoritmos que trabalhem melhor com uma base de dados específica;
- 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 propostos 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* através da análise de anomalias de rede. No final deste capítulo será apresentada uma tabela comparativa considerando alguns dos problemas citados acima, como informações sobre o conjunto de dados utilizado e a adaptabilidade do modelo proposto.

Em 2008, Gu et al. [24] propuseram o BotMiner como uma nova abordagem baseada em análise de tráfego no nível do *host*. A abordagem proposta é dedicada à detecção das atividades de grupos de *botnets*, assumindo que os *bots* dentro da mesma *botnet* realizam atividades maliciosas semelhantes e possuem padrão de comunicação com o canal de C&C semelhante. A ideia principal dessa abordagem é agrupar comunicações semelhantes e tráfego malicioso em *clusters* e, em seguida, executar a correlação entre estes *clusters* para identificar *hosts* que têm padrões de atividades maliciosas. Esta é uma das primeiras abordagens que realiza detecção independente da estrutura e do protocolo de C&C. A abordagem proposta foi avaliada usando dados de rede reais e os resultados mostram que o BotMiner possui uma alta taxa de *True Positive Rate (TPR)* e e gera um baixo número de *False Positive Rate (FPR)*. Uma limitação importante do BotMiner é que ele visa essencialmente grupos de máquinas comprometidas em uma rede monitorada. Sendo assim, se houver somente um *host* comprometido na rede monitorada, o BotMiner pode não ser eficaz na detecção.

Em 2011, Saad et al. [16] exploraram uma técnica para detectar *botnets* por meio de análise de comportamento de rede utilizando aprendizado de máquina, na qual eles tentaram detectar *botnets* na fase de C&C. O *framework* proposto possui duas fases. A primeira fase implementa o pré-processamento do tráfego de rede, extraindo um amplo conjunto de atributos para cada fluxo. A segunda fase implementa modelos supervisionados para classificar os fluxos em tráfego não P2P, tráfego P2P malicioso e tráfego P2P normal. Foram testados os seguintes algoritmos: SVM, ANN, KNN, NB e *Gaussian Classifier*. As seguintes métricas foram utilizadas para avaliação de desempenho: TPR e Taxa de Erro Total. Os autores concluem que todas as cinco técnicas fornecem taxa de detecção (TPR) maior que 89%, porém ANN e SVM requerem mais tempo para serem treinados e para realizar a classificação. A abordagem proposta só considerou *botnets* de arquitetura descentralizada e nenhum dos modelos testados foram capazes de se adaptar às mudanças no tráfego de rede, nem de detectar novos tipos de *botnets* [16].

Em 2012, Bilge et al. [100] propuseram um sistema de detecção de *botnet* independente de protocolo e que possui o objetivo de detectar servidores de C&C (definidos como um par de IP e porta). Os atributos utilizados para detecção foram extraídos a partir dos dados do *Netflow* e foram categorizados em três grupos: baseados no tamanho do fluxo, baseados em padrões de acesso do cliente e baseados em atributos temporais. Os seguintes modelos foram avaliados: RF, J48 *Decision Tree* e SVM. Para avaliação de desempenho, foram utilizadas as métricas Area Under the Curve (AUC), TPR e FPR. Para reduzir a taxa de falsos positivos, foram incorporados uma série de resultados de reputação externa (*blacklists*) no procedimento de detecção. A abordagem foi testada em duas redes do mundo real, com uma taxa de identificação de verdadeiro positivo de 65% e uma taxa

de falso positivo de 1%. Apesar de ser uma das poucas abordagens que consideraram a adaptabilidade a diferentes domínios, alguns detalhes não foram mencionados e há muitas suposições. Além disso, o sistema é vulnerável a várias técnicas de evasão, como por exemplo evasão baseada no tempo [26].

Em 2013, D Zhao et al. [4] propuseram um sistema para detectar *botnet* P2P nas fases de comando e controle e ataque, com base na observação das características do fluxo de rede para intervalos de tempo específicos. Para a avaliação, os autores utilizaram o classificador *Reduced Error Pruning algorithm (REPTree)* para distinguir entre tráfego malicioso e benigno. O desempenho foi avaliado utilizando as métricas TPR e FPR. O modelo conseguiu atingir uma precisão geral de mais de 90%, mantendo a taxa de falsos positivos abaixo de 5%. Porém, no experimento voltado à adaptabilidade do modelo, no qual o conjunto de teste possuía dois tipos de *botnets* que não estavam presentes no conjunto de treino, essa abordagem apresentou alta taxa de falsos positivos (82%). O sistema também é ineficiente quando ocorrem mudanças no comportamento do tráfego de rede, como redirecionar uma máquina específica para outra tarefa, por exemplo.

Em 2014, Beigi, et al. [101] apresentaram um extenso estudo sobre seleção de atributos para a detecção de *botnets*. Neste estudo, Beigi classificou os fluxos de rede usando o classificador C4.5 *Decision Tree*. A abordagem utilizada testa a adaptabilidade do modelo, uma vez que contém mais de 40% de amostras de *botnets* desconhecidas nos dados de teste. Para a seleção dos atributos foi utilizado um algoritmo guloso e os atributos extraídos foram divididos em quatro *clusters*. Foram utilizadas as métricas TPR e FPR. A pesquisa alcançou uma taxa de detecção moderada com 75% e 2,3% de falso positivo, além disso, o método de Beigi é independente de protocolo e arquitetura.

Em 2017, Chen et al. [102] propuseram um mecanismo de detecção baseado em conversação para classificar a comunicação de *botnets* analisando fluxos de rede. Para reduzir a dimensão dos atributos e selecionar os mais promissores, o algoritmo *Random Forest (RF)* foi utilizado. Para verificar a precisão do modelo proposto, o autor aplicou cinco classificadores no conjunto de dados CTU-13, sendo eles: *REPTree, RandomTree, BayesNet, Decision-Tump* e RF. Foram utilizadas as métricas acurácia e FPR para avaliar os modelos, sendo que o RF superou os demais classificadores obtendo uma precisão de 93,6%.

Em 2018, Alauthaman et al. [32] apresentaram um método de detecção de *botnets* P2P baseado em uma rede neural multicamada *feed-forward*. Na abordagem proposta, foi utilizado o algoritmo *Classification and Regression Trees (CART)* como técnica de seleção de atributos para selecionar os atributos mais relevantes, através de dados extraídos do cabeçalho do tráfego TCP. Foi feita uma comparação dessa abordagem de seleção de atributos com outras duas abordagens: *Principal Component Analysis (PCA)* e o algo-

ritmo ReliefF. A avaliação da abordagem proposta foi demonstrada através da realização de experimentos nos conjuntos de dados ISOT [16] e ISCX [103]. Foram utilizadas as seguintes métricas para comparação de desempenho: TPR, FPR, acurácia e F1-score. Os resultados indicaram que o modelo de rede neural com seleção de atributos baseados em árvore de decisão (CART) teve uma melhor precisão de identificação, obtendo uma taxa de detecção média de 99,08% com taxa de falsos positivos de 0,75%. No entanto, como a abordagem rastreia apenas o tráfego TCP para realizar a detecção, as *botnets* que não se comunicam por meio de pacotes TCP não serão detectadas. Além disso, a capacidade de adaptação a diferentes domínios não foi abordada.

Resende et al. [104] propuseram um método de detecção de *botnets* utilizando o classificador *Random Forest*. Foi proposta a utilização de uma arquitetura escalável baseada em tecnologias de processamento de Big Data, suportada por *Hadoop*, *HBase* e *Apache Spark*. Além disso, foram propostos novos atributos para distinguir os canais C&C do tráfego benigno, baseado na observação de que as conexões HTTP estabelecidas para fins de canais de C&C de *botnets* geralmente têm implementações específicas, que se desviam dos padrões usados por acessos HTTP legítimos. Foi observado também que os protocolos IRC e TCP quando usados para criar canais C&C, geram conexões longas e compostas por blocos semelhantes e regulares de transmissão de dados. Foram realizados vários experimentos utilizando somente os novos atributos propostos baseado nas observações mencionadas, utilizando somente os atributos mais comumente utilizados na literatura e ainda, combinando os novos atributos propostos com os mais utilizados na literatura. Para avaliação de desempenho foram utilizadas as métricas acurácia, TPR e FPR. Os resultados mostraram uma melhoria na precisão de detecção quando os atributos que foram propostos são combinados com os atributos mais utilizados na literatura.

Também em 2018, Gadelrab et al. [105] propuseram um modelo de detecção de *botnet* denominado BotCap baseado em técnicas de aprendizado de máquina. Duas questões principais foram discutidas nesse estudo - a inspeção profunda de pacotes do tráfego de rede e a coleta de informações de *hosts* infectados na rede. O conjunto de dados coletado incluiu dois grupos de *botnets* diferenciados por seu modo de operação. O primeiro grupo com *botnets* que se comunicam através de um canal IRC (Aryan, Ngr e Rxbot) e o segundo grupo com *botnets* realizando comunicações HTTP com o servidor C&C (Black Energy, Zeus e Vertexnet). O conjunto de dados utilizado possui amostras de tráfego benigno e malicioso. Os algoritmos J48 *Decision Tree* e SVM foram treinados para distinguir o tráfego de rede como benigno e *botnet*. Dos 55 atributos incluídos no conjunto de dados, um subconjunto de 9 foi selecionado como os mais relevantes para a análise. Em seguida, ambos os algoritmos foram validados pelo cálculo da Precisão, TPR e F1-score. Para otimizar os modelos, o algoritmo *Grid Search* foi aplicado antes de realizar uma análise

de validação cruzada (5 *k-fold*). Os resultados mostraram que a abordagem proposta é capaz de detectar *hosts* infectados individualmente em uma rede local, sem a necessidade de coletar muitas informações dos computadores infectados. Uma desvantagem desta abordagem é que só foi considerada a arquitetura centralizada, sendo portanto dependente do protocolo utilizado para C&C.

Em 2019, R. Khan et al. [106] propuseram uma abordagem de detecção de *botnet* P2P de múltiplas camadas. A estrutura é composta por quatro camadas. A primeira camada realiza a filtragem do tráfego não P2P para reduzir a sobrecarga de processamento. A segunda camada faz a identificação de tráfego P2P e não-P2P através da combinação de consultas DNS, filtragem de portas e uma abordagem de heurística rápida para detecção de tráfego P2P. Em seguida, na terceira camada, a seleção de atributos ajuda a reduzir o *overfitting*, melhorando a taxa de precisão do modelo de classificação. Na camada final, um classificador binário realiza a classificação do tráfego P2P como normal ou *botnet*. Foram utilizadas as métricas acurácia e FPR para avaliação do desempenho. A validação experimental deste estudo mostrou que o algoritmo DT atingiu um nível de precisão médio de 98,7% quando aplicado aos conjuntos de dados CTU-13 e ISOT. Este algoritmo superou os resultados dos outros modelos propostos, como KNN, ANN e *Logistic Regression*. Uma limitação dessa abordagem é ser dedicada à detecção de *botnets* P2P apenas.

Em 2020, Ramos et al. [1] realizaram um abrangente estudo comparativo utilizando os modelos supervisionados mais prevalentes na literatura sobre detecção de *botnets*. Foram selecionados os classificadores: DT, RF, NB, KNN e SVM. Para melhorar o desempenho de cada um dos modelos, foi utilizado o algoritmo *Grid Search*, sendo definida uma gama de valores possíveis para cada parâmetro dos diferentes modelos de AM. Os atributos foram extraídos utilizando a ferramenta *cicflowmeter* e foi realizada uma seleção desses atributos considerando pesquisas anteriores de outros autores. Para validação dos modelos, foram utilizados os conjuntos de dados CIC-AWS-2018 e ISOT HTTP, aplicando as métricas precisão, acurácia e TPR. Os resultados obtidos mostraram que entre os algoritmos escolhidos, os modelos RF e DT foram os mais adequados para detectar diferentes tipos de *botnets*, apresentando maior precisão e menos tempo de processamento. Uma vez que utilizaremos a ferramenta *cicflometer* para extração dos atributos e também utilizaremos o conjunto de dados ISOT HTTP, vamos utilizar os atributos selecionados no trabalho de Ramos et al. [1] como referência para comparação de desempenho com os atributos selecionados pelo EFC.

Por fim, em 2021, Ibrahim et al. [5] propuseram um *framework* multicamadas para detecção dos servidores de C&C de *botnets*, através de algoritmos de aprendizado de máquina. A abordagem consiste em dois módulos principais: o Módulo de Filtragem e o Módulo de Classificação. Ambos os módulos usam atributos baseados em fluxo e

são baseados em comportamento. O objetivo do primeiro módulo é filtrar e reduzir o tráfego de rede para o segundo módulo, utilizando para isso o algoritmo de clusterização *k-means*. O objetivo do segundo módulo é detectar o servidor de C&C, utilizando para isso algoritmos de classificação. Foram avaliados três classificadores: KNN, SVM e MLP. Para avaliar o desempenho dos classificadores foram utilizadas as métricas acurácia, F1-score, precisão, TPR e FNR. O KNN apresentou o melhor resultado com 91,51% de F1-score e uma taxa de falso negativo de 1,5%. A principal vantagem dessa abordagem é ser independente de protocolo e estrutura.

3.1 Comparação dos Trabalhos Relacionados

Nesta seção apresentamos uma comparação dos principais pontos de cada trabalho citado anteriormente. Para esta comparação, apresentamos algumas definições importantes, a seguir:

- Independente de Protocolo: analisa se a abordagem proposta para detecção de *botnets* é limitada à um determinado protocolo específico ou se a detecção é realizada independente do protocolo utilizado pelas *botnets*.
- Independente de Estrutura: analisa se a abordagem proposta é limitada a uma determinada estrutura de C&C específica ou se é capaz de realizar a detecção de *botnets* independente da estrutura de C&C ser centralizada ou descentralizada.
- Adaptabilidade: analisa se foi considerada a capacidade dos modelos em predizer novos tipos de *botnets*, diferentes daquelas presentes no conjunto de treinamento, tanto em relação aos protocolos utilizados, bem como em relação à arquitetura do C&C. Na área de segurança de redes, essa adaptabilidade é especialmente importante devido à existência de ataques *zero-day* e pelo fato das *botnets* serem muito flexíveis e estarem em constante evolução quanto aos protocolos e estruturas utilizadas.
- Domínio Cruzado: analisa se as abordagens propostas consideraram testar a adaptabilidade dos modelos em um domínio diferente daquele no qual foi realizado o treinamento dos modelos. O teste de domínio cruzado testa a capacidade dos modelos em generalizar os seus resultados mesmo com mudanças na distribuição dos dados.

A Tabela 3.1 apresenta a comparação das abordagens propostas nos trabalhos relacionados, considerando as definições apresentadas, os conjuntos de dados utilizados e os algoritmos de AM implementados. Percebe-se que algumas abordagens propostas são

limitadas à um protocolo ou estrutura específica e não consideram a adaptabilidade à mudanças de contexto. Dentre os trabalhos que consideraram a adaptabilidade do modelo, a maioria considerou o mesmo conjunto de dados para treino e teste. O que se pretende nesta dissertação é estender esses testes de adaptabilidade utilizando dois conjuntos de dados heterogêneos, sendo que um deles possui somente *botnets* de arquitetura centralizada e o outro contém *botnets* de arquiteturas centralizadas e descentralizadas e diversos protocolos (IRC, HTTP e P2P). Para tornar os testes mais próximos de uma aplicação real, será realizado o teste de domínio cruzado, utilizando um conjunto de dados para treinamento do modelo proposto e outro para teste. Ao realizar o treinamento do modelo em um conjunto de dados que possui *botnets* de arquitetura centralizada somente (ISOT HTTP) e ao realizar o teste do modelo em um conjunto de dados contendo *botnets* de diferentes arquiteturas e protocolos (CTU-13), testaremos a capacidade do modelo em detectar novos tipos de *botnets*, independente do protocolo e da arquitetura do canal de C&C.

Além disso, observa-se na Tabela 3.1 que todos os trabalhos relacionados utilizaram somente classificadores de classe binária, os quais precisam de dados rotulados da classe maliciosa, para diferenciar o tráfego malicioso do tráfego benigno. Diferentemente, propomos a utilização de um algoritmo *One Class* para a detecção de *botnets* e fizemos um estudo comparativo sobre o desempenho do EFC e o desempenho de diversos classificadores de uma e de duas classes. Por fim, propomos uma nova abordagem para seleção dos atributos mais relevantes para a detecção de *botnets*, utilizando para isto, o algoritmo EFC, o qual gera um modelo estatístico interpretável. Assim, propomos um conjunto de atributos capaz de caracterizar o tráfego de *botnets*.

3.2 Considerações Finais

Neste capítulo, apresentamos os principais trabalhos relacionados ao tema de detecção de *botnets* utilizando fluxos de redes, abordando os principais pontos de cada abordagem proposta e apresentamos, por fim, uma comparação dos trabalhos relacionados com a nossa proposta de pesquisa, em relação aos algoritmos e conjuntos de dados utilizados, a independência de protocolo e estrutura e ainda, em relação à adaptabilidade dos modelos. Muitas abordagens existentes visam a detecção de um tipo específico de *botnet* (e.g., IRC, P2P, HTTP), não sendo capazes de detectar outros tipos de *botnets*. A nossa abordagem, diferentemente, visa diferenciar o tráfego de rede normal do tráfego de *botnet*, independentemente do seu tipo. No próximo capítulo apresentaremos a nossa proposta de solução, bem como a metodologia utilizada nesta pesquisa.

Tabela 3.1: Comparação com os Trabalhos Relacionados.

Autor - Ano	**Classificadores**	**Conjunto de Dados**	**Independente de Protocolo**	**Independente de Estrutura**	**Adaptabilidade**	**Domínio Cruzado**
Gu et al. [24] 2008	Algoritmos de clusterização e X-means	Privado	√	√	-	-
Saad et al. [16] 2011	Gaussian-based, KNN, SVM, NB, NN	ISOT	-	-	-	-
Bilge et al. [100] 2012	RF, J48 DT, SVM	Privado	√	√	√	-
D Zhao et al. [4] 2013	REPTree DT	ISOT	√	√	√	-
Beigi, et al. [101] 2014	C4.5	ISOT e ISCX 2012 IDS	√	√	√	-
Chen et al. [102] 2017	RF, REPTree, RandomTree, BayesNet, DecisionTump	CTU-13	√	√	-	-
Alauthaman et al. [32] 2018	Neural Network (NN)	ISOT e ISCX	-	-	-	-
Resende et al. [104] 2018	Random Forest (RF)	CTU-13 e ISCX	√	√	-	-
Gadelrab et al. [105] 2018	J48 DT, SVM	Privado e CTU-10	-	-	√	-
Khan et al. [106] 2019	KNN, DT, NB, ANN, Regressão Logística	CTU-13 e ISOT	-	-	-	-
Ramos et al. [1] 2020	DT, RF, NB, KNN e SVM	ISOT HTTP e CIC-AWS-2018	√	√	-	-
Ibrahim et al. [5] 2021	KNN, SVM, Multilayer Perceptron	CTU-13	√	√	-	-
Este trabalho	EFC, OCSVM, LOF, iForest, Elenv, NB, DT, RF, KNN, SVM, AD e MLP	ISOT HTTP CTU-13 e ISCX-Bot-2014	√	√	√	√

Capítulo 4

Solução Proposta

Neste capítulo, são apresentados os principais conceitos para entender como funciona o algoritmo EFC. Os três conjuntos de dados utilizados para validação do modelo proposto são apresentados na sequência. Em seguida é apresentada a metodologia proposta. Por fim, a forma de avaliação dos resultados é descrita.

4.1 *Energy-based Flow Classifier* - EFC

A abordagem para detecção de *botnets* proposta nessa dissertação é baseada na utilização do algoritmo EFC, o qual foi desenvolvido por Pontes et al. [20] com o objetivo de superar algumas limitações dos algoritmos de aprendizagem de máquina, como por exemplo a necessidade de grandes quantidades de amostras maliciosas rotuladas, que nem sempre são simples de se obter e principalmente o fato de que a maioria desses algoritmos não são facilmente generalizáveis para outros conjuntos de dados, ou seja, o desempenho é reduzido quando treinados em um conjunto de dados específico e avaliados em outro conjunto de dados [20].

O EFC é um classificador que utiliza técnicas de estatística inversa para durante a etapa de treinamento do modelo inferir uma distribuição de probabilidade para a classe de fluxos a ser detectada, baseado apenas em amostras de fluxos benignas. Na etapa de teste do modelo, a distribuição definida na etapa anterior é usada para classificar novos fluxos, calculando e comparando uma medida denominada energia do fluxo, a qual mede o quão improvável é a ocorrência de um fluxo na distribuição calculada [20].

Um fluxo de rede pode ser definido como uma sequência unidirecional de pacotes que compartilham um conjunto de atributos-chave, sendo que os mais utilizados são: *IP de Origem, IP de Destino, Porta de Origem, Porta de Destino e Protocolo* [100]. Um fluxo benigno é aquele que corresponde ao tráfego de rede legítimo gerado por um usuário ou aplicação ao executar qualquer tarefa não maliciosa [107]. É importante ressaltar que

o EFC é um classificador de anomalias, sendo assim, fluxos maliciosos que não sejam necessariamente relacionados à ataques de *botnets* poderão ser detectados, desde que esses fluxos se diferenciem do tráfego normal utilizado pra inferir o modelo, o que ajuda a melhorar a segurança da rede. Dessa forma, um fluxo classificado como malicioso deverá ser posteriormente analisado para identificar se está relacionado à tráfego de *botnets* ou à qualquer outra atividade maliciosa.

A inferência estatística do EFC é baseada no modelo de *Potts*, o qual faz uma descrição matemática de um conjunto de spins interagindo em uma malha cristalina [20]. Sendo assim, o EFC reutiliza o modelo de *Potts* para caracterizar os fluxos de rede, de forma que um fluxo individual k é representado por uma configuração específica do grafo $G_k(\eta, \varepsilon)$, conforme a figura 4.1. Cada nó representa um atributo $i \in \eta = \{$Porta de Origem, ..., Protocolo$\}$ e cada atributo i assume um valor a_{k_i} do conjunto $\Omega_i = \{1,...,Q\}$, que contém todos os possíveis valores para aquele atributo específico, mapeados em inteiros [20].

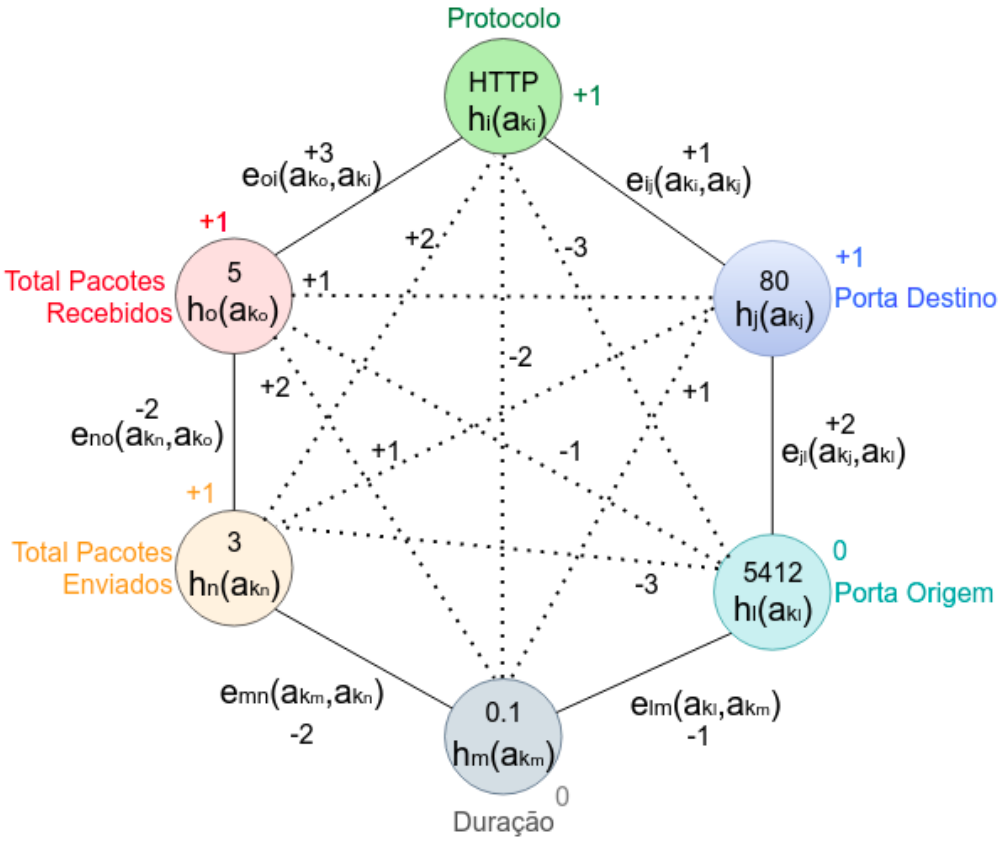


Figura 4.1: Fluxo de Rede Mapeado em um Grafo

Assim como no modelo de *Potts*, cada atributo i possui um campo local $h_i(a_{k_i})$ associado. Além disso, o conjunto de arestas (todos os possíveis pares de atributos) é representado por $\varepsilon = \{(i,j)|i,j \in \eta; i \neq j\}$, sendo que cada aresta possui um valor de

acoplamento associado, o qual é determinado pela função $e_{i_j}(a_{k_i}, a_{k_j})$. Considerando $\mathscr{K}$ o conjunto de todos os fluxos possíveis de uma determinada classe e $\mathscr{L} \subset \mathscr{K}$ o subconjunto do qual queremos inferir a distribuição, os campos locais $h_i(a_{k_i})$ são uma medida do quão provável é que o atributo i assuma o valor a_{k_i} em $\mathscr{L}$. Da mesma forma, para o mesmo fluxo, os valores de acoplamento $e_{i_j}(a_{k_i}, a_{k_j})$ são uma medida do quão provável é que os atributos i e j assumam, ao mesmo tempo, os valores a_{k_i} e a_{k_j} no conjunto $\mathscr{L}$ [20].

Os valores dos campos locais e dos acoplamentos dependem dos valores assumidos por cada atributo individualmente, sendo assim, cada fluxo possuirá uma combinação específica dessas quantidades. Da mesma forma que no modelo de *Potts*, a "energia" total $H(a_{k_1}...a_{k_N})$ de cada fluxo é definida pela soma dos valores dos campos locais e dos acoplamentos associados aos seus atributos e é calculada de forma análoga ao cálculo de energia na Mecânica Quântica, utilizando portanto, o Hamiltoniano. Dessa forma, a energia de um dado fluxo k reflete a probabilidade de que a configuração exata dos valores calculados ocorra em $\mathscr{L}$, refletindo portanto, a similaridade desse fluxo com o subconjunto original $\mathscr{L}$, atributo por atributo. Se a energia do fluxo for alta, significa que ele tem baixa probabilidade, ou seja, não se assemelha aos fluxos que geraram a distribuição. Da mesma forma, se a energia for baixa, é mais provável que esse fluxo exista na distribuição. Cabe salientar que o modelo descrito aqui é discreto, portanto, os atributos contínuos devem ser discretizados. Os detalhes teóricos da inferência do modelo são apresentados em [20].

Para que seja possível decidir se uma energia é alta ou baixa, é definido um limiar, o qual é baseado na distribuição das energias das amostras benignas utilizadas para inferir o modelo. Esse limiar pode ser definido dinâmica ou estaticamente. No nosso estudo de caso, foi utilizado um limiar estático definido pelo 95 percentil das energias das amostras de treinamento. Esse valor foi selecionado de forma empírica considerando uma margem de 5% onde os fluxos benignos e maliciosos não podem ser distinguidos entre si. Dessa forma, se um fluxo tiver energia inferior ao 95 percentil das amostras benignas, ou seja, abaixo do limiar, é considerado normal. Caso contrário, é classificado como tráfego malicioso [20].

Por fim, é importante ressaltar que a complexidade temporal da etapa de treino do EFC é $O((M \times Q)^3 + N \times M^2 \times Q^2)$, onde N é o número de amostras de fluxos, M é o número de atributos de cada fluxo e Q é o tamanho do alfabeto usado para discretização dos atributos. Por sua vez, a complexidade da etapa de classificação para cada amostra é $O(M^2)$. Dessa forma, as complexidades de treinamento e teste são mais dependentes da quantidade de atributos de cada fluxo [20].

4.2 Conjuntos de Dados

Um conjunto de dados ideal para detecção de *botnets* deve representar o tráfego real, fornecendo uma variedade de amostras entre ataques de *malware* e tráfego legítimo [14]. No entanto, há poucos conjuntos de dados disponíveis publicamente e a maioria possui algumas limitações, como tráfego obtido de ambiente simulado e criação de tráfego falso que não reflete o tráfego em tempo real. Os três conjuntos de dados utilizados nessa pesquisa serão descritos a seguir e foram selecionados por serem popularmente utilizados na literatura devido à relevância e realismo dos mesmos [102] [106] [5].

4.2.1 CTU-13

O CTU-13 é um conjunto de dados de tráfego de *botnet* que foi capturado na Universidade CTU, República Tcheca, em 2011 e armazenado em arquivos .PCAP [99]. O conjunto de dados CTU-13 contém 13 arquivos de captura do tráfego, os quais são denominados cenários e são rotulados como Normal, Ataque ou *Background*. Estes arquivos contêm diferentes tipos de *botnets*, conforme mostrado na Tabela 4.1, incluindo estruturas centralizadas (IRC e HTTP) e descentralizadas (P2P) e vários protocolos. Dessa forma, este conjunto de dados atendeu ao nosso propósito de projetar um modelo de detecção de *botnets* que fosse independente de estrutura e protocolo.

Tabela 4.1: Distribuição de *Botnets* CTU-13

Cenário	**Botnet**	**Estrutura**	**N° Bots**	**Duração (h)**
1	Neris	IRC	1	6,15
2	Neris	IRC	1	4,21
3	Rbot	IRC	1	66,85
4	Rbot	IRC	1	4,21
5	Virut	HTTP	1	11,63
6	Menti	IRC	1	2,18
7	Sogou	HTTP	1	0,38
8	Murlo	IRC	1	19,5
9	Neris	IRC	10	5,18
10	Rbot	IRC	10	4,75
11	Rbot	IRC	3	0,26
12	Nsis.ay	P2P	3	1,21
13	Virut	HTTP	1	16,36

Utilizamos nos nossos experimentos os arquivos .PCAP correspondentes aos cenários 1, 3, 5, 6, 7, 8 e 12, de forma que tívessemos um conjunto de dados contendo cada uma das famílias de *botnets* disponibilizadas. Consequentemente, testamos sete tipos de

botnets: Neris, Rbot, Virut, Menti, Sogou, Murlo e Nsis.ay, onde a combinação dessas *botnets* consistia em ambas as estruturas, centralizadas e descentralizadas. Os arquivos .PCAP disponibilizados contém apenas o tráfego malicioso, uma vez que por questões de privacidade, a captura completa contendo todos os dados de *background*, normal e de *botnet* não está disponível. Sendo assim, utilizamos parte do conjunto de dados do projeto ISCX-IDS-2012[1] para obter apenas dados de tráfego normal e complementar o conjunto de dados CTU-13 [103]. Para isso, foi utilizado o arquivo .PCAP referente à captura do dia 12/06/2010 (sábado), o qual possui 4.22 GB de tráfego normal.

