Implementação de novas métricas para avaliação de classificadores de *botnets* dentro do *framework* MOA

Pedro Victor Guerra de Figueiredo

Universidade Federal de Uberlândia
Faculdade de Computação

Uberlândia
2021

Pedro Victor Guerra de Figueiredo

Implementação de novas métricas para avaliação de classificadores de *botnets* dentro do *framework* MOA

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Rodrigo Sanches Miani

Uberlândia
2021

Aos meus pais, Altamir Gomes de Figueiredo e Edilamar Guerra,
a minha esposa, Mariana Barbosa Silva.

Agradecimentos

Agradeço aos meus pais, que sempre se preocuparam e passaram por sacrifícios para garantir que pudesse ter uma boa formação pessoal e profissional e sempre apoiar as escolhas que fiz durante a vida.

A minha melhor amiga e companheira, minha esposa Mariana, que nunca deixou de estar ao meu lado e me incentivar a concluir esse capítulo de minha vida que foi agraduação.

A todos os professores que participaram de minha formação, despertando o amor e interesse pela ciência da computação.

Ao meu orientador, Professor Doutor Rodrigo Miani, pela empatia em um momento complicado em minha vida pessoal e que sem o incentivo e apoio não teria conseguido finalizar a graduação.

Resumo

Com a popularização da Internet, vivemos em uma sociedade interconectada, ou seja, cada vez mais dispositivos presentes em nosso cotidiano estão conectados e expostos em uma rede globalizada. Vários destes dispositivos não são desenvolvidos com os requisitos de segurança em mente, tornando-os alvos para ataques de pessoas mal intencionadas. Estes ataques podem acontecer de diversas maneiras. Uma categoria de ameaça presente nas redes de computadores são as chamadas *botnets*, dispositivos contaminados por *software* malicioso controlados por um mestre a fim de atingir seus objetivos. Um dos caminhos para a detecção e combate às *botnets* acontece por sistemas de detecção de intrusão (IDS - *Intrusion Detection Systems*). Comparar métodos de detecção de *botnets* não é uma tarefa simples, geralmente por falta de documentação dos métodos utilizados e um conjunto de dados rotulado em comum entre as várias pesquisas. Este trabalho se propôs a realizar a implementação de novas métricas para análise dos métodos de detecção proposta por (GARCíA et al., 2014) dentro de uma plataforma de código aberto chamada MOA (*Massive Online Analysis*), que reúne diversos algoritmos de aprendizado de máquina em fluxos de dados contínuos. Como resultado, agora é possível analisar o desempenho de classificadores de *botnets* de outra maneira, expandindo as possibilidades para o campo.

Palavras-chave: Segurança da informação, Sistemas de detecção de intrusão, Mineração de dados, Mineração em fluxos de dados, Aprendizado de máquina, Botnets.

Lista de ilustrações

Lista de tabelas

Sumário

Capítulo 1

Introdução

Após a abertura para uso comercial da Internet, ela vem se fazendo mais presente no dia a dia de milhões de pessoas mundo afora. Com os recentes avanços de *hardware*, um novo paradigma conhecido como Internet das Coisas (IoT - *Internet of Things*) foi criado. A ideia da IoT, é que dispositivos diversos (geralmente com poucos recursos disponíveis) estejam conectados à Internet, trocando informações entre si, de modo a operarem em conjunto.

Toda essa conectividade oferece vários benefícios e ainda é uma área em crescente expansão, mas nos traz novos problemas relacionados à segurança da rede. Afinal com mais pontos de conexão, se aumenta a superfície que pode estar exposta a ataques. Para (STALLINGS, 2015), o campo de redes e segurança da Internet consiste em medidas para deter, prevenir, detectar e corrigir violações de segurança que envolvam a transmissão de informação. Segundo a RFC 4949 (SHIREY, 2007), pode se definir ameaças e ataques à segurança de um sistema da seguinte maneira:

- **Ameaça**: Uma oportunidade de violação da segurança que existe quando há uma circunstância, capacidade, ação ou evento que poderia quebrar a segurança e causar danos. Ou seja, uma ameaça é um possível perigo a explorar uma vulnerabilidade.
- **Ataque**: Um ataque à segurança do sistema, derivado de uma ameaça inteligente; ou seja, um ato inteligente que é uma tentativa deliberada (especialmente no sentido de um método ou técnica) de fugir dos serviços de segurança e violar a política de segurança de um sistema.

Uma ameaça mais recente aos sistemas conectados à Internet são as *botnets*, redes de máquinas contaminadas por programas maliciosos e são controladas por um *botmaster*. Essas redes foram criadas para conduzir atividades ilegais em larga escala (SILVA et al., 2013). Dispositivos IoT são alvos perfeitos para a contaminação por essas *botnets*, pois muitas vezes são desenvolvidos sem a devida atenção à segurança do aparelho, o que os torna vulneráveis a ataques e, consequentemente, ao comando de *botnets*. (SILVA et al., 2020).

Uma das soluções mais comuns para detecção de *botnets* consiste em desenvolver sistemas para analisar o tráfego de rede e identificar os componentes maliciosos. Tais sistemas são chamados Sistemas de Detecção de Intrusão (*IDS - Intrusion Detection System*). A classificação do tráfego malicioso por IDS pode ser feita com o auxílio de bases de dados de ataques conhecidos ou usando técnicas para determinar comportamentos anômalos presentes nos pacotes de rede (RIBEIRO, 2020).

Este trabalho tem enfoque nas técnicas que utilizam de aprendizado de máquina para a detecção de anomalias no tráfego de rede. O emprego de técnicas tradicionais de aprendizado de máquina, onde se usa um conjunto de dados para treinar um modelo de decisão e depois fazer previsões para novas instâncias, acaba não sendo exatamente adequado para a detecção de *botnets*, visto que o comportamento do tráfego de rede muda de forma contínua. Sendo assim, o problema de detecção de *botnets* através do reconhecimento de anomalias no tráfego de rede parece ser um exemplo perfeito para se utilizar mineração em Fluxos Contínuos de Dados (FCD's), ou, no inglês *Data Streams*.

Comparar os métodos de detecção de *botnets* não é uma tarefa fácil, entre os motivos estão a falta de documentação dos modelos, de um conjunto de dados rotulado em comum e de metologia para a comparação dos métodos e métricas de erro adequadas (GARCíA et al., 2014).

1.1 Objetivos

O objetivo deste trabalho é realizar a implementação das novas métricas para avaliação de modelos de classificação propostas por (GARCíA et al., 2014) dentro do *framework opensource Massive Online Analysis* (MOA).

O MOA é um *framework opensource* para algoritmos de mineração de fluxos contínuos de dados escrito em java que inclui uma coleção de algoritmos de aprendizado de máquina (classificação, regressão, agrupamento, detecção de anomalia, detecção de desvio de conceito e sistemas de recomendação) e ferramentas para avaliação.

De acordo com García et al. (2014), as métricas usadas na maioria dos trabalhos geralmente não são homogêneas, eles tendem a usar métricas de erro diferentes e definição de erro diferente. Além disso, as métricas de erro mais comuns como, por exemplo TFP (Taxa de Falsos Positivos) não são suficientes para realizar a comparação entre métodos de detecção de *botnets*, as métricas de erro tradicionais foram definidas a partir de um ponto de vista estatístico e falham em atender as necessidades que um administrador de redes possui (GARCíA et al., 2014). As novas métricas propostas por (GARCíA et al., 2014) serão explicadas em detalhe no Capítulo 3.

1.2 Organização do trabalho

O trabalho se divide em cinco capítulos, sendo eles Introdução 1, Fundamentação teórica 2, Desenvolvimento 3, Experimentos 4 e Conclusão 5. A Introdução busca motivar e contextualizar o leitor sobre o trabalho conforme a literatura, bem como mostrar os objetivos e organização do trabalho. A Fundamentação Teórica exibe o estado da arte e apresenta ao leitor os fundamentos e conceitos de segurança da informação, *botnets*, mineração de dados, mineração de fluxo contínuos de dados e medidas tradicionais de avaliação. Por fim, há os trabalhos relacionados a este. No capítulo Desenvolvimento, será detalhado o passo a passo para implementação das novas métricas de erro dentro do MOA. O capítulo de Experimentos apresenta os resultados obtidos durante o trabalho e, no capítulo seguinte, as conclusões obtidas e propostas para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

O capítulo, dividido em cinco seções, apresenta os principais conceitos para uma boa leitura do presente trabalho. A Seção 2.1 explica a origem e necessidade da segurança da informação com seus pilares. Já na Seção 2.2, é introduzido o conceito e as principais categorias de *botnets*. Na Seção 2.3, se apresenta o conceito de mineração de dados e sua forma tradicional, cenários de mineração em lote. A Seção seguinte 2.4, discute sobre o paradigma de se aplicar conceitos da mineração a fluxos contínuos de dados. A Seção 2.5 apresenta as medidas tradicionais para avaliação de algoritmos de aprendizado de máquina. E, para finalizar, a Seção 2.6 expõe os trabalhos relacionados a este.

