Stacking-based Committees para Detecção de Ataques em Redes de Computadores - Uma abordagem por exaustão

Thiago J. Lucas[1], Kelton A. P. da Costa[1], Eduardo A. Moraes[2], Paulo R. G. Hernandes Júnior[2], Miguel J. das Neves[2]

[1]Departamento de Computação – Univ. Estadual Paulista "Júlio de Mesquita Filho" (UNESP) – Caixa Postal 17033-360 – Bauru – SP – Brasil

[2]Departamento de Segurança da Informação – Faculdade de Tecnologia de Ourinhos (FATEC) – Caixa Postal 19910-206 – Ourinhos – SP – Brasil

{t.lucas,kelton.costa}@unesp.br

{eduardo,galego,miguel}@fatecourinhos.edu.br

Abstract. *The application of Machine Learning techniques in detecting attacks on computer networks has obtained good results, emphasizing the methods of Ensemble, which manage to improve the performance of individual classifiers. The present study was carried out in Dataset CICIDS-2017 and considered classifiers' choice based on a systematic review of the literature, aiming to find the state-of-the-art and trends, both for the classification algorithms and the ensemble techniques. Committees that significantly reduce classification errors are presented by modeling intrusion detectors that are superior to individual methods and related work compared with an average accuracy of 99.92%.*

Resumo. *A aplicação de técnicas de Machine Learning na detecção de ataques em redes de computadores tem obtido bons resultados, com destaque para os métodos de Ensemble, que conseguem melhorar o desempenho de classificadores individuais. O estudo foi realizado utilizando o Dataset CICIDS-2017 e considerou a escolha de classificadores com base em uma revisão sistemática da literatura, objetivando encontrar o estado-da-arte e as tendências, tanto para os algoritmos de classificação quanto para as técnicas de ensemble. Committees que reduzem significativamente os erros de classificação são apresentados modelando detectores de intrusão que são superiores aos métodos individuais e aos trabalhos correlatos comparados obtendo acurácia média de 99.92%.*

1. Introdução

A informação pode ser considerada o maior ativo de uma empresa ou corporação no cenário econômico moderno, de acordo com [Fraimovich et al. 2020], e este fato destaca que os proprietários devem se preocupar com o resguardo de seus dados, principalmente no que tange à confidencialidade, integridade e à disponibilidade destes.

A heterogeneidade dos acessos à Internet no que diz respeito ao conteúdo e à anatomia dos pacotes trafegados na rede mundial torna o processo de identificação de ataques (ou de pacotes maliciosos) um problema de pesquisa sobre o qual a comunidade acadêmica tem se dedicado. Neste sentido, diversos tipos de Sistemas de Detecção de Intrusão (IDS) têm sido empregados para analisar o "comportamento da rede" a fim de

detectar fluxos maliciosos que objetivam, necessariamente, causar danos aos ativos, em especial, a informação.

Este estudo apresenta uma abordagem exaustiva onde diversos *committees* (agrupamento de algoritmos de classificação) foram criados de forma a unir a capacidade de classificadores heterogêneos a fim de se propor a melhor combinação para detectar 6 diferentes tipos de ataque: *Brute-force*, *Infiltration*, DDoS, *Portscan*, ataques provenientes de *Botnets* e ataques Web.

Foram implementados 44 modelos que agrupam 4 diferentes classificadores (k-*Nearest Neighbors* - k-NN, *Support-vector Machines* - SVM, *Decision Trees* - DT e *Multilayer Perceptron* - MLP, que é um tipo de *Neural Network* - NN) considerando todas as possibilidades aceitáveis pelo método de *Ensemble* "*Stacking*". Tanto os classificadores quando a técnica de *Ensemble* foram escolhidos após observação das tendências obtidas por meio da realização de uma revisão sistemática da literatura. Uma relevante contribuição deste estudo é a apresentação do melhor modelo de combinações de classificadores para cada tipo de ataque que propôs-se a detectar com visível queda nos erros de classificação.

O presente trabalho está estruturado da seguinte forma: a Seção 2 ilustra as tendências obtidas por meio da revisão sistemática da literatura, bem como traz a relação de trabalhos correlatos que utilizaram *Stacking* e a descrição teórica dos elementos fundamentais da pesquisa; a Seção 3 apresenta a modelagem dos *committees* e uma visão geral do processo prático percorrido para obtenção dos resultados; a Seção 4 apresenta os números de *Precision*, *Recall*, *F1-Score* e Acurácia bem como uma análise a respeito da diminuição dos erros de classificação na comparação com método individuais e uma comparação dos resultados deste estudo com os trabalhos correlatos; por fim, a Seção 5 apresenta as conclusões e as propostas de trabalhos futuros.

2. Revisão Sistemática da Literatura

Uma revisão sistemática da literatura foi realizada a fim de se obter quais os classificadores e quais técnicas de *Ensemble* são mais utilizadas e de forma com que fosse possível observar algumas tendências. Buscou-se em relevantes bases científicas da área da Ciência da Computação e de Redes de Computadores publicações que continham os termos "*Ensemble*" e "*Intrusion*", seja no Título da publicação ou nas Palavras-chave e cujo ano de publicação estivesse no intervalo entre os anos 2015 e 2020, ou seja, trabalhos recentes que tenham utilizado qualquer técnica de Ensemble aplicada a detecção de intrusão em redes de computadores.

Pelo fato do desenvolvimento desta pesquisa ter-se dado no segundo semestre do ano de 2020, as publicações do ano em questão levaram em consideração apenas o primeiro semestre. Para que os números fossem ajustados, multiplicou-se por 2 os valores do último ano (portanto os números referentes a 2020 são projeções). Ao todo foram encontrados 68 trabalhos (considerando a exclusão de trabalhos duplicados ou relacionados a outras áreas).