4.2.2 ISOT HTTP *Botnet*

O conjunto de dados ISOT HTTP[2] foi disponibilizado pela Universidade de Victória, no Canadá e é composto por dois conjuntos de dados diferentes. O primeiro consiste em tráfego malicioso gerado por diferentes *botnets*, enquanto o segundo consiste em tráfego benigno gerado por diferentes aplicações de software, como antivírus, bate-papo online e aplicativos de mensagens instantâneas (i.e, Skype, Facebook, Messenger) [108]. A captura do tráfego dos dois ambientes (normal e *botnet*) foi realizada no período de 14 a 21 de junho de 2017.

O tráfego malicioso foi coletado a partir de um ambiente virtual, no qual foram implementados diferentes kits de *exploits* para *botnets* HTTP, totalizando 9 (nove) servidores de comando e controle (C&C), um para cada tipo de *botnet*, o que gerou 5 (cinco) arquivos .PCAP. O tráfego benigno também foi capturado de um ambiente virtual simulando tráfego de diversas aplicações instaladas em máquinas virtuais configuradas com o sistema operacional Windows 7, resultando em 3 (três) arquivos .PCAP.

Utilizamos nos nossos experimentos os 3 arquivos contendo tráfego benigno e para o tráfego malicioso utilizamos apenas o arquivo init4.pcap, uma vez que este único arquivo contém tráfego de todos os tipos de *botnets* presentes no conjunto de dados ISOT HTTP. Os seguintes tipos de *botnets* estão presentes no arquivo utilizado: zyklon, blue, liphyra, gaudox, blackout, citadel, be.botnet e zeus. Todas essas *botnets* possuem arquitetura centralizada e utilizam o protocolo HTTP.

4.2.3 ISCX-Bot-2014

O conjunto de dados ISCX-Bot-2014[3] foi disponibilizado pelo Instituto Canadense de Cibersegurança da *University of New Brunswick (UNB)*. No intuito de prover um conjunto

[1] https://www.unb.ca/cic/datasets/ids.html
[2] https://www.uvic.ca/ecs/ece/isot/datasets/botnet-ransomware/index.php
[3] https://www.unb.ca/cic/datasets/botnet.html

de dados que garantisse a generalidade, o realismo e a representatividade, o ISCX-Bot-2014 combina três outros datasets:

- Conjunto de dados ISOT [4]: integra vários projetos, particularmente o *Honeynet Project* [109], *Ericsson Research* [110] e *Lawrence Berkeley National Laboratory Research* [111], sendo que cada um contém traços maliciosos de *botnets* identificadas como Storm e Zeus. O tráfego não malicioso é gerado principalmente por pacotes de jogos, tráfego HTTP e aplicações P2P.
- Conjunto de dados ISCX 2012 IDS [103]: inclui tráfego benigno (HTTP, SMTP, SSH, IMAP, POP3 e FTP) e tráfego malicioso (*Botnet ISCX IRC*) produzido por dispositivos reais.
- *Malware Capture Facility Project* [112]: o objetivo deste conjunto de dados é gerar e capturar variado tráfego de *botnets*, incorporando oito tipos diferentes de *botnets* (Neris, Rbot, Virut, NSIS, Menti, Sogou e Murlo).

São disponibilizados dois arquivos PCAPs, sendo um denominado *Training Dataset*, o qual possui tráfego benigno e tráfego de 4 famílias de *botnets* e o outro denominado *Testing Dataset*, o qual além do tráfego benigno, também disponibiliza tráfego de 16 tipos de *botnets*. Utilizaremos nos nossos experimentos o conjunto de dados *Testing*, por possuir uma maior variedade de tráfego de *botnets*, incluindo *botnets* de diferentes arquiteturas (centralizada e descentralizada) e diversos protocolos (P2P, IRC e HTTP), conforme pode ser visto na Tabela 4.2.

4.3 Metodologia

A abordagem proposta nesta pesquisa é a detecção de *botnets* baseada na análise de fluxos de rede, através da utilização de algoritmos de aprendizado de máquina. Sendo assim, o primeiro desafio é a extração de fluxos de rede a partir dos arquivos PCAPs referentes aos conjuntos de dados descritos acima. Para isso, foi utilizada a ferramenta *CICFlowMeter*[4] que é um gerador e analisador de fluxo de tráfego de rede disponibilizado pelo Instituto Canadense de Segurança Cibernética [113]. O resultado gerado é um arquivo CSV contendo 84 atributos conforme ilustra a Tabela 4.3.

Para o *CICFlowMeter* cada fluxo é definido pelo primeiro pacote que determina as direções *forward* (origem para destino) e *backward* (destino para origem). Além disso, os fluxos TCP geralmente são encerrados na desconexão da conexão (pelo pacote FIN), enquanto os fluxos UDP são encerrados por um tempo limite do fluxo. O valor do tempo

[4] `https://www.unb.ca/cic/research/applications.html#CICFlowMeter`

Tabela 4.2: Distribuição de *Botnets* - *Testing* ISCX-Bot-2014

Botnet	**Estrutura**	**Quantidade**
Neris	IRC	25.967 (5,67%)
Rbot	IRC	83 (0,018%)
Menti	IRC	2.878 (0,62%)
Sogou	HTTP	89 (0,019%)
Murlo	IRC	4.881 (1,06%)
Virut	HTTP	58.576 (12,80%)
NSIS	P2P	757 (0,165%)
Zeus	P2P	502 (0,109%)
SMTP Spam	P2P	21.633 (4,72%)
UDP Storm	P2P	44.062 (9,63%)
Tbot	IRC	1.296 (0,283%)
Zero Access	P2P	1.011 (0,221%)
Weasel	P2P	42.313 (9,25%)
Smoke Bot	P2P	78 (0,017%)
Zeus Control (C2C)	P2P	31 (0.006%)
ISCX IRC bot	P2P	1.816 (0,387%)

limite do fluxo pode ser atribuído arbitrariamente, geralmente 600 segundos para ambos (TCP e UDP).

Após a extração dos fluxos, os arquivos resultantes de cada um dos conjuntos de dados foram rotulados utilizando a linguagem de programação *python* e a biblioteca pandas. A Tabela 4.4 ilustra a quantidade de fluxos benignos e maliciosos extraídos de cada um dos conjuntos de dados e ainda, a quantidade por família de *botnet*. Todos os experimentos foram realizados utilizando a mesma composição de dados, de acordo com a Tabela 4.4

A presente pesquisa possui dois objetivos principais. O primeiro, visa propor uma abordagem que seja adaptável à mudanças na rede, de forma que possa detectar novos tipos de *botnets*, as quais não estarão presentes no treinamento do modelo. O segundo, visa propor um conjunto de atributos que seja capaz de detectar *botnets*, mantendo a adaptabilidade dos modelos. Sendo assim, implementamos o Estudo de Caso 1, voltado para atingir o primeiro objetivo, e o Estudo de Caso 2, projetado para cumprir o segundo objetivo.

No Estudo de Caso 1, utilizamos o algoritmo EFC para a classificação dos fluxos de rede em fluxo normal e fluxo de *botnet*. Testamos o modelo utilizando dois conjuntos de dados distintos, o CTU-13 e o ISOT-HTTP. As quantidades de tráfego benigno e de tráfego malicioso de ambos conjuntos de dados podem ser consultadas na Tabela 4.4. Utilizamos todos os atributos listados na Tabela 4.3, com exceção dos atributos *Flow ID, Source IP, Destination IP e Timestamp*, uma vez que estes atributos são muito específicos de cada

Tabela 4.3: Atributos Extraídos com a Ferramenta *Cicflowmeter*

N°	Feature	N°	Feature	N°	Feature
1	Flow ID	29	Fwd IAT Std	57	ECE Flag Count
2	Source IP	30	Fwd IAT Max	58	Down/Up Ratio
3	Source Port	31	Fwd IAT Min	58	Average Packet Size
4	Destination IP	32	Bwd IAT Total	60	Avg Fwd Segment Size
5	Destination Port	33	Bwd IAT Mean	61	Avg Bwd Segment Size
6	Protocol	34	Bwd IAT Std	62	Fwd Avg Bytes/Bulk
7	Time stamp	35	Bwd IAT Max	63	Fwd Avg Packets/Bulk
8	Flow Duration	36	Bwd IAT Min	64	Fwd Avg Bulk Rate
9	Total Fwd Packets	37	Fwd PSH Flags	65	Bwd Avg Bytes/Bulk
10	Total Backward Packets	38	Bwd PSH Flags	66	Bwd Avg Packets/Bulk
11	Total Length of Fwd Pck	39	Fwd URG Flags	67	Bwd Avg Bulk Rate
12	Total Length of Bwd Pck	40	Bwd URG Flags	68	Subflow Fwd Packets
13	Fwd Packet Length Max	41	Fwd Header Length	69	Subflow Fwd Bytes
14	Fwd Packet Length Min	42	Bwd Header Length	70	Subflow Bwd Packets
15	Fwd Pck Length Mean	43	Fwd Packets/s	71	Subflow Bwd Bytes
16	Fwd Packet Length Std	44	Bwd Packets/s	72	Init_Win_Bytes_Fwd
17	Bwd Packet Length Max	45	Min Packet Length	73	Act_Data_Pkt_Fwd
18	Bwd Packet Length Min	46	Max Packet Length	74	Min_Seg_Size_Fwd
19	Bwd Packet Length Mean	47	Packet Length Mean	75	Active Mean
20	Bwd Packet Length Std	48	Packet Length Std	76	Active Std
21	Flow Bytes/s	49	Packet Len. Variance	77	Active Max
22	Flow Packets/s	50	FIN Flag Count	78	Active Min
23	Flow IAT Mean	51	SYN Flag Count	79	Idle Mean
24	Flow IAT Std	52	RST Flag Count	80	Idle Packet
25	Flow IAT Max	53	PSH Flag Count	81	Idle Std
26	Flow IAT Min	54	ACK Flag Count	82	Idle Max
27	Fwd IAT Total	55	URG Flag Count	83	Idle Min
28	Fwd IAT Mean	56	CWE Flag Count	84	Label

Tabela 4.4: Composição Média dos Conjuntos de Dados

CTU-13		**ISOT HTTP**		**ISCX-Bot-2014**	
Rótulo	**Quantidade**	**Rótulo**	**Quantidade**	**Rótulo**	**Quantidade**
Normal	215.251	Normal	76.360	Normal	153.467
Virut	83.900	Cidatel	145.087	Weasel	67.954
Rbot	46.540	Gaudox	90.970	Virut	42.255
Neris	22.247	Zeus	80.642	Neris	24.070
Murlo	11.536	Be.botnet	13.755	Murlo	12.301
Nsis	7.645	Bluebot	13.593	Menti	4.887
Menti	4.809	Zyklon	12.008	Zero Access	1.816
Sogou	72	Blackout	6.881	Tbot	860
		Liphyra	3.782	Black Hole 2	443
				Zeus	385
				IRC	363
				Black Hole 3	103
				Sogou	81
				Rbot	80
				Smoke Bot	76
				IRCbot	39
				Osx trojan 2	27
Total Malicioso	176.749	**Total Malicioso**	366.718	**Total Malicioso**	155.740
Total Benigno	215.251	**Total Benigno**	76.360	**Total Benigno**	153.467

fluxo. Neste Estudo de Caso, implementamos duas abordagens. A primeira é o teste intra-domínio, onde o treinamento e o teste dos modelos foram realizados no mesmo conjunto de dados. A segunda abordagem é o teste inter-domínio ou teste de domínio cruzado, onde o treinamento dos modelos foi realizado em um conjunto de dados e o teste foi realizado em outro conjunto de dados heterogêneo, tanto em relação aos tipos de *botnets*, quanto em relação aos protocolos e arquiteturas utilizadas, considerando portanto, a mudança de domínio. Os conjuntos de dados CTU-13 e ISOT HTTP são bastante heterogêneos. O CTU-13 é composto por *botnets* que utilizam diferentes protocolos e arquiteturas, já o ISOT HTTP é composto por famílias de *botnets* de arquitetura centralizada e que utilizam somente o protocolo HTTP. O objetivo de trabalhar com dois conjuntos de dados bastante diferentes é testar a capacidade dos modelos de identificar novos tipos de *botnets* independente do protocolo ou arquitetura utilizada.

No Estudo de Caso 2, utilizamos o EFC para seleção dos atributos que mais contribuem para um maior ou menor valor de energia. Uma vez que o EFC é um algoritmo interpretável, é possível verificar quais pares de atributos possuem maior valor de acoplamento e quais pares possuem menor valor. Sendo assim, foram selecionados os atributos de acordo com os valores de acoplamento entre eles. Assim como no Estudo de Caso 1, utilizamos os conjuntos de dados CTU-13 e ISOT HTTP para treinamento e teste e fizemos o teste de domínio cruzado, i.e., treino em um conjunto de dados e teste em outro

conjunto de dados. Por fim, ainda utilizamos um terceiro conjunto de dados, o ISCX-Bot-2014, o qual possui uma variedade maior de *botnets*, no intuito de validar os atributos selecionados em um outro conjunto de dados. Comparamos os atributos selecionados pelo EFC com os atributos selecionados pelo algoritmo *Random Forest* e com um conjunto de atributos utilizados no estudo de Ramos et al., [1].

Em ambos Estudos de Casos, comparamos os resultados obtidos com o modelo EFC com os resultados obtidos pelos algoritmos de aprendizado de máquina mais populares, de acordo com os trabalhos relacionados ao tema. Tais algoritmos realizam a classificação binária, ou seja, precisam que o conjunto de treinamento tenha instâncias maliciosas e benignas para o aprendizado. Como o EFC é um algoritmo de uma classe, ou seja, é treinado apenas com o tráfego benigno, também foram realizados experimentos com os algoritmos de uma classe disponíveis na biblioteca *scikit-learn*[5], com o intuito de comparar os resultados obtidos com o EFC. Cabe salientar que o EFC foi implementado com a sua configuração padrão. Além disso, os demais classificadores foram implementados com a configuração padrão do *scikit-learn*. Outra questão importante é que o EFC trabalha com dados discretizados para realizar a classificação, sendo assim, foi realizada a discretização somente para a implementação do EFC. Para isso, utilizamos o módulo *KBinsDiscretizer*[6] do *scikit-learn* com os seguintes parâmetros: n_bins=30, *encode='ordinal'*, *strategy='quantile'*. Para os demais modelos utilizados nesta pesquisa, foi realizada apenas a normalização dos dados, uma vez que a discretização poderia prejudicar o desempenho destes algoritmos. Na próxima seção apresentamos uma descrição dos algoritmos utilizados para comparação com o EFC.

4.4 Avaliação

O objetivo dos modelos de AM é reconhecer padrões que oferecem um bom resultado de generalização nos dados não vistos, em vez de apenas memorizar os dados apresentados durante o treinamento. Após desenvolver um modelo de aprendizagem, é importante verificar se ele apresenta um bom desempenho em amostras não usadas no treinamento do modelo. Existem várias abordagens em AM para medir a precisão da predição de um modelo. Utilizamos nesta pesquisa, o método comum de validação cruzada (*cross validation*), o qual possui o objetivo de testar a capacidade do modelo de prever novos dados que não foram utilizados no treinamento, a fim de sinalizar problemas como *overfitting* ou viés de seleção e dar uma ideia de como o modelo será generalizado para um conjunto de dados independente [114].

[5] `https://scikit-learn.org/stable/`

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.KBinsDiscretizer.html`

O *k-fold* é o método de validação cruzada mais conhecido e utilizado. Neste processo os dados são divididos em k subconjuntos, contendo quantidades de amostras aproximadamente iguais. A cada vez, um dos k subconjuntos é utilizado como conjunto de teste e os outros $k-1$ subconjuntos formam o conjunto de treinamento. Este método se repete até que todos os subconjuntos tenham sido utilizados como conjunto de teste, sendo que as estatísticas de erro são calculadas em todas as k tentativas. O desempenho final do preditor é dado pela média dos desempenhos observados sobre cada conjunto de teste. Utilizamos o recurso *StratifiedKFold* para preservar a porcentagem de amostras para cada classe, na divisão dos k subconjuntos. Consideramos, nos nossos experimentos, um valor de $k = 5$ e calculamos duas métricas de desempenho padrão: F1-score e a área sobre a curva ROC (AUC). A primeira métrica, F1-score, é a média harmônica das métricas *Precision* e *Recall*, i.e.,

$$F1 = \frac{2}{Precision^{-1} + Recall^{-1}} = \frac{2 * Precision * Recall}{Precision + Recall}$$

onde $Precision = TP/(TP + FP)$ e $Recall = TP/(TP + FN)$. TP é o número de verdadeiros positivos, i.e., fluxos maliciosos classificados como maliciosos, FP é o número de falsos positivos, i.e., fluxos benignos classificados como maliciosos, e FN é o número de falsos negativos, i.e., o número de fluxos maliciosos classificados como benignos.

A segunda métrica, a área sobre a curva ROC (AUC), é obtida plotando a taxa de verdadeiros positivos (TPR), i.e., a proporção de amostras positivas classificadas corretamente, contra a taxa de falsos positivos (FPR) i.e., a proporção de amostras negativas classificadas incorretamente, para diferentes limiares de classificação. Uma AUC de 1,0 indica classificação perfeita e uma AUC de 0,5 indica classificação aleatória. Modelos com desempenho melhor do que o acaso devem atingir valores de AUC na faixa de 0,5 a 1,0 [115]. Uma das maiores vantagens da AUC é que ela é invariante em relação a mudanças na distribuição das classes [115]. Como estamos interessados em investigar a capacidade de adaptação dos classificadores à diferentes domínios, essa métrica é especialmente interessante para esta pesquisa.

Essencialmente, para cada experimento realizado, tanto no Estudo de Caso 1, quanto no Estudo de Caso 2, mensuramos o desempenho dos classificadores quando treinados e testados no mesmo domínio e quando treinados em um domínio e testados em um domínio diferente. O desempenho foi mensurado com base na média dos valores de AUC e F1-score sobre 5 conjuntos de teste (5 *Stratifiedk-fold*) e no erro padrão, com intervalo de confiança de 95%.

4.5 Considerações Finais

Neste capítulo foram abordados conceitos importantes para a compreensão da solução proposta. Também foram apresentados os três conjuntos de dados que serão utilizados nos experimentos, além da metodologia empregada para a extração dos fluxos a partir das capturas de tráfego disponibilizadas. Além disso, foram apresentados detalhes sobre os dois Estudos de Caso que serão implementados no capítulo seguinte, bem como as métricas que serão utilizadas para a avaliação dos resultados obtidos por todos os classificadores utilizados nesta pesquisa. No próximo capítulo apresentaremos os experimentos realizados e os resultados obtidos.

Capítulo 5

Experimentos e Resultados

Neste capítulo são apresentados os resultados obtidos com a utilização do algoritmo EFC para classificação de tráfego relacionado à atividade de *botnets*. Os experimentos foram divididos em dois Estudos de Caso. No primeiro, comparamos o desempenho do algoritmo EFC com o desempenho de outros algoritmos de aprendizado de máquina baseados em uma e duas classes, utilizando para isto, dois conjuntos de dados heterogêneos. Além disso, foi avaliado o desempenho de todos os algoritmos considerando a adaptação de domínio, realizando o treinamento em um conjunto de dados e o teste em outro. No segundo Estudo de Caso, fizemos uma seleção dos atributos mais relevantes para a detecção de *botnets*, através da análise dos acoplamentos entre os pares de atributos e fizemos a comparação dos resultados com outras duas abordagens de seleção de atributos. Além dos dois conjuntos de dados utilizados no Estudo de Caso 1, também testamos os atributos selecionados em um terceiro conjunto de dados.

5.1 Distribuição das Energias Calculadas pelo EFC

Para realizar a classificação de fluxos benignos e fluxos contendo atividades de *botnets*, o EFC infere um modelo estatístico baseado nas amostras de fluxos benignos durante o treinamento do modelo. Esse modelo é então utilizado para calcular as energias de amostras de fluxos benignos e maliciosos contidos no conjunto de teste e a partir dos valores calculados, realizar a classificação do fluxo de rede.

A Figura 5.1 (a) ilustra os valores de energia calculados considerando parte das amostras benignas do conjunto de dados CTU-13 e a figura 5.1 (b) mostra os valores das energias referentes ao conjunto de dados ISOT HTTP. Esses valores são referentes ao teste intra-domínio, ou seja, treino e teste no mesmo conjunto de dados. Pode-se observar que a separação entre as duas classes é clara, i.e., a energia de fluxos benignos está claramente deslocada para a esquerda em relação à distribuição das energias de fluxos

maliciosos. A linha vertical vermelha representa o limiar de classificação do EFC, o qual foi definido como percentil 95 da distribuição de energia obtida na etapa de treino.

5.1 (a) Treino/Teste CTU-13 5.1 (b) Treino/Teste ISOT HTTP

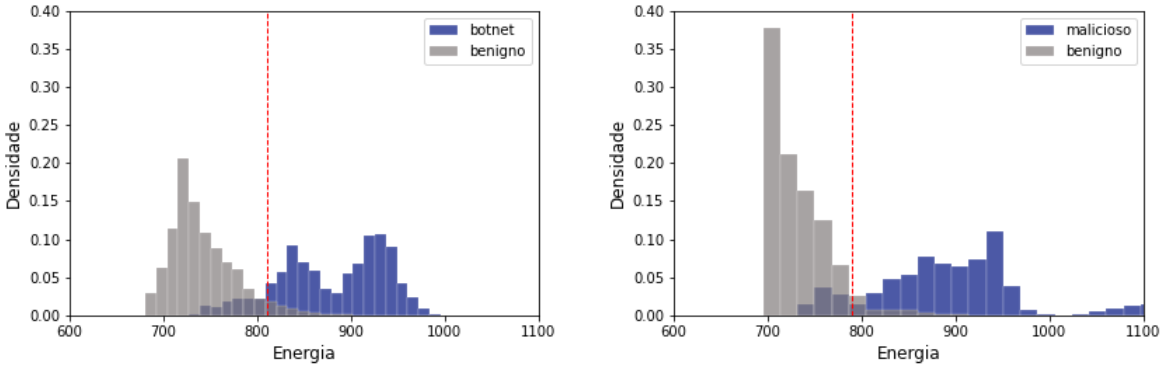


Figura 5.1: Histogramas das Energias Calculadas na Fase de Teste Usando os *Datasets* CTU-13 e ISOT HTTP.

5.2 Estudo de Caso 1: Avaliação do EFC Para Detecção de *Botnets*

O objetivo do Estudo de Caso 1 é avaliar a utilização do algoritmo EFC para a detecção de *botnets* e ainda, verificar a adaptabilidade do modelo inferido em um domínio diferente daquele onde foi realizado o treinamento do modelo. Para isto, utilizamos dois conjuntos de dados heterogêneos, o CTU-13 e o ISOT HTTP. Além disso, utilizamos os 80 atributos extraídos inicialmente e comparamos os resultados obtidos pelo EFC com os resultados obtidos por outros algoritmos de aprendizado de máquina baseados em uma e em duas classes. Por fim, avaliamos o desempenho de todos os algoritmos considerando a adaptação de domínio, i.e., realizando o treinamento em um conjunto de dados e o teste em outro.

Todos os experimentos realizados neste trabalho foram efetuados em um notebook com a seguinte configuração: processador Intel Core I7-7700HQ de 3.8 GHZ, 32 GB de memória RAM e sistema operacional Linux Debian 11. A seguir serão apresentados os resultados obtidos no Estudo de Caso 1.

5.2.1 EFC x Algoritmos de Classificação de Duas Classes

Primeiramente, realizamos a comparação do EFC com diferentes classificadores baseados em duas classes, os quais utilizam a classe benigna e maliciosa para o treinamento dos modelos. Os seguintes algoritmos foram selecionados por serem os mais populares para

detecção de anomalias baseada na análise de fluxos de rede: *k-Nearest Neighbors (KNN), Decision Tree (DT), Multilayer Perceptron (MLP), Naive Bayes (NB)* e *Support Vector Machine (SVM)*, além dos classificadores *ensemble*: *AdaBoost (AD)* e *Random Forest (RF)*.

Foram realizados dois testes principais. O primeiro é o teste intra-domínio, no qual o treinamento e o teste dos modelos foram realizados utilizando o mesmo conjunto de dados. O segundo é o teste inter-domínio, no qual os modelos são treinados em um conjunto de dados e são avaliados em outro conjunto de dados. O objetivo do segundo teste é avaliar a capacidade dos modelos de se adaptarem à mudanças na rede e consequentemente, a capacidade de detectar *botnets* desconhecidas.

5.2.1.1 Resultados ISOT HTTP

Na Tabela 5.1 pode ser visualizado o desempenho médio e o erro padrão (com intervalo de confiança de 95%) de cada classificador, utilizando o conjunto de dados ISOT HTTP como base para o treinamento dos modelos. Na primeira abordagem, que é o teste intra-domínio, todos os modelos propostos obtiveram resultados muito próximos, com um F1-score acima de 0.98 e AUC acima de 0.99. A única exceção foi o algoritmo NB, o qual obteve o menor desempenho, com um F1-score de 0.952 ± 0.000 e AUC de 0.799 ± 0.004.

Tabela 5.1: Desempenho Médio dos Classificadores - ISOT HTTP x CTU-13

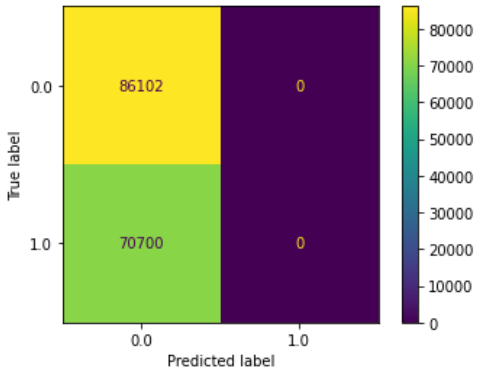
	Treino/Teste ISOT		**Treino ISOT/Teste CTU-13**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.952 ± 0.000	0.799 ± 0.004	0.000 ± 0.000	0.500 ± 0.000
KNN	**0.999 ± 0.000**	0.997 ± 0.000	0.074 ± 0.009	0.333 ± 0.009
DT	**0.999 ± 0.000**	0.996 ± 0.000	0.019 ± 0.031	0.473 ± 0.031
SVM	0.989 ± 0.000	0.998 ± 0.000	0.120 ± 0.002	0.636 ± 0.002
MLP	0.994 ± 0.001	0.999 ± 0.000	0.271 ± 0.240	0.601 ± 0.240
EFC	0.989 ± 0.000	0.995 ± 0.000	**0.663 ± 0.001**	0.535 ± 0.001
Ensemble				
RF	**0.999 ± 0.000**	**1.000 ± 0.000**	0.000 ± 0.000	0.539 ± 0.000
AD	0.998 ± 0.001	**1.000 ± 0.000**	0.000 ± 0.000	**0.756 ± 0.000**

No teste inter-domínio, onde o conjunto de dados ISOT HTTP foi utilizado para treino e o CTU-13 para teste, o EFC obteve resultados surpreendentes quando comparado aos demais modelos, principalmente em relação à métrica F1-score (0.663 ± 0.001). Um fato

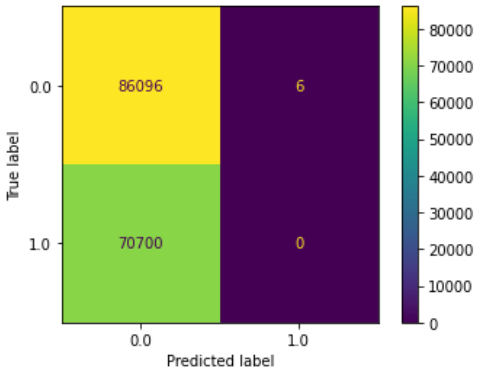
que chamou a atenção foi que os modelos NB, RF e AD obtiveram um F1-score igual a 0. Foi observado através das respectivas matrizes de confusão, conforme a Figura 5.2, que estes modelos classificaram todas as instâncias maliciosas como benignas, obtendo portanto, um taxa de verdadeiro-positivo igual a 0, o que explica o valor do F1-score obtido por estes modelos. Por fim, uma vez que classificadores ensemble utilizam uma combinação das previsões de diversos modelos e geralmente apresentam melhor desempenho em relação à classificadores simples, esperava-se que o desempenho do RF e AD fosse superior ao desempenho dos demais modelos testados.

O teste inter-domínio neste cenário é bastante desafiador, uma vez que o conjunto de dados ISOT HTTP possui instâncias somente de *botnets* que utilizam o protocolo HTTP para comunicação (arquitetura centralizada) e o conjunto de dados CTU-13 possui *botnets* que utilizam os protocolos HTTP, IRC e P2P (arquiteturas centralizadas e descentralizadas). Dessa forma, o resultado obtido pelo EFC foi bastante satisfatório, dada a grande mudança de contexto.

5.2 (a) *Naive Bayes*



5.2 (b) *Random Forest*



5.2 (c) *AdaBoost*

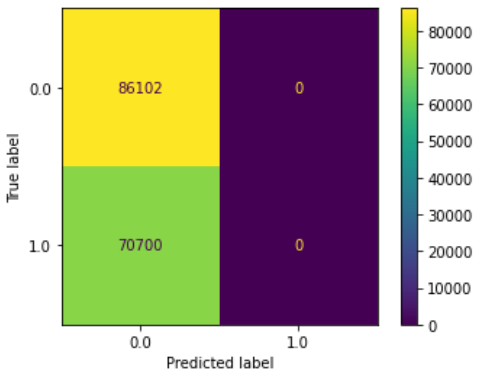


Figura 5.2: Matrizes de Confusão dos Classificadores NB, RF e AD nos Experimentos de Classificação Inter-Domínio.

5.2.1.2 Resultados CTU-13

A Tabela 5.2 mostra o desempenho médio e o erro padrão (com intervalo de confiança de 95%) de cada classificador, utilizando o conjunto de dados CTU-13 como base para o treinamento dos modelos. Na primeira abordagem, que é o teste intra-domínio, os modelos KNN, DT, MLP e RF obtiveram um F1-score e AUC acima de 0.99, enquanto o EFC obteve um F1-score de 0.877 ± 0.000 e AUC de 0.961 ± 0.000. Novamente o NB obteve o menor desempenho, com um F1-score de 0.677 ± 0.001 e AUC de 0.864 ± 0.013.

Tabela 5.2: Desempenho Médio dos Classificadores - CTU-13 x ISOT HTTP

	Treino/Teste CTU-13		**Treino CTU-13/Teste ISOT**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.677 ± 0.001	0.864 ± 0.013	0.109 ± 0.214	0.675 ± 0.214
KNN	0.997 ± 0.000	0.999 ± 0.000	0.005 ± 0.007	0.457 ± 0.007
DT	**0.999 ± 0.000**	0.999 ± 0.000	0.544 ± 0.004	0.275 ± 0.004
SVM	0.912 ± 0.003	0.962 ± 0.001	0.481 ± 0.010	0.665 ± 0.003
MLP	0.994 ± 0.000	**1.000 ± 0.000**	0.325 ± 0.240	0.711 ± 0.240
EFC	0.877 ± 0.000	0.961 ± 0.000	0.758 ± 0.076	**0.729 ± 0.076**
Ensemble				
RF	**0.999 ± 0.000**	**1.000 ± 0.000**	**0.794 ± 0.022**	0.702 ± 0.022
AD	0.980 ± 0.002	0.998 ± 0.000	0.262 ± 0.172	0.573 ± 0.172

No teste inter-domínio, onde o conjunto de dados CTU-13 foi utilizado para treino e o ISOT-HTTP para teste, o RF obteve o maior valor de F1-score (0.794 ± 0.022), enquanto o EFC obteve o melhor AUC (0.729 ± 0.076). O desempenho dos demais classificadores foi bastante inferior. O teste inter-domínio neste cenário é menos desafiador quando comparado ao experimento anterior. Aqui, o treino é realizado em um conjunto de dados contendo *botnets* com protocolos de comunicação variados (HTTP, IRC e P2P) e o teste em um conjunto de dados contendo somente *botnets* HTTP. Sendo assim, todos os modelos obtiveram um desempenho superior ao obtido no experimento inter-domínio da subseção anterior e observa-se ainda, que nenhum modelo obteve F1-score igual a 0.