2.1 Segurança da informação

Abrindo a Internet para uso comercial no começo dos anos 90 ficou evidente como era necessário um conjunto de políticas de segurança para transações remotas. Dados sensíveis passaram a ter que ser protegidos em trânsito, e os nós da Internet tiveram que ser protegidos de acessos indesejáveis (GOLLMANN, 2010). Desde então, o volume de informação gerada anualmente pelos mais variados sistemas não para de crescer. A necessidade que já se evidenciava no começo da década de 90, hoje se torna um imperativo.

Para Stallings (2015), o campo de redes e segurança da Internet consiste em medidas para deter, prevenir, detectar e corrigir violações de segurança que envolvam a transmissão de informação. O NIST define segurança de computadores como: "A proteção proporcionada a um sistema de informação automatizado para atingir os objetivos aplicáveis de preservar a integridade, disponibilidade, e confidencialidade dos recursos do sistema de informação."

Os três pilares da segurança de computadores são detalhados a seguir (STALLINGS, 2015):

- **Confidencialidade:** este item se desdobra em mais dois:

  - **Confidencialidade de dados**: garante que informações privadas ou confidenciais não são acessíveis por indivíduos não autorizados.
  - **Privacidade**: garante que indivíduos tenham controle ou influência sobre quais informações relacionadas a eles podem ser coletadas e armazenadas e por quem e para quem essas informações podem ser reveladas.

- **Integridade:** este item se desdobra em mais dois:
  - **Integridade de dados**: garante que informação (tanto armazenada quanto transmitida) e programas somente são alterados de uma maneira específica e autorizada.
  - **Integridade do sistema**: garante que o sistema opera conforme sua concepção livre de interferências não autorizadas.

- **Disponibilidade:** assegura que o sistema opere prontamente e que não haja indisponibilidade do mesmo para usuários autorizados.

Os três conceitos formam a chamada tríade CIA (do acrônimo para *confidentiality, integrity and availability*). Os conceitos englobam objetivos essenciais de segurança para serviços de informação e computação (STALLINGS, 2015). Apesar de bem estabelecido, alguns, na área de segurança sentem a necessidade de conceitos adicionais, sendo os mais recorrentes, autenticidade e responsabilização. Segue o detalhamento:

- **Autenticidade**: é a propriedade de ser genuíno e a capacidade de ser verificado e confiável; confiança na validação da transmissão de uma mensagem ou na fonte da mesma. Isso significa verificar que usuários são quem eles realmente dizem ser e que as mensagens recebidas chegam de uma fonte confiável.

- **Responsabilização**: é um requerimento de segurança para que ações de uma entidade só possam ser ligadas unicamente a mesma.

2.2 Botnets

Botnets são redes de computadores infectadas por um mesmo *malware* e estão sob o controle de seu criador, o *botmaster*. *Botnets* podem ser utilizadas para uma miríade de ataques à segurança de sistemas, como, por exemplo, lançamento de ataque distribuídos de negação de serviço (*Distributed Denial Of Service - DDoS*), e obtendo lucro com a exploração de informações de usuários normais (NOGUEIRA, 2016). Uma rede de *botnets* normalmente tem a seguinte estrutura:

- ***Bot***: dispositivo alvo infectado pelo código malicioso.

- ***Botmaster***: é o usuário que detém o controle da rede de *bots*.

- **Servidor de Comando e Controle (C&C)**: meio pelo qual o botmaster envia os comandos para os outros *bots*.

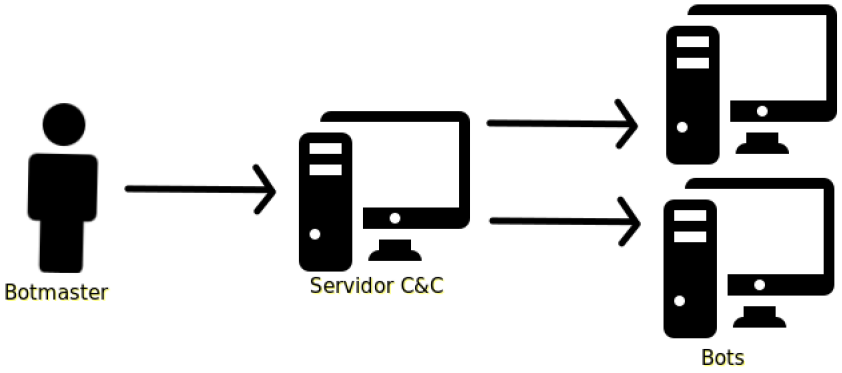

Figura 1 – Arquitetura de uma *botnet*. Adaptado de (RIBEIRO, 2020).

Existem várias classificações de *botnets*, dependendo do protocolo utilizado pelo servidor de C&C. Da mesma forma, também existem várias classificações de *botnets* que não utilizam um servidor de C&C, isto é, que trabalham de maneira descentralizada por protocolos *peer-to-peer* (P2P) (SILVA et al., 2020). As principais topologias das *botnets* são:

- **Centralizada**: baseadas no modelo clássico de rede, o cliente e o servidor. Nessa topologia, todos os *bots* irão estabelecer seu canal de comunicação com um, ou poucos, pontos de conexão, que normalmente são os servidores de C&C. Suas vantagens são rápido tempo de reação e boa coordenação. Entretanto, por possuírem um C&C, ele acaba se tornando seu principal ponto de falha e pode inclusive ajudar na identificação da *botnet* justamente por todos os *bots* se conectarem a um ou poucos pontos.

- **Descentralizada**: comumente é baseada no protocolo P2P. De muito mais difícil combate, visto que descobrir alguns *bots* não necessariamente derruba a rede, dado que ela não possui um ponto de controle.

- **Híbrida**: é uma combinação das duas outras topologias. Como exemplo há o uso de protocolo P2P com *superpeers* (é um nó em uma rede P2P que opera tanto como um servidor para um conjunto de clientes, quanto como um igual em uma rede de *superpeers*).

2.3 Mineração de dados

Mineração de dados é o processo de automaticamente descobrir informações úteis em largos repositórios de dados (KUMAR et al., 2018). As técnicas de mineração de dados podem ser aplicadas em largos conjuntos de dados para se encontrar novos e úteis padrões que poderiam continuar desconhecidos. Elas também podem fornecer a capacidade de se fazer previsões (KUMAR et al., 2018). Além disso, o campo de mineração de dados nasce de uma intersecção de várias outras áreas, que pode ser visualizado na Figura 2.

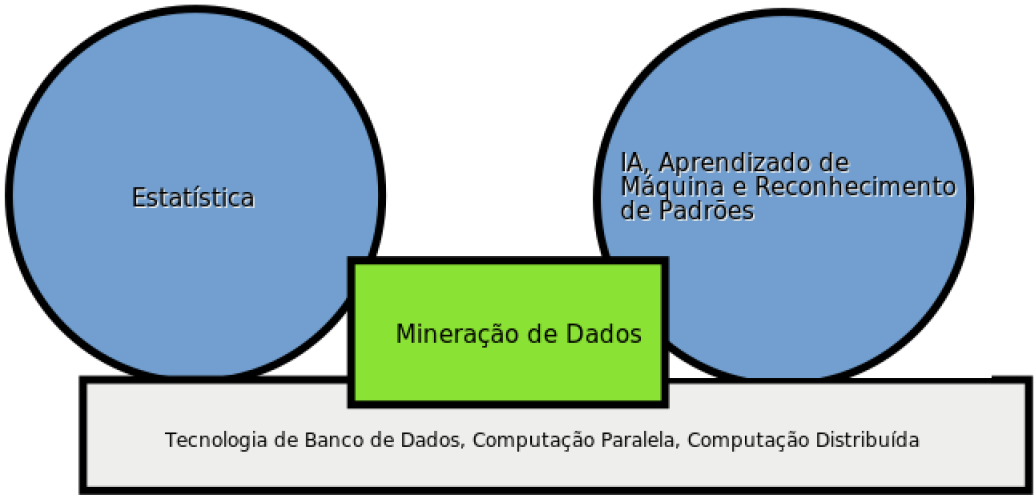

Figura 2 – Mineração de dados e sua relação com outras disciplinas. Adaptado de (KUMAR et al., 2018).