A Figura 1 ilustra a relação de número de publicações por ano no intervalo entre 2015 e 2020 agrupadas pelos quatro classificadores que mais foram utilizados. Mais detalhes acerca dos métodos de classificação estão disponíveis na Seção 2.5. É possível

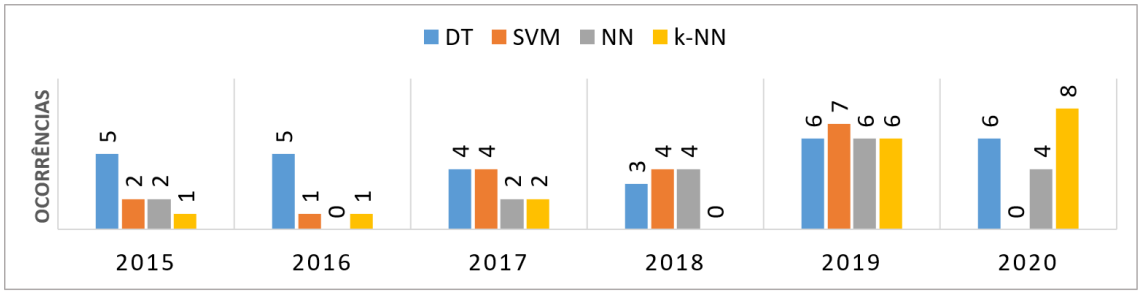

Figura 1. Relação de classificadores por número de ocorrências no intervalo analisado. Fonte: Elaborado pelo Autor.

observar algumas tendências, como por exemplo, o crescente uso do k-NN e, embora oscilante, a implementação de DT e NN, embora se reconheça que uma metodologia mais criteriosa poderia trazer mais respaldo para observação de tendências. Embora não exista registro no último ano para a aplicação de SVM no contexto deste trabalho, é importante observar o crescimento do método nos anos anteriores. Outros classificadores também foram utilizados nos trabalhos correlatos, entretanto estes 4 se destacam pelo número de repetições. A Figura 2 ilustra a relação de número de publicações por ano no intervalo entre 2015 e 2020 agrupadas pelos métodos de *Ensemble* que mais foram utilizados:

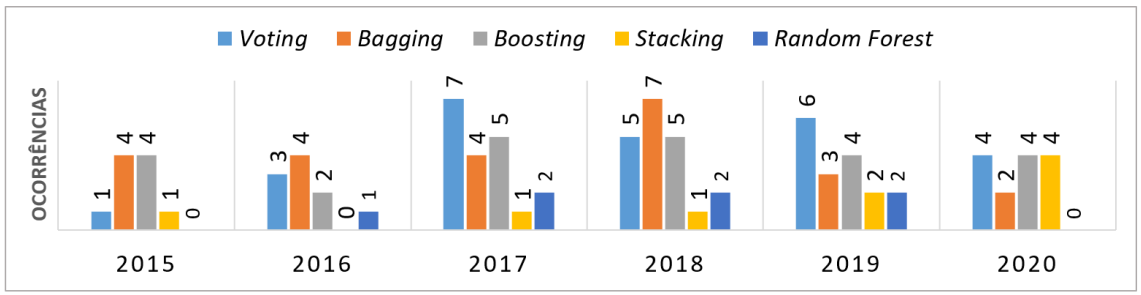

Figura 2. Relação de técnicas de *Ensemble* por número de ocorrências no intervalo analisado. Fonte: Elaborado pelo Autor.

É visível o grande número de implementações usando as técnicas *Voting*, *Bagging* ou *Boosting*. Entretanto destaca-se o crescimento do método denominado *Stacking* (mais detalhes na Seção 2.4), o que motivou sua escolha como método de criação dos *committees* do presente trabalho.

2.1. Trabalhos Correlatos

A presente seção elenca um breve resumo dos trabalhos relacionados que utilizaram *Stacking* para detecção de intrusão.

Um modelo de *Stacking Ensemble* foi proposto por [Milliken et al. 2015]. Os experimentos demonstraram que a análise da entropia entre pares de *features* de diferentes *subsets* pode contribuir para a geração de um *subset* mais adequado para classificação de ataques. Os autores utilizaram o *Dataset* ISCX 2012 como base para os testes juntamente com os classificadores OneR, Conjunctive e Naive Bayes em um modelo de *Stacking* para realização do *Ensemble*. Os experimentos obtiveram como melhor resultado um F-*Score* de 92%.

[Belouch and hadaj 2017] realizaram uma comparação entre três técnicas diferentes de Ensemble: *Booting, Bagging* e *Stacking*. Os autores combinaram quatro diferentes classificadores: *Decision Tree, Naive Bayes, Multilayer Perceptron* e REP*Tree*. Todos os experimentos foram realizados no *Dataset* UNSW-NB15. A publicação documenta todos os resultados, sendo possível observar que o melhor *Ensemble* quanto à acurácia foi um *Stacking* de REP*Tree* seguido por um *Stacking* de *Decision Trees*.

[Sun et al. 2018] apresentaram um modelo de detecção de intrusão que combina três técnicas clássicas de *Ensemble*: *Bagging, Boosting* e *Stacking*. Os classificadores base utilizados foram SVM e k-NN. Com testes realizados no *Dataset* NSL-KDD, após geração de um *subset* por meio de PCA foi possível mensurar como melhores resultados de Acurácia a classificação de pacotes normais com 99.41% e a de pacotes de *Portscan* com 93.13%.