5.2.2 EFC x Algoritmos de Classificação de Uma Classe

Os testes realizados a seguir tiveram como objetivo comparar o desempenho do EFC com o desempenho de diferentes classificadores de uma classe, os quais possuem metodologia de

classificação similar à do EFC, uma vez que utilizam apenas dados benignos para inferir o modelo de classificação. Para atender a este propósito, foram implementados os seguintes classificadores *One Class* disponíveis na biblioteca *scikit-learn*: OCSVM, IForest, LOF e Elenv.

Assim como no experimento anterior, foram realizados os testes intra-domínio, i.e., treino e teste no mesmo conjunto de dados e os testes inter-domínio, i.e., treinamento dos modelos em um conjunto de dados e teste em outro conjunto de dados, avaliando assim, a capacidade dos modelos de detectar *botnets* desconhecidas.

5.2.2.1 Resultados ISOT-HTTP

Na Tabela 5.3 pode ser visualizado o desempenho médio e o erro padrão (com intervalo de confiança de 95%) de cada classificador, utilizando o conjunto de dados ISOT-HTTP como base para o treinamento dos modelos. Na primeira abordagem, que é o teste intra-domínio, o EFC obteve desempenho bem superior aos demais classificadores, tanto em relação ao F1-score (0.989 ± 0.000) quanto ao AUC (0.995 ± 0.000). O menor desempenho foi obtido pelo classificador Elenv com um F1-score de 0.035 ± 0.034 e AUC de 0.781 ± 0.005.

No teste inter-domínio, onde o conjunto de dados ISOT HTTP foi utilizado para treino e o CTU-13 para teste, o EFC manteve o melhor desempenho em relação ao F1-score (0.663 ± 0.001), seguido pelos classificadores OCSVM (0.627 ± 0.005) e LOF (0.621 ± 0.000). Em relação à AUC, o LOF obteve o melhor resultado (0.770 ± 0.000), seguido pelo OCSVM (0.726 ± 0.005) e pelo EFC (0.535 ± 0.001).

Tabela 5.3: Desempenho Médio dos Classificadores *One Class* - ISOT HTTP

	Treino/Teste ISOT		**Treino ISOT/Teste CTU-13**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
EFC	**0.989 ± 0.000**	**0.995 ± 0.000**	**0.663 ± 0.001**	0.535 ± 0.001
OCSVM	0.731 ± 0.001	0.540 ± 0.002	0.627 ± 0.005	0.726 ± 0.005
IForest	0.670 ± 0.040	0.768 ± 0.009	0.443 ± 0.043	0.342 ± 0.043
Elenv	0.035 ± 0.034	0.781 ± 0.005	0.234 ± 0.247	0.466 ± 0.247
LOF	0.630 ± 0.011	0.721 ± 0.029	0.621 ± 0.000	**0.770 ± 0.000**

5.2.2.2 Resultados CTU-13

A Tabela 5.4 mostra o desempenho médio e o erro padrão (com intervalo de confiança de 95%) de cada classificador, utilizando o conjunto de dados CTU-13 como base para o treinamento dos modelos. Na primeira abordagem, que é o teste intra-domínio, o modelo LOF obteve o melhor desempenho, tanto em relação ao F1-score (0.923 ± 0.002), quanto em relação à AUC (0.983 ± 0.000), seguido pelo EFC, o qual obteve um F1-score de 0.879 ± 0.003 e AUC de 0.962 ± 0.001. O classificador IForest obteve o pior desempenho, com um F1-score de 0.084 ± 0.002 e AUC de 0.595 ± 0.025.

Tabela 5.4: Desempenho Médio dos Classificadores *One Class* - CTU13

	Treino/Teste CTU13		**Treino CTU-13/Teste ISOT**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
EFC	0.879 ± 0.003	0.962 ± 0.001	0.705 ± 0.002	**0.736 ± 0.003**
OCSVM	0.762 ± 0.001	0.751 ± 0.003	**0.906 ± 0.000**	0.338 ± 0.001
IForest	0.084 ± 0.002	0.595 ± 0.025	0.732 ± 0.198	0.628 ± 0.002
Elenv	0.257 ± 0.143	0.705 ± 0.043	0.411 ± 0.314	0.412 ± 0.143
LOF	**0.923 ± 0.002**	**0.983 ± 0.000**	0.893 ± 0.001	0.435 ± 0.002

No teste inter-domínio, onde o conjunto de dados CTU-13 foi utilizado para treino e o ISOT-HTTP para teste, o OCSVM obteve o melhor F1-score (0.906 ± 0.000), porém a AUC foi a menor dentre todos os outros classificadores (0.338 ± 0.001). Por outro lado, o EFC obteve a melhor AUC (0.736 ± 0.003). Estes resultados serão melhor investigados em experimentos futuros.

5.2.3 Síntese do Estudo de Caso 1

Os resultados obtidos neste Estudo de Caso demonstraram que os modelos baseados em duas classes sofreram fortes variações nos testes inter-domínio, principalmente no cenário mais desafiador, no qual o conjunto de treinamento possuía somente *botnets* centralizadas (HTTP) e o conjunto de testes possuía *botnets* com arquiteturas centralizadas (HTTP e IRC) e descentralizadas (P2P). Neste cenário o EFC foi bem superior aos demais modelos, obtendo um F1-score de 0.663 ± 0.0010 e AUC de 0.535 ± 0.001.

Em relação aos modelos baseados em uma classe, os resultados demonstraram que o EFC no teste intra-domínio apresentou resultados superiores aos demais algoritmos *One Class*, considerando o conjunto de dados ISOT HTTP, obtendo um F1-score de 0.989 e AUC de 0.995. Nos testes com o *dataset* CTU-13, o EFC obteve um F1-score

de 0.879 e um AUC de 0.962, sendo superado apenas pelo algoritmo LOF. Já no teste inter-domínio, o EFC se mostrou superior, ou em relação ao F1-score (0.663 ± 0.001) em um dos experimentos, ou em relação à AUC (0.736 ± 0.003) em outro experimento. Sendo assim, no contexto geral, o EFC foi o modelo que se mostrou menos sensível a mudanças na distribuição de dados, apresentando resultados mais robustos e se mostrando um classificador promissor para a detecção de novos tipos de *botnets*.

5.3 Estudo de Caso 2: Seleção de Atributos

A quantidade de atributos que descrevem uma instância de dados pode variar em até centenas de atributos. Em geral, espera-se que todos os atributos sejam relevantes, porém nem sempre é possível garantir isso. Por outro lado, a qualidade representativa de um conjunto de atributos influencia muito a eficácia dos algoritmos de AM. Sendo assim, é desejável selecionar cuidadosamente o número e o tipo de atributos utilizados para treinar os algoritmos de AM. Este processo é conhecido como seleção de atributos. Uma das principais vantagens da redução do número de atributos é a diminuição dos tempos de aprendizado e classificação [101]. Além disso, a remoção de atributos irrelevantes ou redundantes também pode aumentar a precisão da classificação [101].

Neste estudo de caso, fizemos uma seleção dos atributos que melhor caracterizam o comportamento do tráfego de rede benigno, através da análise dos acoplamentos entre os pares de atributos calculados pelo algoritmo EFC, durante o treinamento do modelo. Para comprovar a efetividade da seleção de atributos utilizando o EFC, fizemos experimentos com os mesmos atributos propostos no trabalho de Ramos et al. [1], o qual utilizou o conjunto de dados ISOT HTTP como base para a seleção de parte dos atributos utilizados em seu estudo. Além disso, utilizamos o algoritmo RF para seleção dos atributos mais importantes, com o intuito de comparar os resultados dessa seleção com a seleção realizada pelo EFC.

Além de utilizarmos os dois conjuntos de dados do Estudo de Caso 1, utilizamos um terceiro conjunto (ISCX-Bot-2014) com o objetivo de validar a seleção de atributos e avaliar se os atributos selecionados pelo EFC são capazes de manter a generalidade do modelo. A seguir serão apresentados os resultados obtidos no Estudo de Caso 2, para os três cenários, i.e. atributos selecionados pelo EFC, atributos utilizados em Ramos et al. [1] e por fim, atributos selecionados pelo algoritmo RF.

Cabe ressaltar que utilizamos a mesma configuração de hardware do estudo de caso anterior, i.e. processador Intel Core I7-7700HQ de 3.8 GHZ, 32 GB de memória RAM e sistema operacional Linux Debian 11.

5.3.1 Atributos Selecionados pelo EFC

O algoritmo EFC possui uma importante característica que é a possibilidade de se analisar os valores de acoplamento e campo local atribuídos para cada par de atributos ou para cada atributo, respectivamente. Esses valores de acoplamento e campo local compõem a energia total de um fluxo, a qual mede o quão improvável é que um fluxo pertença a uma determinada distribuição de probabilidade. Sendo assim, o acoplamento mede a probabilidade de que a configuração exata dos valores dos pares de atributos ocorra no conjunto de fluxos benignos que geraram o modelo [20]. Dessa forma, é possível identificar quais são os pares de atributos que mais contribuem para um maior ou menor valor de energia. Devido à esta característica, o EFC pode ser considerado um algoritmo interpretável ou caixa branca, ao contrário da maioria dos algoritmos tradicionais existentes [20].

Os experimentos a seguir foram realizados no intuito de verificar, empiricamente, se através da análise dos pares de atributos que mais contribuem para a tarefa de classificação, é possível selecionar um conjunto mínimo de atributos que seja capaz de manter a generalidade do modelo. Para validar esta hipótese, os atributos obtidos a partir dos valores de acoplamento calculados pelo EFC serão avaliados em três conjuntos de dados distintos. Utilizamos o conjunto de dados ISOT HTTP como base para a análise e obtenção dos atributos, uma vez que este conjunto de dados também foi utilizado no estudo de Ramos et al. [1] para a seleção dos atributos, possibilitando assim, a comparação dos resultados obtidos.

Para a geração da matriz de acoplamentos, utilizamos 69 atributos. No Estudo de Caso 1, foram utilizados 80 atributos nos experimentos, porém foi observado que 11 destes atributos continham o valor 0 em todas as linhas do conjunto de dados ISOT HTTP. Optamos, então, por remover estes atributos, uma vez que eles se tornam irrelevantes, já que possuem valor nulo, independente do fluxo ser benigno ou malicioso. Sendo assim, os atributos removidos foram: *'FwdPSHFlags', 'BwdPSHFlags', 'FwdURGFlags', 'BwdURGFlags', 'RSTFlagCount', 'URGFlagCount', 'CWEFlagCount', 'ECEFlagCount', 'FwdAvgBytes-Bulk', 'FwdAvgPackets-Bulk', e 'FwdAvgBulkRate'.*

A Figura 5.3 ilustra a matriz de acoplamentos gerada a partir dos valores de acoplamentos calculados pelo EFC, considerando os 69 atributos. Os pares de atributos com um valor maior de acoplamento (cor vermelha), indicam uma maior probabilidade do fluxo ser classificado como malicioso. Por outro lado, quanto menor o acoplamento (cor azul), maior a probabilidade do fluxo ser classificado como benigno.

Para melhor analisar os valores dos acoplamentos, transformamos o *array* com todas as combinações possíveis de pares de atributos em um arquivo CSV e ordenamos este arquivo baseado nos valores dos acoplamentos calculados para cada par de atributos. Em

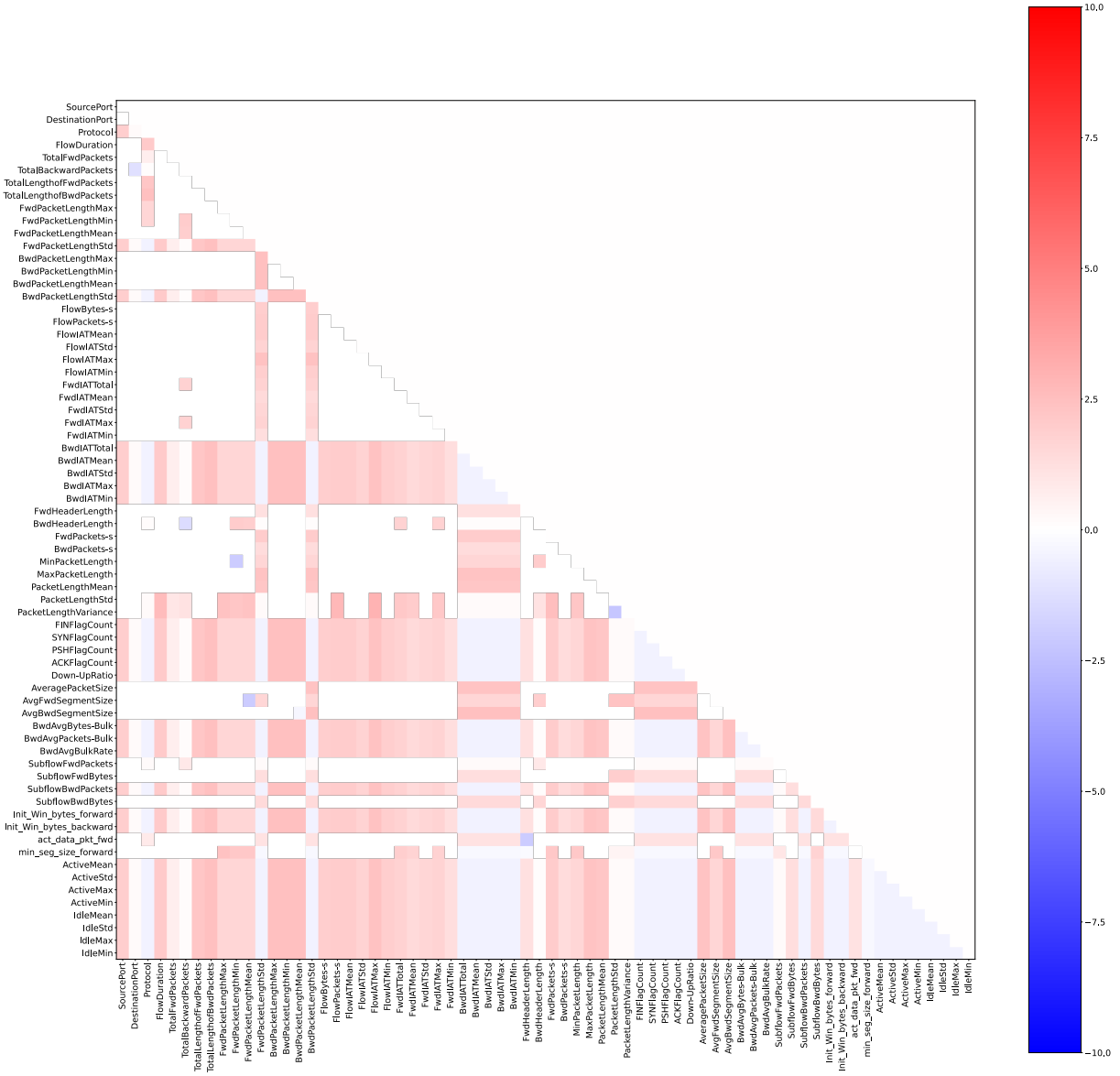


Figura 5.3: Matriz dos Acoplamentos entre Pares de Atributos.

seguida, selecionamos os dez maiores valores de acoplamento (tráfego malicioso) e os 10 menores valores de acoplamento (tráfego benigno), conforme a Tabela 5.5.

Tabela 5.5: Acoplamento entre Pares de Atributos - ISOT HTTP

Atributo 1	**Atributo 2**	**Acoplamento**
Init Win bytes forward	Protocol	-0.471146077382481
IdleMax	Protocol	-0.471146077382481
Init Win bytes backward	Protocol	-0.471146077382484
IdleMean	Protocol	-0.471146077382484
TotalBackwardPackets	DestinationPort	-1.1852260023329
BwdHeaderLength	TotalBackwardPackets	-1.37934354028325
MinPacketLength	FwdPacketLengthMin	-1.95276026388922
AvgFwdSegmentSize	FwdPacketLengthMean	-1.95448435332454
act data pkt fwd	FwdHeaderLength	-2.01219728636031
PacketLengthVariance	PacketLengthStd	-2.22536254946391
PacketLengthVariance	FlowIATMax	3.01094833323442
PacketLengthStd	FlowIATMax	3.01094833323332
PacketLengthVariance	FlowPackets-s	2.87707231577872
PacketLengthStd	FlowPackets-s	2.87707231577644
SubflowFwdPackets	FwdPacketLengthMax	2.83759850250978
IdleMean	FlowBytes-s	2.79154983034748
Init Win bytes forward	FlowBytes-s	2.79154983034743
IdleMin	FlowBytes-s	2.79154983034743
BwdIATMin	FlowBytes-s	2.79154983034583
PacketLengthVariance	FlowDuration	2.58326823567109

A partir da análise dos pares de atributos com os maiores e menores valores de acoplamento, obtemos 25 atributos exclusivos, os quais são apresentados na Tabela 5.6.

Estes atributos serão a base para os experimentos que serão apresentados a seguir, sendo compostos por três abordagens distintas. Na primeira abordagem, apresentamos os resultados obtidos utilizando o conjunto de dados ISOT HTTP como base para o treinamento dos modelos, já que foi a partir deste conjunto de dados que selecionamos os atributos citados. Em seguida, realizamos a análise da adaptabilidade dos modelos, da mesma forma como foi feito no Estudo de Caso 1. Sendo assim, utilizamos o ISOT HTTP para treinamento e o CTU-13 para teste. A segunda abordagem faz exatamente

Tabela 5.6: Atributos Selecionados EFC - ISOT HTTP

N°	Atributo	Descrição
01	PacketLengthVariance	Variância do comprimento de um pacote
02	FlowIATMax	Tempo máximo entre dois pacotes enviados no fluxo
03	PacketLengthStd	Desvio padrão do comprimento de um pacote
04	FlowPackets-s	Número de pacotes em um fluxo por segundo
05	SubflowFwdPackets	O número médio de pacotes em um subfluxo na direção direta
06	FwdPacketLengthMax	Tamanho máximo do pacote na direção direta
07	IdleMean	Tempo médio que um fluxo ficou ocioso antes de se tornar ativo
08	FlowBytes-s	Número de bytes em um fluxo por segundo
09	Init Win Bytes Forward	O número total de bytes enviados na janela inicial na direção direta
10	IdleMin	Tempo mínimo que um fluxo ficou ocioso antes de se tornar ativo
11	BwdIATMin	Tempo mínimo entre dois pacotes enviados no sentido inverso
12	Protocol	Protocolo
13	IdleMax	Tempo máximo que um fluxo ficou ocioso antes de se tornar ativo
14	Init Win Bytes Backward	O número total de bytes enviados na janela inicial na direção inversa
15	TotalBackwardPackets	Total de pacotes na direção inversa
16	BwdHeaderLength	Total de bytes usados para cabeçalhos na direção inversa
17	MinPacketLength	Comprimento mínimo de um pacote
18	AvgFwdSegmentSize	Tamanho médio do segmento observado na direção direta
19	Act Data Pkt Fwd	Contagem de pacotes com pelo menos 1 byte de dados TCP no payload na direção direta
20	DestinationPort	Porta de Destino
21	FwdPacketLengthMin	Tamanho mínimo do pacote na direção direta
22	FwdHeaderLength	Comprimento do cabeçalho do pacote na direção direta
23	FwdPacketLengthMean	Tamanho médio do pacote na direção direta
24	FlowDuration	Duração do fluxo em microssegundos
25	TotalLengthofBwdPackets	Tamanho total do pacote na direção inversa

o contrário. Assim, utilizamos o CTU-13 como base para treinamento dos modelos e testamos a adaptabilidade dos modelos utilizando o ISOT HTTP para teste. Por fim, a terceira abordagem utiliza um outro conjunto de dados (ISCX-Bot-2014), não utilizado no Estudo de Caso 1. Esta abordagem tem o objetivo de avaliar os atributos selecionados pelo EFC em um conjunto de dados distinto, composto por uma variedade maior de

famílias de *botnets*.

5.3.1.1 Resultados ISOT HTTP x CTU-13

A Tabela 5.7 apresenta os resultados da primeira abordagem, na qual utilizamos o conjunto de dados ISOT HTTP como base para o treinamento dos modelos. Comparamos os resultados obtidos utilizando os 25 atributos selecionados pelo EFC com os resultados obtidos no Estudo de Caso 1, no qual foram utilizados 80 atributos. Está sinalizado em azul os resultados que tiveram um aumento igual ou maior a 1%, em relação aos resultados obtidos no Estudo de Caso 1. Por outro lado, sinalizamos em vermelho os resultados com decréscimo maior ou igual a 1%.

Tabela 5.7: Desempenho Médio dos Classificadores - 25 Atributos EFC - ISOT x CTU

	Treino/Teste ISOT HTTP		**Treino ISOT/Teste CTU-13**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.955 ± 0.000	0.870 ± 0.003	0.023 ± 0.008	0.491 ± 0.001
KNN	**0.996** ± 0.000	0.997 ± 0.000	0.036 ± 0.001	0.507 ± 0.003
DT	**0.996** ± 0.000	0.991 ± 0.001	0.029 ± 0.008	0.502 ± 0.005
SVM	0.984 ± 0.002	0.994 ± 0.004	0.329 ± 0.004	0.623 ± 0.109
MLP	0.994 ± 0.000	0.999 ± 0.000	0.139 ± 0.001	**0.706 ± 0.053**
EFC	0.988 ± 0.000	0.995 ± 0.000	**0.653 ± 0.001**	0.560 ± 0.019
Ensemble				
RF	**0.996** ± 0.000	**1.000 ± 0.000**	0.000 ± 0.000	0.513 ± 0.161
AD	0.994 ± 0.000	**1.000 ± 0.000**	0.062 ± 0.017	0.472 ± 0.027

No teste intra-domínio, as variações em relação ao F1-score foram inferiores a 1%. Além disso, os algoritmos KNN, DT e RF mantiveram o melhor valor para esta métrica, assim como ocorreu no Estudo de Caso 1 (Tabela 5.1). Em relação à AUC, a maior variação foi para o NB, o qual obteve um aumento de (+7,1), ou seja, 87% com 25 atributos, contra 79,9% com 80 atributos. Assim, os resultados no teste intra-domínio utilizando os 25 atributos selecionados pelo EFC, ficaram muito próximos aos obtidos com os 80 atributos utilizados anteriormente, sendo que a AUC do NB teve um aumento significativo.

No teste inter-domínio, i.e., treino no ISOT HTTP e teste no CTU-13, apesar das variações terem sido superiores à 1%, no contexto geral os resultados com 25 atributos ficaram próximos aos resultados com 80 atributos, sendo que o EFC manteve o melhor

F1-score, enquanto os demais classificadores mantiveram resultados muito ruins para esta métrica, assim como aconteceu no Estudo de Caso 1. Em relação à AUC, a variação mais relevante foi para o MLP, o qual obteve o melhor resultado (0.706 ± 0.053), com um aumento de +10,5. Já a AUC do AD obteve a maior redução (-28,4).

A comparação dos resultados obtidos pelos classificadores, utilizando os 80 atributos iniciais e utilizando os 25 atributos selecionados pelo EFC, pode ser melhor visualizada na Figura 5.4. Observa-se pelo gráfico do Teste Intra-Domínio que a redução de atributos não afetou o desempenho dos classificadores e que o NB ainda teve um aumento da métrica AUC. No gráfico do Teste Inter-Domínio, observamos que o EFC manteve a adaptabilidade do modelo, com inclusive, um pequeno aumento da AUC. O restante dos classificadores mantiveram o baixo desempenho, principalmente em relação à métrica F1-score. Sendo assim, podemos afirmar que, neste experimento, os 25 atributos selecionados pelo EFC foram capazes de manter a caracterização do tráfego de *botnets*.

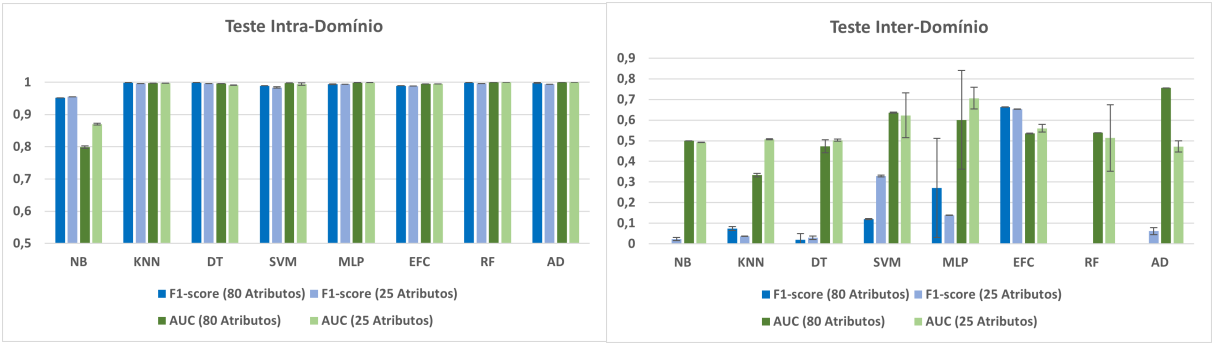


Figura 5.4: Comparação do Desempenho Médio dos Classificadores com 80 e 25 Atributos - ISOT x CTU-13.

A Tabela 5.8 ilustra os tempos de execução dos classificadores com 80 atributos e com 25 atributos. Observa-se que a seleção de atributos proporcionou uma redução significativa nos tempos de treinamento da maioria dos modelos. Em relação ao EFC, uma vez que a sua complexidade é cúbica em relação ao número de *features* para o treinamento do modelo e quadrática para o teste, os resultados na redução dos tempos são consistentes, o que indica que o EFC pode ser um bom classificador a ser empregado em ambientes reais. Por outro lado, o tempo de execução do treinamento para o SVM foi extremamente superior aos demais classificadores (aproximadamente 4 horas). Um fato interessante é que ao contrário dos demais algoritmos, o tempo de treinamento com a redução de atributos aumentou (aproximadamente 6 horas para o treinamento).

Considerando ainda o conjunto de dados ISOT HTTP como base para o treinamento dos modelos, apresentamos na Tabela 5.9 os resultados obtidos pelos classificadores *One Class*, que também foram utilizados no Estudo de Caso 1. Porém, neste experimento utilizamos os 25 atributos selecionados pelo EFC e fizemos uma análise comparativa com

Tabela 5.8: Tempo Médio de Treino e Teste em Segundos - ISOT HTTP x CTU-13

	80 Atributos		**25 Atributos**	
Classificador	**Treino**	**Teste**	**Treino**	**Teste**
NB	0.111 ± 0.011	0.017 ± 0.001	0.084 ± 0.001	0.015 ± 0.002
DT	2.210 ± 0.090	0.008 ± 0.001	1.498 ± 0.035	0.008 ± 0.000
RF	13.768 ± 0.380	0.447 ± 0.008	9.361 ± 0.184	0.212 ± 0.004
EFC	28.066 ± 0.408	1.261 ± 0.008	3.501 ± 0.089	0.165 ± 0.002
AD	35.005 ± 0.614	0.814 ± 0.005	13.132 ± 0.204	0.469 ± 0.010
MLP	228.399 ± 60.201	0.350 ± 0.081	171.323 ± 55.904	0.189 ± 0.021
KNN	357.042 ± 7.625	29.945 ± 2.255	87.414 ± 4.597	9.403 ± 1.175
SVM	15719.610 ± 4119.496	244.507 ± 11.162	22819.454 ± 5084.065	217.473 ± 15.165

os resultados obtidos no Estudo de Caso 1 (Tabela 5.3). Em relação ao teste intra-domínio, i.e., treinamento e teste utilizando o conjunto de dados ISOT HTTP, observa-se que os classificadores OCSVM e iForest tiveram uma redução considerável de desempenho, tanto para o F1-score (-14,3 para OCSVM e -10,2 para o iForest) quanto para AUC (-24,8 para OCSVM e -4,2 para o iForest). Em contra partida, o Elenv obteve uma melhora significativa tanto para a métrica F1-score (+59,4), quanto para a AUC (+11,1). O LOF também obteve um desempenho melhor com a seleção de atributos do EFC, com um aumento no F1-score (+13) e na AUC (+11,9). Os resultados do EFC se mantiveram praticamente os mesmos obtidos no Estudo de Caso 1, mantendo portanto, resultados bem superiores aos demais classificadores.

Tabela 5.9: Desempenho Médio dos Classificadores *One Class* - 25 Atributos EFC

	Treino/Teste ISOT HTTP		**Treino ISOT/Teste CTU-13**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
EFC	**0.987 ± 0.000**	**0.995 ± 0.000**	**0.653 ± 0.001**	0.560 ± 0.019
OCSVM	0.588 ± 0.002	0.292 ± 0.007	0.621 ± 0.000	0.689 ± 0.002
iForest	0.568 ± 0.172	0.726 ± 0.042	0.629 ± 0.004	0.399 ± 0.172
Elenv	0.629 ± 0.168	0.892 ± 0.108	0.620 ± 0.003	0.569 ± 0.065
LOF	0.760 ± 0.059	0.840 ± 0.023	0.622 ± 0.002	**0.744 ± 0.059**

Em relação ao teste inter-domínio, i.e. treinamento no ISOT HTTP e teste no CTU-13, as variações foram mais positivas, evidenciando que os atributos utilizados conseguiram manter a adaptabilidade dos modelos. Assim como ocorreu no estudo de Caso 1, o EFC

manteve o melhor F1-score (0.653 ± 0.001) e o LOF manteve a melhor AUC (0.744 ± 0.059), mesmo tendo sofrido uma pequena redução para esta métrica (-2,6). Cabe ressaltar que o desempenho do Elenv foi bem superior ao obtido no Estudo de Caso 1, tanto para o teste intra-domínio, quanto para o teste de domínio cruzado.

A comparação dos resultados obtidos pelos classificadores *One Class*, utilizando os 80 atributos iniciais e utilizando os 25 atributos selecionados pelo EFC, pode ser melhor visualizada na Figura 5.5. Observa-se no gráfico do Teste Intra-Domínio que a redução de atributos não afetou o desempenho do EFC. Além disso, o Elenv obteve um desempenho muito superior utilizando os atributos selecionados pelo EFC, principalmente em relação à métrica F1-score. O LOF também teve um desempenho ligeiramente superior com a redução de atributos. Por outro lado, o desempenho do IForest e principalmente do OCSVM foi inferior. No gráfico do Teste Inter-Domínio, percebemos que de uma maneira geral, a redução de atributos não afetou o desempenho dos algoritmos, sendo que o desempenho do IForest e do Elenv foi bem superior ao desempenho obtido com os 80 atributos iniciais. Dessa forma, a adaptabilidade dos modelos foi mantida, mesmo com a redução dos atributos.