Mineração de dados é parte integral do *Knowledge Discovery in Databases* (KDD, descoberta de conhecimento em bancos de dados), que é o processo geral de converter dados em informação útil. A Figura 3 ilustra os passos do KDD.

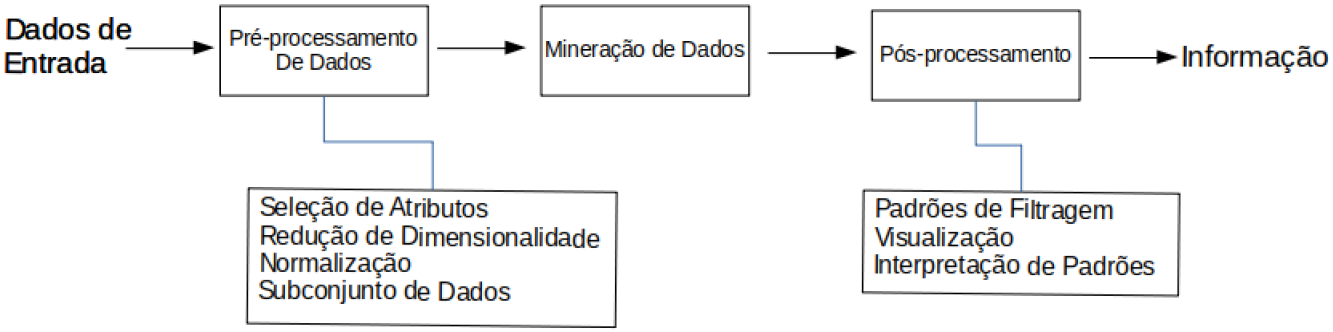

Figura 3 – Descoberta de conhecimento em banco de dados. Adaptado de (KUMAR et al., 2018).

- **Pré-processamento de Dados**: os dados podem estar nos mais variados formatos e padrões. A ideia de fazer um pré-processamento nos dados é justamente preparar esses dados de entrada para poderem ser aplicadas as técnicas de mineração.
- **Mineração de Dados**: a etapa onde vão ser aplicadas as técnicas de mineração nos dados de entrada traduzidos para o formato esperado. Geralmente as tarefas aqui executadas são dividas em duas categorias principais:
  - **Tarefas preditivas**: o objetivo dessas tarefas é prever o valor de um atributo específico baseado nos valores de outros atributos.
  - **Tarefas descritivas**: o objetivo é derivar padrões(correlações, tendências, *clusters*, trajetórias e anomalias) que sumarizem os relacionamentos subjacentes nos dados.
- **Pós-processamento**: fase em que algum processamento pode ser aplicado em cima do resultado da mineração, como, por exemplo decidir se a saída da etapa anterior será integrada em algum modelo.

2.4 Mineração de fluxos contínuos de dados

Segundo (GAMA, 2010) a mineração de dados por muitos anos focou trabalhar em cenários em lote, onde um conjunto de dados para treinamento está disponível para o algoritmo, que produz um modelo de decisão como saída e nunca mais é alterado. Este modelo gerado será utilizado para fazer predições assim que novos exemplos estiverem disponíveis.

A maioria dos algoritmos de mineração de dados (MD) assumem que o conjunto de dados usado para o treinamento é finito, representativo e contempla todas as classes do problema, que os dados são gerados por uma distribuição de probabilidade estacionária e estes poderem ser armazenados fisicamente e analisados por um algoritmo que executa em múltiplos passos (PAIVA, 2014). O que não é suficiente para fluxos contínuos de dados (FCDs), visto que os dados fluem continuamente temporalmente, e a distribuição que gera os exemplos pode mudar com o passar do tempo, ou seja, não é estacionária (PAIVA, 2014). Em mineração de dados em FCDs, o problema que surge é o de manter um modelo de decisão com boa acurácia. Sendo assim, neste cenário o algoritmo deve ser incremental, ou seja, continuar aprendendo conforme novos exemplos forem chegando. Um algoritmo para MD em FCDs deve ter em mente de que a quantidade de dados gerados geralmente é grande e os exemplos não devem ser armazenados, mas utilizados e em seguida descartados (PAIVA, 2014). Outro ponto de atenção é de que como o fluxo de dados é potencialmente infinito o modelo evolui com o passar tempo, sendo necessário esquecer os exemplos do passado e concentrar-se nos exemplos do presente.

Seguem as diferenças entre FCDs e dados convencionais (BABCOCK et al., 2002):

- ❏ Os dados chegam de forma contínua, e em geral, em alta velocidade.
- ❏ O sistema não tem controle sobre a ordem onde os exemplos chegam para serem processados.
- ❏ FCDs têm tamanho ilimitado.
- ❏ Dado que um exemplo foi processado, ele é descartado ou armazenado, não podendo ser recuperado facilmente, a não ser que seja explicitamente armazenado na memória, que é relativamente pequena comparada com o tamanho do FCD.

2.5 Medidas de avaliação

Uma das maneiras mais comuns para a avaliação de classificadores, é se utilizar da chamada matriz de confusão, onde cada linha corresponde as classes reais e as colunas fazem referência as predições realizadas. Um classificador perfeito teria somente positivos e falsos verdadeiros, vale lembrar também que tn + fp + fn + tp = n, onde n é o tamanho da amostra de testes (ELKAN, 2012). Um exemplo de matriz de confusão pode ser encontrado na Figura 4.

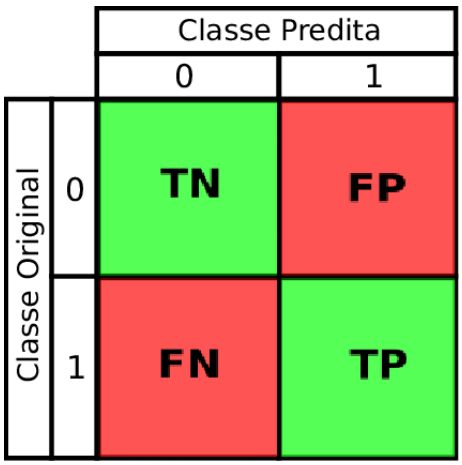

Figura 4 – Matriz de confusão.

A matriz de confusão pode ser obtida aplicando-se um classificador em uma amostra para testes, com os valores em mão, várias estatísticas podem sem extraídas a partir da matriz, entre elas:

$$\text{acurácia} = \frac{(tp + tn)}{n} \tag{1}$$

$$\text{precisão} = \frac{tp}{(tp + fp)} \tag{2}$$

$$\text{revocação} = \frac{tp}{(tp + fn)} \tag{3}$$

Muitas vezes também é conveniente combinar precisão e revocação em uma única métrica chamada F1. Essa métrica é a média harmônica entre a precisão e a revocação, onde a média regular trataria todos os valores igualmente, a média harmônica dá muito mais peso para valores baixos. Como resultado, o classificador só obterá um valor alto para F1 se tanto a precisão quanto a revocação são altos (GéRON, 2019).

$$\text{F1} = 2 \times \frac{precisão \times revocação}{precisão + revocação} \tag{4}$$

2.6 Trabalhos relacionados

Esta seção traz trabalhos anteriores relacionados com o presente trabalho. Em (COSTA et al., 2018), foi proposta uma abordagem baseada em rede para detecção de *botnets* analisando fluxos de rede bidirecionais usando a *Very Fast Decision Tree* (VFDT). A proposta se divide em duas etapas: criação das instâncias e detecção das *botnets*. Informações relacionadas com o endereço IP foram retiradas dos exemplos para evitar enviesamento, os experimentos foram executados nos conjuntos de dados CTU (GARCíA et al., 2014) em todos os cenário exceto o 5 e 7. Foram calculadas a precisão, revocação e a *False Positive Rate* (FPR) para todos os cenários. O custo de memória foi bem baixo com uma média de 0.272MB em todos os cenários.

Em Ribeiro, Paiva e Miani (2020), foi feito um estudo de comparação de desempenho de algoritmos de MD em FCDs, também com uma proposta baseada em rede, aplicados ao mesmo conjunto de fluxos bidirecionais do trabalho anterior, porém com um limitante n de máximo de instâncias utilizadas no treinamento do modelo. Foram avaliados classificadores de duas categorias *Single* e *Ensemble*. Fazendo parte do conjunto dos classificadores sozinhos estão a VFDT e a *Hoeffding Adaptive Tree*. Fazem parte do conjunto *Ensemble*, *Ozabag*, *Ozaboost* e *Oza Bagging With Adwin*. O trabalho conclui que classificadores *Ensemble* tiveram um desempenho melhor que os *Single* e é possível pensar detectores de *botnets* que considerem tanto desempenho quanto o número de instâncias rotuladas.