O sistema de detecção de intrusão apresentado por [Lu et al. 2019] consiste em separar por camadas os pacotes a serem classificados de acordo com os protocolos TCP, UDP e ICMP. Para cada grupo de pacotes são aplicados classificadores binários dispostos em cascata de forma com que, no final do processo, seja possível realizar uma classificação multiclasse completa. Os autores realizaram testes com o *Dataset* NSL-KDD usando os classificadores SVM, DT, Bayes *Network*, REP*Tree*, k-NN, BFTree, SimpleCart e Naive Bayes. Os resultados foram superiores aos comparados *Bagging*, *Boosting* e *Stacking*, atingindo uma acurácia de 95.33% para detectar ataques DoS, 91.14% para ataques *Portscan*, 55.21% para R2L e 1.42 para U2R, além de uma acurácia de 98.02% na detecção de pacotes benignos.

Em experimentos com os *Datasets* NSL-KDD e UNSW-NB15, [Hsu et al. 2019] apresentaram modelos de detecção de intrusão usando *Stacking* seja na classificação ou na seleção de *features*. Agrupou-se usando *Stacking* também no processo de detecção com os classificadores SVM, *Autoencoder* e *Random Forest*. Os resultados de Acurácia indicam 91.7% com NSL-KDD e 91.8% com UNSW-NB15.

[Olasehinde et al. 2020] realizaram experimentos no *Dataset* UNSW-NB15 a fim de se obter qual a melhor hipótese de meta-classificador (classificador da segunda camada) para um modelo de detecção de intrusão usando *Stacking*. Realizando as classificações da primeira camada com k-NN, Naive Bayes e DT os autores testaram algumas possibilidades de meta-classificadores e compararam os resultados. Utilizando três *subsets* diferentes, os melhores resultados foram utilizando DT na segunda camada apontando para uma acurácia média de 98.56%.

Um sistema de detecção de intrusão especialista em detectar ataques web foi apresentado por [Tama et al. 2020]. Um *Ensemble* por meio de *Stacking* foi implementado, entretanto, ao invés de usar classificadores convencionais, os autores usaram outros métodos de *Ensemble* como *Random Forest, Gradient Boost Machine* e XG*boost* para criar os *committess*. Quatro diferentes *Datasets* foram testados: CICIDS-2017, HTTP-CSIC-2010, NSL-KDD e UNSW-NB15. O resultado médio de Acurácia informado pelos autores foi de 96.07% para o modelo proposto, sendo o melhor resultado obtido no CICIDS-2017 com 99.98%.

Em relação aos trabalhos correlatos supracitados, este trabalho apresenta como diferencial uma observação detalhada por meio de exaustão onde o método (documen-

tado na Seção 3) permite a análise de todas as combinações possíveis levando-se em consideração os classificadores utilizados. Para os diferentes tipos de ataque analisados, será apresentado o melhor *committee* implementado via *stacking*.

2.2. Sistemas de Detecção de Intrusão

[Karatas and Sahingoz 2018] definem um IDS como um software desenvolvido para detectar ataques que tenham potencial para causar danos na rede de comunicação ou nos sistemas, sejam eles provenientes de um meio inseguro como a Internet ou da rede local. Tais softwares executam contramedidas de segurança ao detectar alguma anomalia no tráfego de rede ou no comportamento dos hosts que possa caracterizar um ataque. Os autores destacam que muitas abordagens têm sido aplicadas para se aumentar a taxa de detecção, de forma geral, envolvendo técnicas de *Machine Learning* (situações onde o detector aprende por meio de treinamento a detectar anomalias), *Rule-based* (caso onde o detector possui assinaturas características de ataques e apenas compara o tráfego real com as assinaturas de ataques já conhecidos) e/ou métodos estatísticos clássicos (emprego de cálculos de média, desvio padrão, mediana, dentre outros).

2.3. Ensemble Learning

Métodos de *Ensemble Machine Learning*, de acordo com [Zhou 2012], consistem na aplicação conjunta de diferentes algoritmos de classificação ou regressão de forma com que resolvam o mesmo problema. Enquanto técnicas tradicionais constroem um aprendizado baseado em uma técnica específica, as técnicas de *Ensemble* procuram agregar, seja de forma paralela ou de forma sequencial, múltiplas técnicas de aprendizagem, fazendo com que, nas palavras de [Smolyakov 2017], existam melhoras em relação à variância, ao bias ou às previsões.

Segundo [Zhou 2012], existem três formas de organização de um método de *Ensemble* no que diz respeito a como os algoritmos individuais se relacionam, sendo elas: "*combining classifiers*", "*Ensembles of weak*" e "*mixture of experts*". A primeira forma é a mais utilizada na literatura, em especial para solução de problemas de reconhecimento de padrões. Baseia-se na combinação de classificadores com excelente Acurácia e procura-se combiná-los de maneira que a acurácia final (conjunta) na classificação seja aprimorada. A segunda forma é a mais usada pela comunidade de *Machine Learning* e basicamente consiste na aplicação conjunta de algoritmos leves (com boa acurácia) de classificação fazendo com que o classificador final tenha Acurácia excelente extraindo o melhor de cada classificador leve. Por fim, o último método é mais explorado no campo das Redes Neurais Artificiais e, basicamente, parte do princípio de que se pode ter uma solução melhor combinando regras e misturando parâmetros.

Pela observação das tendências extraídas na Revisão Sistemática da Literatura, destacar-se-á o método Stacking, cerne da metodologia desta pesquisa.

2.4. Stacking - Stacked Generalization

Um modelo de *Ensemble* denominado Stacking foi originalmente proposto por [Wolpert 1992]. A técnica consiste em implementar duas camadas de classificação, sendo a primeira composta por n classificadores e a última composta por um único classificador final. Diferentemente das técnicas de bagging e boosting, os classificadores não precisam

necessariamente ser iguais, ou seja, a primeira camada pode ser composta por classificadores heterogêneos.