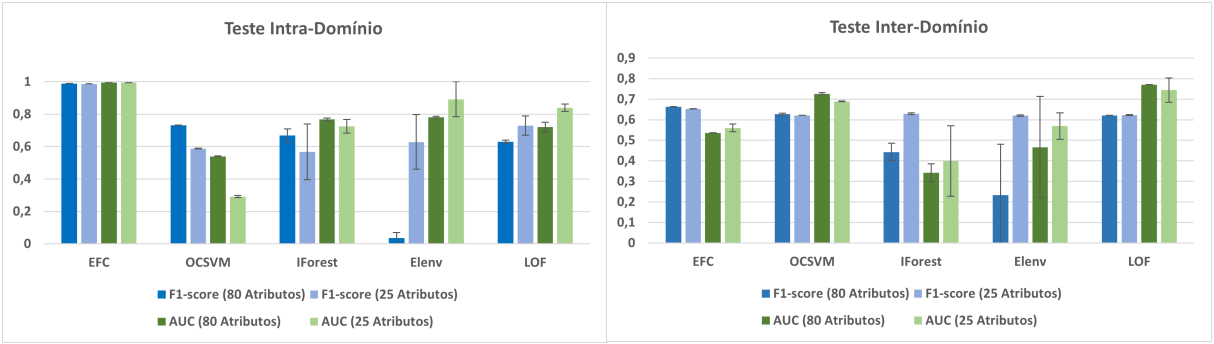


Figura 5.5: Comparação do Desempenho Médio dos Classificadores *One Class* com 80 e 25 Atributos - ISOT x CTU-13.

Apresentamos na Tabela 5.10 o desempenho em relação aos tempos de execução dos classificadores, tanto para treinamento quanto para teste do modelos. Observa-se que os tempos para execução do treinamento e do teste com o classificador OCSVM são muito longos, mesmo com a redução de atributos (i.e. ± 2,7 hs para treinamento e ± 43 minutos para teste). Os demais classificadores realizaram o treinamento dos modelos em menos de 1 minuto, mesmo com os 80 atributos iniciais.

Com estes experimentos, podemos concluir que o conjunto de atributos selecionados a partir da análise dos acoplamentos do EFC, foram capazes de manter o desempenho da maioria dos classificadores, sendo que alguns ainda tiveram desempenho superior ao obtido com os 80 atributos iniciais. Além disso, com exceção do SVM, a redução de

atributos reduziu consideravelmente o tempo de treinamento dos modelos, otimizando assim, o custo computacional para a classificação.

Tabela 5.10: Tempo Médio de Treino e Teste em Segundos - ISOT x CTU-13 - *One Class*

	80 Atributos		**25 atributos**	
Classificador	**Treino**	**Teste**	**Treino**	**Teste**
iForest	1.327 ± 0.004	6.470 ± 0.029	0.504 ± 0.038	3.684 ± 0.225
Elenv	25.119 ± 1.142	0.549 ± 0.004	7.401 ± 0.549	0.053 ± 0.001
EFC	28.066 ± 0.408	1.261 ± 0.008	3.501 ± 0.089	0.165 ± 0.002
LOF	56.852 ± 1.136	157.477 ± 10.611	14.257 ± 0.941	41.595 ± 3.160
OCSVM	17570.797 ± 701.831	3792.085 ± 210.052	9888.109 ± 893.344	1575.646 ± 54.464

5.3.1.2 Resultados CTU-13 x ISOT HTTP

Uma vez que os atributos da Tabela 5.6 foram obtidos a partir do conjunto de dados ISOT-HTTP, vamos avaliar neste experimento, se esses atributos fazem sentido quando utilizamos um conjunto de dados diferente. Assim, utilizaremos o conjunto de dados CTU-13 para o treinamento dos modelos, i.e., faremos o inverso do que foi feito anteriormente. Os resultados podem ser visualizados na Tabela 5.11. Sinalizamos em azul os resultados que tiveram um aumento igual ou maior a 1%, em relação aos resultados obtidos no Estudo de Caso 1 (Tabela 5.2), no qual foram utilizados 80 atributos. Por outro lado, sinalizamos em vermelho os resultados com decréscimo maior ou igual a 1%.

Observa-se que no teste intra-domínio, i.e., treino e teste no CTU-13, alguns classificadores tiveram uma redução de performance, principalmente em relação à métrica F1-score, com destaque para o EFC (-4), MLP (-1,7) e para o AD (-1,5). Em relação à métrica AUC, o EFC teve uma redução de (-2,9). Para os demais classificadores, a variação foi mínima. No teste inter-domínio, i.e., treino no CTU-13 e teste no ISOT-HTTP, com exceção do SVM, o qual teve uma pequena redução no F1-score (-2), os demais classificadores tiveram um aumento para esta métrica, com destaque para o AD (+38,9%), o MLP (+28,4%), o EFC (+4,8) e o RF (+4,4). Essa melhora nos resultados pode ser explicada devido ao fato do teste dos modelos ter sido executado no conjunto de dados ISOT HTTP, o qual foi a base para a seleção dos atributos. Em contra partida, com exceção do DT e do RF, os quais tiveram um aumento da AUC, o restante dos classificadores sofreram uma redução em relação à esta métrica. Por fim, o RF manteve o melhor F1-score (0.838 ± 0.025), assim como ocorreu no Estudo de Caso 1 e diferentemente do que ocorreu no Estudo de Caso 1, superou a AUC do EFC (0.714 ± 0.025).

Tabela 5.11: Desempenho Médio dos Classificadores - 25 Atributos EFC - CTU x ISOT

	Treino/Teste CTU-13		Treino CTU-13/Teste ISOT	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.653 ± 0.008	0.867 ± 0.002	0.351 ± 0.017	0.478 ± 0.017
KNN	0.989 ± 0.000	0.997 ± 0.000	0.364 ± 0.170	0.422 ± 0.170
DT	0.998 ± 0.000	0.998 ± 0.000	0.560 ± 0.134	0.459 ± 0.134
SVM	0.932 ± 0.000	0.985 ± 0.000	0.281 ± 0.320	0.687 ± 0.320
MLP	0.977 ± 0.001	0.998 ± 0.000	0.609 ± 0.006	0.690 ± 0.006
EFC	0.837 ± 0.042	0.932 ± 0.004	0.806 ± 0.001	0.697 ± 0.001
Ensemble				
RF	**0.999 ± 0.000**	**1.000 ± 0.000**	**0.838 ± 0.025**	**0.714 ± 0.025**
AD	0.965 ± 0.002	0.996 ± 0.000	0.651 ± 0.156	0.391 ± 0.156

A comparação dos resultados obtidos pelos classificadores, utilizando os 80 atributos iniciais e utilizando os 25 atributos selecionados pelo EFC, pode ser melhor visualizada na Figura 5.6. No gráfico do Teste Intra-Domínio, observa-se que não houve grande variação no desempenho dos classificadores, o que evidencia que os atributos selecionados pelo EFC fazem sentido para um conjunto de dados diferente do qual os atributos foram extraídos. No gráfico do Teste Inter-Domínio, nota-se que o EFC e o RF mantiveram a adaptabilidade dos modelos. Além disso, o MLP e DT obtiveram um melhor desempenho com a redução de atributos. Também observa-se um aumento significativo da métrica F1-score para o NB, KNN e AD, apesar da AUC desses classificadores ter sofrido uma redução. Apesar dessas variações, pode-se concluir que a redução de atributos não prejudicou o desempenho da maioria dos algoritmos.

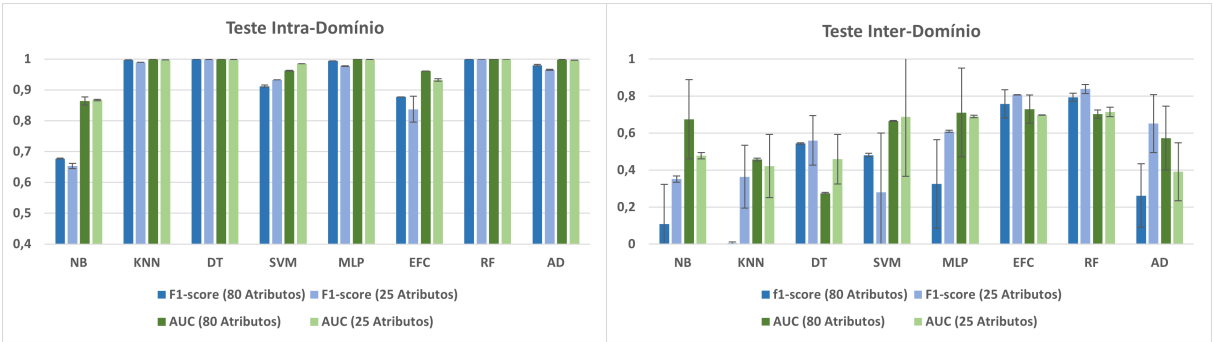


Figura 5.6: Comparação do Desempenho Médio dos Classificadores com 80 e 25 Atributos - CTU-13 x ISOT HTTP.

A Tabela 5.12 apresenta a comparação dos tempos de treinamento e teste com 80 e com 25 atributos. Novamente, com exceção do SVM, todos os classificadores tiveram uma redução nos tempos de treinamento utilizando os 25 atributos. Observa-se que o tempo de treinamento do EFC reduziu de ± 138 segundos para ± 18 segundos, evidenciando que a complexidade tanto do treinamento quanto do teste, são mais dependentes da quantidade de atributos de cada fluxo.

Tabela 5.12: Tempo Médio de Treino e Teste em Segundos - CTU-13 x ISOT HTTP

	80 Atributos		**25 atributos**	
Classificador	**Treino**	**Teste**	**Treino**	**Teste**
NB	0.240 ± 0.003	0.045 ± 0.000	0.087 ± 0.001	0.015 ± 0.000
DT	7.191 ± 0.235	0.013 ± 0.000	2.688 ± 0.163	0.008 ± 0.001
RF	12.042 ± 0.171	0.219 ± 0.005	9.441 ± 0.174	0.212 ± 0.006
AD	40.786 ± 0.250	0.699 ± 0.002	16.720 ± 0.670	0.445 ± 0.024
EFC	138.970 ± 2.732	1.436 ± 0.026	18.313 ± 0.049	0.147 ± 0.001
KNN	255.260 ± 18.017	113.486 ± 8.305	115.183 ± 0.434	17.318 ± 3.892
MLP	444.463 ± 59.629	0.151 ± 0.056	367.182 ± 103.945	0.229 ± 0.058
SVM	9862.926 ± 263.897	221.561 ± 29.827	14805.159 ± 1243.386	242.577 ± 100.042

Considerando ainda o conjunto de dados CTU-13 como base para o treinamento dos modelos, apresentamos na Tabela 5.13 os resultados obtidos pelos classificadores *One Class* que também foram utilizados no Estudo de Caso 1. Porém, neste experimento utilizamos os 25 atributos selecionados pelo EFC e fizemos uma análise comparativa aos resultados obtidos no Estudo de Caso 1 (Tabela 5.4).

Tabela 5.13: Desempenho Médio dos Classificadores *One Class* - 25 Atributos EFC - CTU-13 x ISOT HTTP

	Treino/Teste CTU-13		**Treino CTU-13/Teste ISOT**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
EFC	**0.837 ± 0.042**	**0.932 ± 0.004**	0.806 ± 0.001	**0.697 ± 0.001**
OCSVM	0.743 ± 0.001	0.734 ± 0.001	**0.903 ± 0.000**	0.430 ± 0.001
iForest	0.151 ± 0.007	0.537 ± 0.010	0.727 ± 0.009	0.686 ± 0.007
Elenv	0.103 ± 0.023	0.597 ± 0.005	0.410 ± 0.004	0.438 ± 0.023
LOF	0.729 ± 0.004	0.811 ± 0.022	0.886 ± 0.002	0.393 ± 0.004

Observa-se que no teste intra-domínio, i.e., treino e teste no mesmo dataset, com exceção do algoritmo iForest, os demais classificadores tiveram uma redução para a métrica F1-score, com destaque para o LOF (-19,4) e para o EFC (-4,2). Em relação à AUC, todos os algoritmos tiveram uma redução, sendo que as maiores perdas foram para o LOF (-17,2), Elenv (-10,8) e IForest (-5,8). O LOF foi o algoritmo que mais sofreu redução de desempenho com a seleção de atributos e diferentemente do que aconteceu no Estudo de Caso 1, o EFC obteve o melhor F1-score e a melhor AUC, superando os resultados do LOF.

No teste de domínio cruzado, i.e., treino no CTU-13 e teste no ISOT HTTP, com exceção do EFC que teve uma melhora em relação ao F1-score (+10,1), os demais classificadores tiveram resultados muito próximos aos obtidos utilizando os 80 atributos iniciais. Já em relação à métrica AUC, apesar do EFC e do LOF terem sofrido uma pequena redução (-3,9 e -4,2 respectivamente), os demais classificadores tiveram um aumento para esta métrica, com destaque para OCSVM (+9,2) e iForest (+5,8). Em geral, o OCSVM manteve o melhor F1-score e o EFC manteve a melhor AUC, assim como ocorreu no Estudo de Caso 1. De uma maneira geral, todos os algoritmos mantiveram a adaptabilidade com a utilização dos 25 atributos.

A comparação dos resultados obtidos pelos classificadores *One Class*, utilizando os 80 atributos iniciais e utilizando os 25 atributos selecionados pelo EFC, pode ser melhor visualizada na Figura 5.7.

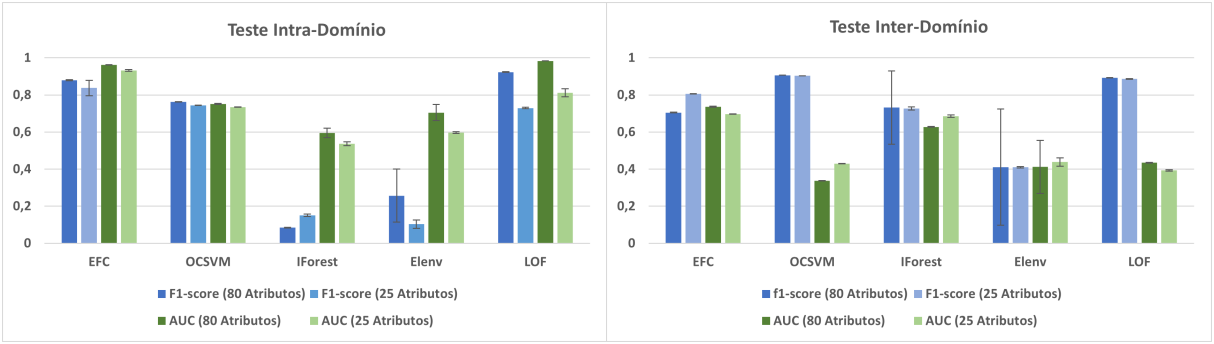


Figura 5.7: Comparação do Desempenho Médio dos Classificadores *One Class* com 80 e 25 Atributos - CTU-13 x ISOT HTTP.

Observa-se no gráfico do Teste Intra-Domínio que a redução de atributos não prejudicou o desempenho do EFC e do OCSVM, apesar de algumas variações. Por outro lado, a redução de atributos não favoreceu o LOF, o qual teve uma redução de desempenho considerável. O iForest e Elenv tiveram desempenho bem inferior aos demais algoritmos, independente da quantidade de atributos utilizada. Já no gráfico do Teste Inter-Domínio, observa-se que o desempenho de todos os algoritmos foi mantido próximo ao desempenho obtido com os 80 atributos iniciais. A maior variação no teste intra-domínio pode ser ex-

plicada pelo fato dos atributos terem sido selecionados utilizando um conjunto de dados diferente daquele onde foi realizado o treino e teste dos modelos.

A Tabela 5.14 apresenta a comparação dos tempos de treinamento e teste com 80 e 25 atributos. Observa-se que os tempos de treino e teste para todos os modelos foram reduzidos, utilizando os 25 atributos selecionados. Mesmo assim, o treinamento do modelo OCSVM levou aproximadamente 37 minutos, o que evidencia que este algoritmo pode não ser o mais indicado para trabalhar com grande quantidade de dados.

Tabela 5.14: Tempo Médio de Treino e Teste em Segundos - CTU-13 x ISOT - *One Class*

	80 Atributos		**25 atributos**	
Classificador	**Treino**	**Teste**	**Treino**	**Teste**
iForest	7.192 ± 0.137	11.797 ± 0.240	1.015 ± 0.150	3.285 ± 0.028
EFC	138.970 ± 2.732	1.436 ± 0.026	18.313 ± 0.049	0.147 ± 0.001
Elenv	171.943 ± 17.928	1.005 ± 0.042	22.932 ± 0.193	0.047 ± 0.000
LOF	1735.382 ± 36.629	1607.213 ± 21.935	117.311 ± 0.730	108.146 ± 1.965
OCSVM	16245.542 ± 683.388	6348.853 ± 178.356	2240.122 ± 86.430	818.896 ± 15.009

5.3.1.3 Resultados ISOT HTTP x ISCX-Bot-2014

A terceira abordagem consiste em utilizar um conjunto de dados diferente dos que foram utilizados no Estudo de Caso 1, para validar os atributos que foram selecionados a partir da análise dos acoplamentos calculados pelo algoritmo EFC. Sendo assim, vamos utilizar o conjunto de dados ISCX-Bot-2014. Primeiramente, fizemos os experimentos com os 80 atributos utilizados no Estudo de Caso 1, para posteriormente, compararmos os resultados utilizando os 25 atributos selecionados pelo EFC.

A Tabela 5.15 apresenta os resultados obtidos utilizando o conjunto de dados ISOT HTTP como base para realizar o treinamento dos modelos e o conjunto de dados ISCX-Bot-2014 para o teste da adaptabilidade dos modelos. Os resultados do teste intra-domínio, i.e., treino e teste com o ISOT HTTP, são os mesmos já apresentados no Estudo de Caso 1 (Tabela 5.1). A novidade é o resultado referente ao teste de adaptabilidade, no qual utilizamos o ISOT HTTP para treino e o ISCX-Bot-2014 para teste. Observa-se que o EFC obteve o melhor F1-score (0.733), enquanto os demais classificadores tiveram um desempenho muito ruim, com alguns classificadores (NB, AD e RF) zerando esta métrica, assim como aconteceu no Estudo de Caso 1. Isto já era esperado, uma vez que o treinamento foi realizado com o ISOT HTTP, o qual é um conjunto de dados bem restrito, com *botnets* de arquitetura centralizada, utilizando o protocolo HTTP somente, enquanto

o teste foi realizado em um conjunto de dados com *botnets* de diferentes arquiteturas (centralizada e descentralizada) e vários protocolos (HTTP, IRC e P2P).

Tabela 5.15: Desempenho Médio dos Classificadores - ISOT x ISCX-Bot - 80 Atributos

	Treino/Teste ISOT HTTP		**Treino ISOT/Teste ISCX-Bot**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.952 ± 0.000	0.799 ± 0.004	0.000 ± 0.000	0.500 ± 0.000
KNN	**0.999 ± 0.000**	0.997 ± 0.000	0.384 ± 0.002	0.531 ± 0.017
DT	**0.999 ± 0.000**	0.996 ± 0.000	0.038 ± 0.042	0.491 ± 0.024
SVM	0.989 ± 0.000	0.997 ± 0.000	0.226 ± 0.083	0.489 ± 0.083
MLP	0.994 ± 0.001	0.999 ± 0.000	0.107 ± 0.052	**0.554 ± 0.021**
EFC	0.987 ± 0.000	0.992 ± 0.000	**0.733 ± 0.000**	0.502 ± 0.003
Ensemble				
AD	**0.998 ± 0.001**	1.000 ± 0.000	0.000 ± 0.000	0.469 ± 0.020
RF	**0.999 ± 0.000**	1.000 ± 0.000	0.000 ± 0.000	0.470 ± 0.035

Agora que possuímos os resultados com os mesmos atributos utilizados no Estudo de Caso 1, podemos realizar os experimentos utilizando a seleção de atributos do EFC. Os resultados estão apresentados na Tabela 5.16.

Tabela 5.16: Desempenho Médio dos Classificadores - ISOT x ISCX-Bot - 25 Atributos

	Treino/Teste ISOT HTTP		**Treino ISOT/Teste ISCX-Bot**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.955 ± 0.001	0.869 ± 0.003	0.017 ± 0.002	0.497 ± 0.001
KNN	**0.996 ± 0.000**	0.997 ± 0.000	0.198 ± 0.004	0.522 ± 0.002
DT	**0.996 ± 0.000**	0.991 ± 0.001	0.027 ± 0.034	0.483 ± 0.037
SVM	0.984 ± 0.002	0.994 ± 0.004	0.361 ± 0.058	0.419 ± 0.008
MLP	0.993 ± 0.000	0.999 ± 0.000	0.004 ± 0.007	0.464 ± 0.104
EFC	0.988 ± 0.000	0.995 ± 0.000	**0.733 ± 0.000**	0.495 ± 0.004
Ensemble				
AD	0.994 ± 0.000	**1.000 ± 0.000**	0.092 ± 0.032	**0.547 ± 0.039**
RF	**0.996 ± 0.000**	**1.000 ± 0.000**	0.000 ± 0.000	0.498 ± 0.102

Observa-se que os resultados do teste intra-domínio. i.e., treino e teste no ISOT HTTP, ficaram muito parecidos com os resultados utilizando 80 atributos (Tabela 5.15), com exceção do NB, o qual teve um aumento na AUC (+7). Sendo assim, a redução dos atributos realizada através dos acoplamentos do EFC foi muito positiva para o teste intra-domínio. Já no teste inter-domínio, i.e., treino no ISOT e teste no ISCX-Bot-2014, houve uma pequena variação, tanto em relação ao F1 quanto à AUC, mas ainda assim, o EFC obteve o melhor desempenho, enquanto os demais classificadores continuaram obtendo um desempenho muito ruim. Em relação ao EFC, apesar do F1-score ter se mantido igual ao resultado com 80 atributos, a AUC sofreu uma pequena redução (-0,7).

A comparação dos resultados obtidos pelos classificadores utilizando os 80 atributos iniciais e utilizando os 25 atributos selecionados pelo EFC, pode ser melhor visualizada na Figura 5.8. Observa-se no gráfico do Teste Intra-Domínio que a redução de atributos não afetou o desempenho dos classificadores. No teste Inter-Domínio, o EFC conseguiu manter a adaptabilidade do modelo, mesmo com a redução de atributos.

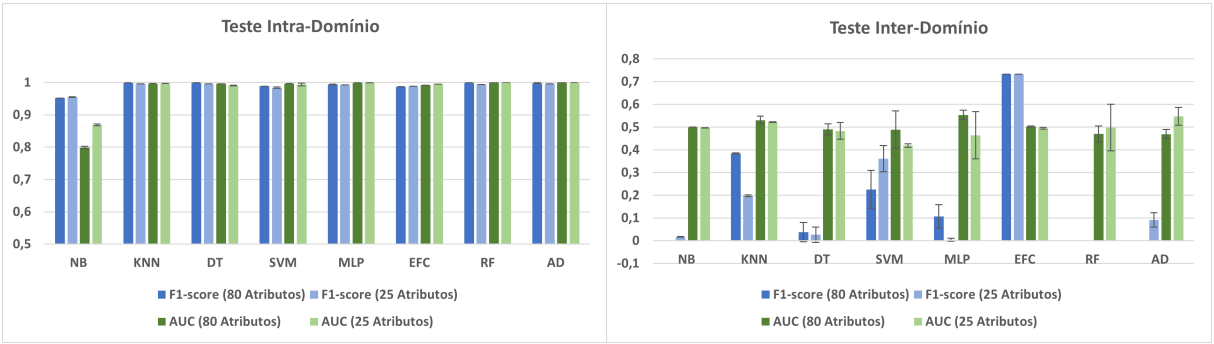


Figura 5.8: Comparação do Desempenho Médio dos Classificadores com 80 e 25 Atributos - ISOT HTTP x ISCX-Bot.

A Tabela 5.17 apresenta a comparação dos tempos de treinamento e teste com 80 e com 25 atributos. Observa-se que os algoritmos NB e DT tiveram o menor tempo de treino e teste, tanto com 80 atributos quanto com 25 atributos. Por outro lado, o KNN, MLP e SVM tiveram o maior tempo de treino e teste, sendo que o SVM apresentou tempos de treinamento e teste muito superiores aos demais, mesmo com a redução dos atributos. Observa-se novamente que a redução de atributos para o classificador EFC leva à uma redução considerável no tempo de treinamento do modelo.

Para comparação dos resultados obtidos pelos classificadores *One Class*, utilizando os conjuntos de dados ISCX-Bot e ISOT HTTP, primeiramente realizamos os experimentos com os 80 atributos utilizados no Estudo de Caso 1. Os resultados são apresentados na Tabela 5.18. Os resultados do teste intra-domínio, i.e., treino e teste no conjunto de dados ISOT-HTTP, são praticamente iguais aos resultados apresentados no Estudo de Caso 1 (Tabela 5.3). A novidade neste experimento é em relação ao teste inter-domínio,

Tabela 5.17: Tempo Médio de Treino e Teste em Segundos - ISOT x ISCX-Bot

	80 Atributos		**25 atributos**	
Classificador	**Treino**	**Teste**	**Treino**	**Teste**
NB	0.240 ± 0.003	0.045 ± 0.000	0.111 ± 0.011	0.017 ± 0.001
DT	7.191 ± 0.235	0.013 ± 0.000	1.498 ± 0.035	0.008 ± 0.000
RF	12.042 ± 0.171	0.219 ± 0.005	9.361 ± 0.184	0.212 ± 0.004
AD	40.786 ± 0.250	0.699 ± 0.002	13.132 ± 0.204	0.469 ± 0.010
EFC	138.970 ± 2.732	1.436 ± 0.026	3.501 ± 0.089	0.165 ± 0.002
KNN	357.042 ± 7.625	29.945 ± 2.255	255.260 ± 18.017	113.486 ± 8.305
MLP	367.182 ± 103.945	0.229 ± 0.058	171.323 ± 55.904	0.189 ± 0.021
SVM	18569.612 ± 3935.621	243.471 ± 4.413	13951.569 ± 1249.079	255.388 ± 28.015

i.e., treino no ISOT HTTP e teste no ISCX-Bot. Neste teste, os classificadores EFC, OCSVM e LOF obtiveram o mesmo valor de F1-score (0.733). Sobre a métrica AUC, o maior valor foi para o OCSVM (0.586), seguido pelo LOF (0.530) e pelo EFC (0.502). Por outro lado, o desempenho dos algoritmos iForest e Elenv foi bem inferior aos demais classificadores *One Class*.

Tabela 5.18: Desempenho Médio dos Classificadores *One Class* - 80 Atributos

	Treino/Teste ISOT HTTP		**Treino ISOT/Teste ISCX-Bot**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
EFC	**0.987 ± 0.000**	**0.992 ± 0.000**	**0.733 ± 0.002**	0.502 ± 0.003
OCSVM	0.736 ± 0.002	0.549 ± 0.001	**0.733 ± 0.000**	**0.586 ± 0.002**
iForest	0.674 ± 0.007	0.767 ± 0.023	0.581 ± 0.005	0.397 ± 0.007
Elenv	0.031 ± 0.017	0.785 ± 0.024	0.005 ± 0.005	0.386 ± 0.017
LOF	0.630 ± 0.016	0.719 ± 0.044	**0.733 ± 0.001**	0.530 ± 0.016

Em seguida, fizemos os experimentos utilizando os atributos selecionados pelo EFC, e então, realizamos a análise comparativa com os resultados obtidos no experimento anterior. A Tabela 5.19 apresenta os resultados obtidos com a seleção dos atributos. Os resultados do testes intra-domínio, i.e., treino e teste utilizando o conjunto de dados ISOT HTTP, são praticamente os mesmos já apresentados na Tabela 5.9. Observa-se que o EFC manteve os resultados obtidos com os 80 atributos iniciais. Por outro lado, o OCSVM e iForest sofreram uma redução no desempenho, enquanto o Elenv e o LOF tiveram um melhor desempenho com a redução de atributos.

Tabela 5.19: Desempenho Médio dos Classificadores *One Class* - 25 Atributos EFC

	Treino/Teste ISOT HTTP		**Treino ISOT/Teste ISCX-Bot**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
EFC	**0.988 ± 0.000**	**0.995 ± 0.000**	**0.733 ± 0.000**	0.495 ± 0.004
OCSVM	0.588 ± 0.003	0.291 ± 0.011	0.733 ± 0.000	**0.594 ± 0.003**
iForest	0.568 ± 0.091	0.727 ± 0.024	0.628 ± 0.008	0.539 ± 0.007
Elenv	0.629 ± 0.082	0.892 ± 0.002	0.562 ± 0.019	0.438 ± 0.082
LOF	0.761 ± 0.082	0.847 ± 0.032	0.703 ± 0.019	0.488 ± 0.082

A novidade neste experimento é em relação ao teste de domínio cruzado. Pode-se observar que em relação à métrica F1-score, os classificadores EFC e OCSVM mantiveram o mesmo resultado obtido no experimento com os 80 atributos iniciais. Por outro lado, o iForest e o Elenv obtiveram um aumento em relação à esta métrica (+4,7 e +55,7 respectivamente). Por fim, o F1-score do LOF sofreu uma pequena redução (-3). Sobre a métrica AUC, com exceção dos classificadoress EFC e LOF, os quais tiveram uma redução (-0,7 e -4,2 respectivamente), os demais classificadores tiveram um aumento para esta métrica, com destaque para iForest (+14,2) e Elenv (+5,2). De uma maneira geral, com exceção do OCSVM e iForest, os quais tiveram o desempenho bastante reduzido no teste intra-domínio, os demais classificadores tiveram resultados satisfatórios com a redução de atributos.

A comparação dos resultados obtidos pelos classificadores *One Class*, utilizando os 25 atributos selecionados pelo EFC e utilizando os 80 atributos iniciais, pode ser melhor visualizada na Figura 5.9.

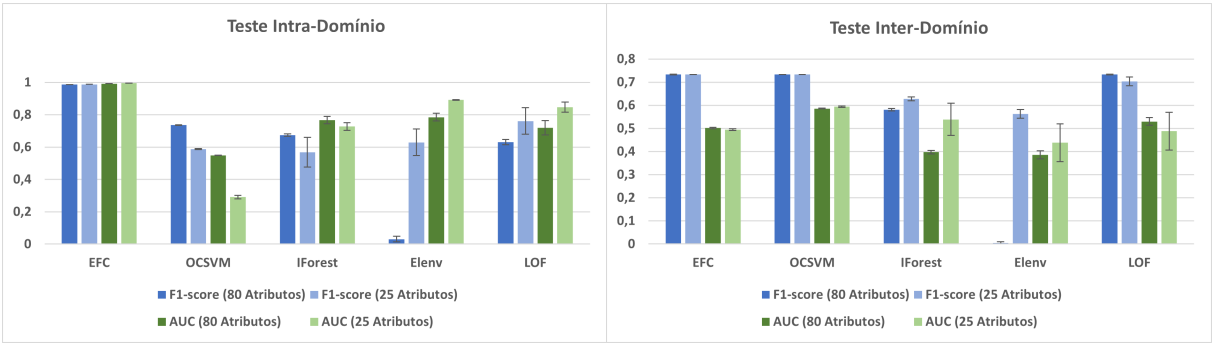


Figura 5.9: Comparação do Desempenho Médio dos Classificadores *One Class* com 80 e 25 Atributos - ISOT HTTP x ISCX-Bot.