Em García et al. (2014) também é proposto um estudo para comparação de algoritmos para detecção de *botnets*, foram comparados 3 métodos de detecção, sendo eles o CAMNEP, BCLus e *BotHunter*. Uma grande adição desse trabalho foi criar um conjunto de dados para realização dos testes para comparação, visto que essa é uma das maiores dificuldades em se comparar algoritmos para detecção de *botnets*, uma base de dados real em que vários trabalhos possam operar em cima. Também foram sugeridas novas métricas de erro para comparação dos métodos. O estudo conclui que a comparação dos métodos

utilizando uma base de dados real ajudou muito a melhorar a pesquisa dos autores. Também é apontado como é interessante para o campo todo definir melhor uma metodologia de comparação dos métodos e métricas de erros melhores.

O objetivo deste trabalho é que, com a implementação das métricas propostas por (GARCíA et al., 2014) dentro da plataforma *opensource* MOA, seja possível analisar trabalhos como os de (RIBEIRO; PAIVA; MIANI, 2020) e (COSTA et al., 2018) de outra forma, levando em consideração os computadores infectados ao invés de simplesmente avaliar a quantidade de fluxos maliciosos que um determinado modelo de classificação acertou.

Capítulo 3

Desenvolvimento

O presente capítulo pretende detalhar o processo de desenvolvimento do trabalho. A Seção 3.1 explica as novas métricas para comparação entre métodos de detecção de *botnets* que foi proposta por (GARCíA et al., 2014). A Seção 3.2 explica o passo a passo para implementação das métricas dentro do MOA.

3.1 Novas Métricas

De acordo com García et al. (2014), realizar a comparação entre métodos de detecção de *botnets* nem sempre é uma tarefa simples, pois a falta de: i) documentação sobre os métodos, ii) conjuntos de dados rotulados abertos ao público, iii) metodologia para comparação e iv) métricas de erro apropriadas; são fatores que impedem esse processo de comparação. No trabalho (GARCíA et al., 2014), foi criado um conjunto de dados público com tráfego real de rede (CTU-13), que possui fluxos rotulados como normais, de fundo e *botnet*. Além disso, foram propostas métricas de erro mais adequadas para um administrador de rede, visto que as métricas de erro normalmente usadas para analisar os resultados foram pensadas por um ponto de vista estatístico. Os autores propuseram o seguinte conjunto de princípios para fundamentar as novas métricas:

- Erros devem considerar endereços IP ao invés de fluxos de rede.
- É melhor detectar um endereço IP de uma *botnet*(TP) cedo do que tarde.
- Não detectar um endereço IP de uma *botnet*(FN) cedo é pior do que tarde.
- O valor de detectar um endereço normal de IP(TN) não é afetado pelo tempo.
- O valor de não detectar um endereço normal de IP(FP) não é afetado pelo tempo.

Como o objetivo era atender os princípios listados acima, foi necessário incorporar o tempo no cálculo das métricas. Isso foi feito através da noção de uma janela de tempo

para comparações. Estas janelas de tempo somente são utilizadas no cálculo das métricas e não interferem no modelo de detecção.

Também era necessário atrelar as medidas de erro tradicionais (TP, FP, TN, FN) ao endereço IP. Para isso novas métricas foram definidas da seguinte forma:

- **cTP:** Um verdadeiro positivo deve ser contado quando um endereço IP de uma *botnet* é detectado como *botnet* pelo menos uma vez durante uma janela de tempo.
- **cTN:** Um verdadeiro negativo deve ser contado quando um endereço IP normal é detectado como não *botnet* durante toda a janela de tempo.
- **cFP:** Um falso positivo deve ser contado quando um endereço IP normal é detectado como um endereço de uma *botnet* pelo menos uma vez durante a janela de tempo.
- **cFN:** Um falso negativo deve ser contado quando um endereço IP de uma *botnet* é detectado como um endereço normal durante toda a janela de tempo.

Também foi necessária a definição de uma função baseada no tempo para correção dos valores acumulados durante a janela de tempo:

$$\text{função de correção} = e^{-\alpha \times n} + 1 \tag{5}$$

onde n será a janela de tempo atual no momento do cálculo das métricas finais. Com essa função de correção foram criadas quatro variáveis dependentes do tempo. Elas são computadas da seguinte forma:

- **tTP:**

$$\frac{cTP \times \text{função de correção}}{\text{Quantidade de endereços IP de } botnets \text{ únicos na janela de tempo}} \tag{6}$$

- **tFN:**

$$\frac{cFN \times \text{função de correção}}{\text{Quantidade de endereços IP de } botnets \text{ únicos na janela de tempo}} \tag{7}$$

- **tFP:**

$$\frac{cFP}{\text{Quantidade de endereços IP normais únicos na janela de tempo}} \tag{8}$$

- **tTN:**

$$\frac{cTN}{\text{Quantidade de endereços IP normais únicos na janela de tempo}} \tag{9}$$

Com os valores corrigidos, atrelados tanto ao tempo quanto aos endereços IP agora é possível realizar o cálculo das medidas tradicionais.

- **FPR:**
$$\frac{tFP}{tTN + tFP} \tag{10}$$
- **TPR:**
$$\frac{tTP}{tTP + tFN} \tag{11}$$
- **TNR:**
$$\frac{tTN}{tTN + tFP} \tag{12}$$
- **FNR:**
$$\frac{tFN}{tFN + tTP} \tag{13}$$
- **Precisão:**
$$\frac{tTP}{tTP + tFP} \tag{14}$$
- **Acurácia:**
$$\frac{tTP + tTN}{tTP + tTN + tFP + tFN} \tag{15}$$
- **Taxa de erro:**
$$\frac{tFP + tFN}{tTP + tTN + tFP + tFN} \tag{16}$$
- ***F1*:**
$$2 \times \frac{Precisão \times TPR}{Precisão + TPR} \tag{17}$$

3.2 Implementação no MOA

Para realizar a implementação das novas métricas no MOA, primeiro foi feito um estudo de um código escrito em Python disponibilizado pelo próprio autor. [1]

Em sua implementação, os autores criaram uma classe para representar cada janela de tempo para comparação e uma classe chamada *algorithms* responsável pelo cálculo final das estatísticas para cada algoritmo. Como a ideia deste trabalho é conseguir utilizar as novas métricas propostas por (GARCíA et al., 2014) no escopo de algoritmos de mineração de fluxos contínuos de dados, expandindo a avaliação de modelos feita em trabalhos anteriores como (RIBEIRO, 2020) e (OLIMPIO et al., 2021).

O trabalho de (RIBEIRO, 2020) se propõe a comparar diversos algoritmos de mineração em FCD's presentes no MOA, tanto algoritmos isolados como classificadores em conjunto. A Figura 5 demonstra em maior detalhe como é o funcionamento genérico de um classificador em conjunto.

A proposta pode ser verificada na Figura 6. Um conjunto específico de características é extraído de um conjunto de fluxos de tráfego de rede para a obtenção do conjunto de dados que será utilizado na comparação. Uma iteração começa quando a primeira

[1] https://sourceforge.net/projects/botnetdetectorscomparer/files/

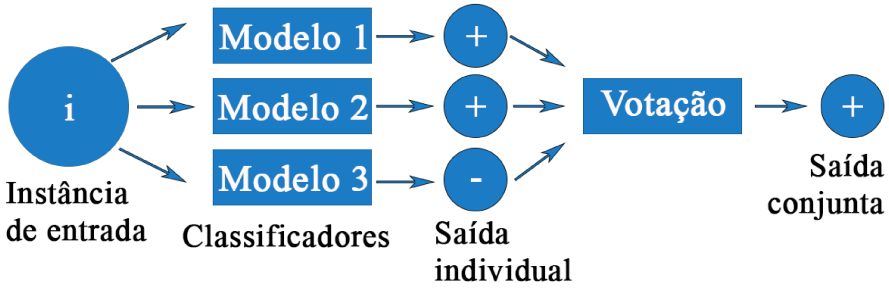

Figura 5 – Exemplo de um classificador em conjunto. Adaptado de (RIBEIRO; PAIVA; MIANI, 2020).