[Agarwal and Chowdary 2020] definem o processo de *Stacking* como uma combinação de predições realizadas por múltiplos métodos de aprendizagem $(L_1, L_2, ..., L_n)$ gerados por múltiplos classificadores $(l_1, l_2, ..., l_n)$ numa camada inicial. Tais classificadores são treinados com a mesma amostragem de treinamento D_{Train} que contém amostras no formato $s_i =< x_1, y_i >$ onde x_i é o vetor de características contendo valores para todas as features de D_{Train} e y_i é o rótulo da classe a qual pertence o vetor.

Na primeira fase os classificadores $l_1, l_2, ..., l_n$ realizam predições para um vetor x_q. Na segunda fase um meta-classificador M faz a predição final da classe levando em consideração a predição realizada na camada anterior. [Agarwal and Chowdary 2020] destacam a importância da escolha do meta-classificador como fundamental para incremento da performance de classificação e [Džeroski and Ženko 2004] complementam afirmando que a utilização de um meta-classificador se justifica apenas quando a performance do *Ensemble* é superior à performance do classificador singular de forma com que deve-se observar no processo de implementação qual classificador individual tem a melhor performance para comparação com a performance do *Ensemble*.

[Aggarwal 2014] e [Rocca 2019] definem o algoritmo do *Stacking* como sendo o seguinte processo:

1. Divida a amostra de treinamento D_{Train} em duas partições $D_{Train}A$ e $D_{Train}B$;
2. Defina os classificadores individuais $l_1, l_2, ..., l_n$ da primeira camada e os treine usando a partição $D_{Train}A$;
3. Para cada classificador $l_1, l_2, ..., l_n$ faça predições na partição $D_{Train}B$; e
4. Treine o meta-classificador na partição $D_{Train}B$ usando como parâmetros de entrada um novo dataset que é $D_{Train}B$ concatenado com as predições da etapa anterior feitas por $l_1, l_2, ..., l_n$.

Como forma de análise do processo, considere as letras (de A à I) da Figura 3.

A - Representa a amostra de treinamento que é parte de um *Dataset* hipotético;
B - Após a etapa de particionamento, têm-se em B uma amostra de A;
C - Da mesma forma, têm-se em C uma amostra de A;
D - Representa o processo de treinamento dos classificadores individuais $\{1, 2, ..., n\}$ com o objetivo de gerar modelos preditores $\{1, 2, ..., n\}$;
E - Etapa de testes, onde os modelos preditores $\{1, 2, ..., n\}$ fazem previsões usando como amostragem de testes a segunda partição obtida no processo C (C1). As predições são concatenadas na partição B como novas features (C2);
F - Representa o processo de treinamento do meta-classificador que é realizado com a partição obtida no processo C (C3) já concatenada com as predições da etapa C2;
G - Processo de testes do meta-modelo onde as predições são realizadas na amostragem de testes de um *Dataset* hipotético;
H - Representa a amostragem de testes de um dataset hipotético; e
I - Processo de análise de performance do Stacking onde se obtém, por exemplo, métricas de acurácia, precision, recall, dentre outros.

Por fim, nas palavras de [Wolpert 1992], define-se o método de *Stacking* como sendo um esquema de encaminhamento de informações de um conjunto de classificadores

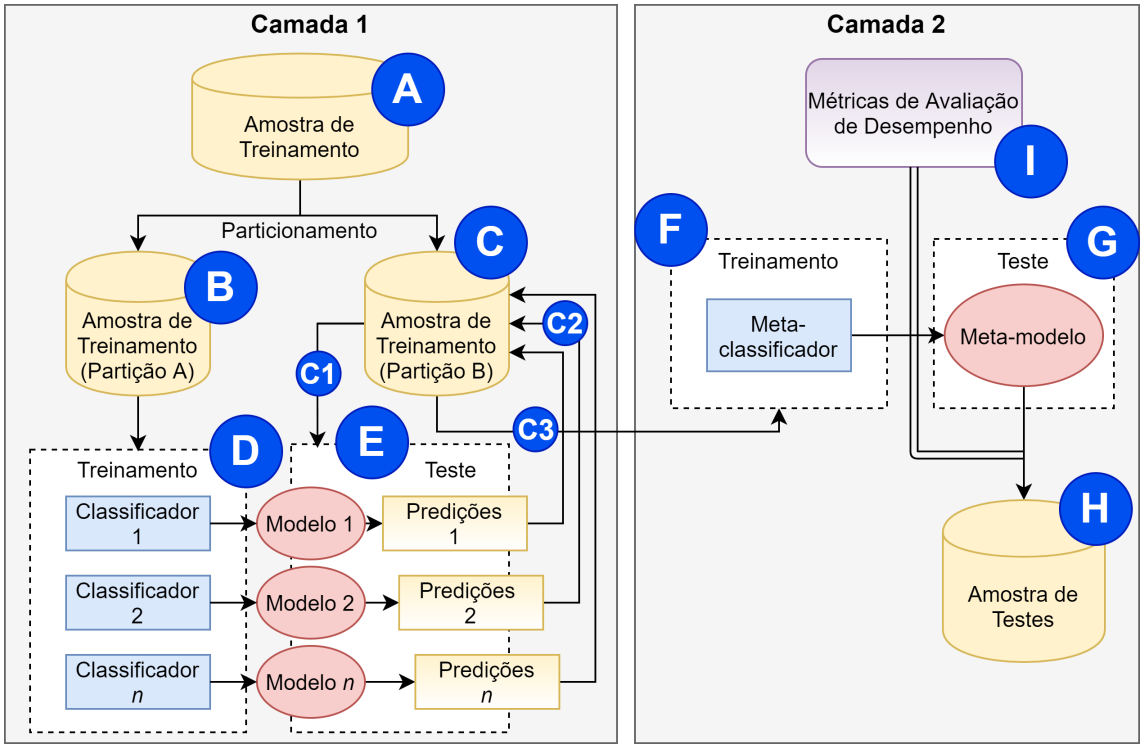

Figura 3. Fluxo de funcionamento do Stacking. Fonte: Elaborada pelo Autor.

para outro, antes de obter-se a classificação final e que pode ser útil como forma de reduzir a taxa de erro num modelo de predições.