O gráfico do Teste Intra-Domínio é igual ao da Figura 5.5, portanto, já foi discutido

anteriormente. No gráfico do Teste Inter-Domínio, observa-se que o EFC, OCSVM e LOF mantiveram a adaptabilidade dos modelos, mesmo com a redução de atributos. O destaque neste gráfico, é em relação ao desempenho do Elenv, o qual foi muito superior ao desempenho obtido com os 80 atributos iniciais, principalmente em relação à métrica F1-score. Por fim, o IForest também teve uma melhora em seu desempenho.

A Tabela 5.20 apresenta a comparação dos tempos de treinamento e teste com 80 atributos e com 25 atributos, considerando os classificadores *One Class*. Observa-se que o tempo para o treinamento do modelo OCSVM é muito superior ao tempo de treinamento dos demais modelos, mesmo com a redução de atributos (± 2,5 horas com 25 atributos). Em relação ao EFC, observa-se que o tempo de treinamento do modelo reduziu bastante com a seleção de atributos.

Tabela 5.20: Tempo Médio de Treino e Teste em Segundos - ISOT x ISCX-Bot - *One Class*

	80 Atributos		**25 atributos**	
Classificador	**Treino**	**Teste**	**Treino**	**Teste**
iForest	1.280 ± 0.116	6.133 ± 0.425	0.474 ± 0.033	3.559 ± 0.305
LOF	46.759 ± 3.601	136.296 ± 12.422	13.426 ± 1.297	39.594 ± 4.287
Elenv	47.200 ± 2.201	0.494 ± 0.016	6.771 ± 0.119	0.046 ± 0.003
EFC	138.970 ± 2.732	1.436 ± 0.026	3.501 ± 0.089	0.165 ± 0.002
OCSVM	18114.873 ± 952.081	3699.799 ± 111.119	9113.386 ± 1402.392	1546.364 ± 161.204

5.3.1.4 Resultados ISCX-Bot-2014 x ISOT HTTP

Nos próximos experimentos fizemos exatamente o inverso. Utilizamos o conjunto de dados ISCX-Bot-2014 como base para o treinamento dos modelos e o ISOT HTTP para testar a adaptabilidade dos modelos. Inicialmente fizemos os experimentos com os mesmos atributos utilizados no Estudo de Caso 1, para posteriormente, compararmos com os resultados obtidos utilizando os 25 atributos selecionados pelo EFC. A Tabela 5.21 ilustra os resultados dos experimentos com os 80 atributos do Estudo de Caso 1.

Observa-se que no teste intra-domínio, a maioria dos classificadores obteve F1-score e AUC acima de 99%. As exceções foram para o EFC (F1=0.858 e AUC=0.941) e para o NB (F1= 0.764 e AUC=0.746), mas ainda assim, os resultados obtidos pelo EFC são comparáveis aos resultados obtidos pelos outros classificadores. Já no teste inter-domínio, i.e., treino no ISCX-Bot-2014 e teste no ISOT HTTP, o EFC obteve o melhor desempenho (F1=0.900 e AUC=0.647), seguido pelo MLP (F1=0.801 e AUC=0.506) e pelo KNN

Tabela 5.21: Desempenho Médio dos Classificadores 80 atributos - ISCX-Bot x ISOT

	Treino/Teste ISCX-Bot		**Treino ISCX-Bot/Teste ISOT**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.764 ± 0.002	0.746 ± 0.057	0.537 ± 0.002	0.370 ± 0.002
KNN	0.998 ± 0.000	0.999 ± 0.000	0.708 ± 0.003	0.629 ± 0.003
DT	**0.999 ± 0.000**	0.999 ± 0.000	0.187 ± 0.135	0.373 ± 0.135
SVM	0.984 ± 0.000	0.996 ± 0.000	0.700 ± 0.007	0.584 ± 0.007
MLP	0.997 ± 0.000	**1.000 ± 0.000**	0.801 ± 0.007	0.506 ± 0.007
EFC	0.858 ± 0.003	0.941 ± 0.001	**0.900 ± 0.002**	**0.647 ± 0.002**
Ensemble				
AD	0.992 ± 0.000	0.999 ± 0.000	0.794 ± 0.133	0.416 ± 0.133
RF	**0.999 ± 0.000**	**1.000 ± 0.000**	0.249 ± 0.211	0.435 ± 0.211

(F1=0.708 e AUC=0.629). Por outro lado, o DT (F1=0.187 e AUC=0.373) e o RF (F1=0.249 e AUC=0.435) obtiveram o pior desempenho.

A partir dos experimentos realizados com os mesmos atributos utilizados no Estudo de Caso 1, fizemos então, os experimentos utilizando a seleção de atributos do EFC, mantendo o ISCX-Bot-2014 como base para o treinamento dos modelos e o ISOT HTTP para o teste de adaptabilidade dos modelos. Os resultados estão apresentados na Tabela 5.22.

Observa-se que no teste intra-domínio, i.e., treino com o ISCX-Bot-2014 e teste com o ISOT HTTP, a maioria dos classificadores obteve desempenho muito parecido ao obtido utilizando os 80 atributos iniciais. As exceções foram o NB, o qual teve uma redução em relação à métrica F1-score (-2,8) e o SVM, o qual sofreu uma redução tanto em relação ao F1-score (-5,1) quanto em relação à AUC (-2,2), assim como o EFC (-3,5 para o F1-score e -4,8 para AUC). Apesar dessas reduções de desempenho, de maneira geral os resultados são satisfatórios, uma vez que estamos testando os atributos que foram selecionados a partir de um conjunto de dados específico, em um conjunto de dados diferente.

No teste inter-domínio, houve variação positiva para alguns classificadores e variação negativa para outros. Mas no geral, o EFC manteve o melhor F1 score (0.906), assim como no experimento com 80 atributos. Da mesma forma, apesar de terem sofrido uma redução do F1-score, o MLP (0.737) e o KNN (0.610) seguem o EFC, com F1-score superior ao restante dos classificadores. Em relação à AUC, exceto o EFC e o KNN, os quais sofreram uma redução, os demais classificadores tiveram um aumento em relação à esta métrica,

Tabela 5.22: Desempenho Médio dos Classificadores 25 Atributos - ISCX-Bot x ISOT

	Treino/Teste ISCX-Bot		**Treino ISCX-Bot/Teste ISOT**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.736 ± 0.005	0.752 ± 0.002	0.566 ± 0.007	0.498 ± 0.007
KNN	0.997 ± 0.000	0.999 ± 0.000	0.610 ± 0.016	0.526 ± 0.016
DT	**0.998 ± 0.000**	0.998 ± 0.000	0.349 ± 0.209	0.411 ± 0.209
SVM	0.933 ± 0.001	0.974 ± 0.002	0.608 ± 0.000	0.523 ± 0.000
MLP	0.987 ± 0.002	0.998 ± 0.000	0.737 ± 0.148	0.530 ± 0.148
EFC	0.823 ± 0.003	0.893 ± 0.004	**0.906 ± 0.000**	**0.568 ± 0.000**
Ensemble				
AD	0.983 ± 0.009	0.997 ± 0.001	0.538 ± 0.008	0.488 ± 0.008
RF	**0.998 ± 0.000**	**1.000 ± 0.000**	0.547 ± 0.011	0.446 ± 0.011

mas ainda assim, o EFC obteve a melhor AUC (0.568).

A comparação dos resultados obtidos pelos classificadores utilizando os 25 atributos selecionados pelo EFC e utilizando os 80 atributos iniciais, pode ser melhor visualizada na Figura 5.10.

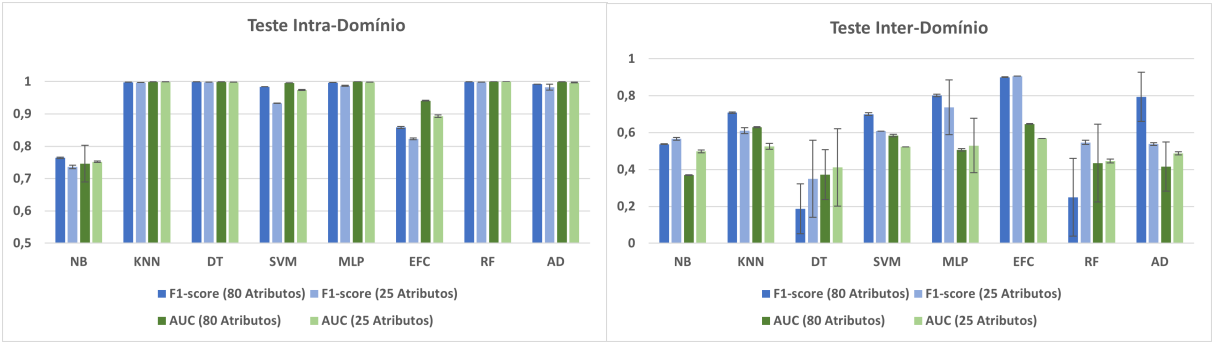


Figura 5.10: Comparação do Desempenho Médio dos Classificadores com 80 e 25 Atributos - ISCX-Bot x ISOT.

Observa-se no Gráfico do Teste Intra-Domínio, que a maioria dos classificadores mantiveram o mesmo desempenho obtido com os 80 atributos iniciais. Por outro lado, o EFC e o SVM foram os classificadores que tiveram uma pequena redução de desempenho. Já no gráfico do Teste Inter-Domínio, observa-se que o EFC manteve a adaptabilidade do modelo, mesmo com a redução de atributos. O SVM, KNN, e MLP apesar de terem sofrido uma pequena redução de desempenho, também obtiveram resultados próximos aos obtidos com os 80 atributos iniciais. Por fim, o AD foi o classificador que sofreu a maior

variação negativa, com uma redução de -25,6 em relação à métrica F1-score, enquanto o RF teve a maior variação positiva para esta métrica (+29,8). Apesar das variações nestes experimentos, no contexto geral, o resultado obtido com o conjunto de atributos selecionado pelo EFC foi bastante satisfatório, uma vez que nestes experimentos estamos utilizando os atributos que foram extraídos a partir de um conjunto de dados específico, em outro conjunto de dados.

A Tabela 5.23 apresenta a comparação dos tempos de treinamento e teste com 80 atributos e com 25 atributos. Observa-se que com exceção do SVM, os tempos de treinamento de todos os modelos reduziu significativamente com a seleção de atributos, sendo que o NB foi o algoritmo com menor tempo de treino e teste, independente da quantidade de atributos utilizados. Por outro lado, o SVM obteve o maior tempo de treinamento, mesmo com a redução dos atributos (± 3 horas).

Tabela 5.23: Tempo Médio de Treino e Teste em Segundos - ISCX-Bot x ISOT

	80 Atributos		**25 atributos**	
Classificador	**Treino**	**Teste**	**Treino**	**Teste**
NB	0.230 ± 0.038	0.037 ± 0.007	0.072 ± 0.019	0.013 ± 0.005
DT	5.867 ± 0.924	0.013 ± 0.001	1.563 ± 0.182	0.006 ± 0.002
RF	9.089 ± 1.609	0.199 ± 0.045	5.509 ± 0.494	0.158 ± 0.102
KNN	20.647 ± 6.131	46.343 ± 8.753	10.567 ± 2.420	13.996 ± 1.664
EFC	31.275 ± 0.571	0.953 ± 0.017	4.301 ± 0.598	0.106 ± 0.012
AD	31.616 ± 5.098	0.639 ± 0.121	11.966 ± 1.985	0.328 ± 0.057
MLP	689.532 ± 247.656	0.122 ± 0.040	582.095 ± 154.532	0.122 ± 0.005
SVM	7241.064 ± 1602.511	130.398 ± 10.637	10679.592 ± 398.539	220.665 ± 73.748

Para compararmos os resultados obtidos pelos classificadores *One Class* utilizados no Estudo de Caso 1, porém agora utilizando os conjuntos de dados ISCX-Bot-2014 e ISOT HTTP, foi necessário, primeiramente, realizar os experimentos com os 80 atributos iniciais, para então realizar os experimentos com os atributos selecionados pelo EFC. Os resultados com os 80 atributos também utilizados no Estudo de Caso 1 podem ser visualizados na Tabela 5.24, onde o conjunto de dados ISCX-Bot-2014 foi utilizado como base para o treinamento dos modelos e o ISOT HTTP foi utilizado para o teste de adaptabilidade.

No teste intra-domínio, i.e., treino e teste no mesmo conjunto de dados, observa-se que o EFC obteve o melhor desempenho (F1=0.858 e AUC=0.941), seguido pelo OCSVM (F1=0.844 e AUC=0.861) e pelo LOF (F1=0.801 e AUC=0.910). Por outro lado, o iForest e o Elenv apresentaram resultados muito inferiores. Já no teste de domínio cruzado, i.e., treino no ISCX-Bot-2014 e teste no ISOT HTTP, o Elenv obteve o melhor F1-score (0.914),

Tabela 5.24: Desempenho Médio dos Classificadores *One Class* - 80 Atributos - ISCX-Bot x ISOT

	Treino/Teste ISCX-Bot		**Treino ISCX-Bot/Teste ISOT**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
EFC	**0.858 ± 0.003**	**0.941 ± 0.001**	0.900 ± 0.002	**0.647 ± 0.002**
OCSVM	0.844 ± 0.002	0.861 ± 0.006	0.905 ± 0.000	0.389 ± 0.000
iForest	0.121 ± 0.007	0.589 ± 0.066	0.906 ± 0.000	0.444 ± 0.000
Elenv	0.509 ± 0.197	0.796 ± 0.118	**0.914 ± 0.010**	0.600 ± 0.010
LOF	0.801 ± 0.001	0.910 ± 0.023	0.901 ± 0.000	0.301 ± 0.000

seguido dos demais classificadores, os quais alcançaram ± 0.900 para esta métrica. Em relação à AUC, o EFC obteve o melhor desempenho (0.647), seguido pelo Elenv (0.600). Por outro lado, a AUC obtida pelos demais classificadores foi bastante inferior.

Por fim, fizemos os experimentos utilizando os 25 atributos selecionados pelo EFC, para então, realizar a análise comparativa com os resultados obtidos no experimento anterior. Os resultados desta abordagem podem ser visualizados na Tabela 5.25. No teste intra-domínio, i.e., treino e teste no ISCX-Bot-2014, com exceção do iForest, que teve um melhora discreta em relação à metrica F1-score (+2,8), os demais classificadores tiveram uma redução para esta métrica, com destaque para Elenv (-40,2), OCSVM (-8,6) e LOF (-6,9). O mesmo ocorreu para a métrica AUC, onde o IForest também obteve uma melhora (+4,1), enquanto os demais classificadores tiveram uma redução, com destaque para Elenv (-17,3), OCSVM (-9,8) e EFC (-4,8). Apesar dessas reduções, o EFC obteve o melhor desempenho, assim como no experimento com os 80 atributos iniciais (Tabela 5.24), seguido pelo OCSVM e LOF. No teste de domínio cruzado, i.e., treino no ISCX-Bot e teste no ISOT HTTP, o classificador Elenv obteve uma redução considerável em relação ao F1-score (-34,9), apesar de ter obtido um pequeno aumento da AUC (+4,3). O F1-score dos demais classificadores ficou muito próximo ao obtido com 80 atributos. Em relação à métrica AUC, todos os classificadores tiveram um aumento, com destaque para LOF (+22,5), OCSVM (+13,9) e iForest (+5,6).

A comparação dos resultados obtidos pelos classificadores *One Class* utilizando os 25 atributos selecionados pelo EFC e utilizando os 80 atributos iniciais, pode ser melhor visualizada na Figura 5.11. No gráfico do Teste Intra-Domínio podemos visualizar que apesar de uma pequena redução no desempenho, o EFC e LOF obtiveram resultados próximos aos obtidos com os 80 atributos iniciais. O iForest obteve um desempenho ligeiramente superior com a redução de atributos, porém ainda assim, o seu desempenho foi bem in-

Tabela 5.25: Desempenho Médio dos Classificadores *One Class* - 25 Atributos EFC - ISCX-Bot x ISOT

	Treino/Teste ISCX-Bot		**Treino ISCX-Bot/Teste ISOT**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
EFC	**0.823 ± 0.003**	**0.893 ± 0.004**	**0.906 ± 0.000**	**0.668 ± 0.000**
OCSVM	0.758 ± 0.004	0.763 ± 0.003	0.906 ± 0.003	0.528 ± 0.003
iForest	0.149 ± 0.019	0.630 ± 0.014	0.906 ± 0.000	0.500 ± 0.000
Elenv	0.107 ± 0.095	0.623 ± 0.095	0.565 ± 0.086	0.643 ± 0.086
LOF	0.732 ± 0.099	0.866 ± 0.032	0.893 ± 0.015	0.526 ± 0.015

ferior ao desempenho da maioria dos classificadores. Por fim, o desempenho do Elenv foi bastante reduzido utilizando os atributos selecionados pelo EFC e o OCSVM também teve uma redução significativa, principalmente em relação à métrica AUC. No gráfico do Teste Inter-Domínio, podemos verificar que com exceção do Elenv, os demais classificadores mantiveram o desempenho obtido com os 80 atributos iniciais, com inclusive uma melhora da métrica AUC. Sendo assim, com exceção do Elenv, os demais classificadores mantiveram a adaptabilidade com a utilização dos atributos selecionados pelo EFC.

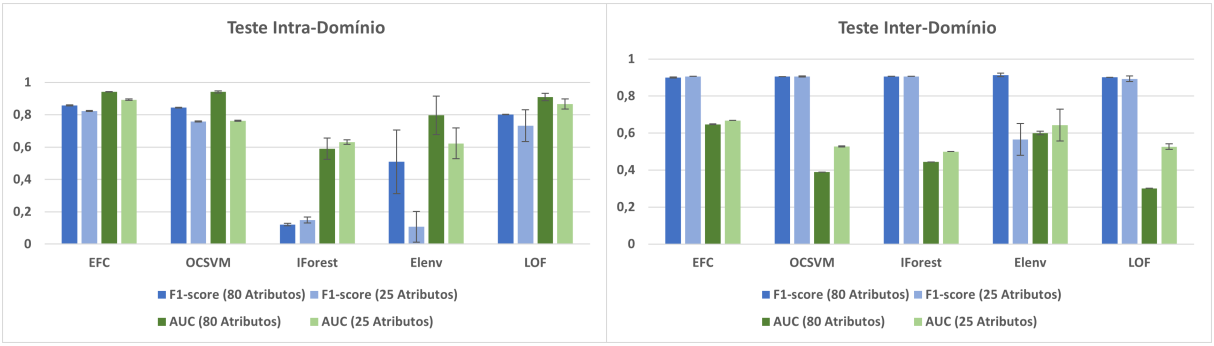


Figura 5.11: Comparação do Desempenho Médio dos Classificadores *One Class* com 80 Atributos e com 25 Atributos - ISCX-Bot x ISOT.

A Tabela 5.26 apresenta a comparação dos tempos de treinamento e teste com 80 atributos e com 25 atributos, considerando os classificadores *One Class*. Observa-se que o iForest apresentou o menor tempo para treinamento do modelo, independente da quantidade de atributos utilizados, seguido pelo EFC, o qual obteve um tempo de treinamento com 25 atributos bem inferior ao obtido com 80 atributos. Por outro lado, o OCSVM apresentou o maior tempo tanto para o treinamento quanto para o teste do modelo, mesmo com a redução de atributos.

Tabela 5.26: Tempo Médio de Treino e Teste em Segundos - ISCX-Bot x ISOT - *One Class*

	80 Atributos		**25 atributos**	
Classificador	**Treino**	**Teste**	**Treino**	**Teste**
iForest	1.971 ± 0.041	4.016 ± 0.083	0.534 ± 0.054	2.290 ± 0.173
EFC	31.275 ± 0.571	0.953 ± 0.017	4.301 ± 0.598	0.106 ± 0.012
Elenv	76.605 ± 0.817	0.308 ± 0.005	13.359 ± 0.177	0.033 ± 0.001
LOF	112.202 ± 0.872	139.423 ± 7.297	31.186 ± 2.810	38.036 ± 3.971
OCSVM	1208.025 ± 163.791	724.559 ± 23.922	446.070 ± 30.456	429.109 ± 158.775

Com base nos inúmeros experimentos realizados até o momento, percebe-se que os 25 atributos selecionados pelo EFC, através da análise dos acoplamentos entre os pares de atributos, utilizando o conjunto de dados ISOT HTTP como referência, se mostraram capazes de manter resultados próximos aos alcançados com os 80 atributos iniciais, principalmente nos testes em que o ISOT HTTP foi utilizado como base para o treinamento dos modelos.

Nos demais testes, onde o CTU-13 e o ISCX-Bot-2014 foram utilizados como base para o treinamento, houve algumas variações positivas ou negativas para determinados classificadores, mas ainda assim, no contexto geral, consideramos que os atributos trabalhados nestes experimentos são bastante representativos para identificação de atividades de *botnets*. Além disso, dependendo da quantidade de dados a serem submetidos para o treinamento, uma pequena redução no desempenho pode ser aceita, considerando que o treinamento da maioria dos classificadores utilizando uma menor quantidade de atributos, consumirá menos recurso computacional e consequentemente, levará menos tempo. Os próximos experimentos terão o objetivo de avaliar outros conjuntos de atributos e comparar os resultados obtidos com os atributos selecionados pelo EFC.

5.3.2 Atributos Utilizados no Estudo de Referência [1]

No intuito de comparar a eficiência dos atributos selecionados pelo EFC, pesquisamos na literatura estudos que fizeram a seleção de atributos a partir do conjunto de atributos extraídos com a ferramenta *cicflowmeter*, a qual foi utilizada na presente pesquisa. Sendo assim, vamos utilizar o estudo de Ramos et al. [1] como base, uma vez que nesse estudo também foi utilizado o conjunto de dados ISOT HTTP para seleção de parte dos atributos. Além disso, alguns dos algoritmos que utilizamos em nossos experimentos, também foram

utilizamos no trabalho de referência, i.e., *Decision Tree (DT), Random Forest (RF), Naive Bayes (NB), k-Nearest Neighbors (KNN) e Support Vector Machine (SVM).*

Ramos et al. [1] utilizou 20 atributos no total, os quais foram selecionados considerando pesquisas anteriores de outros autores. Estes atributos foram divididos em dois grupos. O primeiro grupo contém oito atributos, os quais foram escolhidos seguindo a metodologia apresentada por Sharafaldin et al. [116], onde foi utilizado o algoritmo *Random Forest Regressor* para obter parte das métricas comportamentais descritas na Tabela 5.27.

O segundo grupo, composto por nove atributos, foi selecionado baseado na metodologia apresentada por Gonzalez-Cuautle et al. [117], onde foi utilizado o conjunto de dados ISOT HTTP Botnet como referência para seleção daqueles atributos considerados mais importantes por diversos autores, considerando o estado da arte. Por fim, Ramos et al. [1] selecionaram mais dois atributos utilizando o critério *"feature_importances"* do algoritmo *Random Forest* e ainda incluíram o atributo "Protocolo", por considerar que este atributo poderia trazer informações revelantes.

A Tabela 5.27 ilustra todos os atributos que foram utilizados no estudo de Ramos et al. [1]. Estes atributos serão a base para os experimentos que serão apresentados a seguir, sendo compostos por três abordagens distintas. Na primeira abordagem, apresentamos os resultados obtidos utilizando o conjunto de dados ISOT HTTP como referência, já que foi a partir deste conjunto de dados que grande parte dos atributos foram selecionados. Realizamos também, a análise da adaptabilidade dos modelos, utilizando o ISOT HTTP para treinamento dos modelos e o CTU-13 para teste. Na segunda abordagem, fizemos exatamente o contrário, i.e. utilizamos o CTU-13 como base para treinamento dos modelos e testamos a adaptabilidade dos modelos utilizando o ISOT HTTP para teste. Por fim, a terceira abordagem utiliza o conjunto de dados ISCX-Bot-2014 como base para o treinamento dos modelos e o ISOT HTTP para o teste de adaptabilidade e em seguida faz o inverso, utilizando o ISOT HTTP para o treinamento dos modelos e o ISCX-Bot-2014 para o teste de domínio cruzado. Os resultados obtidos com todas essas abordagens serão comparados aos resultados obtidos utilizando os atributos selecionados pelo EFC.

5.3.2.1 Resultados ISOT HTTP x CTU-13

A Tabela 5.28 apresenta os resultados da primeira abordagem, na qual utilizamos o conjunto de dados ISOT HTTP como base para o treinamento dos modelos e o CTU-13 para o teste de adaptabilidade. Está sinalizado em azul os resultados que tiveram um aumento igual ou maior a 1%, em relação aos resultados obtidos com a seleção de atributos do EFC. Por outro lado, sinalizamos em vermelho os resultados com decréscimo maior ou igual a 1%.

Tabela 5.27: Atributos Utilizados no Estudo de Referência [1]

Tipo	**Atributo**	**Descrição**
Descritivos	SourcePort	Porta de origem
	DestinationPort	Porta de destino
	Protocol	Protocolo
Comportamentais	FlowDuration	Duração do fluxo em microssegundos
	TotalForwardPackets	Total de pacotes transmitidos na direção direta
	TotalBackwardPackets	Total de pacotes transmitidos na direção inversa
	TotalLengthofBwdPackets	Tamanho total dos pacotes transmitidos na direção inversa
	TotalLengthofFwdPackets	Tamanho total dos pacotes transmitidos na direção direta
	FwdPacketLengthMax	Tamanho máximo do pacote na direção direta
	FwdPacketLengthMin	Tamanho mínimo do pacote na direção direta
	FwdPacketLengthMean	Tamanho médio dos pacotes transmitidos na direção direta
	BwdPacketLengthMean	Tamanho mínimo dos pacotes transmitidos na direção direta
	FlowBytes-s	Número de bytes em um fluxo por segundo
	FlowPackets-s	Número de pacotes de fluxo por segundo
	FwdPackets-s	Taxa de pacotes transferidos por segundo na direção direta
	BwdPackets-s	Taxa de pacotes transferidos por segundo na direção inversa
	SubflowFwdBytes	O número médio de bytes em um subfluxo na direção direta
	SubflowBwdBytes	O número médio de bytes em um subfluxo na direção inversa
	SubflowBwdPackets	O número médio de pacotes em um subfluxo na direção inversa
	SubflowFwdPackets	O número médio de pacotes em um subfluxo na direção direta

No teste intra-domínio, as variações tanto em relação à métrica F1-score quanto em relação à métrica AUC foram mínimas, com resultados muito próximos aos obtidos utilizando os 25 atributos selecionados pelo EFC (Tabela 5.7). A maior variação foi para o NB, o qual teve uma pequena redução (-1,5) na métrica F1-score. No teste inter-domínio, i.e., treino no ISOT e teste no CTU-13, houve uma variação negativa para a maioria dos

Tabela 5.28: Desempenho Médio dos Classificadores - ISOT x CTU-13 - Atributos Segundo Ramos et al. [1]

	Treino/Teste ISOT HTTP		**Treino ISOT/Teste CTU-13**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.940 ± 0.005	0.870 ± 0.006	0.021 ± 0.004	0.452 ± 0.004
KNN	0.997 ± 0.001	0.995 ± 0.001	0.051 ± 0.002	0.512 ± 0.002
DT	**0.999 ± 0.000**	0.997 ± 0.000	0.002 ± 0.000	0.498 ± 0.000
SVM	0.983 ± 0.003	0.996 ± 0.002	0.153 ± 0.024	0.628 ± 0.024
MLP	0.994 ± 0.000	0.999 ± 0.000	0.020 ± 0.007	**0.697 ± 0.007**
EFC	0.986 ± 0.000	0.996 ± 0.000	**0.657 ± 0.003**	0.547 ± 0.007
Ensemble				
RF	**0.999 ± 0.000**	**1.000 ± 0.000**	0.000 ± 0.000	0.455 ± 0.000
AD	0.998 ± 0.000	1.000 ± 0.000	0.000 ± 0.001	0.410 ± 0.001

classificadores, seja em relação ao F1-score ou em relação à AUC, com destaque para os classificadores SVM e MLP, os quais tiveram uma redução significativa em relação à métrica F1-score (-17,6 e -11,9 respectivamente). Porém, de maneira geral, o EFC manteve o melhor F1-score (0.657) e o MLP manteve a melhor AUC (0.697), assim como aconteceu nos experimentos utilizando os 25 atributos selecionados pelo EFC. Concluiu-se, portanto, que utilizando os atributos selecionados pelo EFC, obtivemos resultados ligeiramente superiores, principalmente no teste de domínio cruzado.

A comparação dos resultados obtidos pelos classificadores utilizando os 25 atributos selecionados pelo EFC e utilizando os 20 atributos selecionados no Estudo de Referência [1], pode ser melhor visualizada na Figura 5.12. Observa-se pelos gráficos que os resultados do Teste Intra-Domínio ficaram muito próximos. No Teste Inter-Domínio, o EFC manteve a adaptabilidade, apesar de uma pequena redução na métrica AUC. Houve uma redução também para o SVM, em relação à métrica F1-score, mas no geral, podemos concluir que o conjunto de atributos selecionados pelo EFC, são tão representativos de tráfego de *botnets*, quanto os atributos selecionados considerando o Estado da Arte.

5.3.2.2 Resultados CTU-13 x ISOT HTTP

A Tabela 5.29 apresenta os resultados da segunda abordagem, na qual utilizamos o conjunto de dados CTU-13 como base para o treinamento dos modelos. No teste intra-domínio, o NB teve uma pequena melhora em relação ao F1-score (+1,1), mas obteve

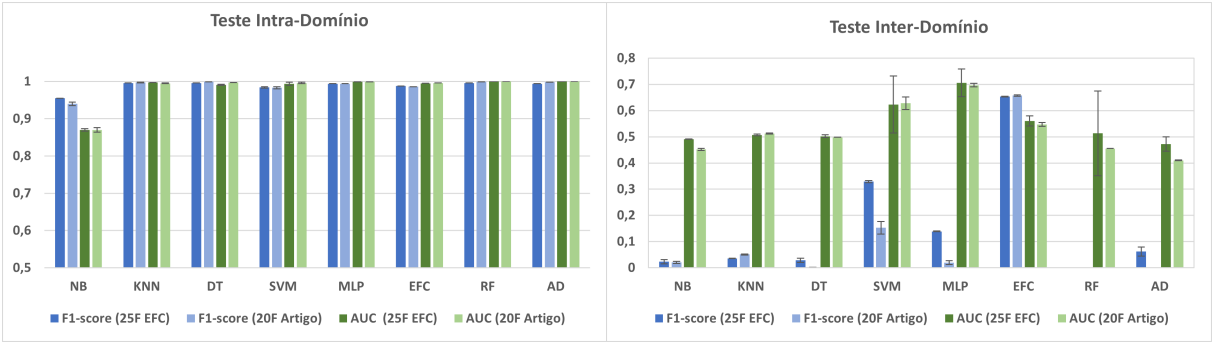


Figura 5.12: Comparação do Desempenho Médio dos Classificadores com Atributos do EFC e Atributos Segundo Ramos et al. [1] - ISOT x CTU-13.

uma redução significativa em relação à métrica AUC (-11,6), quando comparado com os resultados obtidos utilizando os atributos selecionados pelo EFC (Tabela 5.11). Além disso, o EFC e o SVM obtiveram uma redução para o F1-score (-6,4 e -10,5 respectivamente) e ainda, o SVM teve uma redução também na métrica AUC (-6). Para o restante dos classificadores, a variação foi mínima, não ultrapassando o limite estabelecido de 1% para a comparação dos resultados.