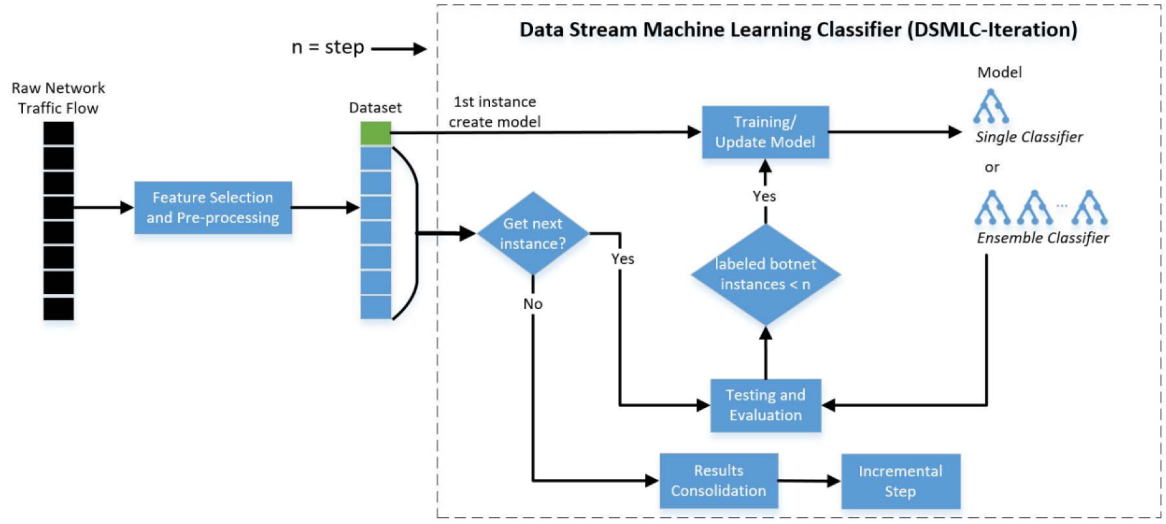

Figura 6 – Proposta de experimento. Extraído de (RIBEIRO; PAIVA; MIANI, 2020).

instância disponível inicia o algoritmo de mineração. A partir deste ponto o algoritmo já está pronto para fazer predições. Vale ressaltar que a próxima predição vai estar relacionada a única classe que ele aprendeu até o momento. Se outra instância estiver disponível o algoritmo irá fazer uma predição, este resultado é utilizado para computar as métricas de desempenho do mesmo. Se a instância possuir um rótulo verdadeiro sobre sua classe, esta então é utilizada para atualização do modelo. Esse processo de prover um rótulo verdadeiro irá acontecer até que se alcance um limite máximo n pré definido (RIBEIRO, 2020).

Como no trabalho original (RIBEIRO; PAIVA; MIANI, 2020) não se utilizava o endereço IP dos fluxos como um atributo, de modo a evitar um enviesamento no processo de

treinamento. O que foi um impasse, visto que para conseguir realizar o cálculo das novas métricas, é necessário conseguir relacionar cada instância com seu endereço. Assim, foi necessário realizar uma modificação no *script* de pré processamento dos dados para não ser mais excluído o IP de origem do fluxo. Com esse intuito, foi aplicada uma função *hash* (SHA-256) em cada endereço IP, para gerar um inteiro como identificador de cada IP, evitando conflitos. Logo antes da classificação e treinamento começarem a acontecer, é realizada uma deleção lógica com o identificador de modo a continuar garantido que não aconteça algum viés por IP. O IP de origem é extraído e logo em seguida seu valor é atribuído como 0, como todas as instâncias terão o mesmo valor para esse atributo, o viés é evitado. A implementação pode ser verificada no Código 3.1.

Código 3.1 – Deleção lógica IP origem

```
// ALPrequentialEvalutaionTask.java
...

public class ALPrequentialEvaluationTask extends ALMainTask {
    ...

    @Override
   protected Object doMainTask(TaskMonitor monitor, ObjectRepository
       repository) {
       ...
       Example<Instance> trainInst = stream.nextInstance();
       double srcIpValue = trainInst.getData().value(2);
       trainInst.getData().setValue(2, 0);
   }
}
```

Como foi dito que a intenção era evoluir o trabalho anterior dentro do escopo do cálculo das métricas de avaliação, foi realizado um estudo do que já havia sido implementado em (RIBEIRO, 2020). A principal classe que havia sido implementada para o nosso objetivo foi *ALWindowClassificationPerformanceEvaluator* que se localiza na estrutura de diretório do MOA em *moa.src.java.moa.evaluation*. Esta classe simplesmente estendia o classificador base já existente no MOA e adiciona o controle de quantos rótulos foram utilizados para o treinamento do modelo.

Assim, ficou claro em qual parte do projeto deveria ser realizada nossa implementação. Foi então criada uma classe estendendo *ALWindowClassificationPerformanceEvaluator*, chamada *TimeWindowClassificationPerformanceEvaluator*. Diferente da implementação realizada por García et al. (2014) onde cada janela de tempo foi representada pela classe janela de tempo, aqui o controle dos quadros de tempo foi feito de modo lógico através de

um parâmetro de inicialização do avaliador que é a largura da janela de tempo. Foi considerado que a cada nova instância avaliada, corresponderia uma unidade de tempo. Sendo assim, com um contador que começa em 0 e vai até a largura informada previamente, foi possível simular essa mudança de uma janela de tempo para outra. Esta implementação pode ser visualizada em Código 3.2.

Código 3.2 – Controle troca de janela de tempo

```
// TimeWindowClassificationPerformanceEvaluator.java
...

public class TimeWindowClassificationPerformanceEvaluator extends
    AbstractOptionHandler implements ALClassificationPerformanceEvaluator {
    ...

    @Override
    public void addResult(Example<Instance> example, double[] classVotes,
        double srcIpAddress) {
        Instance inst = example.getData();

        posWindow++;

        if (!inst.classIsMissing()) {
            int trueClass = (int) inst.classValue();
            int predictedClass = Utils.maxIndex(classVotes);


            if(!botnetIps.contains(srcIpAddress) &&
                !normalIps.contains(srcIpAddress)) {
                if(trueClass == 0) {
                    normalIps.add(srcIpAddress);
                    if(predictedClass == trueClass) {
                        this.cTN++;
                    } else {
                        this.cFP++;
                    }
                } else if(trueClass == 1) {
                    botnetIps.add(srcIpAddress);
                    if(predictedClass == trueClass) {
                        this.cTP++;
                    } else {
                        this.cFN++;
                    }
```

```
                }
            }

            if(posWindow == widthOption.getValue()) {
                this.calculateTimeFrameMetrics();
                this.resetTimeFrame();
            }
        }
    }
}
```

Logo que é identificado o momento de troca entre quadros de tempo, as estatísticas (tTP, tTN, tFP, tFN) referentes ao mesmo são calculadas e armazenadas em um vetor em memória. Quando as instâncias do fluxo se encerram, é calculada a média dos valores que estavam sendo guardados em memória. Com as médias calculadas, é possível calcular as estatísticas finais (FPR, TPR, TNR, FNR, Precisão, Acurácia, Taxa de erro e *F1*).

```
// TimeWindowClassificationPerformanceEvaluator.java
...

public class TimeWindowClassificationPerformanceEvaluator extends
    AbstractOptionHandler implements ALClassificationPerformanceEvaluator {

...

private void calculateTimeFrameMetrics() {
        this.qtyTimeFrames++;

        if(botnetIps.size() > 0) {
            alltTP.add(cTP*this.correctingFunction()/botnetIps.size());
            alltFN.add(cFN*this.correctingFunction()/botnetIps.size());
        } else {
            alltTP.add(0.0);
            alltFN.add(0.0);
        }

        if(normalIps.size() > 0) {
            alltFP.add(cFP/normalIps.size());
            alltTN.add(cTN/normalIps.size());
        } else {
            alltFP.add(0.0);
            alltTN.add(0.0);
```

```
        }
    }

private void calculateFinalValues() {
        for (Double tp : alltTP) {
            this.tTP += tp;
        }
        for (Double fn : alltFN) {
            this.tFN += fn;
        }
        for (Double fp : alltFP) {
            this.tFP += fp;
        }
        for (Double tn : alltTN) {
            this.tTN += tn;
        }

        this.tTP = this.tTP / this.alltTP.size();
        this.tFN = this.tFN / this.alltFN.size();
        this.tFP = this.tFP / this.alltFP.size();
        this.tTN = this.tTN / this.alltTN.size();
    }
}
```

Capítulo 4

Experimentos e Análise dos Resultados

Este capítulo é dedicado a mostrar os experimentos elaborados para validar a implementação das métricas no MOA. O primeiro experimento utiliza um número de instâncias reduzido de modo a confirmar a corretude da implementação (Seção 4.2). E por fim, é realizado na Seção 4.3 o mesmo experimento feito por (RIBEIRO; PAIVA; MIANI, 2020), de modo a comparar os resultados obtidos com as diferentes métricas de avaliação.