2.5. Classificação

A escolha dos algoritmos para implementação dos Stackings levou em consideração as tendências obtidas na Revisão Sistemática da Literatura. Pode-se perceber na Figura 1 que há quatro algoritmos com utilização em destaque representando o estado-da-arte para técnicas de Ensemble na criação de sistemas de detecção de intrusão. A presente seção objetiva descrever o funcionamento dos métodos DT, k-NN, MLP e SVM.

2.5.1. *Decision Trees*

De acordo com [Rezende 2003], os métodos que se utilizam de *Decision Trees* são pertencentes à família *Top Down Induction of Decision Trees* (TDIDT) que é uma estrutura de dados que define "nós folhas" e "nós de decisão". Ao primeiro nó compete representar uma classe enquanto ao segundo compete representar um teste sobre algum atributo. Para cada exemplo a ser classificado, os atributos são condicionados pelos "nós de decisão" de forma com que se obtenha um caminho até uma classe "nó folha". Trata-se de uma sequência de "*ifs*" lógicos que irão determinar o caminho a ser percorrido na árvore para cada exemplo.

[Sharma and Kumar 2016] definem DT como uma estrutura similar a de um fluxograma onde cada nó interno representa um teste para um atributo, cada ramificação representa o resultado de um teste e cada folha representa uma classe. Os autores desta-

cam que as DT podem ser construídas com relativamente menos tempo e menos custo de processamento se comparada a outros métodos de classificação.

2.5.2. k-*Nearest Neighbors*

O método k-*Nearest Neighbors* funciona de uma forma bastante simples. Ao estabelecer um valor para k o algoritmo, no espaço de parâmetros, calcula a distância da amostra de testes para todas as amostras de treinamento e as ordena da menor distância para a maior. Por fim, para os k vizinhos mais próximos é observada a classe que mais se repete, que será, por fim, atribuída ao objeto analisado.

Segundo [Tchaye-Kondi et al. 2020], embora seja simples, o k-NN é capaz de obter excelentes resultados para classificação de dados. O melhor valor do hiperparâmetro k é obtido mediante testes. [Deng et al. 2016] afirmam que, pelo fato do k-NN necessitar computar a distância de todas as amostras de treinamento para cada uma das amostras de testes, o custo computacional deve ser considerado no processo de implementação.

2.5.3. *Multilayer Perceptron*

Uma *Artificial Neural Network* é uma estrutura similar ao cérebro humano no sentido de comunicação entre os neurônios. O cérebro humano responde a estímulos nas entradas dos seus neurônios e a organização celular faz com que os demais sejam ativados (ou não) em resposta ao processamento daquele estímulo. Da mesma forma, uma Rede Neural Artificial normalmente possui a quantidade de entradas correspondente ao número de *features* de um *Dataset* a ser processado. A estrutura das demais camadas e a forma como a propagação dos estímulos acontece dependem do tipo da Rede Neural criada. A última camada de neurônios corresponde às classes dos dados analisados, de forma com que seja possível, dados os parâmetros de entrada e a propagação dos estímulos, acionar o neurônio correspondente à classe estimada.

Um perceptron, que é um tipo de Rede Neural, produz uma saída binária ao receber várias entradas. Trata-se de um modelo matemático que, ao usar uma única camada de neurônios consegue resolver apenas problemas linearmente separáveis. Por essa limitação do Perceptron de camada única, segundo [Tinós 2020], é comum se usar o Perceptron Multicamadas, uma vez que a segunda camada pode combinar as saídas da primeira camada de forma a produzir uma classificação final resolvendo problemas linearmente não separáveis.

2.5.4. *Support-vector Machines*

Um classificador baseado em SVM busca, apresentadas no espaço de parâmetros duas classes linearmente separáveis (a e b), traçar o melhor hiperplano de forma com que o elemento da classe a mais próximo ao primeiro elemento da classe b sejam considerados os limites. O hiperplano, portanto, será exatamente construído na metade de tais limites. Aos limite dá-se o nome de "vetores de suporte" pois representam os elementos a e b mais próximos.

Destaca-se também a capacidade do SVM em lidar com situações de separação de classes em espaços n-dimensionais. [Baeza-Yates and Ribeiro-Neto 2013] destacam que na existência de mais dimensões, o método é capaz de resolver problemas de classificação binária com igual eficácia. Se numa situação bi-dimensional a divisão das classes é realizada traçando-se uma reta, numa situação tri-dimensional, por exemplo, um plano seria desenhado de forma a realizar a classificação. Em suma, o SVM é capaz de lidar com diversas dimensões, destacando-se o custo computacional exigido à medida em que mais dimensões (e *features*) são analisadas.

2.6. CICIDS-2017 *Dataset*

Para que fosse possível validar a proposta do presente estudo, seis diferentes *Datasets* disponibilizados pelo *Canadian Institute for Cybersecurity*[1] por meio da *University of New Brunswick*[2] foram utilizados. Todos os seis datasets fazem parte de um projeto denominado CICIDS-2017 publicado por [Sharaf. et al. 2018].