Tabela 5.29: Desempenho Médio dos Classificadores - CTU-13 x ISOT - Atributos Segundo Ramos et al. [1]

	Treino/Teste CTU-13		**Treino CTU-13/Teste ISOT**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.664 ± 0.006	0.751 ± 0.002	0.108 ± 0.212	0.265 ± 0.212
KNN	0.981 ± 0.001	0.994 ± 0.000	0.105 ± 0.038	0.467 ± 0.038
DT	**0.995 ± 0.000**	**0.996 ± 0.000**	0.458 ± 0.318	0.537 ± 0.318
SVM	0.827 ± 0.001	0.925 ± 0.001	0.104 ± 0.013	0.639 ± 0.013
MLP	0.962 ± 0.009	0.993 ± 0.003	0.652 ± 0.243	0.583 ± 0.243
EFC	0.773 ± 0.001	0.930 ± 0.001	0.699 ± 0.003	**0.703 ± 0.003**
Ensemble				
RF	**0.996 ± 0.000**	**1.000 ± 0.000**	**0.714 ± 0.006**	0.588 ± 0.006
AD	0.953 ± 0.001	0.990 ± 0.000	0.387 ± 0.323	0.556 ± 0.323

No teste inter-domínio, i.e., treino no CTU-13 e teste no ISOT HTTP, com exceção do MLP, o qual obteve um aumento em relação à métrica F1-score (+4,3), os demais classificadores tiveram uma redução para esta métrica, com destaque para AD (-26,4), KNN (-25,9) e NB (-24,3). Por outro lado, alguns classificadores tiveram um aumento em

relação à métrica AUC, com destaque para AD (+16,5) e DT (+7,8), enquanto outros tiveram uma redução em relação à esta métrica, como por exemplo o NB (-21,3) e RF (-12,6). Apesar dessas variações, o RF manteve o melhor F1-score, assim como aconteceu nos experimentos utilizando os atributos selecionados pelo EFC. Porém, diferentemente, o EFC obteve a melhor AUC (0.703), superando a AUC obtida pelo RF. Novamente, os resultados utilizando os atributos selecionados pelo EFC, superaram a maioria dos resultados obtidos utilizando os atributos do estudo de referência.

A comparação dos resultados obtidos pelos classificadores utilizando os 25 atributos selecionados pelo EFC e utilizando os 20 atributos selecionados no Estudo de Referência, pode ser melhor visualizada na Figura 5.13. No gráfico do Teste Intra-Domínio, observamos que para o KNN, DT, MLP, RF e AD o desempenho obtido foi muito próximo do desempenho com os atributos selecionados pelo EFC. Porém, o NB, SVM e EFC tiveram um desempenho menor utilizando os atributos propostos por Ramos et al. [1]. No gráfico do Teste Inter-Domínio, observamos que todos os classificadores, com exceção do MLP, tiveram uma redução da métrica F1-score utilizando os atributos propostos no Estudo de Referência [1]. Em relação à métrica AUC, alguns modelos sofreram uma redução (NB, SVM, MLP e RF), enquanto outros tiveram um aumento (KNN, DT e AD). Considerando os dois testes (intra e inter-domínio), podemos concluir que no contexto geral, os classificadores tiveram melhor desempenho utilizando os atributos selecionados pelo EFC.

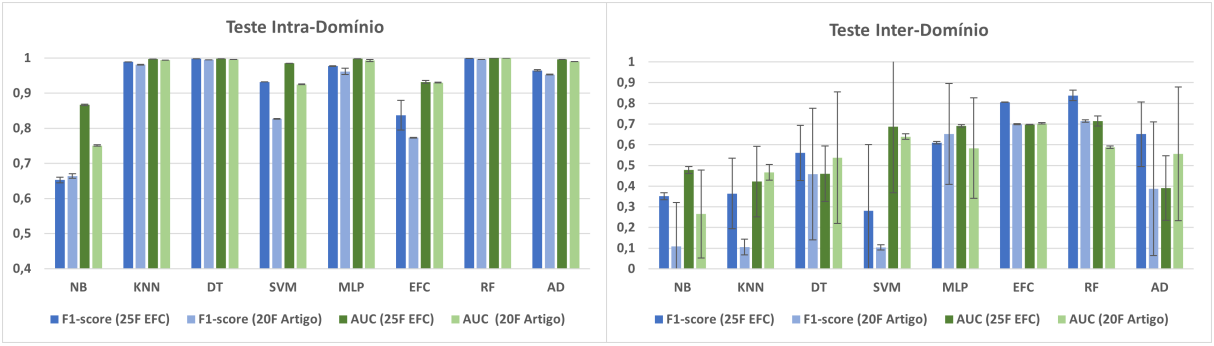


Figura 5.13: Comparação do Desempenho Médio dos Classificadores com Atributos do EFC e Atributos Segundo Ramos et al. [1] - CTU-13 x ISOT.

5.3.2.3 Resultados ISOT HTTP x ISCX-Bot-2014

Ainda utilizando os atributos obtidos no Estudo de Referência [1], realizamos experimentos utilizando o conjunto de dados ISCX-Bot-2014, assim como foi feito nos experimentos com os atributos selecionados pelo EFC. Na Tabela 5.30 apresentamos os resultados obtidos utilizando primeiramente, o conjunto de dados ISOT HTTP como base para o treinamento dos modelos e o ISCX-Bot-2014 para o teste de adaptabilidade. Em seguida,

fizemos a comparação dos resultados obtidos com os resultados utilizando os atributos selecionados pelo EFC.

Tabela 5.30: Desempenho Médio dos Classificadores - Atributos Segundo Ramos et al. [1] - ISOT x ISCX-Bot

	Treino/Teste ISOT HTTP		**Treino ISOT/Teste ISCX-Bot**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.936 ± 0.013	0.875 ± 0.010	0.001 ± 0.000	0.485 ± 0.000
KNN	0.998 ± 0.002	0.996 ± 0.003	0.006 ± 0.008	0.440 ± 0.008
DT	**0.999 ± 0.000**	0.997 ± 0.001	0.002 ± 0.001	0.500 ± 0.001
SVM	0.983 ± 0.003	0.996 ± 0.002	0.207 ± 0.307	0.377 ± 0.307
MLP	0.994 ± 0.001	0.999 ± 0.000	0.089 ± 0.174	0.670 ± 0.174
EFC	0.986 ± 0.000	0.996 ± 0.001	**0.731 ± 0.003**	0.391 ± 0.004
Ensemble				
AD	0.998 ± 0.000	**1.000 ± 0.000**	0.000 ± 0.000	0.539 ± 0.000
RF	**0.999 ± 0.000**	**1.000 ± 0.000**	0.001 ± 0.000	0.391 ± 0.000

Os resultados obtidos no teste intra-domínio, i.e., treino e teste utilizando o conjunto de dados ISOT-HTTP já são conhecidos, uma vez que já foram apresentados na Tabela 5.28. A novidade é em relação ao teste de domínio cruzado, no qual utilizamos o conjunto de dados ISOT HTTP para o treinamento e o conjunto de dados ISCX-Bot-2014 para o teste dos modelos. Observa-se que com exceção do EFC, o qual manteve o valor de F1-score alcançado com os atributos selecionados pelo EFC (Tabela 5.16), a maioria dos classificadores sofreu uma redução para esta métrica. A AUC da maioria dos classificadores também reduziu com a utilização dos atributos propostos por Ramos et al. [1]. No contexto geral, apesar da redução em relação à métrica AUC (-10,4), o EFC obteve o melhor desempenho, enquanto os demais classificadores obtiveram desempenhos muito ruins, assim como aconteceu nos experimentos utilizando os atributos selecionados pelo EFC. Isso se deve ao fato do treinamento ter sido realizado em um conjunto de dados bem restrito, o qual possui *botnets* de arquitetura centralizada somente, enquanto o teste dos modelos foi realizado em um conjunto de dados mais variado, contendo *botnets* de arquitetura centralizada e descentralizada.

A comparação dos resultados obtidos pelos classificadores utilizando os 25 atributos selecionados pelo EFC e utilizando os 20 atributos selecionados no Estudo de Referência, pode ser melhor visualizada na Figura 5.14. Observa-se no Gráfico do Teste Intra-Domínio que o desempenho dos classificadores foi praticamente igual ao desempenho ob-

tido utilizando os atributos selecionados pelo EFC. Já no Gráfico do Teste Inter-Domínio, observa-se que o EFC manteve a adaptabilidade do modelo, apesar de ter sofrido uma redução em relação à métrica AUC. Todos os outros classificadores tiveram um desempenho muito ruim, assim como aconteceu nos experimentos utilizando os atributos selecionados pelo EFC, mas ainda assim, percebe-se que o KNN e o SVM tiveram um desempenho um pouco melhor utilizando os atributos do EFC. Dessa forma, podemos concluir que os atributos selecionados pelo EFC geraram resultados ligeiramente superiores, principalmente no teste inter-domínio.

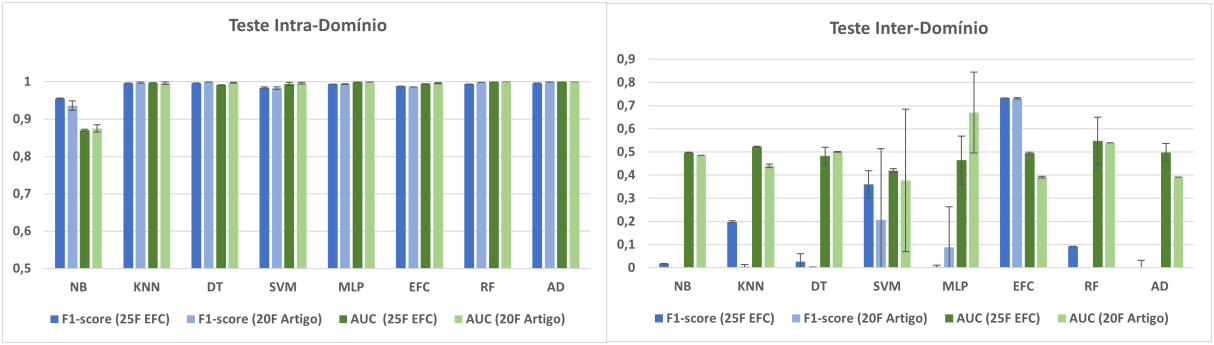


Figura 5.14: Comparação do Desempenho Médio dos Classificadores com Atributos do EFC e Atributos Segundo Ramos et al. [1] - ISOT x ISCX-Bot.

5.3.2.4 Resultados ISCX-Bot-2014 x ISOT HTTP

A última abordagem com os atributos obtidos no Estudo de Referência, foi exatamente o inverso do experimento anterior, i.e., utilizamos o conjunto de dados ISCX-Bot-2014 como base para o treinamento dos modelos e o ISOT HTTP para o teste de domínio cruzado. Os resultados obtidos podem ser visualizados na Tabela 5.31.

No teste intra-domínio, alguns classificadores mantiveram os resultados para a métrica F1-score, enquanto outros sofreram uma redução, com destaque para EFC (-8,8) e AD (-4,8). Em relação à AUC, o NB obteve um aumento para esta métrica (+7,3), enquanto houve uma redução para o SVM (-2,9), EFC (-2,4) e AD (-1,8). Para os demais classificadores, os resultados se mantiveram muito próximos aos obtidos utilizando os atributos selecionados pelo EFC (Tabela 5.22).

No teste de domínio cruzado, houve uma redução em relação à métrica F1-score para o NB (-2,3) e DT (-9,7), enquanto para os demais classificadores houve um aumento desta métrica, com destaque para AD (+14,3), KNN (+7,2) e MLP (+6,7). Em relação à métrica AUC, houve uma variação positiva para o MLP (+26) e para o RF (+5,3). Para o restante dos classificadores houve uma redução, com destaque para AD (-21,2), NB (-8,8) e EFC (-5,7). Apesar dessas variações, o EFC manteve o melhor F1-score, assim como

Tabela 5.31: Desempenho Médio dos Classificadores - Atributos Segundo Ramos et al. [1] - ISCX-Bot x ISOT

	Treino/Teste ISCX-Bot		Treino ISCX-Bot/Teste ISOT	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.728 ± 0.000	0.825 ± 0.003	0.542 ± 0.003	0.410 ± 0.003
KNN	0.980 ± 0.000	0.994 ± 0.000	0.682 ± 0.001	0.521 ± 0.001
DT	0.991 ± 0.000	0.990 ± 0.000	0.252 ± 0.328	0.370 ± 0.328
SVM	0.904 ± 0.001	0.945 ± 0.001	0.613 ± 0.000	0.508 ± 0.000
MLP	0.965 ± 0.001	0.991 ± 0.000	0.804 ± 0.004	**0.790 ± 0.004**
EFC	0.735 ± 0.003	0.869 ± 0.002	**0.906 ± 0.001**	0.511 ± 0.002
Ensemble				
AD	0.935 ± 0.005	0.979 ± 0.001	0.681 ± 0.218	0.276 ± 0.218
RF	**0.993 ± 0.000**	**1.000 ± 0.000**	0.553 ± 0.019	0.499 ± 0.019

no experimento utilizando os atributos do EFC. Porém, diferentemente, o MLP obteve a melhor AUC. No contexto geral, os resultados obtidos com os atributos selecionados pelo EFC foram superiores aos resultados obtidos com os atributos extraídos do artigo de referência, para a maioria dos classificadores. As exceções foram os algoritmos MLP e RF, que no teste de domínio cruzado obtiveram resultado superior para ambas as métricas (F1-score e AUC).

A comparação dos resultados obtidos pelos classificadores utilizando os 25 atributos selecionados pelo EFC e utilizando os 20 atributos propostos no Estudo de Referência, pode ser melhor visualizada na Figura 5.15.

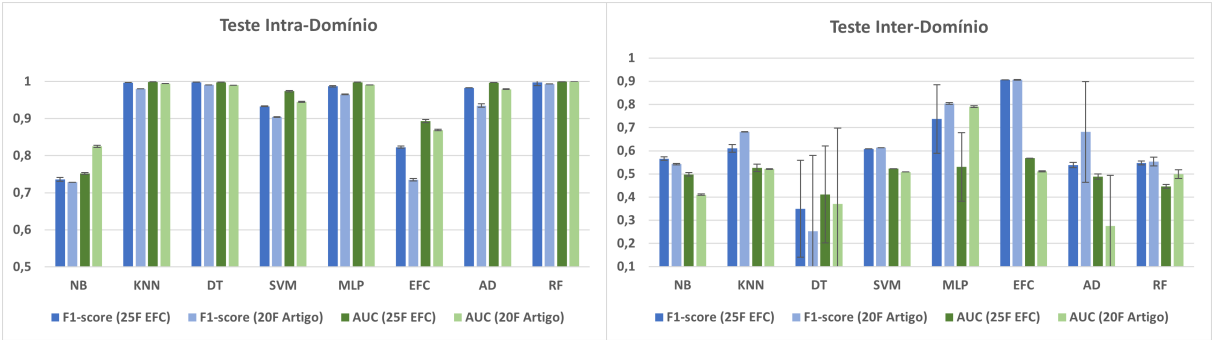


Figura 5.15: Comparação do Desempenho Médio dos Classificadores com Atributos do EFC e Atributos Segundo Ramos et al. [1] - ISCX-Bot x ISOT.

No gráfico do Teste Intra-Domínio, observa-se que exceto o NB, os demais classificadores tiveram uma variação negativa, principalmente o EFC e SVM. Esta variação pode ser explicada pelo fato de estarmos realizando o treinamento e teste dos modelos em um conjunto de dados diferente do qual a maioria dos atributos propostos por Ramos et al. [1] foram extraídos. No gráfico do Teste Inter-Domínio, observa-se que o NB, DT e EFC tiveram um desempenho ligeiramente superior utilizando os atributos selecionados pelo EFC. Por outro lado, o desempenho do KNN, MLP e RF foi melhor utilizando os atributos propostos no Estudo de Referência [1].

Considerando todos os experimentos realizados com os atributos propostos por Ramos et al., percebe-se que quando o treinamento e teste dos modelos foram realizados no mesmo conjunto de dados utilizado para a seleção da maioria dos atributos propostos (ISOT HTTP), os resultados obtidos ficaram muito parecidos aos resultados utilizando os atributos selecionados pelo EFC. Porém, quando o treinamento ou o teste foram realizados em um conjunto de dados diferente (i.e., CTU-13 ou ISCX-Bot-2014), houve uma redução de desempenho para a maioria dos classificadores. Sendo assim, pode-se afirmar que o conjunto de atributos selecionados pelo EFC se mostrou robusto e que os resultados obtidos a partir da utilização desses atributos, foram superiores aos resultados obtidos com os atributos propostos por Ramos et al. [1], para a maioria dos classificadores.

5.3.3 Atributos Selecionados pelo *Random Forest*

A última abordagem de seleção de atributos foi através da utilização do algoritmo RF, o qual faz uma estimativa da importância de cada atributo durante o ajuste do modelo (*fit*). Essa estimativa pode ser visualizada através do recurso *feature_importances_*, o qual gera uma pontuação (*score*) de importância para cada atributo, onde quanto maior a pontuação, mais importante é o atributo para a predição.

Na Tabela 5.32 podemos visualizar os 20 atributos mais importantes (maior pontuação), bem como os respectivos *scores* calculados pelo RF. Também utilizamos o conjunto de dados ISOT HTTP para seleção dos atributos, assim como foi feito na seleção de atributos com o EFC e no estudo de referência [1]. Realizamos os mesmos experimentos da subseção anterior, i.e., utilizamos primeiramente os conjuntos de dados ISOT HTTP e CTU-13 para os testes de adaptabilidade dos modelos, e em seguida, fizemos os experimentos com os conjuntos de dados ISOT HTTP e ISCX-Bot-2014. Por fim, os resultados obtidos com os 20 atributos mais importantes selecionados pelo RF foram comparados com os resultados obtidos utilizando os 25 atributos selecionados pelo EFC.

Tabela 5.32: Atributos Selecionados pelo Random Forest

N°	Atributo	Score
01	AveragePacketSize	0.151291
02	PacketLengthMean	0.080177
03	MaxPacketLength	0.076698
04	TotalLengthofBwdPackets	0.054258
05	BwdPacketLengthMax	0.052149
06	BwdPacketLengthMean	0.050071
07	AvgFwdSegmentSize	0.046841
8	FwdPacketLengthMax	0.045578
9	PacketLengthStd	0.045408
10	FwdPacketLengthMin	0.045406
11	BwdPacketLengthMin	0.039770
12	AvgBwdSegmentSize	0.037554
13	FwdPacketLengthMean	0.035193
14	PacketLengthVariance	0.034817
15	MinPacketLength	0.031845
16	FlowBytes-s	0.018559
17	TotalLengthofFwdPackets	0.017074
18	SourcePort	0.016763
19	DestinationPort	0.011706
20	FlowPackets-s	0.010226

5.3.3.1 Resultados ISOT HTTP x CTU-13

A Tabela 5.33 apresenta os resultados da primeira abordagem, na qual utilizamos o conjunto de dados ISOT HTTP como base para o treinamento dos modelos e o CTU-13 para o teste de adaptabilidade. Sinalizamos em azul os resultados que tiveram um aumento igual ou maior a 1%, em relação aos resultados obtidos utilizando os atributos selecionados pelo EFC. Por outro lado, sinalizamos em vermelho os resultados com decréscimo maior ou igual a 1%.

No teste intra-domínio, observa-se que os resultados foram muito próximos aos obtidos utilizando os atributos selecionados pelo EFC (Tabela 5.7). A exceção foi para o resultado do classificador EFC, o qual obteve uma pequena redução no F1-score (-1,1), bem como na AUC (-3,7). Por outro lado, o classificador NB obteve um aumento na métrica AUC (+4,8). Já no teste de domínio cruzado, todos os classificadores tiveram um desempenho muito ruim, inclusive o EFC, o qual teve uma redução tanto em relação à métrica F1-score (-42,3), quanto em relação à métrica AUC (-24,1). Com isso, concluímos que para este cenário, utilizando os atributos selecionados pelo RF, o desempenho do EFC no teste de domínio cruzado, foi muito inferior ao desempenho obtido tanto com os atributos

Tabela 5.33: Desempenho Médio dos Classificadores - 20 Atributos RF - ISOT x CTU-13

	Treino/Teste ISOT HTTP		**Treino ISOT/Teste CTU-13**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.963 ± 0.001	0.918 ± 0.003	0.185 ± 0.014	0.532 ± 0.007
KNN	**0.999 ± 0.000**	0.997 ± 0.000	0.016 ± 0.006	0.484 ± 0.001
DT	**0.999 ± 0.000**	0.997 ± 0.000	0.004 ± 0.000	0.498 ± 0.000
SVM	0.989 ± 0.001	0.995 ± 0.000	0.062 ± 0.017	0.542 ± 0.017
MLP	0.993 ± 0.001	0.999 ± 0.000	0.031 ± 0.021	**0.672 ± 0.007**
EFC	0.977 ± 0.000	0.958 ± 0.001	**0.230 ± 0.001**	0.319 ± 0.002
Ensemble				
RF	**0.999 ± 0.000**	1.000 ± 0.000	0.000 ± 0.000	0.308 ± 0.025
AD	**0.999 ± 0.001**	1.000 ± 0.000	0.000 ± 0.000	0.429 ± 0.018

selecionados pelo EFC (Tabela 5.7), quanto com os atributos propostos no Estudo de Referência (Tabela 5.28).

A comparação dos resultados obtidos pelos classificadores utilizando os 25 atributos selecionados pelo EFC e utilizando os 20 atributos selecionados pelo RF, pode ser melhor visualizada na Figura 5.16. Observamos no gráfico referente ao Teste Intra-Domínio, que a maioria dos classificadores mantiveram praticamente o mesmo desempenho obtido com os atributos selecionados pelo EFC. Além disso, o NB teve um aumento da métrica AUC utilizando os atributos selecionados pelo RF. Já no gráfico do Teste Inter-Domínio, observa-se que o desempenho do EFC utilizando os atributos selecionados pelo RF, foi muito inferior ao desempenho obtido utilizando os atributos selecionados pelo EFC, não sendo capaz portanto, de manter a adaptabilidade do modelo.

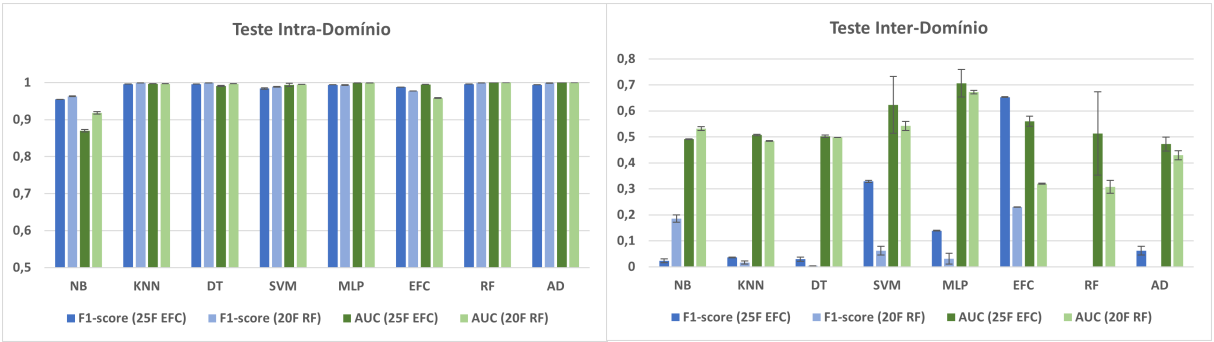


Figura 5.16: Comparação do Desempenho Médio dos Classificadores com Atributos do EFC e Atributos do RF - ISOT x CTU-13.

5.3.3.2 Resultados CTU-13 x ISOT HTTP

A Tabela 5.34 apresenta os resultados da segunda abordagem, na qual utilizamos os atributos selecionados pelo RF e o conjunto de dados CTU-13 como base para o treinamento dos modelos. No teste intra-domínio, com exceção do NB, o qual obteve uma melhora para a métrica F1-score (+5,5), os demais classificadores tiveram uma redução desta métrica, com destaque para EFC (-42,7), MLP (-14,8) e SVM (-9,7). Em relação à métrica AUC, os classificadores KNN e RF mantiveram o mesmo desempenho obtido com os atributos do EFC (Tabela 5.11), enquanto os demais classificadores sofreram uma redução para esta métrica, com destaque para NB (-11,8), EFC (-6,4) e MLP (-5,6). Observa-se que o desempenho do EFC neste cenário foi bastante inferior ao desempenho obtido com os atributos selecionados pelo EFC.

Tabela 5.34: Desempenho Médio dos Classificadores - 20 Atributos RF - CTU-13 x ISOT

	Treino/Teste CTU-13		**Treino CTU-13/Teste ISOT**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.708 ± 0.005	0.749 ± 0.001	0.552 ± 0.012	0.572 ± 0.012
KNN	0.974 ± 0.000	0.990 ± 0.000	0.224 ± 0.233	0.480 ± 0.233
DT	**0.982 ± 0.000**	0.986 ± 0.000	0.637 ± 0.217	**0.655 ± 0.217**
SVM	0.835 ± 0.001	0.932 ± 0.001	0.166 ± 0.090	0.461 ± 0.090
MLP	0.829 ± 0.034	0.942 ± 0.003	0.571 ± 0.095	0.550 ± 0.095
EFC	0.410 ± 0.002	0.868 ± 0.000	**0.692 ± 0.003**	0.587 ± 0.003
Ensemble				
RF	0.981 ± 0.000	**0.998 ± 0.000**	0.662 ± 0.054	0.460 ± 0.054
AD	0.941 ± 0.001	0.984 ± 0.000	0.624 ± 0.013	0.541 ± 0.013

No teste inter-domínio, o NB obteve um aumento em relação à métrica F1-score (+20,1), assim como o DT (+7,7). Os demais classificadores sofreram uma redução para esta métrica, com destaque para RF (-17,6), KNN (-14) e EFC (-11,4). Por outro lado, alguns classificadores tiveram uma melhora da métrica AUC, com destaque para DT (+19,6), AD (+15) e NB (+9,4). Apesar disso, outros classificadores sofreram uma redução para esta métrica, com destaque para RF (-25,4), MLP (-14) e EFC (-11). No contexto geral, o EFC obteve o melhor F1-score (0.692) e o DT obteve a melhor AUC (0.655). Apesar disso, podemos concluir que o desempenho do EFC foi bastante reduzido utilizando os atributos selecionados pelo RF, considerando os testes intra e inter-domínio. O SVM e o MLP também tiveram desempenho bastante inferiores. Por outro lado, a

seleção de atributos do RF favoreceu os classificadores NB e DT, principalmente no teste de domínio cruzado.

A comparação dos resultados obtidos pelos classificadores utilizando os 25 atributos selecionados pelo EFC e utilizando os 20 atributos selecionados pelo RF, pode ser melhor visualizada na Figura 5.17. Observa-se no Gráfico do Teste Intra-Domínio, que os atributos selecionados pelo RF não favoreceram o classificador EFC, o qual teve o desempenho bastante reduzido, principalmente em relação à métrica F1-score. Também houve redução no desempenho para os classificadores MLP e SVM. No geral, essa variação de desempenho neste experimento pode ser explicada pelo fato do treino e teste estar sendo realizado em um conjunto de dados diferente daquele a partir do qual os atributos foram selecionados. No gráfico do Teste Inter-Domínio, percebe-se que houve variação para todos os classificadores, sendo que o NB e DT tiveram um aumento tanto para o F1-score quanto para a AUC. Por outro lado, os classificadores SVM, MLP, EFC e RF tiveram uma redução também em relação às duas métricas (F1-score e AUC).

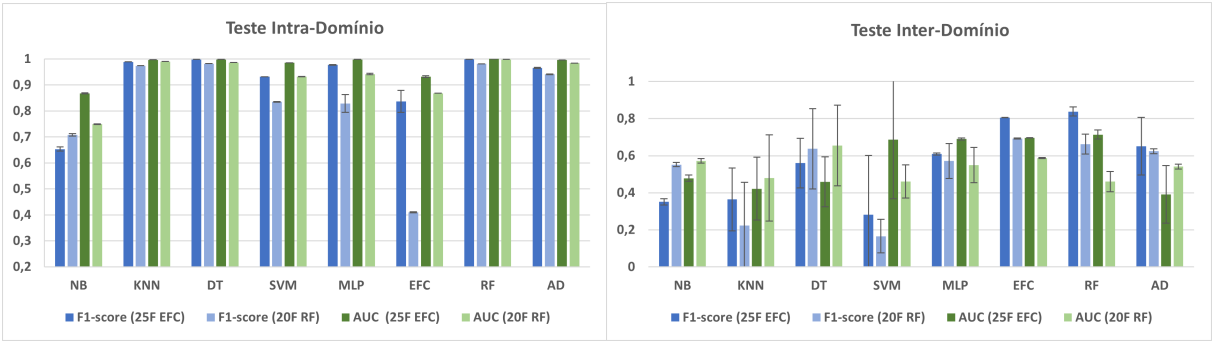


Figura 5.17: Comparação do Desempenho Médio dos Classificadores com Atributos do EFC e Atributos do RF - CTU-13 x ISOT.

5.3.3.3 Resultados ISOT HTTP x ISCX-Bot-2014

O próximo experimento avaliou o desempenho com os atributos selecionados pelo RF, utilizando o conjunto de dados ISOT HTTP como base para o treinamento dos modelos e o ISCX-Bot-2014 para o teste de adaptabilidade. Em seguida, os resultados foram comparados aos resultados obtidos utilizando os atributos selecionados pelo EFC. A Tabela 5.35 apresenta o desempenho dos classificadores em ambos os testes (intra e inter-domínio).

Percebe-se que em relação ao teste intra-domínio, i.e., treino e teste utilizando o ISOT-HTTP, os resultados são os mesmos já apresentados na Tabela 5.33. A novidade neste experimento é em relação ao teste de domínio cruzado, i.e., treino no ISOT HTTP e teste no ISCX-Bot-2014. Observa-se que alguns classificadores tiveram uma melhora significativa em relação à métrica F1-score, com destaque para NB (+51,1), MLP (+27,7)

e SVM (+21). A AUC de quase todos os classificadores também teve uma pequena melhora, com destaque para SVM (+8,3), MLP (+3,9) e KNN (+2,9). Porém, alguns classificadores tiveram uma redução para esta métrica, com destaque para o EFC (-10,6). Apesar da melhora de desempenho de alguns classificadores (NB e SVM principalmente), o EFC manteve o melhor F1-score, assim como aconteceu no experimento utilizando os atributos do EFC (Tabela 5.16). De um modo geral, os classificadores NB e SVM se beneficiaram da seleção de atributos do RF, principalmente se considerarmos o teste de domínio cruzado. Em relação ao EFC, apesar do F1-score no teste de domínio cruzado ter se mantido igual ao obtido utilizando os atributos do EFC, houve uma redução significativa no valor da AUC, além da redução também no teste intra-domínio, o que indica que esta abordagem de seleção de atributos não favoreceu o desempenho do EFC.