4.1 Conjunto de dados e suas características

Para realização de todos os experimentos foram utilizados os conjuntos de dados presentes no CTU-13 (GARCíA et al., 2014). Esse conjunto de dados foi escolhido por ser público e conter dados reais sobre o tráfego de rede, com tráfegos classificados como *botnet*, normal e de fundo. Os conjuntos possuem 13 cenários que incluem diferentes tipos de protocolos e *botnets*. Fluxos normais e de fundo foram ambos tratados como sendo fluxo normal, resultando em duas classes restantes possíveis, normal e *botnet*. As características utilizadas no arquivo arff pode ser conferida na Tabela 4.1. Os testes foram executados em cima dos cenários 1 e 2. Ambos os cenários possuíam fluxos da *botnet* Neris. O primeiro cenário possui um total de 2.824.636 fluxos, destes 40.959 (1,45%) são fluxos de *botnet* e 2.783.677 (98,55%) fluxos normais. O segundo cenário possui um total de 1.808.122 fluxos, sendo 20.941 (1,15%) fluxos de *botnet* e 1.787.181 (98,85%) fluxos normais.

4.2 Primeiro Experimento

O primeiro experimento consiste em avaliar a corretude da implementação das novas métricas de erro (GARCíA et al., 2014) no MOA. O experimento foi executado utilizando o algoritmo de classificação *moa.classifiers.active.ALLimitedInstances* tendo como base o *drift.SingleClassifierDrift* com seus parâmetros padrões, o avaliador utilizado foi *moa.evaluation.TimeWindowClassificationPerformanceEvaluator*, implementado por este trabalho, tendo tamanho de janela de tempo 6 e alpha 0,01. Para isso, foi criado um

Característica	**Descrição**
Duração	Duração total do fluxo em segundos
Protocolo	Protocolo transação. Por exemplo, TCP ou UDP
Ip de origem	Identificador inteiro obtido após aplicar um *hash* no IP de origem
Porta de origem	Porta de origem da conexão
Porta de destino	Porta de destino da conexão
Direção	Direção do fluxo. Faz referência a origem da conexão e também combina informações sobre o estado da transação.
Estado	Relacionado ao estado básico da transação. Por exemplo REQ|INT(requisitado|inicial), ACC(aceito), etc.
sTos	Tipo de serviço da origem para o destino
dTos	Tipo de serviço do destino para a origem
Total de pacotes	Quantidade de pacotes no fluxo
Total de bytes	Soma dos bytes no fluxo
Bytes da origem	Soma dos bytes da origem
Média de bytes por pacote	Média de bytes por pacote

Tabela 1 – Lista das características

arquivo arff de tamanho reduzido, somente 30 instâncias, retiradas do CTU-11 de maneira aleatória, que possui três endereços IP infectados 147.32.84.165, 147.32.84.191 e 147.32.84.192. Como o tamanho de janela de tempo é igual à 6, no total são 5 janelas de tempo calculando suas respectivas estatísticas. A Figura 7 mostra os fluxos de rede, suas respectivas janelas de tempo e seus rótulos verdadeiros.

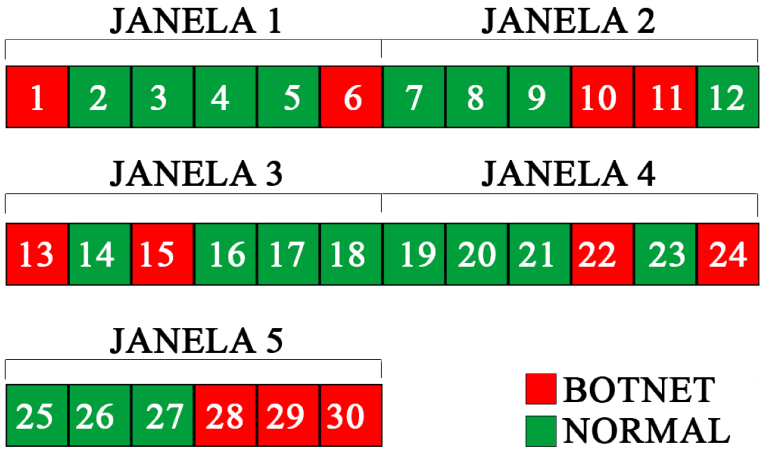

Figura 7 – Janelas de tempo e rótulos verdadeiros

A Figura 8 ilustra o comportamento do classificador de acordo com as novas métricas, e os valores das métricas para cada janela com a média final pode ser visualizado na

Tabela 2. Na primeira janela de tempo, tivemos dois falsos negativos e quatro falsos positivos, que geraram esse FN 1,99 em consequência da função de correção. Na segunda janela, o classificador apontou um verdadeiro positivo e três verdadeiros negativos. É interessante notar como o classificador já conseguiu identificar o fluxo malicioso, porém este só foi contabilizado uma vez, pois o IP de origem do fluxo 11 era o mesmo. A terceira janela possui um verdadeiro positivo e quatro verdadeiros negativos e, novamente, o comportamento da janela anterior se repetiu, os fluxos maliciosos presentes nesta janela possuíam o mesmo IP de origem, fazendo com que somente um verdadeiro positivo fosse contabilizado. Na quarta janela, dois verdadeiros positivos e três verdadeiros negativos foram encontrados. Como aqui os fluxos maliciosos possuíam IP de origem distintos, ambas as classificações foram contabilizadas. Nesse caso, o fluxo que não foi contabilizado é um fluxo normal, pelo mesmo motivo anterior, este fluxo possui o mesmo endereço de origem de um fluxo que já foi contabilizado. Por fim, na quinta janela, dois verdadeiros positivos e três verdadeiros negativos foram encontrados.

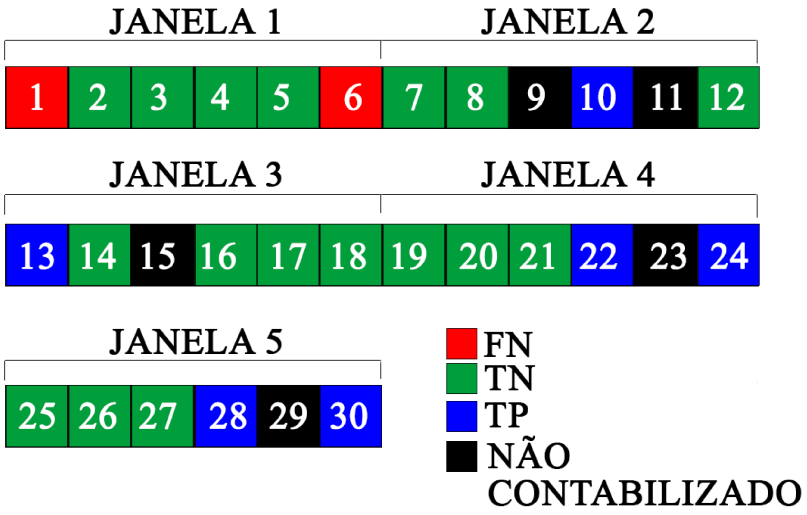

Figura 8 – Classificação por fluxo de rede

	Janela 1	Janela 2	Janela 3	Janela 4	Janela 5	Média
TP	0	1,9801	1,9704	1,9607	1,9512	1,5725
FN	1,99	0	0	0	0	0,3980
FP	0	0	0	0	0	0
TN	1	1	1	1	1	1

Tabela 2 – Métricas por janela de tempo e média final.

Com a média final calculada para todas as métricas coletadas, seguindo as fórmulas propostas por (GARCíA et al., 2014), foi realizado o cálculo dos valores finais, que podem ser visualizadas na Tabela 3.

	Valor
FPR	0
TPR	0,7980
TNR	1
FNR	0,2020
Precisão	1
Acurácia	0,8660
Taxa de erro	0,1340
F1	0,8877

Tabela 3 – Resultados finais Experimento 1.

Podemos observar nos resultados finais que o valor da Precisão foi igual à 1, isso quer dizer que, todos os endereços IP maliciosos foram classificados corretamente. O valor do TNR também foi igual à 1, ou seja, nenhum endereço IP normal foi classificado como malicioso.

4.3 Segundo Experimento

O objetivo do segundo experimento é executar o mesmo experimento por (RIBEIRO; PAIVA; MIANI, 2020), porém se utilizando das novas métricas implementadas de modo a analisar se houve alguma diferença entre o uso de estatísticas convencionais e as novas métricas elaboradas especificamente para comparação entre detectores de *botnet*. Os experimentos foram executados para os cenários 1 e 2 do conjunto de dados CTU-13 mantendo-se o algoritmo de classificação, *OzaBoost*, e seus respectivos parâmetros, mas utilizando as novas medidas de avaliação com um tamanho da janela de tempo igual a 10000 e alpha (parâmetro da função de correção) 0,01. Os resultados obtidos para o experimento podem ser visualizados na Tabela 4, e a comparação com as métricas originais, que foram calculadas da forma tradicional, na Tabela 5.