Cada dataset contém dados para um tipo de ataque específico acompanhado de dados de tráfego benigno. Todos possuem 78 *features* com os respectivos rótulos na coluna 79. Embora os dados sejam disponibilizados contendo diferentes tipos de um mesmo ataque eles foram transformados em dados binários de forma a permitir criar um sistema de detecção de intrusão que não tem o objetivo de caracterizar os ataques detectados mas sim diferenciar o tráfego legítimo de um tráfego malicioso. A Tabela 1 apresenta mais detalhes sobre a estrutura dos datasets utilizados:

Tabela 1. Detalhes do *Dataset* CICIDS-2017. Fonte: Elaborado pelo Autor.

ID	Tipo de ataque	Número de pacotes
1	*Brute-force*	432074 benígnos 13835 de ataque.
2	*Infiltration*	288566 benígnos 36 de ataque.
3	DDoS	97718 benígnos 128027 de ataque.
4	*Portscan*	127537 benígnos 158930 de ataque.
5	Botnet	189067 benígnos 1966 de ataque.
6	Web	168186 benígnos 2180 de ataque.

O conjunto de datasets CICIDS-2017 é dividido em oito arquivos, de acordo com [Panigrahi and Borah 2018] e [Stiawan et al. 2020]. Dois arquivos foram descartados pois o primeiro apresentava apenas tráfego benígno e o segundo apresentava uma quantidade de registros muito grande impossibilitando seu processamento no método por exaustão.

3. Metodologia

A presente Seção elenca a relação de materiais e métodos utilizados neste estudo. O hardware utilizado nos testes tinha as seguintes características: 8 processadores Intel(R) Core(TM) i7 CPU 960 @3.20GHz e 16 GB de Memória RAM. A codificação foi realizada em com a linguagem Python utilizando as bibliotecas MLX*tend*, Pandas e *Sci-kit Learn*.

[1] *Canadian Institute for Cybersecurity* - https://www.unb.ca/cic/
[2] *University of New Brunswick* - https://www.unb.ca/

3.1. Desenvolvimento dos Stackings

De forma a implementar os modelos otimizados, os classificadores k-NN, MLP, DT e SVM foram definidos de acordo com os melhores hiperparâmetros obtidos por meio de grid search tunning. A Tabela 2 organiza a relação dos melhores *Ensembles* criados por meio de *Stacking*. Combinando dois ou mais classificadores na camada 1 e um meta-classificador na camada 2 foram criados 44 modelos. O método de abordagem por exaustão se mostrou bastante trabalhoso e trouxe um alto custo computacional. Os modelos elencados na Tabela 2 refletem os melhores *committees* em termos de acurácia para cada tipo de ataque:

Tabela 2. Arquitetura dos melhores *Stackings* criados pelo método de axaustão. Fonte: Elaborado pelo Autor.

Ataque	**ID do *Committee***	**Camada 1**	**Camada 2**
Brute-force	Stacking #1	k-NN + MLP	k-NN
Infiltration	Stacking #2	k-NN + DT + SVM	k-NN
DDoS	Stacking #3	MLP + DT	MLP
Portscan	Stacking #4	k-NN + DT + SVM	k-NN
Botnet	Stacking #5	k-NN + MLP + DT + SVM	k-NN
Web	Stacking #6	MLP + DT + SVM	DT

A fim de resumir o processo metodológico adotado neste trabalho para a concepção dos modelos de IDS, a Figura 4 ilustra todas as etapas percorridas para a obtenção dos *Stackings* e compilação dos resultados. Portanto, as letras (de A à N) da Figura 4 representam cada processo descrito na sequência:

A - Tratou-se do processo de obtenção dos dados para treinamento e testes considerando os dados brutos, ainda sem discretização, normalização e binarização das classes.

B, C, D e E - Processos relevantes para adequação dos dados aos padrões necessários de entrada de dados para os métodos de criação dos classificadores pelos algoritmos de *machine learning*.

Tais etapas que consistiram em (“B”) tranformar todos os dados em números mantendo-se a proporcionalidade das *features*; (“C”) adequar os dados numéricos num mesmo intervalo de forma a garantir a não influência negativa de determinada(s) *feature*(s) em específico; (“D”) criação de duas classes para garantir o objetivo principal de um detector de intrusão (que é mais detectar os ataques do que detectar o tipo deles) e (“E”) representando os dados pré-processados já adequados para a criação dos modelos classificadores e dos testes para extração das métricas de desempenho.

F, G, H e I - A utilização de métodos para obtenção dos melhores hiperparâmetros permitem criar modelos mais adequados de classificação em termos de desempenho nas taxas de detecção. Tratou-se de um processo de repetição onde foram testados diversos hiperparâmetros para os diferentes classificadores.

J, K e L - Processo de treinamento e testes utilizando 10-*fold cross validation* (“J”) e, consequentemente, criação dos modelos (“K”) para futura extração das métricas de desempenho (“L”).

M e N - Para que fosse possível sugerir o melhor *committee* de classificadores para cada