Tabela 5.35: Desempenho Médio dos Classificadores - 20 Atributos RF - ISOT x ISCX-Bot

	Treino/Teste ISOT HTTP		**Treino ISOT/Teste ISCX-Bot**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.963 ± 0.001	0.918 ± 0.003	0.528 ± 0.016	0.506 ± 0.007
KNN	**0.999 ± 0.000**	0.997 ± 0.000	0.241 ± 0.030	0.551 ± 0.007
DT	**0.999 ± 0.000**	0.997 ± 0.000	0.016 ± 0.001	0.403 ± 0.000
SVM	0.989 ± 0.001	0.995 ± 0.000	0.571 ± 0.029	0.502 ± 0.029
MLP	0.993 ± 0.000	0.999 ± 0.000	0.281 ± 0.051	0.503 ± 0.011
EFC	0.977 ± 0.000	0.952 ± 0.000	**0.733 ± 0.000**	0.389 ± 0.008
Ensemble				
AD	**0.999 ± 0.001**	1.000 ± 0.000	0.000 ± 0.000	**0.571 ± 0.018**
RF	**0.999 ± 0.000**	1.000 ± 0.000	0.000 ± 0.000	0.339 ± 0.044

A comparação dos resultados obtidos pelos classificadores utilizando os 25 atributos selecionados pelo EFC e utilizando os 20 atributos selecionados pelo RF, pode ser melhor visualizada na Figura 5.18. Observa-se pelo gráfico do Teste Intra-Domínio que quando os experimentos são realizados utilizando o mesmo conjunto de dados utilizado para a seleção dos atributos mais importantes, o desempenho de todos os classificadores fica muito próximo ao desempenho obtido utilizando os atributos selecionados pelo EFC. Já no gráfico do Teste Inter-Domínio, percebe-se que os atributos selecionados pelo RF beneficiaram os classificadores NB, MLP e SVM, os quais tiveram um aumento significativo, principalmente em relação à métrica F1-score. Por outro lado, apesar do EFC ter mantido a adaptabilidade do modelo com os atributos selecionados pelo RF, percebe-se que houve uma redução em seu desempenho, principalmente em relação à métrica AUC,

tanto no teste intra como no teste inter-domínio. Dessa forma, podemos concluir que para este experimento, especificamente, o desempenho da maioria dos classificadores foi maior utilizando os atributos selecionados pelo RF.

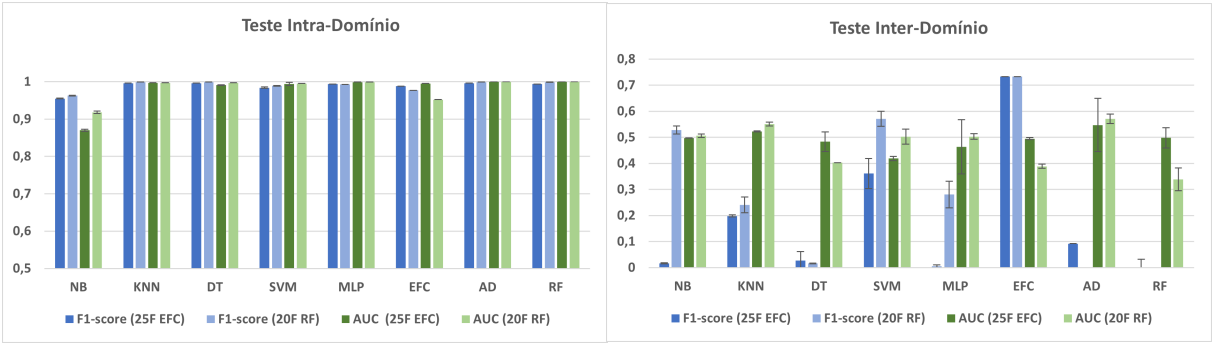


Figura 5.18: Comparação do Desempenho Médio dos Classificadores com Atributos do EFC e Atributos do RF - ISOT x ISCX-Bot.

5.3.3.4 Resultados ISCX-Bot-2014 x ISOT HTTP

Nosso último experimento se baseia exatamente no inverso do experimento anterior, i.e., utilizaremos o conjunto de dados ISCX-Bot-2014 como base para o treinamento dos modelos e o conjunto de dados ISOT HTTP para o teste de domínio cruzado. Os resultados podem ser visualizados na Tabela 5.36.

Tabela 5.36: Desempenho Médio dos Classificadores - 20 Atributos RF - ISCX-Bot x ISOT

	Treino/Teste ISCX-Bot		**Treino ISCX-Bot/Teste ISOT**	
Classificador	**F1-score**	**AUC**	**F1-score**	**AUC**
NB	0.404 ± 0.186	0.809 ± 0.003	0.630 ± 0.008	0.578 ± 0.008
KNN	0.977 ± 0.000	0.993 ± 0.001	0.633 ± 0.013	0.424 ± 0.013
DT	0.982 ± 0.000	0.984 ± 0.000	0.331 ± 0.145	0.413 ± 0.145
SVM	0.906 ± 0.001	0.949 ± 0.001	0.593 ± 0.000	0.309 ± 0.000
MLP	0.959 ± 0.005	0.985 ± 0.003	0.775 ± 0.331	0.607 ± 0.331
EFC	0.548 ± 0.003	0.741 ± 0.002	**0.901 ± 0.004**	0.314 ± 0.004
Ensemble				
AD	0.931 ± 0.003	0.977 ± 0.000	0.527 ± 0.109	0.281 ± 0.109
RF	**0.984 ± 0.000**	**0.999 ± 0.000**	0.572 ± 0.030	**0.680 ± 0.030**

No teste intra-domínio, i.e., treino e teste no mesmo conjunto de dados, todos os classificadores tiveram uma redução em relação à métrica F1-score, com destaque para NB (-33,2), EFC (-27,5) e AD (-5,2). Já para a métrica AUC, o NB obteve um aumento (+5,7), enquanto os outros classificadores tiveram uma redução, com destaque para EFC (-15,2) e SVM (-2,5).

No teste inter-domínio, apesar do classificador EFC ter obtido o mesmo valor de F1-score alcançado no experimento utilizando os atributos do EFC (Tabela 5.22), houve uma redução considerável em relação à métrica AUC (-25,4) para este classificador. Por outro lado, os classificadores NB, MLP e RF, tiveram um aumento tanto em relação à métrica F1-score, quanto em relação à AUC. No contexto geral, apesar das variações nos resultados, o EFC manteve o melhor F1-score (0.901), assim como ocorreu nos experimentos utilizando os atributos do EFC. Porém, diferentemente, o RF superou o EFC em relação ao melhor valor de AUC (0.680).

A comparação dos resultados obtidos pelos classificadores utilizando os 25 atributos selecionados pelo EFC e utilizando os 20 atributos selecionados pelo *Random Forest*, pode ser melhor visualizada na Figura 5.19.

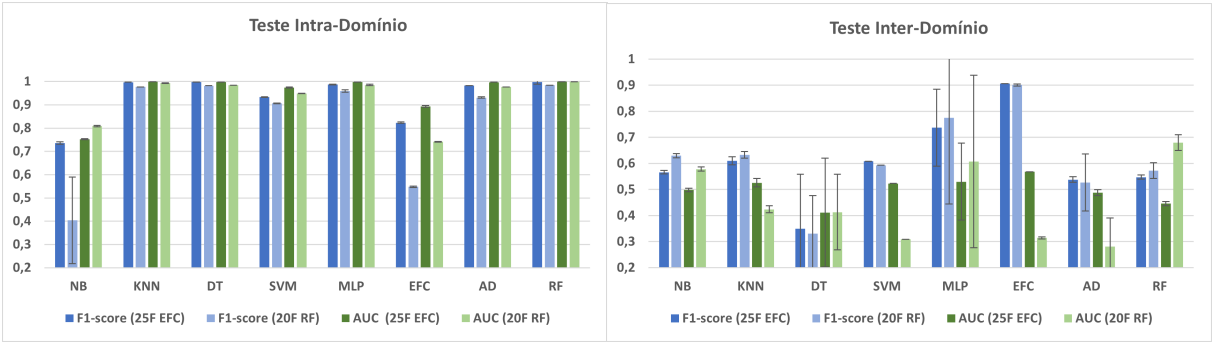


Figura 5.19: Comparação do Desempenho Médio dos Classificadores Com 25 Atributos do EFC e os 20 Atributos do RF - ISCX-Bot x ISOT.

No gráfico do Teste Intra-Domínio, nota-se que o desempenho do NB e do EFC foi bastante inferior com a utilização dos atributos selecionados pelo RF, principalmente em relação à métrica F1-score. Para os demais classificadores, os resultados ficaram muito próximos aos obtidos utilizando os atributos selecionados pelo EFC, apesar da pequena redução em relação à métrica F1-score. No gráfico do Teste Inter-Domínio, podemos visualizar que apesar do EFC ter mantido o valor de F1-score, houve uma redução significativa em relação à métrica AUC. O SVM e o AD também tiveram uma redução significativa em relação à AUC. Por outro lado, o desempenho do MLP, RF e NB, foi superior utilizando os atributos selecionados pelo RF, tanto em relação ao F1-score quanto em relação à AUC.

Considerando todos os experimentos realizados com os atributos selecionados utilizando o algoritmo RF, percebe-se que quando o treinamento e teste dos modelos foram realizados no mesmo conjunto de dados utilizado para a seleção dos atributos (ISOT HTTP), os resultados obtidos ficaram muito parecidos aos resultados utilizando os atributos selecionados pelo EFC. Porém, quando o treinamento ou o teste foram realizados em um conjunto de dados diferente (i.e., CTU-13 ou ISCX-Bot-2014), o desempenho do EFC reduziu significativamente, prejudicando inclusive, a adaptabilidade do modelo, em um dos experimentos. Sendo assim, para o EFC, a seleção de atributos utilizando o RF prejudica o desempenho do classificador. Para os demais classificadores, houve uma grande variação no desempenho. Dessa forma, em alguns dos experimentos, os resultados obtidos utilizando os atributos do RF superaram os resultados com a seleção de atributos do EFC, enquanto em outros cenários houve uma redução significativa no desempenho para alguns classificadores. Com isso, concluímos que o conjunto de dados selecionado pelo EFC se mostrou mais robusto, mantendo resultados mais estáveis, principalmente quando utilizamos um conjunto de dados diferente daquele utilizado para seleção dos atributos.

5.3.4 Síntese do Estudo de Caso 2

Podemos analisar os resultados obtidos com os experimentos do Estudo de Caso 2 em duas etapas. Na primeira etapa, o objetivo era explorar a interpretabilidade do modelo inferido pelo EFC e propor uma abordagem para seleção dos atributos que mais caracterizassem o tráfego benigno e assim detectar atividades de *botnets*. Os resultados obtidos demonstraram que os 25 atributos selecionados à partir do conjunto de dados ISOT HTTP, foram capazes de manter resultados próximos aos alcançados com os 80 atributos iniciais, principalmente nos testes em que o ISOT HTTP foi utilizado como base para o treinamento dos modelos. Apesar de ter havido redução no desempenho quando utilizamos outros conjuntos de dados, essa redução pode ser compensada pela economia de recurso computacional e consequentemente, economia de tempo para o treinamento dos modelos, principalmente quando se pensa em trabalhar com uma grande quantidade de dados, que é o que acontecerá em um cenário real.

O objetivo da segunda etapa do Estudo de Caso 2 era comparar o desempenho obtido utilizando o conjunto de atributos selecionados pelo EFC com o desempenho obtido utilizando outros conjuntos de atributos, avaliando assim, até que ponto a redução de atributos era aplicável a outros algoritmos. Dessa forma, utilizamos o conjunto de atributos proposto por Ramos et al. [1], o qual considerou o Estado da Arte e o conjunto de dados ISOT HTTP para a seleção de grande parte desses atributos. Além disso, utilizamos um outro conjunto de atributos, os quais foram selecionados através do recurso

feature importance do RF, também utilizando o conjunto de dados ISOT HTTP como base para a seleção. Apresentaremos a seguir, a comparação do desempenho utilizando os três conjuntos de atributos mencionados, sendo que a análise será realizada considerando primeiramente o teste intra-domínio e em seguida o teste inter-domínio.

5.3.4.1 Comparação de Desempenho no Teste Intra-Domínio

Os gráficos a seguir ilustram a comparação do desempenho de todos os classificadores utilizando os três conjuntos de atributos utilizados no Estudo de Caso 2: atributos do EFC, atributos do Estudo de Referência [1] e atributos selecionados através do recurso *feature importance* do RF.

No Figura 5.20 temos os resultados obtidos utilizando o conjunto de dados ISOT HTTP para treino e teste dos modelos. Observa-se que neste caso, estamos utilizando o mesmo conjunto de dados a partir do qual os conjuntos de atributos foram selecionados. As barras em roxo representam os valores de F1-score e AUC utilizando o conjunto de atributos selecionados pelo EFC, as barras em verde representam os valores de F1-score e AUC utilizando o conjunto de atributos do Estudo de Referência [1] e por fim, as barras em alaranjado representam as métricas utilizando o conjunto de atributos do RF. Neste cenário, a maioria dos classificadores obtiveram desempenho muito parecido, independente do conjunto de atributos utilizado. A exceção foi o NB, o qual obteve um desempenho superior utilizando os atributos do RF, principalmente em relação à AUC e o EFC, o qual obteve um desempenho inferior utilizando os atributos do RF.

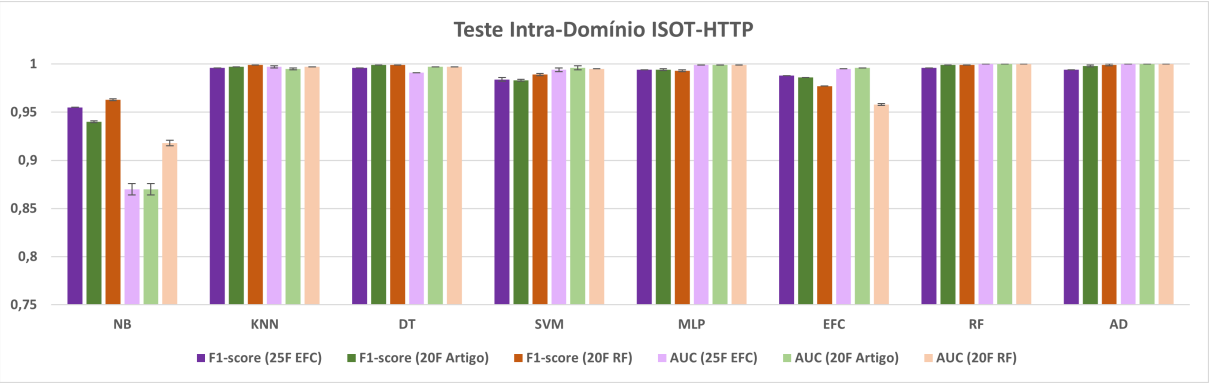


Figura 5.20: Comparação do Desempenho Médio dos Classificadores Com Atributos do EFC, Atributos Segundo Ramos et al. [1] e Atributos do RF - ISOT HTTP.

A figura 5.21 apresenta os desempenho dos classificadores utilizando o conjunto de dados CTU-13 como base para o treino e teste dos modelos. Neste cenário, estamos testando o conjunto de atributos selecionados a partir de um conjunto de dados (ISOT HTTP), em outro conjunto de dados (CTU-13). Percebe-se que nesse cenário, o desempenho utilizando o conjunto de atributos selecionados pelo EFC foi superior para a maioria dos

classificadores, com destaque para o SVM, EFC e MLP. Observa-se também que o desempenho do EFC foi muito inferior utilizando o conjunto de atributos do RF, principalmente em relação à métrica F1-score.

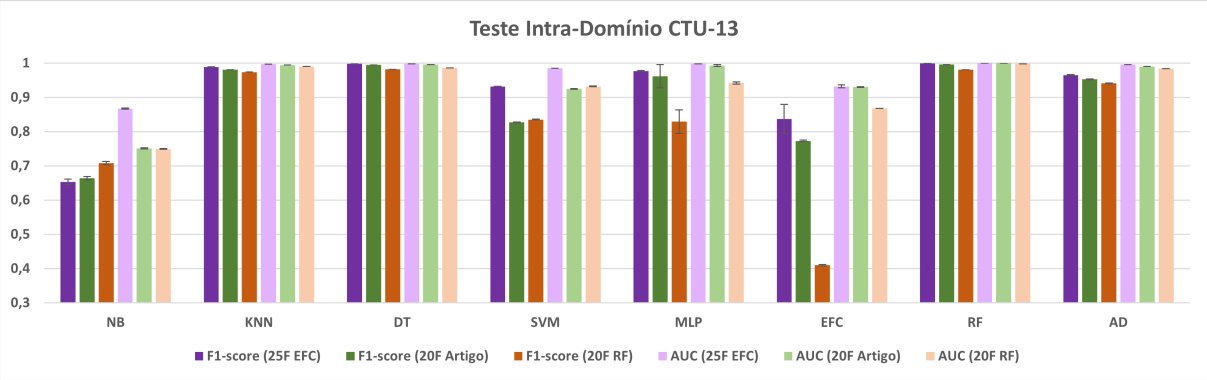


Figura 5.21: Comparação do Desempenho Médio dos Classificadores Com Atributos do EFC, Atributos Segundo Ramos et al. [1] e Atributos do RF - CTU-13.

Por fim, a Figura 5.22 apresenta os desempenho dos classificadores utilizando o conjunto de dados ISCX-Bot 2014 como base para o treino e teste dos modelos. Neste cenário, também estamos testando o conjunto de atributos selecionados a partir de um conjunto de dados (ISOT HTTP), em outro conjunto de dados (ISCX-Bot 2014). Novamente, com exceção do NB, o desempenho utilizando o conjunto de atributos selecionados pelo EFC foi superior para todos os classificadores, com destaque para o EFC, SVM e AD. Observa-se, ainda, que o pior desempenho para o classificador EFC foi utilizando o conjunto de atributos do RF. O NB também sofreu uma redução considerável em relação ao F1-score, utilizando conjunto de atributos do RF.

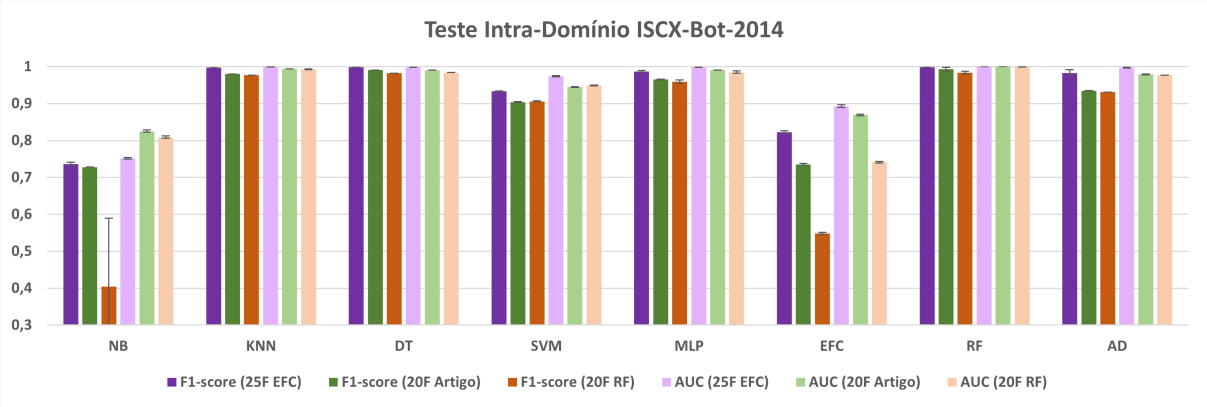


Figura 5.22: Comparação do Desempenho Médio dos Classificadores Com Atributos do EFC, Atributos Segundo Ramos et al. [1] e Atributos do RF - ISCX-Bot-2014.

Com esses resultados podemos concluir que, para a maioria dos classificadores, o desempenho obtido utilizando os atributos selecionados pelo EFC foram superiores ao de-

sempenho obtido utilizando tanto o conjunto de atributos propostos por Ramos et al. [1], quanto utilizando o conjunto de atributos selecionados pelo RF, principalmente quando o treinamento ou o teste foram realizados em um conjunto de dados diferente daquele a partir do qual os conjuntos de atributos foram selecionados. Dessa forma, pode-se afirmar que, o conjunto de atributos selecionado pelo EFC se mostrou mais robusto, mantendo resultados mais estáveis do que os resultados obtidos com os outros dois conjuntos de atributos.

5.3.4.2 Comparação de Desempenho no Teste Inter-Domínio

Os gráficos a seguir ilustram a comparação do desempenho de todos os classificadores em relação ao teste inter-domínio, o qual possui o objetivo de avaliar a adaptabilidade dos modelos quando testados em um domínio diferente daquele onde foi realizado o treinamento. A comparação foi realizada considerando os três conjuntos de atributos utilizados no Estudo de Caso 2: atributos do EFC, atributos do Estudo de Referência [1] e atributos do RF.

No Figura 5.23 temos os resultados obtidos utilizando o conjunto de dados ISOT HTTP para o treino e o conjunto de dados CTU-13 para o teste dos modelos. Observa-se que o EFC conseguiu manter a adaptabilidade do modelo, tanto utilizando o conjunto de atributos do EFC, quanto utilizando o conjunto de atributos do Estudo de Referência [1]. Por outro lado, o desempenho do classificador EFC foi bastante reduzido utilizando o conjunto de atributos do RF. O desempenho do restante dos classificadores foi muito ruim, independente do conjunto de atributos utilizado, assim como aconteceu com os 80 atributos iniciais.

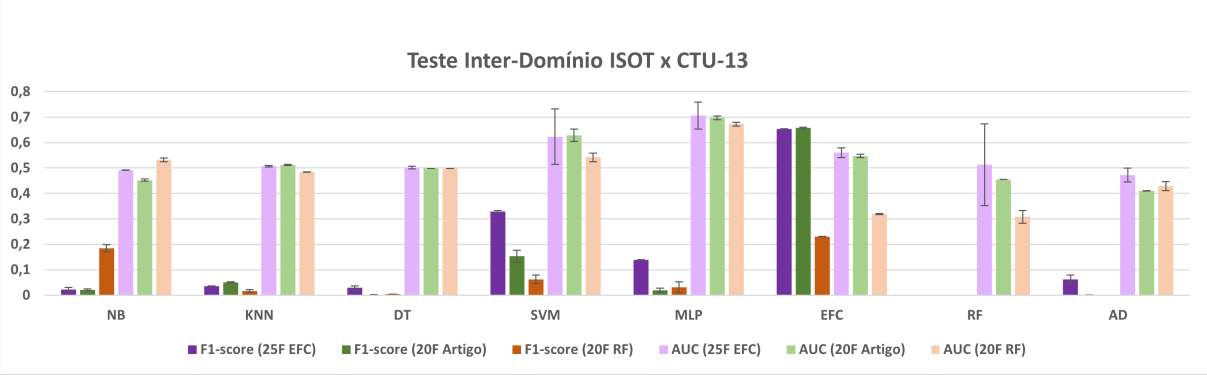


Figura 5.23: Comparação do Desempenho Médio dos Classificadores Com Atributos do EFC, Atributos Segundo Ramos et al. [1] e Atributos do RF - ISOT x CTU-13.

A figura 5.24 apresenta os resultados obtidos utilizando o conjunto de dados CTU-13 para o treino e o conjunto de dados ISOT HTTP para o teste dos modelos. Observa-se que os classificadores EFC e RF obtiveram um melhor desempenho utilizando o conjunto

de atributos do EFC, enquanto o pior desempenho para estes classificadores foi utilizando o conjunto de atributos do RF. O MLP também obteve desempenho inferior utilizando os atributos do RF. Por outro lado, o NB e DT obtiveram melhor desempenho utilizando o conjunto de atributos do RF.

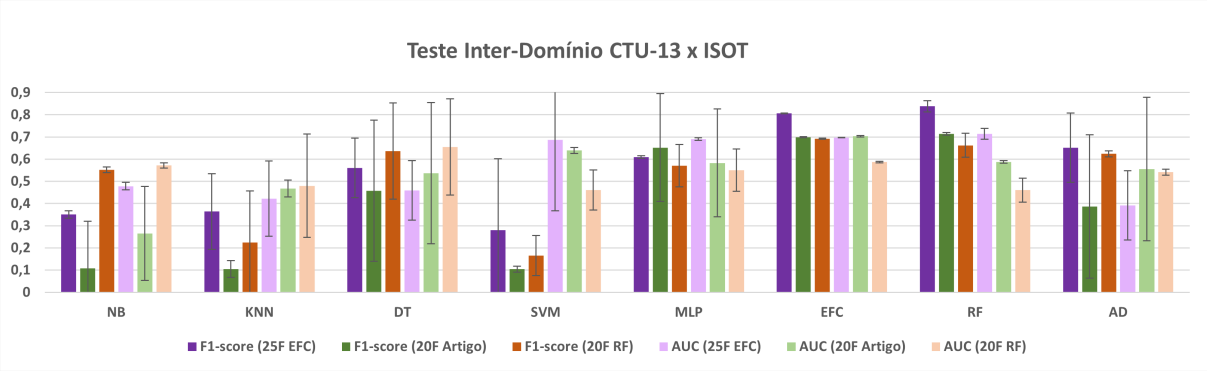


Figura 5.24: Comparação do Desempenho Médio dos Classificadores Com Atributos do EFC, Atributos Segundo Ramos et al. [1] e Atributos do RF - CTU-13 x ISOT.

A figura 5.25 apresenta os resultados obtidos utilizando o conjunto de dados ISOT HTTP para o treino e o conjunto de dados ISCX-Bot-2014 para o teste de adaptabilidade dos modelos. Observa-se que, independente do conjunto de atributos utilizado e com exceção do EFC, os demais classificadores obtiveram um desempenho muito ruim, assim como aconteceu quando foi utilizado os 80 atributos iniciais. Em relação ao classificador EFC, observa-se que o mesmo manteve a adaptabilidade do modelo, independente do conjunto de dados utilizado e que o melhor desempenho foi utilizando os atributos selecionados pelo EFC.

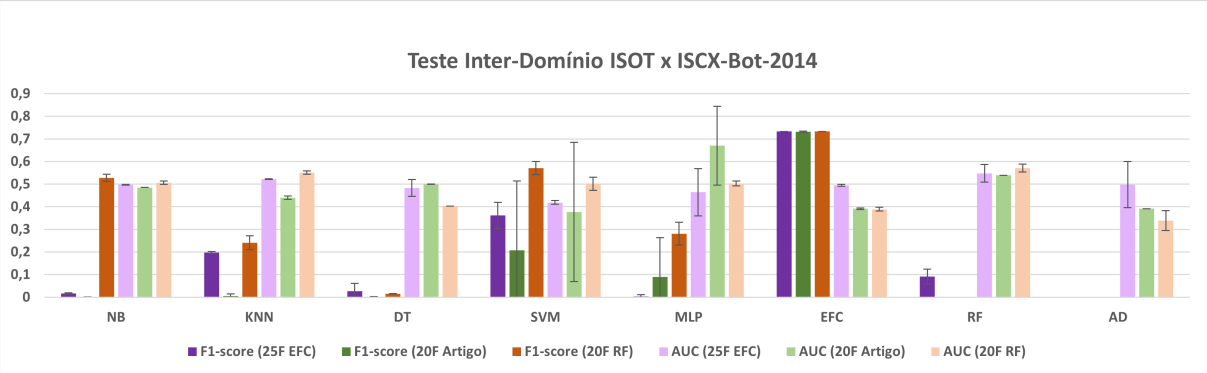


Figura 5.25: Comparação do Desempenho Médio dos Classificadores Com Atributos do EFC, Atributos Segundo Ramos et al. [1] e Atributos do RF - ISOT x ISCX-Bot.

Por fim, a Figura 5.26 apresenta os resultados obtidos utilizando o conjunto de dados ISCX-Bot-2014 para o treino e o conjunto de dados ISOT HTTP para o teste de adaptabilidade dos modelos. Observa-se que o EFC manteve a adaptabilidade do modelo,

independente do conjunto de atributos utilizado, mas ainda assim, o desempenho obtido utilizando os atributos do EFC foi superior ao desempenho obtido utilizando os outros conjuntos de atributos, principalmente em relação à métrica AUC. Por outro lado, os classificadores NB e RF obtiveram desempenho superior utilizando os atributos do RF, enquanto o KNN e o MLP tiveram um melhor desempenho utilizando os atributos do Estudo de Referência [1].

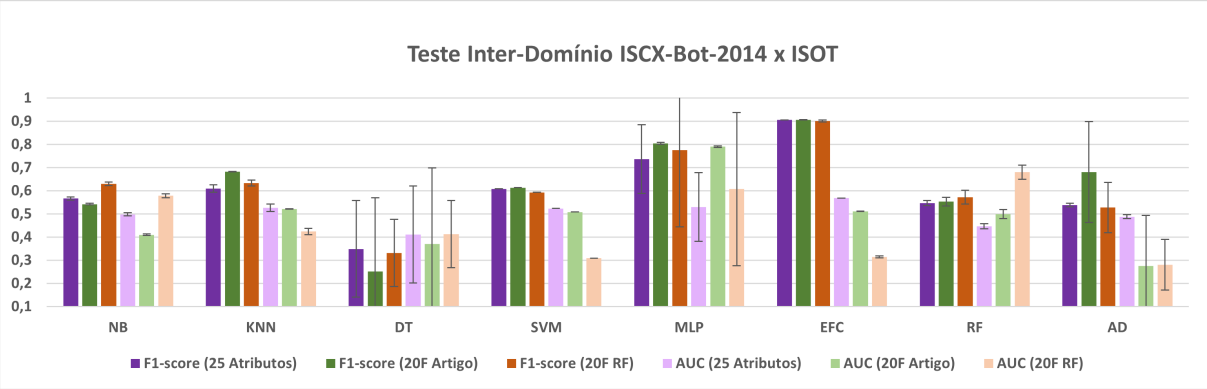


Figura 5.26: Comparação do Desempenho Médio dos Classificadores Com Atributos do EFC, Atributos Segundo Ramos et al. [1] e Atributos do RF - ISCX-Bot x ISOT.

A partir dos resultados obtidos no teste inter-domínio, concluímos que o classificador EFC conseguiu manter a adaptabilidade do modelo em todos os experimentos realizados e que os melhores resultados foram obtidos utilizando o conjunto de atributos selecionado pelo EFC, enquanto o pior desempenho do EFC foi utilizando o conjunto de atributos do RF. Para o restante dos classificadores houve uma variação, sendo que alguns classificadores obtiveram melhor desempenho utilizando os atributos do EFC em alguns dos experimentos, enquanto em outros experimentos os resultados obtidos utilizando o conjunto de atributos do RF ou do Estudo de Referência [1] foram superiores.

Capítulo 6

Conclusão e Trabalhos Futuros

Esta dissertação explorou o problema relacionado à detecção de *botnets*, as quais são consideradas uma das ameaças cibernéticas mais danosas e complexas. A detecção desse tipo de ameaça é uma tarefa desafiadora, devido à constante evolução das *botnets*, tanto em relação às arquiteturas e protocolos utilizados, quanto ao seu funcionamento. A maioria das abordagens que utilizam o aprendizado de máquina para a detecção de *botnets* empregam algoritmos binários, os quais necessitam que a base de treinamento tenha amostras benignas e maliciosas para gerar o modelo de detecção. Dessa forma, somente as *botnets* que estejam presentes no conjunto de treinamento ou que tenham comportamento muito parecido, serão detectadas.

Sendo assim, propomos a utilização de um classificador *One Class*, denominado *Energy-Based Flow Classifier (EFC)*, o qual infere um modelo estatístico baseado apenas em amostras de tráfego benignas. Dessa forma, não há necessidade de conhecer o tráfego malicioso para gerar o modelo de detecção. Comparamos os resultados obtidos pelo EFC, com os resultados obtidos por diversos classificadores de duas e de uma classe. Além disso, exploramos a interpretabilidade do modelo estatístico inferido pelo EFC, para propor uma metodologia de seleção dos atributos mais representativos para classificação do tráfego relativo à atividades de *botnets*, sendo esta metodologia a principal contribuição desta pesquisa.