	Resultados							
CTU	Precisão	Revocação	FPR	TNR	FNR	Acurácia	F1	Erro
1	0,999	0,896	0,0001	0,999	0,104	0,951	0,945	0,049
2	0,999	0,559	0.0008	0,999	0,441	0,772	0,716	0,228

Tabela 4 – Resultados experimentos finais

	Base de comparação (métricas originais)		Novas métricas	
CTU	Precisão	Revocação	Precisão	Revocação
1	0,981	0,954	0,999	0,896
2	0,781	0,899	0,999	0,559

Tabela 5 – Comparação com trabalho base

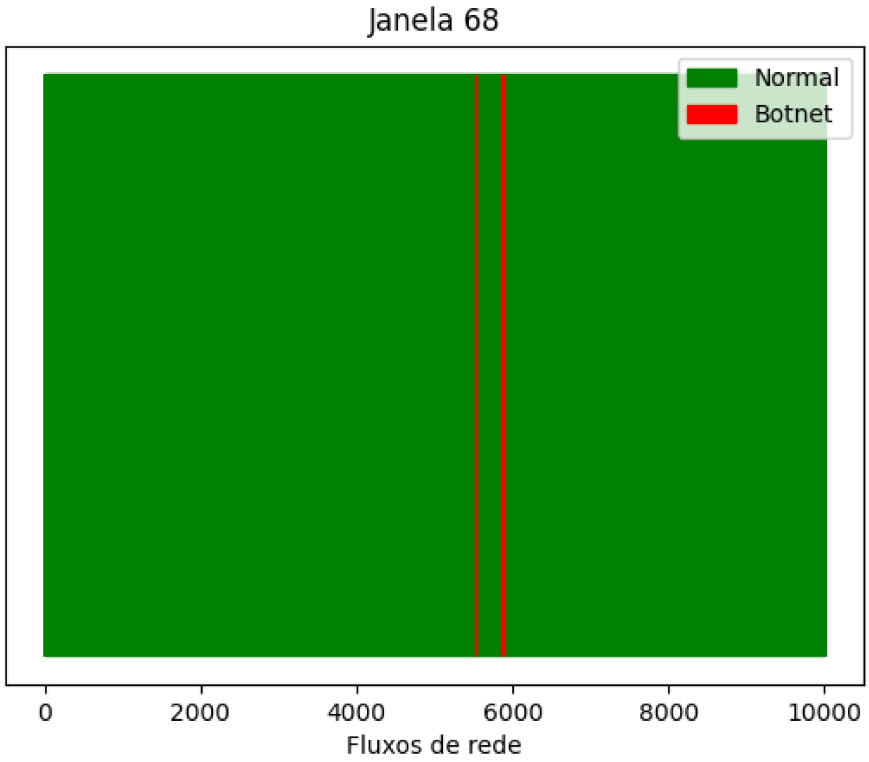

Figura 9 – Janela de tempo 68 do CTU-1

	Resultados							
Janela	Precisão	Revocação	FPR	TNR	FNR	Acurácia	F1	Erro
68	0	0	0	1	1	0,399	0	0,601
71	0,998	1	0,002	0,998	0	0,999	0,999	0,0007

Tabela 6 – Resultados por janela de tempo CTU-1

Para o CTU-1, a primeira janela de tempo em que apareceu um fluxo de *botnet* foi a de número 68. A Figura 9 ilustra a distribuição dos 10000 fluxos da janela. Porém, como se seguiu um longo tempo sem a ocorrência de um fluxo de *botnet*, o algoritmo ainda não conseguiu classificar estas instâncias corretamente, a primeira janela em que uma *botnet* é classificada corretamente se dá na janela de número 71 observável na Figura 10. Isso equivale a um total de 26355 fluxos entre a primeira aparição de um fluxo *botnet* até a detecção correta de um, onde 58 eram fluxos maliciosos. As estatísticas para ambas as janelas citadas podem ser conferidas na Tabela 6.

Vale mencionar que como nossa coleta de estatísticas se baseia na aparição única de um IP fonte por janela de tempo e o CTU-1 só possui um único IP infectado, na primeira tentativa de classificação deste IP malicioso na janela de tempo, este resultado será o determinante para o cálculo das estatísticas que envolvem tanto verdadeiros positivos quanto falsos negativos.

Observando o segundo conjunto de dados utilizado para realização dos experimentos,

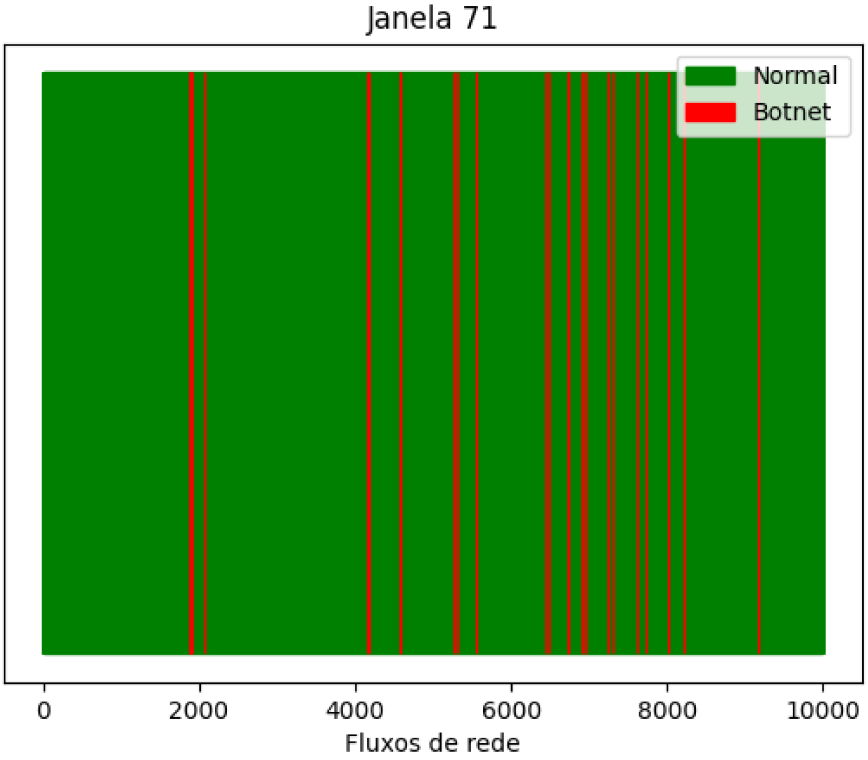

Figura 10 – Janela de tempo 71 do CTU-1

	Resultados							
Janela	Precisão	Revocação	FPR	TNR	FNR	Acurácia	F1	Erro
36	0	0	0	1	1	0,370	0	0,629
44	0,998	1	0,002	0,997	0	0,999	0,999	0.0008

Tabela 7 – Resultados por janela de tempo CTU-2

o CTU-2, podemos perceber um comportamento semelhante ao CTU-1, no sentido de que ambos passam um longo tempo sem a ocorrência de *botnets* e demoram algumas janelas de tempo para começarem a acertar a predição dos fluxos positivos. No CTU-2, a primeira ocorrência de uma *botnet* se da na janela de tempo número 36, conforme a Figura 11. Porém, a primeira detecção só acontece na janela de número 44, que pode ser conferida na Figura 12, isso equivale à um total de 76986 fluxos entre a primeira aparição de um fluxo *botnet* até a detecção correta de um, onde 127 eram fluxos maliciosos. As estatísticas referente a cada janela de tempo podem ser visualizadas na Tabela 7.

É interessante notar, na Tabela 5, que essa queda alta entre a revocação calculada pela base de comparação para o CTU-2 e o experimento corrente, pode se dar pela demora da detecção do primeiro endereço IP da *botnet*, visto que entre a aparição do primeiro endereço malicioso até sua detecção se passaram 8 janelas de tempo.

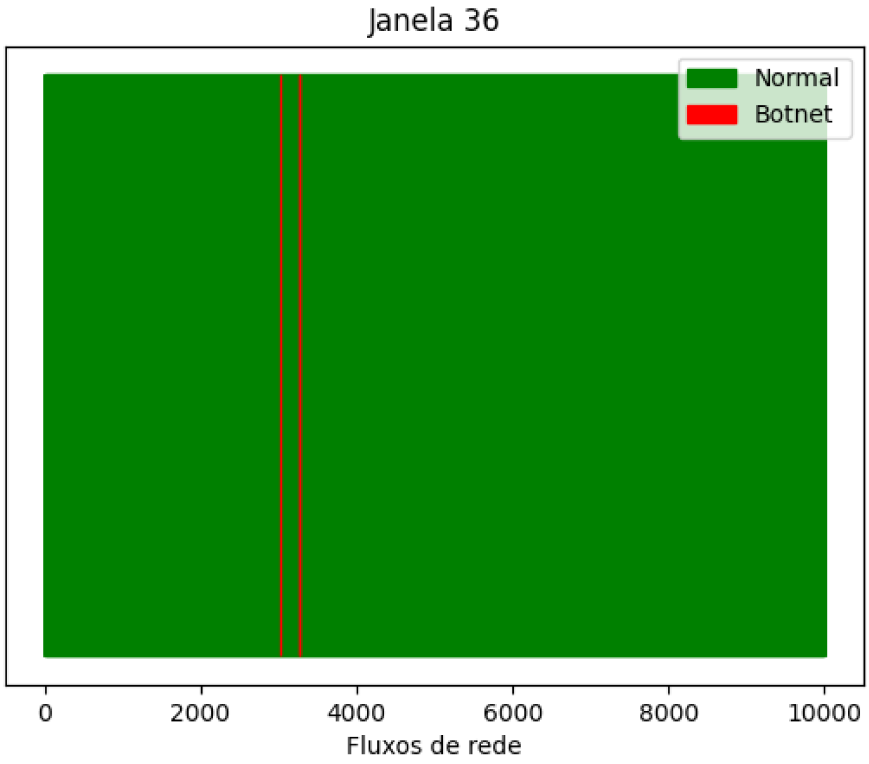

Figura 11 – Janela de tempo 36 do CTU-2

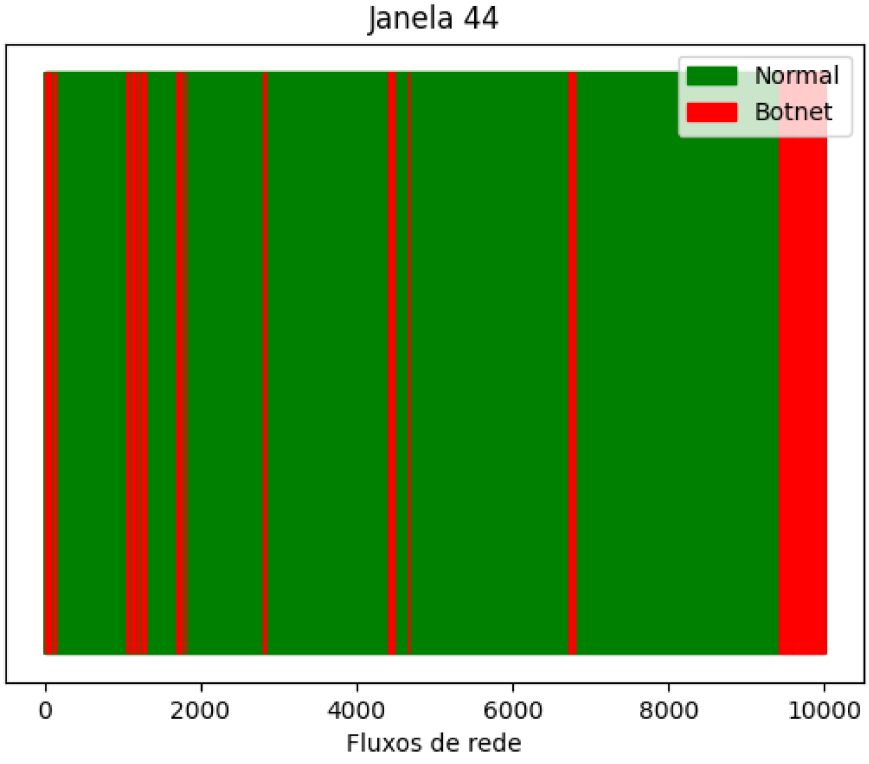

Figura 12 – Janela de tempo 44 do CTU-2

Capítulo 5

Conclusão

Uma sociedade cada vez mais conectada à rede é uma realidade. Isso traz à tona diversos desafios na área de segurança, mais dispositivos conectados, potencialmente tendo sido desenvolvidos sem a devida precaução com segurança, sendo um alvo para *botnets*. Se utilizar de técnicas de mineração de dados e aprendizados de máquina para detectar possíveis fluxos maliciosos observando-se anomalias nesse tráfego é um caminho possível. Porém, pela própria natureza de um tráfego de rede, em que pacotes fluem continuamente por tempo indeterminado, as técnicas mais tradicionais podem não ser adequadas para a detecção e combate das *botnets*. Logo, a mineração em fluxos contínuos de dados se faz muito mais adequada para atacar este tipo de problema.

No momento de se avaliar esses algoritmos empregados a fluxos contínuos, as métricas tradicionais para comparação de algoritmos também não estão perfeitamente ajustadas para o tipo de problema, medidas como acurácia e precisão dizem muito pouco sobre este tipo de problema visto que o volume de dados é muito grande, e a classe que está se tentando identificar, *botnet*, representa uma porcentagem muito baixa desde volume total. Logo, é esperado que tanto precisão quanto acurácia sejam altas.

O trabalho desenvolvido por (GARCíA et al., 2014) dá um importante passo na direção de melhorar a discussão acadêmica em torno da comparação de métodos de detecção de *botnets*, ao prover um conjunto de dados real e rotulado disponível para toda comunidade além de propor novas métricas para a realização dessa comparação.

Como complemento deste trabalho, pode-se realizar uma investigação em como determinar o tamanho ótimo para a janela de comparação, pois este tem impacto direto nas métricas finais obtidas. Outro trabalho possível é a realização de mais experimentos em conjuntos de dados que possuam uma quantidade maior de *botnets*.

Referências

BABCOCK, B. et al. Models and issues in data stream systems. In: **Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems**. New York, NY, USA: Association for Computing Machinery, 2002. (PODS '02), p. 1–16. ISBN 1581135076. Disponível em: <https://doi.org/10.1145/543613.543615>.

COSTA, V. et al. Online detection of botnets on network flows using stream mining. In: **Anais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**. Porto Alegre, RS, Brasil: SBC, 2018. p. 225–238. ISSN 2177-9384. Disponível em: <https://sol.sbc.org.br/index.php/sbrc/article/view/2418>.

ELKAN, C. Evaluating classifiers. **San Diego: University of California**, Citeseer, 2012.

GAMA, J. **Knowledge Discovery from Data Streams**. [S.l.]: CRC Press, 2010.

GARCíA, S. et al. An empirical comparison of botnet detection methods. **Computers Security**, v. 45, p. 100–123, 2014. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404814000923>.

GOLLMANN, D. Computer security. **WIREs Computational Statistics**, v. 2, n. 5, p. 544–554, 2010. Disponível em: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.106>.

GéRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**. [S.l.: s.n.], 2019.

KUMAR, V. et al. **Introduction to Data Mining**. 2. ed. [S.l.]: Pearson, 2018. (What's New in Computer Science). ISBN 2017048641,9780133128901,0133128903.

NOGUEIRA, M. Anticipating moves to prevent botnet generated ddos flooding attacks. **CoRR**, abs/1611.09983, 2016. Disponível em: <http://arxiv.org/abs/1611.09983>.

OLIMPIO, G. et al. Intrusion detection over network packets using data stream classification algorithms. In: **2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)**. [S.l.: s.n.], 2021. p. 985–990.

PAIVA, E. R. d. F. **Detecção de novidade em fluxos contínuos de dados multiclasse**. Tese (Doutorado) — Universidade de São Paulo, 2014.

RIBEIRO, G. H.

Detecção de botnets utilizando classificação de fluxos contínuos de dados, 2020.

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

SHIREY, R. W. **Internet Security Glossary, Version 2**. RFC Editor, 2007. RFC 4949. (Request for Comments, 4949). Disponível em: <https://www.rfc-editor.org/info/rfc4949>.

SILVA, L. et al. Study on machine learning techniques for botnet detection. **IEEE Latin America Transactions**, v. 18, n. 05, p. 881–888, 2020.

SILVA, S. S. et al. Botnets: A survey. **Computer Networks**, v. 57, n. 2, p. 378–403, 2013. ISSN 1389-1286. Botnet Activity: Analysis, Detection and Shutdown. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128612003568>.

STALLINGS, W. **Cryptography and Network Security principles and practices**. [S.l.]: Prentice Hall, 2015.