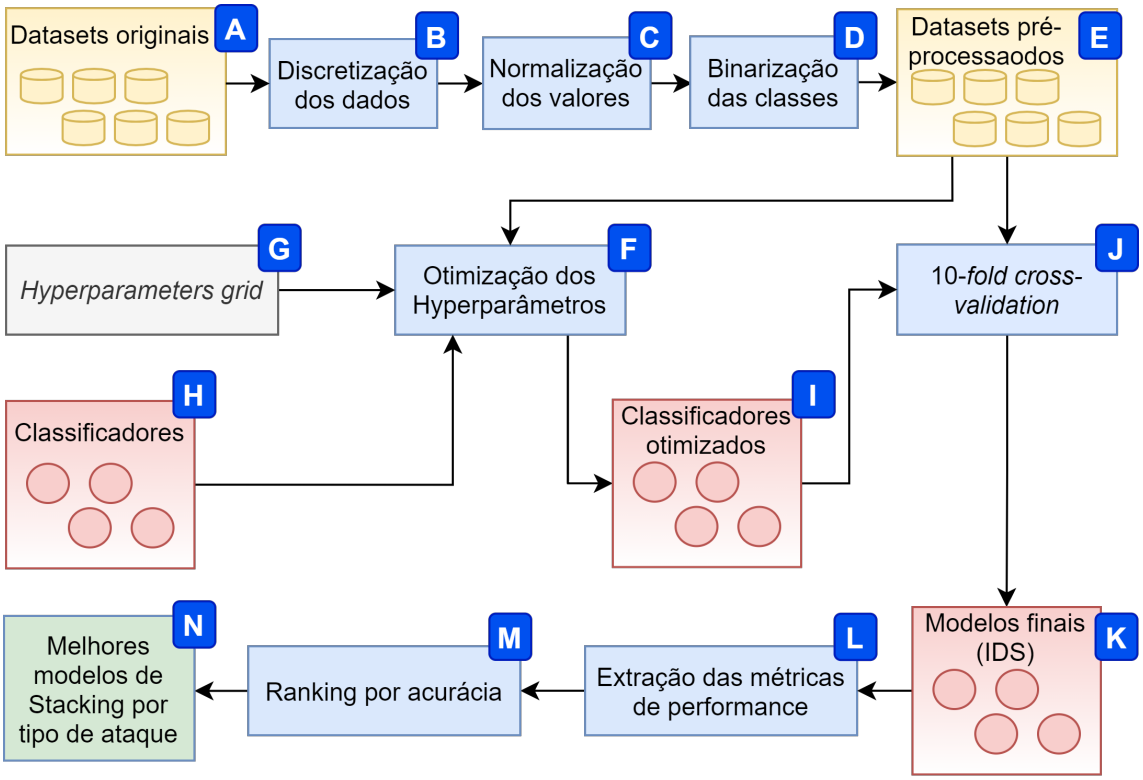

Figura 4. Visão Geral da Metodologia. Fonte: Elaborado pelo Autor.

um dos seis tipos de ataque ("N") utilizou-se como regra o *Stacking* que apresentou melhor desempenho ("M").

4. Resultados

De forma a organizar os resultados obtidos, foram calculados os índices de Acurácia, *Precision*, *Recall* e F1-*Score*. Os valores absolutos de True Positive (TP), True Negative (TN), False Positive (FP) e False Negative (FN) obtidos pelo emprego de uma *Confusion Matrix* foram usados para os cálculos.

A Tabela 3 apresenta os melhores resultados obtidos pelo método de exaustão. Os IDs dos Stackings são baseados na Tabela 2 portanto representam o desempenho de cada *committee* em relação a cada tipo de ataque presente no *Dataset* objeto deste estudo:

Tabela 3. Compilação dos resultados para os melhores *Stackings* criados. Fonte: Elaborado pelo Autor.

Stacking #ID	Acurácia	*Precision*	*Recall*	F1-*Score*
#1	0.99982	0.99993	0.99988	0.99991
#2	0.99991	0.99992	0.99998	0.99995
#3	0.99946	0.99934	0.99942	0.99938
#4	0.99956	0.99962	0.99940	0.99951
#5	0.99720	0.99773	0.99943	0.99858
#6	0.99950	0.99972	0.99977	0.99975

Como forma de observar a melhoria no desempenho no sentido de diminuição do número de classificações incorretas, a Tabela 4 elenca, em porcentagem, para todos os classificadores individuais e para cada *Stacking* criado as taxas de erro de classificação:

Tabela 4. Compilado com todas as taxas de erro individuais e com o *Stacking* escolhido pela aplicação da combinação proposta pelo autor organizados de acordo com os ataques. Fonte: Elaborado pelo autor.

Classificador	*Bruteforce*	*Infiltration*	DDoS	*Portscan*	*Botnet*	Web
Stackings	**0.017**%	**0.008**%	**0.053**%	**0.026**%	**0.196**%	**0.031**%
k-NN	0.018%	0.009%	0.085%	0.044%	0.282%	0.032%
DT	0.029%	0.011%	**0.053**%	0.031%	0.210%	0.049%
SVM	0.528%	0.131%	1.250%	0.464%	0.478%	1.256%
MLP	0.235%	0.131%	0.320%	0.408%	0.448%	0.161%

É nítido observar na Tabela 4 que os *Stackings* criados obtiveram, para todos os ataques que se propuseram detectar, taxas de erro menores em comparação aos classificadores individuais. A diferença é maior quando comparados aos classificadores MLP e SVM, sendo mínima em relação a k-NN e DT. Ademais, a Tabela 5 elenca uma comparação dos resultados obtidos pela aplicação do presente estudo com os trabalhos correlatos citados na Seção 2.1:

Tabela 5. Comparação entre os resultados do presente estudo e os trabalhos correlatos. Fonte: Elaborado pelo Autor.

Proposta	Acurácia	F1-*Score*	*Precision*	*Recall*
[Milliken et al. 2015]		0.98		
[Belouch and hadaj 2017]	0.8572			
[Sun et al. 2018]	0.5727			
[Lu et al. 2019]	0.7076			
[Hsu et al. 2019]	0.9175		**0.931**	**0.921**
[Olasehinde et al. 2020]	0.9856			
[Tama et al. 2020]	0.9604			
Este trabalho	**0.9992**	**0.999**	**0.9995**	**0.9995**

Para os trabalhos que se propuseram a detectar vários tipos de ataques, como o presente estudo, foi calculada a média dos resultados obtidos, para efeitos de comparação. Nem todos os trabalhos traziam as informações de Acurácia, *Precision*, *Recall* e F1-*Score* simultaneamente, portanto, a Tabela 5 considerou somente os valores informados nos trabalhos correlatos. Destaca-se que, conforme supracitado na Seção de trabalhos correlatos, apenas [Tama et al. 2020] utilizaram o mesmo *dataset* usado na presente pesquisa. Entretanto, para fins de comparação mais abrangente, optou-se por não utilizar apenas um único trabalho como comparativo.