Os experimentos do Estudo de Caso 1, demonstraram que os classificadores binários sofreram fortes variações nos testes inter-domínio, i.e., treino em um conjunto de dados e teste em outro conjunto de dados, o que indica que esses classificadores não seriam capazes de detectar novos tipos de *botnets*, em um ambiente real. Por outro lado, os classificadores *One Class* se mostraram mais eficientes nos testes inter-domínio, mas ainda assim, o EFC foi o modelo que se mostrou menos sensível à mudanças na distribuição de dados, podendo ser considerado um classificador promissor para a detecção de novos tipos de *botnets*.

Os experimentos do Estudo de Caso 2, demonstraram que através da análise dos

acoplamentos entre os pares de atributos, é possível selecionar um conjunto de atributos capaz de representar o tráfego de *botnets*, reduzindo o volume de dados a ser analisado, sem prejudicar a análise. Os diversos testes realizados, bem como a análise comparativa com outras abordagens de seleção de atributos, comprovaram que os atributos selecionados pelo EFC se mostraram mais robustos, mantendo resultados mais estáveis, principalmente quando utilizamos um conjunto de dados diferente daquele utilizado para seleção dos atributos.

Concluímos portanto que o EFC pode ser utilizado não somente como um classificador de fluxos de rede, mas também como uma técnica para análise e seleção dos atributos que melhor representem um determinado problema. Por fim, destaca-se a intenção de implementar a proposta desta dissertação na rede do Centro de Coordenação de Operações Móvel (CCOp Mv), de forma a contribuir com a segurança das operações coordenadas pelo Exército Brasileiro, por meio da detecção de *botnets*.

6.1 Contribuições em Produção Bibliográfica

A presente pesquisa gerou até o momento, duas publicações, a saber:

- O artigo "*Botnet detection based on network flow analysis using inverse statistic*" (Lopes, D. A., Marotta, M. A., Ladeira, M., & Gondim, J. J., 2022) que apresenta a análise comparativa do classificador EFC com os classificadores tradicionais de uma e duas classes, para a detecção de *botnets*, analisando também, a capacidade de adaptação à diferentes domínios, foi publicado e apresentado na conferência CISTI 2022 (17th *Iberian Conference on Information Systems and Technologies*) [118].
- O artigo "Detecção de botnets baseada na análise de fluxos de rede utilizando estatística inversa" (LOPES, Daniele A. G.; MAROTTA, Marcelo A.; LADEIRA, Marcelo; GONDIM, João J. C., 2022) que apresentou uma pequena evolução em relação ao artigo anterior, apresentando a análise dos pares de atributos com maior e menor valor de acoplamento, foi publicado e apresentado no XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos [119].

6.2 Trabalhos futuros

A partir dos experimentos realizados e dos resultados alcançados nesta pesquisa, vislumbra-se como trabalhos futuros ampliar os experimentos, explorando os hiper-parâmetros tanto do EFC quanto dos demais classificadores utilizados, no intuito de obter melhor desempenho. Um outro desafio, é estudar opções de como colocar o presente estudo em um

ambiente em produção e simular um ataque real em um ambiente controlado, para analisar a eficiência do EFC, em um cenário real.

Outra possibilidade é expandir a avaliação do modelo proposto, utilizando outras métricas como por exemplo precisão e revocação e ainda avaliar a detecção por tipo de *botnet*, no caso dos conjuntos de dados ISOT HTTP e ISCX-Bot-2014 e por cenário, no caso do conjunto de dados CTU-13.

Referências

[1] Ramos, Katherinne Shirley Huancayo, Marco Antonio Sotelo Monge e Jorge Maestre Vidal: *Benchmark-based reference model for evaluating botnet detection tools driven by traffic-flow analytics*. Sensors (Switzerland), 20(16):1–31, 2020, ISSN 14248220. x, xii, xiv, 37, 40, 50, 60, 61, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 101, 102, 103, 104, 105, 106

[2] Brasileiro, Exército: *Ministério da defesa exército brasileiro estado-maior do exército portaria nº 011 - eme, de 14 de janeiro de 2019.* 2019. 1, 2

[3] Hoang, Xuan Dau e Quynh Chi Nguyen: *Botnet detection based on machine learning techniques using DNS query data*. Future Internet, 10(5):1–11, 2018, ISSN 19995903. 2

[4] Zhao, David, Issa Traore, Bassam Sayed, Wei Lu, Sherif Saad, Ali Ghorbani e Dan Garant: *Botnet detection based on traffic behavior analysis and flow intervals*. Computers and Security, 39(PARTA):2–16, 2013, ISSN 01674048. `http://dx.doi.org/10.1016/j.cose.2013.04.007`. 2, 35, 40, 46

[5] Ibrahim, Wan Nur Hidayah, Syahid Anuar, Ali Selamat, Ondrej Krejcar, Ruben Gonzalez Crespo, Enrique Herrera-Viedma e Hamido Fujita: *Multilayer Framework for Botnet Detection Using Machine Learning Algorithms*. IEEE Access, 9:48753–48768, 2021, ISSN 21693536. 2, 3, 9, 10, 11, 12, 23, 37, 40, 44

[6] Council to Secure the Digital Economy: *International Botnet and Iot Security Guide 2020*. 2019. `https://securingdigitaleconomy.org/wp-content/uploads/2019/11/CSDE_Botnet-Report_2020_FINAL.pdf`. 2

[7] Wainwright, Polly e Houssain Kettani: *An analysis of botnet models*. ACM International Conference Proceeding Series, páginas 116–121, 2019. 2

[8] Antonakakis, Manos, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas e Yi Zhou: *Understanding the mirai botnet*. Em *26th USENIX Security Symposium (USENIX Security 17)*, páginas 1093–1110, Vancouver, BC, agosto 2017. USENIX Association, ISBN 978-1-931971-40-9. `https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis`. 2, 19

[9] European Union Agency for Network and Information Security: *Botnet - ENISA Threat Landscape 2019/20*. (April), 2020. `https://www.enisa.europa.eu/publications/enisa-threat-landscape-2020-botnet`. 2, 17

[10] Silva, Sérgio S.C., Rodrigo M.P. Silva, Raquel C.G. Pinto e Ronaldo M. Salles: *Botnets: A survey*. Computer Networks, 57(2):378–403, 2013, ISSN 13891286. 3, 8, 9, 12, 13, 14, 21, 22, 32

[11] Vormayr, Gernot, Tanja Zseby e Joachim Fabini: *Botnet Communication Patterns*. IEEE Communications Surveys and Tutorials, 19(4):2768–2796, 2017, ISSN 1553877X. 3, 23

[12] Yadav, Jagdish e Jawahar Thakur: *BotEye: Botnet detection technique via traffic flow analysis using machine learning classifiers*. PDGC 2020 - 2020 6th International Conference on Parallel, Distributed and Grid Computing, páginas 154–159, 2020. 3

[13] Sperotto, Anna, Gregor Schaffrath, Ramin Sadre, Cristian Morariu, Aiko Pras e Burkhard Stiller: *An overview of IP flow-based intrusion detection*. IEEE Communications Surveys and Tutorials, 12(3):343–356, 2010, ISSN 1553877X. 3

[14] García, Sebastián, Alejandro Zunino e Marcelo Campo: *Survey on network-based botnet detection methods*. Security and Communication Networks, 7(5):878–903, 2014. `https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.800`. 3, 44

[15] Spamhaus: *Botnet Threat Report 2019*. 2019. `https://www.spamhaus.org/news/images/full-2019/spamhaus-botnet-threat-report-2019.pdf`. 4

[16] Saad, Sherif, Issa Traore, Ali Ghorbani, Bassam Sayed, David Zhao, Wei Lu, John Felix e Payman Hakimian: *Detecting P2P botnets through network behavior analysis and machine learning*. 2011 9th Annual International Conference on Privacy, Security and Trust, PST 2011, páginas 174–180, 2011. 4, 34, 36, 40

[17] Li, Hao, Zhenxiang Chen, Riccardo Spolaor, Qiben Yan, Chuan Zhao e Bo Yang: *DART: Detecting Unseen Malware Variants using Adaptation Regularization Transfer Learning*. IEEE International Conference on Communications, 2019-May, 2019, ISSN 15503607. 4

[18] Zolanvari, Maede, Marcio A. Teixeira e Raj Jain: *Effect of imbalanced datasets on security of industrial IoT using machine learning*. 2018 IEEE International Conference on Intelligence and Security Informatics, ISI 2018, páginas 112–117, 2018. 4

[19] Rudin, Cynthia: *Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead*. Nature Machine Intelligence, 1(5):206–215, 2019, ISSN 25225839. 4

[20] Pontes, Camila F.T., Manuela M.C. De Souza, Joao J.C. Gondim, Matt Bishop e Marcelo Antonio Marotta: *A New Method for Flow-Based Network Intrusion Detection Using the Inverse Potts Model*. IEEE Transactions on Network and Service Management, 18(2):1125–1136, 2021, ISSN 19324537. 4, 23, 24, 41, 42, 43, 61

[21] Karim, Ahmad, Rosli Bin Salleh, Muhammad Shiraz, Syed Adeel Ali Shah, Irfan Awan e Nor Badrul Anuar: *Botnet detection techniques: review, future trends, and issues.* Journal of Zhejiang University: Science C, 15(11):943–983, 2014, ISSN 1869196X. 5, 15, 21, 22

[22] Bissell, Kelly, Ryan M Lasalle e Paolo Dal Cin: *The cost of cybercrime: Ninth annual cost of cybercrime study.* Ponemon Institute LLC and jointly developed by Accenture. Online verfügbar unter https://www. accenture. com/_acnmedia/pdf-96/accenture-2019-cost-ofcybercrime-study-final. pdf, 2019. 5

[23] Xing, Ying, Hui Shu, Hao Zhao, Dannong Li e Li Guo: *Survey on Botnet Detection Techniques: Classification, Methods, and Evaluation.* Mathematical Problems in Engineering, 2021, 2021, ISSN 15635147. 5, 14, 22

[24] Gu, Guofei, Roberto Perdisci, Junjie Zhang e Wenke Lee: *BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection.* Proceedings of the 17th USENIX Security Symposium, páginas 139–154, 2008. 8, 13, 14, 34, 40

[25] Rodríguez-Gómez, Rafael A, Gabriel Maciá-Fernández e Pedro García-Teodoro: *Survey and taxonomy of botnet research through life-cycle.* ACM Computing Surveys (CSUR), 45(4):1–33, 2013. 9, 10, 18, 19

[26] Stevanovic, Matija e Jm Pedersen: *Machine learning for identifying botnet network traffic.* Forskningsbasen.Deff.Dk, 2013. `http://forskningsbasen.deff.dk/Share.external?sp=S12d2f5d1-eba2-45f7-bc2a-cc7487941bd7&sp=Saau`. 10, 11, 12, 14, 15, 21, 22, 23, 35

[27] Khattak, Sheharbano, Naurin Rasheed Ramay, Kamran Riaz Khan, Affan A. Syed e Syed Ali Khayam: *A Taxonomy of botnet behavior, detection, and defense.* IEEE Communications Surveys and Tutorials, 16(2):898–924, 2014, ISSN 1553877X. 11, 12, 14

[28] Alieyan, Kamal, Ammar Almomani, Ahmad Manasrah e Mohammed M. Kadhum: *A survey of botnet detection based on DNS.* Neural Computing and Applications, 28(7):1541–1558, 2017, ISSN 09410643. 12, 21, 22

[29] Amini, Pedram, Muhammad Amin Araghizadeh e Reza Azmi: *A survey on Botnet: Classification, detection and defense.* Proceedings - 2015 International Electronics Symposium: Emerging Technology in Electronic and Information, IES 2015, páginas 233–238, 2016. 12

[30] Shetu, Syeda Farjana, Mohd Saifuzzaman, Nazmun Nessa Moon e Fernaz Narin Nur: *A survey of botnet in cyber security.* 2019 2nd International Conference on Intelligent Communication and Computational Techniques, ICCT 2019, páginas 174–177, 2019. 12

[31] Shinan, Khlood, Khalid Alsubhi, Ahmed Alzahrani e Muhammad Usman Ashraf: *Machine learning-based botnet detection in software-defined network: A systematic review.* Symmetry, 13(5):1–28, 2021, ISSN 20738994. 12, 13, 14, 21, 22

[32] Alauthaman, Mohammad, Nauman Aslam, Li Zhang, Rafe Alasem e M. A. Hossain: *A P2P Botnet detection scheme based on decision tree and adaptive multilayer neural networks*. Neural Computing and Applications, 29(11):991–1004, 2018, ISSN 09410643. 13, 35, 40

[33] Feily, Maryam, Alireza Shahrestani e Sureswaran Ramadass: *A survey of botnet and botnet detection*. Proceedings - 2009 3rd International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2009, páginas 268–273, 2009. 13

[34] Alothman, Basil: *Similarity-Based Instance Transfer Learning for Botnet Detection*. 9(1):880–889, 2018. 14, 25

[35] Kirubavathi, G. e R. Anitha: *Botnet detection via mining of traffic flow characteristics*. Computers and Electrical Engineering, 50:91–101, 2016, ISSN 00457906. http://dx.doi.org/10.1016/j.compeleceng.2016.01.012. 15

[36] Wang, Ping, Sherri Sparks e Cliff C. Zou: *An advanced hybrid peer-to-peer botnet*. IEEE Transactions on Dependable and Secure Computing, 7(2):113–127, 2010, ISSN 15455971. 15

[37] Plohmann, Daniel e Elmar Gerhards-Padilla: *Case study of the miner botnet*. Em *2012 4th International Conference on Cyber Conflict (CYCON 2012)*, páginas 1–16. IEEE, 2012. 15

[38] Andriesse, Dennis, Christian Rossow, Brett Stone-Gross, Daniel Plohmann e Herbert Bos: *Highly resilient peer-to-peer botnets are here: An analysis of gameover zeus*. Em *2013 8th International Conference on Malicious and Unwanted Software:" The Americas"(MALWARE)*, páginas 116–123. IEEE, 2013. 15

[39] Falliere, Nicolas: *Sality: Story of a peer-to-peer viral network*. Rapport technique, Symantec Corporation, 32, 2011. 15

[40] Mahjabin, Tasnuva, Yang Xiao, Guang Sun e Wangdong Jiang: *A survey of distributed denial-of-service attack, prevention, and mitigation techniques*. International Journal of Distributed Sensor Networks, 13(12):1550147717741463, 2017. 16, 17

[41] Mirchev, Mircho Jordanov e Seferin Todorov Mirtchev: *System for ddos attack mitigation by discovering the attack vectors through statistical traffic analysis*. International Journal of Information and Computer Security, 13(3-4):309–321, 2020. 16

[42] Peng, Tao, Christopher Leckie e Kotagiri Ramamohanarao: *Survey of network-based defense mechanisms countering the dos and ddos problems*. ACM Computing Surveys (CSUR), 39(1):3–es, 2007. 16

[43] Barford, Paul e Vinod Yegneswaran: *An inside look at botnets*. Em *Malware detection*, páginas 171–191. Springer, 2007. 16

[44] Banitalebi Dehkordi, Afsaneh, MohammadReza Soltanaghaei e Farsad Zamani Boroujeni: *The ddos attacks detection through machine learning and statistical methods in sdn*. The Journal of Supercomputing, 77(3):2383–2415, 2021. 16

[45] Garber, Lee: *Denial-of-service attacks rip the internet*. Computer, 33(04):12–17, 2000. 16

[46] Wang, An, Wentao Chang, Songqing Chen e Aziz Mohaisen: *A data-driven study of ddos attacks and their dynamics*. IEEE Transactions on Dependable and Secure Computing, 17(3):648–661, 2018. 16

[47] Saxena, Utkarsh, JS Sodhi e Yaduveer Singh: *An analysis of ddos attacks in a smart home networks*. Em *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, páginas 272–276. IEEE, 2020. 16

[48] Dobbins, Roland e Steinthor Bjarnason: *High-profile ddos extortion attacks — september 2020*. `https://www.netscout.com/blog/asert/high-profile-ddos-extortion-attacks-september-2020`, 2020. 16

[49] Cisco, The e Annual Internet: *Cisco: 2020 CISO Benchmark Report*, 2020. ISSN 1361-3723. 16

[50] Lesk, Michael: *The new front line: Estonia under cyberassault*. IEEE Security & Privacy, 5(4):76–79, 2007. 17

[51] Khan, Wazir Zada, Muhammad Khurram Khan, Fahad T Bin Muhaya, Mohammed Y Aalsalem e Han Chieh Chao: *A comprehensive study of email spam botnet detection*. IEEE Communications Surveys & Tutorials, 17(4):2271–2295, 2015. 17, 18

[52] THE RADICATI GROUP INC: *Email Statistics Report, 2021-2025*, 2021. `https://www.radicati.com/wp/wp-content/uploads/2020/12/Email-Statistics-Report-2021-2025-Executive-Summary.pdf`. 17

[53] Akinyelu, Andronicus A: *Advances in spam detection for email spam, web spam, social network spam, and review spam: Ml-based and nature-inspired-based techniques*. Journal of Computer Security, 29(5):473–529, 2021. 17

[54] Hossain, Fahima, Mohammed Nasir Uddin e Rajib Kumar Halder: *Analysis of optimized machine learning and deep learning techniques for spam detection*. Em *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, páginas 1–7. IEEE, 2021. 17

[55] Beek, Christiaan: *Necurs botnet leads the world in sending spam traffic*. `https://www.mcafee.com/blogs/other-blogs/mcafee-labs/necurs-botnet-leads-the-world-in-sending-spam-traffic/`, 2018. 17

[56] Jain, Ankit Kumar e Brij B Gupta: *Phishing detection: analysis of visual similarity based approaches*. Security and Communication Networks, 2017, 2017. 18

[57] Abdelhamid, Mohamed *et al.*: *The role of health concerns in phishing susceptibility: Survey design study*. Journal of medical Internet research, 22(5):e18394, 2020. 18

[58] Stavroulakis, Peter e Mark Stamp: *Handbook of information and communication security*. Springer Science & Business Media, 2010. 18

[59] Alabdan, Rana: *Phishing attacks survey: types, vectors, and technical approaches*. Future Internet, 12(10):168, 2020. 18

[60] Basit, Abdul, Maham Zafar, Xuan Liu, Abdul Rehman Javed, Zunera Jalil e Kashif Kifayat: *A comprehensive survey of ai-enabled phishing attacks detection techniques*. Telecommunication Systems, 76(1):139–154, 2021. 18

[61] Reports, Phishing E mail, Phishing Site Trends, Brand domain Pairs Measurement, E mail Phishing Attacks, Most Targeted e Industry Sectors: *Quarter*. (June):1–13, 2022. 18

[62] *IBM: X-Force Threat Intelligence Index*, 2022. ISSN 1361-3723. 18

[63] Porras, Phillip, Hassen Saidi e Vinod Yegneswaran: *A multi-perspective analysis of the storm (peacomm) worm*. Relatório Técnico, Technical report, Computer Science Laboratory, SRI International, 2007. 19

[64] Cimpanu, Catalin: *The satori botnet is mass-scanning for exposed ethereum mining rigs*, 2018. 19

[65] Xiao, Yang, Jing Liu, Kaveh Ghaboosi, Hongmei Deng e Jingyuan Zhang: *Botnet: Classification, attacks, detection, tracing, and preventive measures*. Eurasip Journal on Wireless Communications and Networking, 2009, 2009, ISSN 16871472. 19

[66] Ogu, Emmanuel C, Olusegun A Ojesanmi, Oludele Awodele e 'Shade Kuyoro: *A botnets circumspection: The current threat landscape, and what we know so far*. Information, 10(11):337, 2019. 19, 20

[67] Haddadi, Hamed: *Fighting online click-fraud using bluff ads*. ACM SIGCOMM Computer Communication Review, 40(2):21–25, 2010. 20

[68] Gu, Guofei, Junjie Zhang e Wenke Lee: *BotSniffer : Detecting Botnet Command and Control Channels in Network Traffic*. Proceedings of the 15th Annual Network and Distributed System Security Symposium., 53(1):1–13, 2008, ISSN 03601315. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.8092&rep=rep1&type=pdf`. 20

[69] Amoli, Payam Vahdani: *A Taxonomy of Botnet Detection Techniques Hossein Rouhani Zeidanloo , Moh amm ad Jorjor Zadeh M . Safari , Mazdak Zamani B . Intrusion Detection System (IDS)*. Industrial Engineering, páginas 158–162, 2010. `http://www.mendeley.com/research/a-taxonomy-of-botnet-detection-techniques/`. 21, 22

[70] GHAFIR, IBRAHIM, JAKUB SVOBODA e VACLAV PRENOSIL: *A Survey on Botnet Command and Control Traffic Detection*. 5(2):55–60, 2015. 21

[71] Gu, Guofei, Phillip Porras, Vinod Yegneswaran, Martin Fong e Wenke Lee: *BotHunter: Detecting malware infection through IDS-driven dialog correlation.* 16th USENIX Security Symposium, páginas 167–182, 2007. 21

[72] Zhang, Harry: *Exploring conditions for the optimality of naïve bayes.* International Journal of Pattern Recognition and Artificial Intelligence, 19(2):183–198, 2005, ISSN 02180014. 25

[73] Metsis, Vangelis, Ion Androutsopoulos e Georgios Paliouras: *Spam filtering with Naive Bayes - Which Naive Bayes?* 3rd Conference on Email and Anti-Spam - Proceedings, CEAS 2006, 2006. 25

[74] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay: *Scikit-learn: Machine learning in Python.* Journal of Machine Learning Research, 12:2825–2830, 2011. 25, 26, 27, 28, 29, 30, 31

[75] Da Luz, Pedro Marques: *Botnet detection using passive dns.* Radboud University: Nijmegen, The Netherlands, 2014. 25, 27

[76] Pandit, Shraddha, Suchita Gupta *et al.*: *A comparative study on distance measuring approaches for clustering.* International journal of research in computer science, 2(1):29–31, 2011. 26

[77] Hu, Li Yu, Min Wei Huang, Shih Wen Ke e Chih Fong Tsai: *The distance function effect on k-nearest neighbor classification for medical datasets.* SpringerPlus, 5(1):1–9, 2016. 26

[78] Bueno Silva, Luis Felipe, Luan Nunes Utimura, Kelton Augusto Pontara Da Costa, Marcia Aparecida Zanoli Meira E Silva e Simone Das Gracas Domingues Prado: *Study on Machine Learning Techniques for Botnet Detection.* IEEE Latin America Transactions, 18(5):881–888, 2020, ISSN 15480992. 26

[79] Pachghare, V. K. e Parag Kulkarni: *Pattern based network security using decision trees and support vector machine.* Em *2011 3rd International Conference on Electronics Computer Technology*, volume 5, páginas 254–257, 2011. 26

[80] Basheer, Imad A e Maha Hajmeer: *Artificial neural networks: fundamentals, computing, design, and application.* Journal of microbiological methods, 43(1):3–31, 2000. 27

[81] Chatterjee, Ankita, Jayasree Saha e Jayanta Mukherjee: *Clustering with multilayered perceptron.* Pattern Recognition Letters, 155:92–99, 2022. 27

[82] Maseer, Ziadoon Kamil, Robiah Yusof, Nazrulazhar Bahaman, Salama A Mostafa e Cik Feresa Mohd Foozy: *Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset.* IEEE access, 9:22351–22370, 2021. 27

[83] Funchal, Gustavo Silva: *Development of security mechanisms for a multi-agent cyber-physical conveyor system using machine learning.* Tese de Doutoramento, 2020. 27

[84] Cortes, Corinna e Vladimir Vapnik: *Support-vector networks.* Machine learning, 20(3):273–297, 1995. 28

[85] Vapnik, Vladimir: *The nature of statistical learning theory.* Springer science & business media, 1999. 28

[86] James, Gareth, Daniela Witten, Trevor Hastie e Robert Tibshirani: *An introduction to statistical learning*, volume 112. Springer, 2013. 28

[87] Breiman, Leo: *Random forests.* Machine learning, 45(1):5–32, 2001. 28

[88] Resende, Paulo Angelo Alves e André Costa Drummond: *A survey of random forest based methods for intrusion detection systems.* ACM Computing Surveys (CSUR), 51(3):1–36, 2018. 29

[89] Shu, Xinqing e Pan Wang: *An improved adaboost algorithm based on uncertain functions.* Em *2015 International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration*, páginas 136–139. IEEE, 2015. 29

[90] Rojas, Raúl *et al.*: *Adaboost and the super bowl of classifiers a tutorial introduction to adaptive boosting.* Freie University, Berlin, Tech. Rep, 2009. 29

[91] Hastie, Trevor, Saharon Rosset, Ji Zhu e Hui Zou: *Multi-class adaboost.* Statistics and its Interface, 2(3):349–360, 2009. 29

[92] Xiao, Yingchao, Huangang Wang e Wenli Xu: *Parameter selection of gaussian kernel for one-class svm.* IEEE transactions on cybernetics, 45(5):941–953, 2014. 29

[93] Kampmann, Geritt e Oliver Nelles: *One-class ls-svm with zero leave-one-out error.* Em *2014 IEEE Symposium on Computational Intelligence in Control and Automation (CICA)*, páginas 1–6. IEEE, 2014. 29

[94] Raj, Mehedi Hasan, ANM Asifur Rahman, Umma Habiba Akter, Khayrun Nahar Riya, Anika Tasneem Nijhum e Rashedur M Rahman: *Iot botnet detection using various one-class classifiers.* Vietnam Journal of Computer Science, 8(02):291–310, 2021. 29, 30

[95] Maglaras, Leandros A e Jianmin Jiang: *A novel intrusion detection method based on ocsvm and k-means recursive clustering.* EAI Endorsed Transactions on Security and Safety, 2(3):e5–e5, 2015. 30

[96] Liu, Fei Tony, Kai Ming Ting e Zhi Hua Zhou: *Isolation forest.* Em *2008 eighth ieee international conference on data mining*, páginas 413–422. IEEE, 2008. 30, 31

[97] Xu, Siying, Huiyi Liu, Liting Duan e Wenjing Wu: *An improved lof outlier detection algorithm.* Em *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, páginas 113–117. IEEE, 2021. 31

[98] Breunig, Markus M, Hans Peter Kriegel, Raymond T Ng e Jörg Sander: *Lof: identifying density-based local outliers*. Em *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, páginas 93–104, 2000. 31

[99] García, S., M. Grill, J. Stiborek e A. Zunino: *An empirical comparison of botnet detection methods*. Computers and Security, 45:100–123, 2014, ISSN 01674048. 33, 44

[100] L. Bilge, D. Balzarotti, W. Robertson E. Kirda e C. Kruegel: *DISCLOSURE: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis*. Annals of Internal Medicine, 152(11):751–752, 2010, ISSN 15393704. 34, 40, 41

[101] Beigi, Elaheh Biglar, Hossein Hadian Jazi, Natalia Stakhanova e Ali A. Ghorbani: *Towards effective feature selection in machine learning-based botnet detection approaches*. 2014 IEEE Conference on Communications and Network Security, CNS 2014, páginas 247–255, 2014. 35, 40, 60

[102] Chen, Ruidong, Weina Niu, Xiaosong Zhang, Zhongliu Zhuo e Fengmao Lv: *An Effective Conversation-Based Botnet Detection Method*. Mathematical Problems in Engineering, 2017, 2017, ISSN 15635147. 35, 40, 44

[103] Shiravi, Ali, Hadi Shiravi, Mahbod Tavallaee e Ali A. Ghorbani: *Toward developing a systematic approach to generate benchmark datasets for intrusion detection*. Computers and Security, 31(3):357–374, 2012, ISSN 01674048. `http://dx.doi.org/10.1016/j.cose.2011.12.012`. 36, 45, 46

[104] Resende, Paulo Angelo Alves e André Costa Drummond: *HTTP and contact-based features for Botnet detection*. Security and Privacy, 1(5):e41, 2018. 36, 40

[105] Gadelrab, Mohammed S., Muhammad ElSheikh, Mahmoud A. Ghoneim e Mohsen Rashwan: *BotCap: Machine learning approach for botnet detection based on statistical features*. International Journal of Communication Networks and Information Security, 10(3):563–579, 2018, ISSN 2073607X. 36, 40

[106] Khan, Riaz Ullah, Xiaosong Zhang, Rajesh Kumar, Abubakar Sharif, Noorbakhsh Amiri Golilarz e Mamoun Alazab: *An adaptive multi-layer botnet detection technique using machine learning classifiers*. Applied Sciences (Switzerland), 9(11), 2019, ISSN 20763417. 37, 40, 44

[107] Campazas-Vega, Adrián, Ignacio Samuel Crespo-Martínez, Ángel Manuel Guerrero-Higueras e Camino Fernández-Llamas: *Flow-data gathering using netflow sensors for fitting malicious-traffic detection models*. Sensors (Switzerland), 20(24):1–13, 2020, ISSN 14248220. 41

[108] Alenazi, Abdelraman, Issa Traore, Karim Ganame e Isaac Woungang: *Holistic Model for HTTP Botnet Detection Based on DNS Traffic Analysis*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10618 LNCS:1–18, 2017, ISSN 16113349. 45

[109] *The honeynet project*. `https://www.honeynet.org/`. 46

[110] Szabó, Géza, Dániel Orincsay, Szabolcs Malomsoky e István Szabó: *On the validation of traffic classification algorithms*. Em *International conference on passive and active network measurement*, páginas 72–81. Springer, 2008. 46

[111] *Berkeley lab bringing science solutions to the world*. `https://www.lbl.gov/`. 46

[112] Stratosphere, T: *Malware capture facility project*, 2015. `https://www.stratosphereips.org/datasets-malware`. 46

[113] Lashkari, Arash Habibi, Gerard Draper Gil, Mohammad Saiful, Islam Mamun e Ali A Ghorbani: *Characterization of Tor Traffic using Time based Features*. (Cic):253–262, 2017. 46

[114] Cawley, Gavin C e Nicola LC Talbot: *On over-fitting in model selection and subsequent selection bias in performance evaluation*. The Journal of Machine Learning Research, 11:2079–2107, 2010. 50

[115] Brzezinski, Dariusz e Jerzy Stefanowski: *Prequential auc: properties of the area under the roc curve for data streams with concept drift*. Knowledge and Information Systems, 52(2):531–562, 2017. 51

[116] Sharafaldin., Iman, Arash Habibi Lashkari. e Ali A. Ghorbani.: *Toward generating a new intrusion detection dataset and intrusion traffic characterization*. Em *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP,*, páginas 108–116. INSTICC, SciTePress, 2018, ISBN 978-989-758-282-0. 85

[117] Gonzalez-Cuautle, David, Aldo Hernandez-Suarez, Gabriel Sanchez-Perez, Karina Toscano, Jose Portillo-Portillo, Jesus Olivares Mercado, Hector Perez-Meana e Ana Sandoval-Orozco: *Synthetic minority oversampling technique for optimizing classification tasks in botnet and intrusion-detection-system datasets*. Applied Sciences, 10:794, janeiro 2020. 85

[118] Lopes, Daniele AG, Marcelo A Marotta, Marcelo Ladeira e João JC Gondim: *Botnet detection based on network flow analysis using inverse statistics*. Em *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, páginas 1–6. IEEE, 2022. 108

[119] Lopes, Daniele, Marcelo Marotta, Marcelo Ladeira e João Gondim: *Detecção de botnets baseada na análise de fluxos de rede utilizando estatística inversa*. Em *Anais do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, páginas 182–195, Porto Alegre, RS, Brasil, 2022. SBC. `https://sol.sbc.org.br/index.php/sbrc/article/view/21170`. 108