5. Conclusão e Trabalhos Futuros

Os resultados apresentados neste trabalho evidenciam o bom desempenho do método *Stacking*. Escolher os melhores classificadores que irão compor um *committee* de forma

exaustiva possibilitou obter a melhor combinação de algoritmos para cada tipo de ataque presente no *Dataset* analisado. Os números de taxa de erro evidenciam que a aplicação dos *committees* da forma como sugeridos reduz significativamente os números de classificações realizadas incorretamente. Destaca-se que, em termos gerais, optar pelos *stackings* apresentados em relação a implementação individual de k-NN e DT traz melhores resultados, porém com pouca diferença siginificativa. O ganho mais evidente é na escolha dos *stackings* em relação a implementação individual dos classificadores SVM e NN, o que fica evidente pela maior diferença nas taxas de erros de classificação.

Como proposta de trabalhos futuros pretende-se implementar métodos de *Ensemble Pruning* de forma com que seja possível encontrar uma combinação de classificadores que seja adequada também em termos de custo computacional. Pode ser inviável, em alguns ambientes, o método exaustivo, pois depende da execução de todas as possibilidades. Então, teoricamente, aplicar uma forma de *Pruning*, pode ser vantajosa.

Referências

Agarwal, S. and Chowdary, C. R. (2020). A-stacking and a-bagging: Adaptive versions of ensemble learning algorithms for spoof fingerprint detection. *Expert Systems with Applications*, 146:113160.

Aggarwal, C. (2014). Data classification: Algorithms and applications, ser. *Frontiers in physics. Chapman and Hall/CRC*.

Baeza-Yates, R. and Ribeiro-Neto, B. (2013). *Recuperação de Informação-: Conceitos e Tecnologia das Máquinas de Busca*. Bookman Editora.

Belouch, M. and hadaj, S. E. (2017). Comparison of ensemble learning methods applied to network intrusion detection. In *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing*, pages 1–4.

Deng, Z., Zhu, X., Cheng, D., Zong, M., and Zhang, S. (2016). Efficient knn classification algorithm for big data. *Neurocomputing*, 195:143–148.

Džeroski, S. and Ženko, B. (2004). Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273.

Fraimovich, D. Y., Donichev, O. A., Grachev, S. A., and Gundorova, M. A. (2020). The role of information and digital resources in regional development. In *Growth Poles of the Global Economy: Emergence, Changes and Future Perspectives*, pages 1305–1316. Springer.

Hsu, Y.-F., He, Z., Tarutani, Y., and Matsuoka, M. (2019). Toward an online network intrusion detection system based on ensemble learning. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pages 174–178. IEEE.

Karatas, G. and Sahingoz, O. K. (2018). Neural network based intrusion detection systems with different training functions. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pages 1–6. IEEE.

Lu, L., Teng, S., Zhang, W., Zhang, Z., Liu, D., and Fang, X. (2019). Error-correcting ability based collaborative multi-layer selective classifier ensemble model for intrusion detection. In *2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 4–9. IEEE.

Milliken, M., Bi, Y., Galway, L., and Hawe, G. (2015). Ensemble learning utilising feature pairings for intrusion detection. In *2015 World Congress on Internet Security (WorldCIS)*, pages 24–31. IEEE.

Olasehinde, O. O., Johnson, O. V., and Olayemi, O. C. (2020). Evaluation of selected meta learning algorithms for the prediction improvement of network intrusion detection system. In *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*, pages 1–7. IEEE.

Panigrahi, R. and Borah, S. (2018). A detailed analysis of cicids2017 dataset for designing intrusion detection systems. *International Journal of Engineering & Technology*, 7(3.24):479–482.

Rezende, S. O. (2003). *Sistemas inteligentes: fundamentos e aplicações*. Editora Manole Ltda.

Rocca, J. (2019). Ensemble methods: bagging, boosting and stacking - towards data science. `https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205`. (Accesso en 23/04/2020).

Sharaf., I., Lashkari, A., Habibi, and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, pages 108–116.

Sharma, H. and Kumar, S. (2016). A survey on decision tree algorithms of classification in data mining. *International Journal of Science and Research (IJSR)*, 5(4):2094–2097.

Smolyakov, V. (2017). Ensemble learning to improve machine learning results. `https://blog.statsbot.co/ensemble-learning-d1dcd548e936`. (Acesso em 30/09/2019).

Stiawan, D., Idris, M. Y. B., Bamhdi, A. M., Budiarto, R., et al. (2020). Cicids-2017 dataset feature analysis with information gain for anomaly detection. *IEEE Access*, 8:132911–132921.

Sun, C., Lv, K., Hu, C., and Xie, H. (2018). A double-layer detection and classification approach for network attacks. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–8. IEEE.

Tama, B. A., Nkenyereye, L., Islam, S. R., and Kwak, K.-S. (2020). An enhanced anomaly detection in web traffic using a stack of classifier ensemble. *IEEE Access*, 8:24120–24134.

Tchaye-Kondi, J., Zhai, Y., and Zhu, L. (2020). A new hashing based nearest neighbors selection technique for big datasets. *arXiv preprint arXiv:2004.02290*.

Tinós, R. (2020). Perceptron multicamadas. `https://edisciplinas.usp.br/pluginfile.php/4445475/mod_resource/content/1/rn_5_mlp_1.pdf`. (Acesso em 04/15/2020).

Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC.