SÃO PAULO STATE UNIVERSITY – UNESP
"JÚLIO DE MESQUITA FILHO"

Thiago José Lucas

About Intrusion Detection in Computer Networks and Computational Systems: A Pruning Proposal to Reduce Computational Cost and Gain Performance using Ensemble Learning

Bauru – SP – Brazil
2023

Thiago José Lucas

About Intrusion Detection in Computer Networks and Computational Systems: A Pruning Proposal to Reduce Computational Cost and Gain Performance using Ensemble Learning

Doctoral Thesis presented for obtaining the title of Computer Science's Ph.D., together with the Postgraduate Program in Computer Science of São Paulo State University "Júlio de Mesquita Filho", Bauru's Campus.

Advisor: Kelton Augusto Pontara da Costa

Bauru – SP – Brazil
2023

Lucas, Thiago José.
About Intrusion Detection in Computer Networks and Computational Systems: A Pruning Proposal to Reduce Computational Cost and Gain Performance using Ensemble Learning / Thiago José Lucas, 2023
96 f. : il.

Orientador: Kelton Augusto Pontara da Costa

Tese (Doutorado)–Universidade Estadual Paulista (Unesp). Faculdade de Ciências, Bauru, 2023

1. Machine Learning 2. Ensemble Learning 3. Intrusion Detection System 4. Computers Network 5. Ensemble Learning 6. Ensemble Pruning I. Universidade Estadual Paulista. Faculdade de Ciências. II. About Intrusion Detection in Computer Networks and Computational Systems: A Pruning Proposal to Reduce Computational Cost and Gain Performance using Ensemble Learning.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Bauru

ATA DA DEFESA PÚBLICA DA TESE DE DOUTORADO DE THIAGO JOSÉ LUCAS, DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO, DA FACULDADE DE CIÊNCIAS - CÂMPUS DE BAURU.

Aos 19 dias do mês de maio do ano de 2023, às 09:00 horas, por meio de Videoconferência, realizou-se a defesa de TESE DE DOUTORADO de THIAGO JOSÉ LUCAS, intitulada **About Intrusion Detection in Computer Networks and Computational Systems: A Pruning Proposal to Reduce Computational Cost and Gain Performance using Ensemble Learning**. A Comissão Examinadora foi constituida pelos seguintes membros: Prof. Dr. KELTON AUGUSTO PONTARA DA COSTA (Orientador(a) - Participação Virtual) do(a) FC / UNESPBauru SP, Prof. Dr. DANIEL CARLOS GUIMARÃES PEDRONETTE (Participação Virtual) do(a) IGCE / UNESP/Rio Claro (SP), Prof. Dr. JOSE REMO FERREIRA BREGA (Participação Virtual) do(a) Departamento de Computação / UNESP/Câmpus de Bauru, Prof. Dr. MURILO VARGES DA SILVA (Participação Virtual) do(a) Instituto Federal de São Paulo, Prof. Dr. MÁRCIO ANDREY TEIXEIRA (Participação Virtual) do(a) IFSP / Catanduva (SP). Após a exposição pelo doutorando e arguição pelos membros da Comissão Examinadora que participaram do ato, de forma presencial e/ou virtual, o discente recebeu o conceito final:_ _ _ _ _ _ _ _ _ _ _ _ _ _ . Nada mais havendo, foi lavrada a presente ata, que após lida e aprovada, foi assinada pelo(a) Presidente(a) da Comissão Examinadora.

Prof. Dr. KELTON AUGUSTO PONTARA DA COSTA

Acknowledgments

To the owner of all science and wisdom. The only one who is omnipotent, omniscient and omnipresent. Gratitude always and first to God, our father. I thank you for the gift of life, for health, and for my family. Everything happens by your authorization, and everything in my life is by your infinite grace and mercy.

For my beloved wife, Nathália. For the patience, for the advice, for the company, and for love. Nothing positive happens to me without you being together, walking beside me. You are the greatest gift God has given me. Thank you for sharing your dreams and projects and supporting my adventures. The world would be much better if every man could have a wife like you. Thanks for existing. I love you lots.

For my little Manuela. When you grow up and understand life, I want you to be proud of your daddy. Not for what he is, because he is nothing, but for what he does for you. I strive daily to be the best man you know until I can find another who deserves your love more than me in the future. You are my motivation, my anxiety, and my pride. I love you.

To my parents, Silvio César and Ângela Maria, for their concern, example, and affection. Thanks for believing in me when I didn't. Love you.

To my advisor, Prof. Kelton. You've been with me since graduation and believed in me through my master's and again through my doctorate. The technical guidance you gave me is nothing compared to the life guidance and advice I received from you, especially the last year when I thought about giving up. May God bless you and your family always with good health and peace.

To my brothers, Larissa Gabrieli and Gustavo Henrique, for their companionship and support. To my wife's parents, Gilberto and Fátima, for all the help and recognition. Love you.

All the effort in preparing this Thesis and all the work I spent on the Ph.D. program I dedicate to the love of my life, my wife Nathália Lucas. All this was for you and for you.

Although the fig tree shall not blossom, neither shall fruit be in the vines; the labor of the olive shall fail, and the fields shall yield no meat; the flock shall be cut off from the fold, and there shall be no herd in the stalls:

Yet I will rejoice in the LORD, I will joy in the God of my salvation.

Holy Bible – Habakkuk's book – 3:17,18

Resumo

Manter os requisitos de Confidencialidade, Integridade e Disponibilidade é um desafio muito relevante para empresas, governos e corporações no que diz respeito à segurança de suas informações. Ataques a redes e sistemas de computadores vêm se intensificando recentemente, tornando-se mais recorrentes e sofisticados. Os Sistemas de Detecção de Intrusão (IDS) são responsáveis por analisar o tráfego de rede ou o comportamento dos sistemas operacionais para detectar comportamentos anômalos e bloquear ataques. Os IDS tradicionais, no entanto, têm dificuldade em detectar padrões de ataque mais complexos, pois seus métodos de detecção (por anomalia ou por assinatura) são antigos e os ataques modernos são robustos e heterogêneos. Neste sentido, a área de inteligência artificial, com ênfase na área de aprendizado de máquina, entrega algoritmos de classificação capazes de reconhecer padrões complexos, permitindo assim a construção de IDS inteligentes que cometem menos erros. A área de aprendizado de máquina também consegue unir diferentes classificadores (ensemble learning) focados em resolver o mesmo problema, aumentando o desempenho quanto aos acertos de classificação, mas com um problema relevante: o alto custo computacional. Esta tese de doutorado está organizada como uma "compilação de artigos" e apresenta uma forma de estimar os melhores classificadores para compor um ensemble com base na diversidade entre eles. Esta escolha permitiu encontrar uma maneira mais aceitável e menos dispendiosa de criar um IDS baseado em ensemble learning que pudesse diminuir os erros de classificação enquanto reduzia o custo computacional. Os materiais e métodos escolhidos foram baseados no estado-da-arte para a área obtido por uma revisão sistemática abrangente da literatura, e os experimentos foram realizados nos cinco conjuntos de dados de intrusão mais relevantes, usando o algoritmo de ensemble "stacking" e os quatro classificadores supervisionados mais comuns na área. Os resultados obtidos estão organizados nos artigos desta compilação e demonstram que a poda pela diversidade resolve o problema estipulado nesta tese: redução de custo computacional e aumento de acertos de classificação de ataques.

Palavras-chave: Aprendizagem de Máquina; Ensemble Learning; Sistemas de Detecção de Intrusão; Redes de Computadores; Ensemble Pruning.

Abstract

Maintaining Confidentiality, Integrity, and Availability requirements is a very relevant challenge for companies, governments, and corporations concerning the security of their information. Attacks on computer networks and systems have been intensifying recently, becoming more recurrent and sophisticated. Intrusion Detection Systems (IDS) are responsible for analyzing network traffic or operating systems' behavior to detect anomalous behavior and block attacks. Traditional IDS, however, have difficulty detecting more complex attack patterns, as their detection methods (by anomaly or by signature) are old and modern attacks are robust and heterogeneous. In this sense, the area of artificial intelligence, with emphasis on the field of machine learning, delivers classification algorithms capable of recognizing complex patterns, thus allowing the construction of intelligent IDS that make fewer mistakes. The field of machine learning also manages to unite different classifiers (ensemble learning) focused on solving the same problem, increasing performance concerning classification successes, but with a common problem: the high computational cost. This doctoral thesis is organized as a "compilation of articles" and presents a way to estimate the best classifiers to compose an ensemble based on the diversity between them. This choice allowed finding a more acceptable and less costly way to create an IDS based on ensemble learning that could decrease classification errors while reducing the computational cost. The materials and methods chosen were based on the state-of-the-art for the area obtained by a comprehensive systematic review of the literature, and the experiments were carried out on the five most relevant intrusion datasets, using the ensemble "stacking" method and the four supervised classifiers most common to the area. The results obtained are organized in the articles of this compilation and demonstrate that pruning for diversity solves the problem stipulated in this thesis: reduction of computational cost and increase of attacks classification hits.

Keywords: Machine Learning; Ensemble Learning; Intrusion Detection System; Computers Network; Ensemble Learning; Ensemble Pruning.

Contents

1 Introduction

Aristotle (n.d.) in your book "Politics" written 350 B.C.E introduced a thought that would later be called "Wisdom of the crowd". The central idea focuses on the possibility that many, although not individually good men become better when they are together, not from an individual point of view, but from a collective point of view.

It is assumed that there is idiosyncratic noise related to each decision-making and that the process, when observed collectively, tends to be less prone to biased decisions due to individual noise. Decision-making, in this sense, with an independent and diverse group of individuals rather than a single specialist, brings the aforementioned "Wisdom of the crowd" as a product. The theme was the subject of a relevant publication by Galton (1907) and became popular in various segments of human knowledge by the book by Surowiecki (2005).

In the context of classification (one of the possibilities of applying machine learning methods), separating one object from another and assigning it a class of belonging is challenging. Several classification algorithms have achieved good results in several areas of human knowledge, such as medicine – classifying a computed tomography image as healthy or sick; as in traffic – classifying a moving object as a car, motorcycle, bus, or bicycle; or as in biometric authentication processes – classifying an iris image, a fingerprint, or even a voice capture, as belonging (or not) to a certain individual.

Although such classification algorithms tend to make little mistakes, even though the training data are complex and abundant, errors, depending on the context, can bring considerable damage to a company, corporation, or even an individual. The cybersecurity area is an example in this case. Attack (thus malicious) packets entering a network or a server can cause enormous damage, even if they are just a few packets. So, applying robust algorithms in the presence of good and adjusted data tends to decrease classification errors correctly, but this has a computational cost. That is, the fewer mistakes, the better.

In theory, the more complex the algorithm, the more robust the training process (epochs), and the more quality data available, the higher the computational cost. In short: more time will be devoted to training. This problem of high computational cost becomes more evident when several classification algorithms are used together (ensemble learning) since the training time for a single classification algorithm is multiplied by the number of algorithms that will compose the ensemble. This is a generalization, as training data can be partitioned to reduce the time (and it usually is), but still, the computational cost is a concern when using ensemble learning.

The challenge that guided this doctoral thesis was to estimate which classifiers should be part of an ensemble so that classification errors are reduced, and computational cost is not a problem. Furthermore, homogeneous ensemble methods (such as bagging or boosting) do

not allow different types of classifiers to perform the same task (be part of the same ensemble), so in these cases, the challenge is usually to choose the most appropriate hyperparameters. However, in the case of a heterogeneous ensemble (such as stacking), it is possible to use different classification algorithms (one can use, for example, a neural network with a decision tree and a support vectors machine). Therein lies the challenge: estimating which are the best classifiers to compose a stacking in the context of cybersecurity.

One hypothesis to estimate the optimal classifiers that will compose an ensemble is by the exhaustive method: choose n classifiers and combine all of them; extract the results and, for the lowest error rate, define that this is the most suitable combination of classifiers. This simple solution is not trivial because this process has a very high computational cost. In this sense, the research problem that this thesis seeks to answer is: is it possible to automate the choice of classifiers that will compose the ensemble to obtain a process that delivers a reduction in classification errors and, at the same time, reduce computational training cost?

We seek to apply the theory of pruning for diversity in the context of cybersecurity, presented by Kuncheva e Whitaker (2003). In our tests, we estimate that the more different the classifiers that will compose the ensemble (stacking) in the first layer, the better the results tend to be. We performed tests on several datasets, and the details can be observed by organizing this compilation of scientific publications:

- Chapter 2 presents the work that guided the choice of methods. This work is the result of a comprehensive survey where we looked for works related to our theme and made several cuts in order to find trends and the state-of-the-art itself;
- Chapter 3 documents the application of the method by exhaustion, which, although it brings the optimal combination of classifiers, is unfeasible due to its high computational cost. Even so, this work brought a challenge for future research the cost reduction by applying to prune for diversity (we added the original version written in Brazilian Portuguese in the Appendix);
- Chapter 4 is our first publication of the diversity pruning method, solving the future research problem left by the work presented in Chapter 3. We document here the results of detecting six different types of attacks on computer networks and computational systems;
- Chapter 5 is the application of the diversity pruning method in a broader scenario, where we perform tests on four different datasets that cover a large and diverse number of attacks on computer networks and computational systems ;
- Chapter 6 documents the last work of this compilation of publications, bringing an alternative in the choice of classifiers where, in an unsupervised way, the most repre-

sentative algorithms are chosen to solve the pruning problem due to concern with the computational cost using the Optimum-path Forest classifier;

- Chapter 7 presents the final considerations about all the works and an overview of the experience in conducting the works documented in this compilation.

The General Objective of this doctoral thesis was: to present pruning for diversity as a possibility of estimate the best classifiers for composing an ensemble so that the computational cost is reduced along with classification errors. The Specific Objectives were as follows:

- Find the state-of-the-art and usage trends of classifiers, datasets, and ensemble methods by performing a comprehensive systematic literature review (documented in Chapter 2);
- Perform the experiments (we use R with the Caret library[1] and Python with the Pandas [2] , NumPy[3], Scikit-Learn[4] and MLXTend[5] libraries) covering pre-processing, training, and classifications;
- Extract the absolute results (false positive, false negative, true positive, and true negative rates) so that it was possible to observe the relevant statistics by traditional methods, such as accuracy, precision, and recall, among others;
- Compare the results in two ways: with the results obtained by the individual classifiers and then with the results obtained by related works;
- Document all procedures to make scientific publications possible and compile the works in this doctoral thesis.

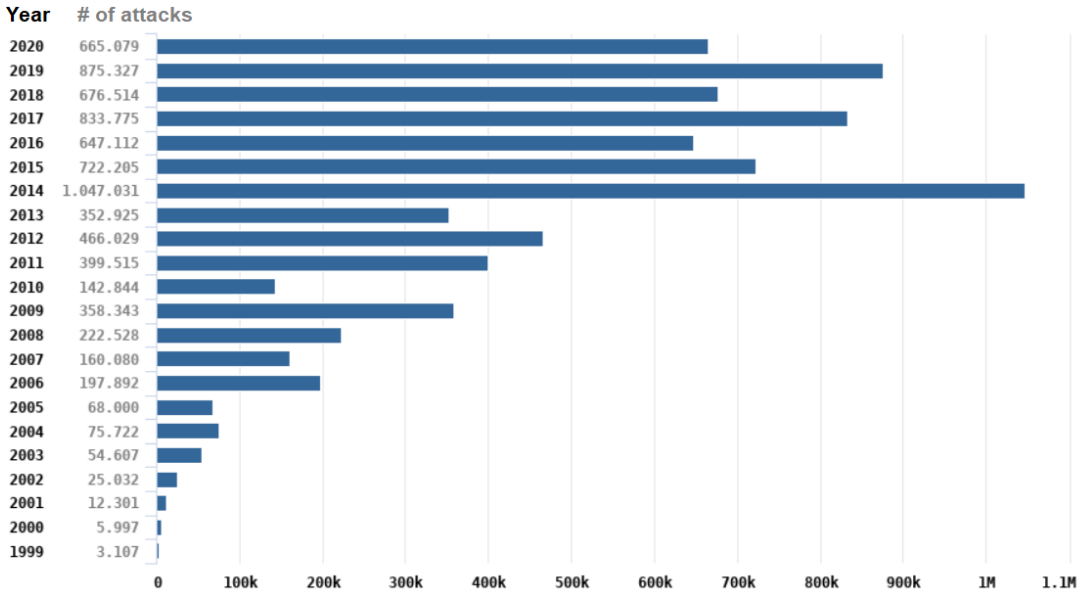
The justification for the efforts in carrying out the research that makes up this collection of works lies in the growing and worrying scenario of computer systems and network attacks. A cybersecurity report by Forbes (2022) compiles projections about the global information security scenario, highlighting the main threats and estimates and suggesting priorities for risk mitigation. According to Forbes (2022), ahead of risks of legal uncertainty (19%), pandemics (22%), natural disasters (25%), and interruption of services (42%), the risks with information security incidents represent in 2022 a total of 44% for businesses globally.

Projections of Morgan (2020) estimate a loss of 10.5 trillion dollars in 2025. For comparative purposes, in 2015, it was 3 trillion dollars, and in 2021, 6 trillion. Cybersecurity investments will maintain an annual level of US$1.75 trillion annually until 2025.

[1] Caret: Classification and Regression Training – https://cran.r-project.org/web/packages/caret/index.html
[2] Pandas: Python Data Analysis Library – https://pandas.pydata.org/
[3] NumPy: The fundamental package for scientific computing with Python – https://numpy.org/
[4] Scikit-learn: Machine Learning in Python – https://scikit-learn .org/
[5] MLXTend: Python machine learning extensions – http://rasbt.github.io/mlxtend/

A snippet of the scenario where we analyze the data from Brazil can be seen in Figure 1:

Figure 1 – General statistics about cybersecurity scenario in Brazil.



Source: Adapted from CERT.br (2020)

Analyzing Figure 1, regardless of the variation between years, it is evident the growth in the number of attacks notified to CERT.br (entity belonging to the Internet Management Committee in Brazil and responsible for organizing data and creating general national cybersecurity policies) year by year. There was a peak in 2014 due to political demonstrations and the FIFA Soccer World Cup in Brazil. In a normalized view, from 1999 to 2020, growth is important and worrying. In this sense, efforts to create security mechanisms by implementing robust intrusion detection systems are justifiable.

The guiding hypothesis of this thesis was that estimating the best classifiers to compose the Ensemble using the exhaustive method would be computationally infeasible due to the high training time. So, finding a logic that could perform such an estimate with a lighter algorithm would be ideal as we would benefit from error reduction of Ensemble methods with the non-dependence of a high computational cost. Hypothetically, by applying diversity pruning, we would arrive at the optimal estimate in all or most cases. The results obtained corroborate this hypothesis in the vast majority of scenarios.

The next chapters, as mentioned above, bring the publications in their original format, preserving the characteristics of the journals and conferences.

2 A Comprehensive Survey on Ensemble Learning-based IDS Approaches in Computer Networks

This survey was submitted to the "Knowledge-based Systems Journal"[6] evaluated by Brazilian CAPES with A1 Qualis score. This extensive research aims to understand the details of the application of ensemble learning techniques in the cybersecurity scenario, considering what has been published in recent years. We got the state-of-the-art and described the details as follows:

- There were 238 works published between 2015 and 2021 that the Systematic Literature Review collected;
- After filtering, we reached 138 works directly related to the research objective, which were analyzed in order to extract details that could document both the state-of-the-art and trends in the use of classifiers, datasets, and ensemble methods;
- The works were briefly summarized and segmented by year, and microdata was extracted so that it was possible to create taxonomies and observe trends;
- We detailed the four most used classifiers that showed some tendency, and the same was done for the five datasets and the four most popular ensemble methods;

Although the graphic clipping focuses only on the four most used classifiers, it is also noticed a tendency to use the supervised classification algorithms Decision Tree, k-Nearest Neighbors, Neural Networks, and Support vectors Machine. It will be possible to perceive a tendency to use the Stacking method in front of the other ensemble algorithms (bagging, boosting, and voting). Of the five most repeated datasets in the works, two present old network traffic (not contemplating modern attacks), and the other three differ for being constituted of relatively recent attacks.

The work can be seen in sequence, in full, and in its original format to maintain the characteristics of the journal.

[6] Knowledge-based Systems is an international and interdisciplinary journal in the field of artificial intelligence – https://www.sciencedirect.com/journal/knowledge-based-systems

A Comprehensive Survey on Ensemble Learning-based IDS Approaches in Computer Networks

Abstract

Machine learning algorithms present a robust alternative for building Intrusion Detection Systems due to their ability to recognize attacks in computer network traffic by recognizing patterns in large amounts of data. Typically, classifiers are trained for this task. Together, ensemble learning algorithms have increased the performance of these detectors, reducing classification errors and allowing computer networks to be more protected. This research presents a comprehensive Systematic Review of the Literature where works related to intrusion detection with ensemble learning were obtained from the most relevant scientific bases. We present 138 works and several compilations of datasets, classifiers, and ensemble algorithms, in addition to documenting the experiments that stood out in their performance. A characteristic of this research is its originality. We did not find surveys in the literature specifically focusing on the relationship between ensemble techniques and intrusion detection. We present a timeline-based view of the works studied to highlight evolutions and trends. The results obtained by our survey show a growing area, with excellent results in detecting attacks, but with needs for improvement in pruning for choosing classifiers, which makes this work unprecedented for this context.

Keywords: Intrusion Detection Systems, Machine Learning, Ensemble Learning, Cibersecurity

1. Introduction

Information can be considered the greatest asset of a company or corporation in the modern economic scenario, according to [1], and this fact highlights that owners should be concerned about safeguarding their data, especially about confidentiality, integrity, and availability. The fourth industrial revolution, or "Industry 4.0", a term created by the President of the World Economic Forum, [2], describes the paradigm shift in the way people live and relate to each other and, in particular, how companies and corporations need to know how to manage information in order to survive in a competitive market, reinforcing the need as mentioned earlier to protect data through the implementation of information security methods.

A cybersecurity report by [3] compiles projections about the global information security scenario, highlighting the main threats, estimates, and suggesting priorities for risk mitigation. According to [4] and [3], ahead of risks of legal uncertainty (19%), pandemics (22%), natural disasters (25%) and interruption of services (42%), the risks with information security incidents represent in 2022 a total of 44% for businesses globally.

Projections of [5] and [3] estimate a loss of 10.5 trillion dollars in 2025. For comparative purposes, in 2015, it was 3 trillion dollars, and in 2021, 6 trillion. Investment in cybersecurity tends to maintain the annual level of 1.75 trillion annually until 2025.

Also, according to [6] and [3], it is worth noting that prioritizing the protection against attacks based on people (social engineering), investing in limiting data loss and business interruption, and in particular, applying advanced analytics and intelligence technologies for security are essential for the survival of companies in an environment as critical as information security.

The statistics mentioned above justify the efforts employed in elaborating this research.

Given the context presented, it is essential to apply intelligent methods to detect attacks to prevent damage. Intrusion Detection and Prevention Systems (IDS/IPS) based on machine learning can analyze large amounts of data coming from network traffic and, through the training step, create robust models capable of recognizing attack patterns.

Although not trivial, creating an IDS model is a challenge already known and debated in cybersecurity community. The challenge is increasingly related to reducing the computational cost of training and increasing detection results by reducing false-positive and false-negative alarms.

The general objectives of this work is to obtain state-of-the-art ensemble learning techniques applied to intrusion detection. The specific objectives were as follows:

- Conducting a comprehensive systematic review of the literature to obtain related works from the pe-

riod between the years 2015 and 2021 (to work with years already ended, allowing to organize complete annual statistics and present a timeline-based view of the works);

- Extraction of the main characteristics for the works obtained so that it was possible to observe trends in the use of classifiers, datasets, and ensemble algorithms.

The remainder of this manuscript is organized in such a way that Section 2 contains an explanation concerning the method used to produce this work. Section 4 covers brief descriptions of the main components of an IDS. Section 3 contains detailed reports on each of the works listed in this review. In Section 5 we describe our interpretation of the area in light of the works listed, and finally, in Section 6 we report the conclusion of this work.

2. Methodology

A systematic literature review aims to identify, analyze and interpret available evidence related to a particular research topic or phenomenon of interest. For the production of this review, the set of instructions related in the [7] study were followed. The main tasks listed can be seen in Figure 1.

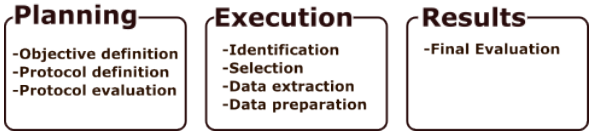


Figure 1: Systematic review procedure, adapted from [7]

The planning task consists of elaborating a protocol that is the central element of the work that will guide the execution. The development of a suitable protocol aims to eliminate biases that may be committed by the authors [8].

The main objectives of this study can be summarized as follows: obtaining the State of the Art (SOTA) in the research area relevant to IDS after year 2014, obtaining an overview of the main works related to the components and techniques used in elaborating an IDS, and reporting the main contributions.

A systematic literature review was carried out, where articles registered in the scientific bases IEEEExplore, ACM Digital Library, Springer, Wiley, Hindawi and MDPI were collected by searching in the intervals between the years 2015 and 2021 for records that contained the terms ”ensemble“ and ”intrusion“ in both the title and the keywords.

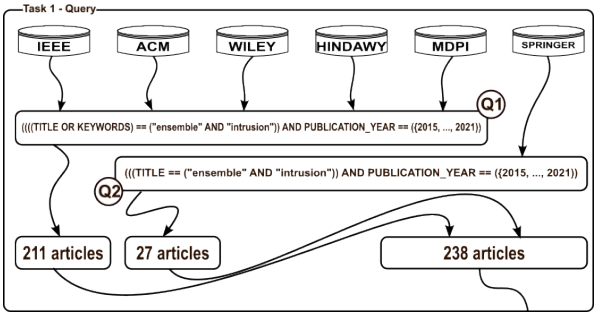


Figure 2: Acquisition Queries.

Figure 3: Filtering Process.

The search protocol can be seen in Table 1. The acquisition process of related articles is shown in Figure 2; it is possible to notice that only for the Springer repository the search criterion did not take into account the occurrences of keywords in the abstract, this was since if we added the abstract as a search criterion the number of articles returned exceeded five thousand articles. Thus, the acquisition of articles was restricted to the title for the Springer repository.

Once the phase of acquiring articles related to the area of interest was completed, which totaled 238 articles segregated by repositories. Then the filtering phase of the list of articles took place as illustrated in Figure 3. The refinement was performed using the reading technique proposed by [9], and as a criterion for the inclusion of articles, we selected all articles that denoted

Figure 4: Wordcloud obtained by all abstracts.

Table 1: Research protocol applied to perform this work.

General Info	
Title	A survey on Ensemble Learning applied to intrusion detection in computers networks.
Objectives	Extract SOTA of Ensemble Learning applied to intrusion detection.
Research Question	
Keywords	ensemble; intrusion
Search string	Q1 ->((((TITLE OR KEYWORDS) == (”ensemble” AND ”intrusion”)) AND PUBLICATION_YEAR == ({2015, ..., 2021)) Q2 ->(((TITLE == (”ensemble” AND ”intrusion”)) AND PUBLICATION_YEAR == ({2015, ..., 2021))
Sources Selection Criteria	Digital repositories of relevant scientific articles withinr computer science academia.
List of sources	IEEE, ACM, MDPI, Springer, Hindawi, WILEY
Selection and Evaluation of Studies	
Study selection strategy and assesment of quality	We adopted the procedure described in [9] to identify studies concerning the research topic of interest to this study.
Inclusion criteria	Considering the first step of [9], all articles that denoted covering the area of security and computer networks were included in the scope of this study.
Exclusion criteria	Considering the first step of [9], all articles that did not cover the area of security and computer networks were excluded from the scope of this study.
Data synthesis and presentation of results	
Data Extraction Strategy	Analysis of essential topics was based on analyzing relevant terms extracted from the abstracts of all articles, with the help of wordcloud formulation. Among the relevant terms, it was possible to note terms such as MODEL, DATASET, ACCURACY, DETECT, and LEARNING. These keywords guided the compilation of studies.
Summarization strategy	The summarization of the results was based on the relevant terms contained in the abstracts, which can be viewed in the wordcloud as noted in Figure 4. With the selection of these relevant terms, we segregated the studies by using Datasets, Ensemble Models, and consequently the Base Classifiers, already making up the rest of the article reading process according to [9].

covering the area of security and computer networks included in the scope of this study. As exclusion criteria, we drop all studies out of inclusion criteria. At the end of this stage, we have 138 selected articles related to the topic of interest.

Once the selection of articles is finalized, and in order to obtain insights into the research area of interest, we use a word cloud that provides data and information to management and support decisions [10]. Exploration of essential topics was based on analyzing relevant terms extracted from the abstracts of all articles, with the help of word cloud formulation, as noted in 4. Among the relevant terms, it was possible to note terms such as MODEL, DATASET, ACCURACY, DETECT, and LEARNING. These keywords guided the compilation of the studies.

The summary of the results was based on the relevant terms contained in the abstracts, which can be viewed in the word cloud presented in Figure 4. With the selection of these relevant terms, we divided the studies by used Datasets, Ensemble Models, and consequently the Base Classifiers, already making up the rest of the article reading process according to [9]. Still regarding the total volume of articles, it can be seen in Figure 5 that the IEEE and Springer repositories represent 84%

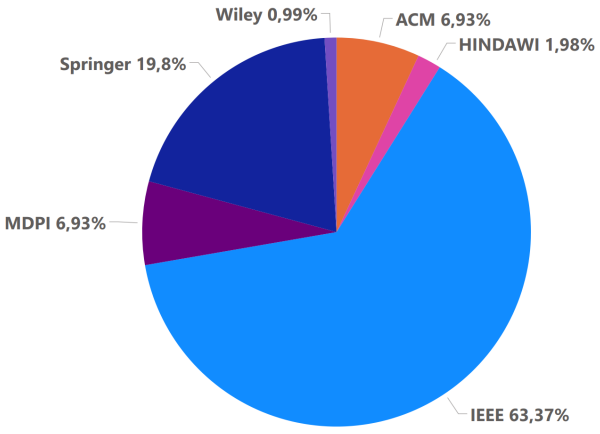


Figure 5: Distribution of works obtained by publisher.

of the total number of articles listed in this study.

In this section, we described the elaboration of the research protocol and the tasks performed to complete the research protocol. At the end of the tasks, we obtained 138 articles related to the subject of Intrusion Detection Systems using Ensemble techniques. With the articles in hand, we created a word cloud that gave us insights into this work's subjects and relevant components. In the next section, we produce a careful and synthesized article-by-article review of those selected by the research protocol section.

3. Literature Review

In this section, we will discuss article by article those chosen after the selection process, as presented in Table 1. To conduct a chronological development line, we segregated the reports by year of publication in which we described the major findings for each manuscript.

3.1. Works published in 2015

[11] carried out an experiment using the KDD-CUP'99 dataset, which was initially processed by the C4.5 algorithm to generate Decisions Trees that, in a second step, were grouped by the adaboost ensemble learning algorithm. Finally, Snort integration was carried out to perform adequate tests in real-time. The authors highlight the achievement of a Detection Rate of 89.56%, while the value of the False Alarm Rate was 0.1% for the tests performed.

[12] used Ensemble learning techniques for the feature selection process and the alarm classification. A union and intersection logic between the features obtained by the Best First, Genetic Search, and Rank Search algorithms allowed an adequate choice of the best features in the NSL-KDD dataset. The Ensemble was performed through Majority Voting among the Bayesian Network, Naive Bayes, random forest, and J48 classifiers in the classification process. The proposed model obtained a True Positive Rate of 98.0% against a False Positive Rate of 0.021%.

[13] proposed a Stacking Ensemble model. The authors used the ISCX 2012 dataset as a basis for the tests and the OneR, Conjunctive, and Naive Bayes classifiers in a Stacking model to perform the Ensemble. The experiments showed that the analysis of entropy between pairs of features from different subsets could contribute to the generation of a subset more suitable for attack classification. The experiments obtained an F-Score of 92% as the best result.

[14] proposed a combination of Core Vector Machine as a classifier and Bagging as an Ensemble technique. Other tests were performed by the authors and are documented in their publication, such as the application of adaboost to Ensemble and Naive Bayes, decision tree, and random forest as classifiers. However, the best results in the KDD-CUP'99 dataset pointed to an accuracy of 98.7% for DoS detection, 98.78% for probe attacks, 98.16% for R2L, and 98.41% for U2R using the combination of Bagging and Core Vector Machine.

[15] presented an expert model for detecting R2L and U2R attacks. It combines Naive Bayes, multilayer perceptron, decision tree, support vectors machine, and k-nearest neighbors classifiers using the Ensemble adaboost technique. The best features in the KDD-CUP'99 dataset were selected through correlation analysis. The best results obtained were a sensitivity of 76% for Naive Bayes and multilayer perceptron and a specificity of 99.05% for a combination of different Naive Bayes classifiers.

[16] presented a model that uses Bagging as an Ensemble technique for reptree classifiers. The authors selected the best features in the NSL-KDD dataset using BIRCH Hierarchical Clustering. The results obtained were 99.67% accurate, while a False Positive Rate of only 0.3% was observed. Although they did not specify which classifier was used as a base, [17] also modeled an intrusion detector using Bagging on the same NSL-KDD, and the measurement accuracy in detection was 97.85%.

Through the one-vs-all Ensemble technique, [18] combined three types of classifiers to improve the performance in intrusion detection for the proposed model. The authors used decision tree, neural network, and support vectors machines algorithms as classifiers. The dataset used in the tests was the NSL-KDD and the feature selection technique used was not specified. The accuracy obtained in the classification was 97.35%.

[19] presented two models that are experts in detecting DDoS attacks. Using random forests as classifiers, the authors modeled two Ensembles: one with adaboost and the other with bagging to compare them. The bagging-based model obtained the best results when observing the values obtained for False Negative Rate, False Positive Rate, and Precision. The only evaluation metric where adaboost was superior was Recall. All tests were performed on the LLS-DDoS1.0, CAIDA 2007, and CAIDA Conficker datasets.

As it is a standard protocol for remote access on Unix and Linux servers, the SSH protocol is widely used for dictionary attacks to gain remote privileges to control vulnerable servers. This is what [20] claims in their experiment, where they tested different Ensemble methods (bagging, boosting, adaboost, Multiboost-

ingAB, and Rotation Forest) to classify alerts from the Euskalert honeypot. The best results were with Bagging, where it was possible to measure a True Positive Rate of 99.93%.

[21] experiment with particle swarm optimization and tree-based classifiers for network intrusion detection. PSO is used for feature selection, and the classifiers (C4.5, RF, and CART) were tested in multiple ensemble formats. The NSL-KDD dataset was used. The best accuracy (99.80%) and false-positive rate (0.02) were obtained with PSO-50 and the Average Probability ensemble scheme.

3.2. Works published in 2016

An Ensemble with C4.5, reptree, and RTree classifiers combined with Bagging was presented by [22] In the pre-processing, the authors chose the best features based on correlation for the ISOT dataset, and the tests performed pointed to an accuracy of 99.97%.

[23] presented an intrusion detector for wireless networks. Using decision trees as classifiers, the authors modeled an Ensemble technique by majority voting where the most voted label among Bagging, random forest, and an extra tree is elected. The tests were performed on the AWID dataset, and the results showed an accuracy of 96.32% and a precision/recall of 96%.

[24] highlights two problems that Machine Learning methods usually present when dealing with alert classification situations: unbalanced classes and speed of streams for real-time detection. To create a model capable of dealing with these problems more effectively, the authors modeled an Ensemble using boosting to combine the following classifiers: J48, JRIP rule learner, Naïve Bayes tree, Naïve Bayes, OneR, logistic model trees, logistic regression, decision stumps, and k-nearest neighbor. The best results in tests performed with the ISCX IDS 2012 dataset obtained a precision of 88.28%, a recall of 80.79%, and an area under the precision/recall curve of 89%.

Using an Ensemble proposal to implement rotation forests, [25] experimented with 20 classifiers to create a model for intrusion detection in wireless networks. Several combinations were tried in the GPRS dataset, and the best results for detection in WEP/WPA and WPA2 bases (which are protocols for wireless networks) were obtained with the best first decision trees grouped by the Ensemble rotation forest algorithm. The area under the precision/recall curve calculated in the tests was 96.01%.

[26] presented an intrusion detector for cloud computing that distributes the task of capturing packets among the network hosts and, by sampling, unites in a central controller the Ensemble routine that was implemented using Bagging. The classifier implemented by the authors was the C-Fuzzy decision trees. In tests with the KDD-CUP'99 dataset, the proposed model obtained an accuracy of 99.47%.

A combination of two feature selection methods, PCA and LDA, combined with the SVM classification algorithm was proposed by [27] The pairs of standard features obtained by the PCA and LDA methods applied to the KDD-CUP'99 dataset are processed by a weighted majority voting Ensemble for ten SVM classifiers. The authors obtained an accuracy of 92.1% in the classification of alerts combined with a false positive rate of 1.9% and a false negative rate of 10.8%.

[28] presented an expert intrusion detection system for cloud computing focusing on DDoS attacks. With its dataset. The proposed Ensemble method was called Consensus Cluster Plus, which united hierarchical cluster classifiers. The authors did not present results that could measure the quality of the presented system.

[29] presented an Ensemble using Random Committee and voting. Combining Bayesian network and random tree classifiers, the method obtained excellent results when observed by the area under the precision/recall curve in the KDD-CUP'99 dataset: 99.9% for Probe and R2L; 100% for DoS, and 99.5% for U2R.

DAREnsemble is an IDS architecture developed by [30] that uses rule learners and decision trees, implemented with the voting role combination method. Compared to other ensemble models, such as Bagging and adaboost, DAREnsemble presents better results when using Average Probability as a combination rule, reaching 99.88% accuracy. For training and testing, the NSL-KDD dataset was utilized.

Working with KDD99 and ISCX IDS datasets, [31] propose a distributed intrusion detection framework using ensemble learning and non-trainable functions, CAGE-MetaCombiner. On the KDD99 dataset, the model performs better than the comparison model for minority classes and almost as well on the majority classes. The model also performs better on the ISCX dataset than the non-specialized algorithm with a specialized ensemble.

A system called SCDNN is proposed by [32], and it combines spectral clustering and deep neural network algorithms. SC separates features by similarity, facilitating the identification of unknown attack types. The KDDCup99 and NSL-KDD are divided into six subsets for testing the model. Analysis of performance is done by dataset, and results vary for each class and dataset. The system outperforms the comparison mod-

els in many instances.

3.3. Works published in 2017

[33] performed several experiments on the NSL-KDD dataset to compare the following Ensemble techniques: majority vote, weighted vote, naive Bayes combination, Bagging, boosting, rotation forest, and random forest, all using Naive Bayes classifiers. The best results were obtained with the naive Bayes combination, where it was possible to measure 84.13% of correctly classified alerts while 15.86% of erroneously classified alerts were observed.

[34] proposed an expert intrusion detection system for web attacks. The Ensemble LogitBoost technique increased the ability of random forest classifiers to detect attacks. Performing tests on the NSL-KDD and UNSW-NB15 datasets, the authors performed the selection of features combining wrapper and filter techniques. The observed detection rates were 89.75% and 99.10% for the datasets mentioned above, respectively.

[35] presented a sequential ensemble for alert classification. In the first layer, an EMSVM classifier is applied, and later (considering that only data labeled as benign in the current step go to the next step), the other packets are classified by k-NN and by SMO in the order presented. The selection of the best features is made with an Intelligent Agent-based Attribute Selection Algorithm (IAASA). The results obtained in the KDD-CUP'99 dataset were approximately 97% accurate for Probe, 97.5% for DoS, 99% for U2R, and 86% for R2L.

[36] presented an unsupervised intrusion detector that creates an Ensemble by voting between four clustering algorithms: DBSCAN, One-SVM, Agglomerative Clustering, and Expectation Maximization. The results obtained in the experiments with the NSL-KDD dataset were a detection rate of 91.03% and a false positive rate of 2.26%.

After selecting the best features in the NSL-KDD dataset using PSO and correlation-based selection, [37] implemented three Ensemble techniques for SVM classifiers: majority voting between adaboost and RSM. The proposed model obtained an accuracy of 85.01% and a false positive rate of 12.6%.

[38] created majority voting ensembles for random forest classifiers to detect intrusion in three different datasets: NSL-KDD, UNSW-NB15, and GPRS. For the datasets above, the observed accuracy was 99.57%, 95.5%, and 91.8%.

[39] proposed an Ensemble-based intrusion detection system of clusters. Using the Gure KDD dataset and the ADTree, k -Means, and k-NN classifiers through weighted majority voting for binary classification, the tests obtained an accuracy of 99.93% and a detection rate of 99.8%. [40] performed a similar experiment were using several distance metrics. The authors obtained 90% accuracy using k-means in the UNSW-NB15 dataset. In the same vein, [41] proposed a detector that addresses the application of Ensemble methods through voting using several unsupervised classifiers in the NSL-KDD dataset; however, the authors did not disclose the results obtained.

[42] performed a comparison between three different Ensemble techniques: booting, Bagging, and Stacking. The authors combined four different classifiers: decision tree, Naive Bayes, multilayer perceptron, and reptree. All experiments were performed on the UNSW-NB15 dataset. The publication documents all the results, and it is possible to observe that the best Ensemble in terms of accuracy was a Stacking of reptree followed by a Stacking of decision trees.

[43] presented an intrusion detector that combines a selection of the best features using an information gain ratio and a voting scheme between Bagging, Random Committee, and decision tree. Considering that the first two use random trees as classifiers, the authors performed tests on the NSL-KDD and KDD-CUP'99 datasets, were both pointed to a valid positive rate of 100% and a false positive rate of 0%.

[44] presented their Ensemble model using Very Fast decision trees classifiers and selecting the best features with Hoeffding bound. The presented model obtained a detection rate of 98.58%. All tests were performed on the KDD-CUP'99 dataset.

Three different datasets were used to test an Ensemble of support vectors machines proposed by [45] Using Rough set theory to obtain the best features, the authors obtained 99.95%, 100%, and 99.98% accuracy for the KDD-CUP'99, HTTP CSIC 2010, and UNB ISCX datasets, respectively.

[46] created an intrusion detection model that combines several Ensemble methods using the decision tree as the base classifier. Each cluster uses a technique: Bagged tree (decision trees based on Bagging), adaboost, logitboost, gentleboost (derivation of adaboost), and RUSboost (boosting algorithm for class balancing). The best detection rates were obtained with bagged tree and Gentleboost, obtaining 99.1% when tested on the UNSB-NB15 dataset.

Seeking to detect Neptune-type attacks (a kind of DoS) on the NSL-KDD dataset, [47] created a binary classifier capable of detecting intrusion by differentiating normal from abnormal connections. The authors im-

plemented an Ensemble via adaboost joining decision stump classifiers which allowed them to obtain an accuracy of 87.83%. The feature selection method used in the experiment was the Information Gain Ratio.

[48] presented an intrusion detector that uses a neural network to create an Ensemble containing an autoencoder, a deep belief neural network, a deep neural network, and an Extreme Learning Machine. The presented detector is capable of performing binary classification. The author performed the experiments on the NSL-KDD dataset and obtained a detection rate of 97.9% in addition to an accuracy of 92.4%

[49] propose an IDS focused on enhancing the detection rate by treating minority classes. KDDCup99 dataset is utilized to evaluate the system's performance on multi-class classification, using different base classifiers: Logistic regression, J48, and Naive Bayes. All classifiers perform better when applied with the proposed method.

3.4. Works published in 2018

An Ensemble of SVM classifiers using Bagging as clustering techniques was proposed by [50] The intrusion detector obtained a subset through PCA applied to the NSL-KDD dataset. The authors divided the classification task between three detectors according to three computer network protocols present in the dataset mentioned above. A genetic algorithm collaborates in the process of choosing the weights in order to potentiate a particular classifier to the detriment of another in the cluster. The accuracy obtained in the tests was 88.28%.

The purpose of the intrusion detector proposed by [51] is real-time data processing. The authors point out that if statistical properties of data alterations are observed throughout monitoring, particularly the variance between attacks, it may be possible to build a more robust classification system. In this sense, the authors proposed an Ensemble-based on the concept of incremental drift learning. The classifiers used were Naive Bayes, Stochastic Gradient Descent, and multilayer perceptron. In tests performed on the KDD-CUP'99 dataset, the observed accuracy was 94.91%.

[52] presented an intrusion detection model that combines three classic Ensemble techniques: Bagging, boosting, and Stacking. The base classifiers used were SVM and k-NN. With tests performed on the NSL-KDD dataset after generating a subset through PCA, it was possible to measure the best accuracy results for the classification of regular packages with 99.41% and that of probe packages with 93.13%.

[53] presented a semi-supervised classification model. Consisting of two steps, the classifier uses an Ensemble through weighted voting composed of decision trees and neural networks trained in a supervised way, that is, with access to data labels. In a second moment, there is the removal of redundant and noisy data in the dataset sample without labels so that the resulting subset is processed by the same Ensemble of the first moment. The authors used a subset obtained by applying PCA to the NSL-KDD dataset, and the accuracy obtained was 84.54%.

[54] proposed an intrusion detection model that groups, through Bagging, three classifiers, namely Naive Bayes, Adaptive Boost, and PART. The dataset used by the authors was the KDD-CUP'99, from which the best features were obtained by observing analyzes based on entropy and filtering. The results obtained point to an accuracy of 99.97%.

[55] modeled an intrusion detection system for wireless networks. Using decision trees for classification, the authors tested four different Ensemble techniques: Bagging, random forests, extra-trees, and XGboost. After pre-processing the AWID dataset, the resulting subset was better processed by combining decision trees and random forests, which showed an accuracy of 95.87% in multi-class detection.

[56] presented an intrusion detector based on an Extreme Learning Machine and optimized with a bat algorithm (inspired by the behavior of bats), whose function was to eliminate the worst classifier among the four different subsets that were generated through bootstrap selection (more details on bootstrapping are covered under "Bagging" in Section [sec:machinelearning]) in the original dataset. The Ensemble was applied by majority voting among the three classifiers not eliminated by the meta-heuristic optimization algorithm. The authors performed tests on the KDD-CUP'99, NSL-KDD, and Kyoto datasets. The measured accuracies were 98.94%, 97.46%, and 99.19% for the datasets mentioned above.

[57] proposed an Ensemble by majority voting. The label most voted by three different classifiers, namely logistic regression, neural network, and decision trees, is attributed to the analyzed data. The subset processed by the proposed method was obtained after applying the PCA method to the KDD-CUP'99 dataset. The accuracy obtained in the tests was 96.14%. Another proposal of Ensemble by majority voting was presented by [58] where the authors reduce the dimensionality of the datasets using a deep belief network and classify the data through a grouping of support vector machines. This model was tested in four different datasets: NSL-KDD, KDD-CUP'99, UNSW-NB15, and CICIDS 2017, where the areas under the precision/recall curves were, respectively, 98.56%, 98.44%, 98%, and 96.30%.

[59] proposed a multi-class intrusion detector with k-SVM, a combination of k-means with support vector machines. The model presented a detection rate of 99.7%. The best features in the NSL-KDD dataset are obtained after applying Genetic Linear Discriminant Analysis in the pre-processing stage. The Ensemble was implemented through voting.

[60] compared adaboost and bagging to build a more effective intrusion detection system. The authors used different classifiers in the tests with the NSL-KDD dataset: decision tree, random forest, decision stump, reptree, and random tree. The authors used two methods to select the best features: information gain ratio and leave-one-out. The best results were obtained with the features selected through information gain ratio and classified with grouping via bagging decision tree classifiers, obtaining an accuracy of 84.25%.

[61] modeled an intrusion detector that uses a sparse autoencoder network to reduce the dimensionality of the NSL-KDD dataset. An ensemble of neural networks using xgboost is responsible for classifying the packets. The authors also used the SMOTE technique to create synthetic packages seeking to increase the data of classes with fewer populations. The classification layers were organized in the form of a binary tree. For DoS, Probe, U2R, and R2L attacks, the observed F1-score was 98.86%, 86.02%, 77.89%, and 98.14%, respectively. For regular packages, it was 98.98%.

[62] compares several simple classifiers with the Ensemble bagging, adaboost, and random forest techniques. Using the UNSW-NB15 dataset, the authors obtained the best result in an Ensemble of random forests reaching 98.1% of the area under the ROC curve. The other classifiers used in the tests were multilayer perceptron, decision tree, Bayesian network, and support vector machine.

[63] presented an intrusion detector tested on Darpa DDoS, Darpa DDoS Malware, and DoS DNS datasets. The grouping of holding trees classifiers using the Accuracy Updated Ensemble was responsible for obtaining an accuracy of 100%.

[64] propose a novel ensemble classifier for intrusion detection based on Naive Bayes and ADTree (a combination of decision tree and boosting), with a majority voting ensemble. NSL-KDD dataset is used in the experiments. Interquartile Range (IQR) is used to remove outliers from the dataset. The average accuracy achieved was 97.11%.

[65] propose a 2 phase NIDS which applies binary classifiers to solve multi-class classification in a One-vs-All strategy. A decision tree is used for the aggregation of predicted values. Features are selected from the NSL-KDD dataset using domain knowledge. Accuracies of 99.99% and 99.89% were observed for binary and five-class classification, respectively.

[66] present an ensemble technique using supervised and unsupervised learning for detecting network anomalies and misuse. The experiments were performed in three different datasets, the NSL-KDD, Kyoto 2006+, and KDD Cup99. Seven different combinations of base classifiers were tested in a clustering ensemble approach, and K-means + J48 and K-means + RF showed the best performances for all datasets.

Aiming to identify the critical features for the construction of an intrusion detection model with high accuracy, [67] propose the use of Chi-Square feature selection and SVM, modified Naive Bayes, and LPBoost classifiers with the majority voting for an IDS. The model's analysis is done with the NSL-KDD dataset. It can detect DoS and R2L attacks with 99% accuracy, probe attacks with 98% accuracy, and u2R with 100% accuracy.

3.5. Works published in 2019

[68] modeled an intrusion detection system with adaboost. The authors' focus was on creating a specific model for IoT environments. The selection of features was performed by applying the correlation coefficient between the data from the UNSW-NB15 and NIMS datasets. The Ensemble created was composed of three classifiers: naive Bayes, decision tree, and neural network. The binary classifier obtained an accuracy of 98.97% and 98.29% for the tests on the datasets in the order they were mentioned.

[69] proposed a cascade intrusion detection system. It is a sequence of neural networks that detect a type of attack in the NSL-KDD and KDD-CUP'99 datasets. Sequentially, the subset resulting from each detection is then processed by the subsequent specialist detector so that, at the end of the cascade processing, it is possible to detect each one of the classes present in the datasets. The measured accuracies were 95.27% for the KDD-CUP'99 dataset and 97.77% for the NSL-KDD.

[70] proposed an intrusion detector that integrates an unsupervised classifier using k-means with a supervised classifier using support vector machines. The proposed Ensemble is given sequentially, where the model initially groups the data with k-means and then classifies them using support vector machines. In tests performed with the NSL-KDD dataset, it was possible to observe an accuracy of 99.45% and a detection rate of 99.04%.

The classifier presented by [71] can balance the size of training samples in order to build an adaptive model

of intrusion detection. The authors implemented an Ensemble through weighted voting between decision tree, random forest, k-NN, logistic regression, support vector machines, and deep neural network classifiers. PCA was used to reduce the dimensionality of the NSL-KDD dataset. The accuracy observed in the tests was 84.23%.

[72] proposed an Ensemble for selecting the best features in the Kyoto dataset. The standard features (and) among the information gain, correlation, and significance methods were used to generate a subset processed by the k-means, support vector machine, k-NN, random forests, and quadratic discriminant analysis classifiers. The best results were obtained with k-NN, where accuracy of 82.28% was observed.

[73] compared several Ensemble methods: adaboost, gradient boost, random forest, and extra tree. With tests performed on a subset generated by selecting the best features and observing the standard deviation between the data from the NSL-KDD dataset, the authors obtained the same results of accuracy, precision, and Recall for the Ensemble methods all of them being 99.9%.

An intrusion detection system focused on DDoS attacks was modeled by [74] The authors created an Ensemble through majority voting among the perceptron, support vector machine, k-NN, and decision tree multilayer classifiers. Using the NSL-KDD dataset in tests, it was possible to obtain an accuracy of 99.77% and a false positive rate of 0.23%.

[75] proposed an Ensemble model for feature selection. According to the authors, the model consists of applying an odd number of feature selection algorithms and, through simple voting, generating a subset containing the model's choices. The experiments used three datasets: KDD-CUP'99, CICIDS 2017, and UNSW-NB15. Four different classifiers were used to observe the improvement in the results before and after the Ensemble application: decision tree, k-NN, multilayer perceptron, and support vector machine. The best result obtained pointed to an accuracy of 99.4%. Another approach along the same lines was taken by [76] where Ensemble selected the best features using ReliefF, information gain ratio, consistency, and correlation. The authors used random forests for classification. In tests with the KDD-CUP'99, NSL-KDD, UNSW-NB15, and CICIDS 2017 datasets, the accuracy results were 99.97%, 99.89%, 95.87%, and 99.88%, respectively.

The intrusion detection system presented by [77] consists of layering packets classified according to TCP, UDP, and ICMP protocols. For each group of packages, binary classifiers arranged in cascade are applied so that, at the end of the process, it is possible to perform a complete multi-class classification. The authors performed tests with the NSL-KDD dataset using the support vector machine, decision tree, Bayes network, reptree, k-NN, BFTree, SimpleCart, and Naive Bayes classifiers. Compared to Bagging, boosting, and Stacking, the results were superior, reaching 95.33% for detecting DoS attacks, 91.14% for probe attacks, 55.21% for R2L, and 1.42 for U2R, in addition to an accuracy of 98.02% for detecting benign packets.

[78] was motivated to create an IoT intrusion detection system with the lowest possible false-positive rate. Using several possible combinations between k-NN and decision tree classifiers and the Ensemble bagging, boosting, random forest, and voting methods, the authors presented several configuration scenarios and their respective results in tests with the NSL-KDD dataset. The best accuracy values for binary classification were 85.81% (decision tree + Bagging) and for multi-class, 83.83% (voting between k-NN, random forest, Bagging, and decision tree boosting).

[79] developed a three-stage NIDS with feature selection, weighted extreme learning machines as classifiers, and softmax aggregation. Feature selection is made by extra trees classifier. The model performs multi-class classification. When tested on UNSW and KDDCup99 datasets, it obtained 98.24% and 99.76% accuracy.

Tackling both intrusion detection and feature selection problems, [80] propose a method using gradually feature removal, ant colony algorithm, and ensemble DT that presents high accuracy (99.92%) with only 16 features of the KDDCup99 dataset.

[81] propose a method for intrusion detection based on Dynamic Deep Forest. The system is composed of two distinct parts. The first, responsible for feature selection, is built with random forest and Extra Trees, and the other, for classification, uses ZGBoost and LightGBM as base classifiers and linear regression for the final output. For training, KDD 99 dataset was chosen. Compared to xgboost and DBN independently, the proposed model performs better in all metrics, presenting 0.917 precious, 0.862 recall, 0.831 F1 scores, and 0.028 false alarm.

In experiments with the NSL-KDD and UNSW-NB15 datasets, [82] and [83] presented intrusion detection models using Ensembles either in classification or classification in feature selection. The first grouped, using Stacking in the detection process, the support vector machine, autoencoder, and random forest classifiers. The second used an Ensemble of majority voting for rotation forest and bagging in the classification process and applied a feature selector that makes an Ensemble of particle swarm optimization, ant colony, and ge-

netic algorithm. The accuracy results of the first authors indicate 91.7% with NSL-KDD and 91.8% with UNSW-NB15, while the other authors obtained 85.79% and 91.27% in the same sequence.

3.6. Works published in 2020

Mixing multi-objective genetic algorithms and neural networks, [84] presents a new approach for network intrusion detection with a majority voting ensemble. The proposed model performs well in detecting both majority and minority classes and increases the detection rate on both the datasets used for development, reaching 97% accuracy on the ISCX-2012 dataset and 88% on NSL-KDD.

[85] propose a stacking ensemble of Decision trees and Recursive Feature Elimination for selecting features and classifying intrusion data. Tests are made on the KDDCup99 and NSL-KDD datasets. The method presents 0.9921 average accuracies on the KDDCup99 dataset and 0.9923 average accuracies on the NSL-KDD dataset.

[86] present in their article a stacking ensemble model using logistic regression, K-nearest neighbor, Random forest, and SVM. The UNSW-NB15 dataset is used for testing binary and multi-class classification, along with the UGR'16 dataset. The best features of each dataset were identified with a combination of information gain and hashing. 97.19 overall accuracy was observed with the UGR'16 dataset and 94.00 with the UNSW-NB15.

[87] felt a lack of recent and complete datasets for training effective IDS systems and presented the creation of a realistic dataset (UMaT-OD-20), together with a Multi-layer Stack Ensemble model for intrusion detection. The dataset was created with the help of the ONDaSCA framework and used in the training of the proposed model. Five different machine learning algorithms were used in the creation of the IDS: KNN, decision tree, logistic regression, random forest, and Naive Bayes. The predictions made by these algorithms were learned and fed into another KNN and RF via ten-fold cross-validation, and the final results showed that the model had a better performance than other work referenced in the authors' research, with 97.93% accuracy and 0.22% false alarm rate.

In a differentiated approach to intrusion detection systems, [88] propose an IDS that utilizes natural language processing and supervised ensemble machine learning (NLPIDS) to analyze natural language HTTP requests and detect anomalous traffic in a network without having to learn existing attack methods. The researchers utilized the HTTP dataset CSIC 2010 to train the ML framework, which applies five different classifiers in the first stage (logistic regression, support vector machines, naive Bayes with gaussian function, decision tree, and neural networks) and then feed the obtained results into an ensemble classifier that applies majority voting, LR, NB, NN, DT, and SVM. The performance of each algorithm is then analyzed, and the best one is aggregated to the NLDS. The results of this experiment show a 99.96 detection rate, 0.07% false alarm rate, and 99.96 f1 score.

Concerned with detecting network intrusion but also with the time it takes between identifying them and taking a measure to avoid any damage, [89] propose an intrusion detection system using machine learning and analyze its performance not only in regard to accuracy but also training time. The used dataset is the UNSW-NB 15, and it is processed by an ensemble algorithm composed of 10 classifier methods, which are evaluated according to their performances in metrics like accuracy and training time. This ensemble selects the method with faster training time and higher accuracy. Through the experiments conducted, Extreme Gradient Boosting (XGB) showed the best overall results, with 95.54% accuracy and 190 seconds of training.

[90] propose an ensemble learning approach as a solution to the detection accuracy, detection time, false alarms, and unknown attacks problems faced by IDS. One of their approaches to reduce the detection time and improve the detection process was to work with the CIC-IDS2018 dataset, selecting only the relevant features according to statistical measurements (chi-squared and Spearman's rank correlation). The choice of classification algorithms to compose the ensemble learning model was made by evaluating their individual performances on the chosen dataset, and the final model was built using gradient boosting, decision trees, and logistic regression, obtaining the following results over the selected data: 98.8% accuracy, 98.8% precision, 97.9% f1 score, 97.1% recall and 00:10:54 execution time.

[91] focus on the problem of detecting multiple attacks at the same time or detecting attacks that are a combination of other existing ones. The NSL-KDD dataset is processed for feature reduction, using gain ratio, information gain, and correlation coefficient method algorithms in sequence, reducing to 25 features. For the multi-attack signatures, the authors captured some scripted attack packets from the network. The dataset is used for training three classifiers for a bagging ensemble model: naive Bayes, decision tree, and random forest, and this ensemble model outperforms the individual methods (99% precision, 0.09 false alarm rate, and 98.76% accuracy.

[92] presented an authorial Ensemble model for two probability-based classifiers. Using MLP and k-NN, the authors performed experiments on the NSL-KDD and CIC-IDS2017 datasets. In multi-class detection for the datasets mentioned above, the results showed, respectively, the accuracy of 97.05% and 99.68%, a precision of 98% and 100%, a recall of 97% and 100%, and an F1-Score of 97% and 100%.

Also dealing with the problem of unbalanced data, [93] turn their attention to software-defined networks and develop an IDS based on online integrated learning. The ensemble method applied is bagging, and the results are better than those of adaboost and C4.5 algorithms, with 72.26% accuracy. But the method also shows great performance fluctuation, with a variation of 20.63%.

Using binary classification, [94] propose an ensemble-inspired IDS. To increase classification accuracy, the best 15 features of the NSL-KDD dataset were selected through an ensemble of feature selection techniques: information gain, gain ratio, and reliefF. For the classification step, the J48, decision table, and random forest algorithms were compared and obtained 99.39%, 98.83%, and 99.58% accuracy, respectively.

Using RF, DBSCAN, and RBM as base classifiers, [95] propose an ensemble model to improve the classification accuracy for machine learning IDS. Along with the classifiers, the researchers aimed to apply both DBCC and IBCC methods to maximize the system's accuracy. The proposed model works with a stream of data that is divided into two portions that are fed into two identical parallel ensemble methods (consisting of the previously mentioned classifiers). The outputs are combined by BCC-based algorithms, namely the DBCC method. For training, 38 features of the KDDTrain+ dataset were used. Results showed approximately 100% detection and a 1.0 accuracy rate.

Using the H2O python library, [96] presents comparisons between ensemble learning algorithms to evaluate the effect that feature selection has on their performance. Using a genetic algorithm, the author extracted the main 43 features from the NSL-KDD dataset after performing one-hot encoding and standardization. From the three ensemble methods implemented, DRF presented weaker performance with feature selection, while GBM and xgboost showed improvements. For comparison, a DNN was also implemented, and though it had a worse performance than all ensemble-based algorithms, it also showed improvement with FS.

[97] tested four different machine learning methods against their proposed bagging ensemble learning model, and theirs showed better performance, with 84.93% accuracy and 2.45% false alarm rate. The system developed by the authors had tree-based algorithms as base classifiers for a bagging ensemble model and was trained on the NSL-KDD dataset. From the dataset, the top 30 features were selected according to the information gain algorithm.

Also working with bagging ensemble learning, [98] propose an approach using improved extreme trees as base classifiers and combining it with quadratic discrimination analysis through maximization for the final prediction results. The training was conducted on KDD-CUP99 and UNSW-NB15 datasets, whose best features were selected by Extra-trees. The model outperforms other algorithms on both datasets, reaching 92.88% accuracy and 0.9538 F1 scores on the KDD and 92.45% accuracy and 0.9462 F1 scores on UNSW-NB15, which shows it also has good adaptability.

[99] proposed system uses Gradient Boosting decision tree-Paralleled quadratic ensemble and Gated Recurrent Unit as a way to work with both spatial and temporal data for intrusion detection. CIC-IDS2017 and CAS2018's (ICN traffic data) best features are selected with random forest and PCA algorithms. The model presents great classification capabilities, reaching 99.9% accuracy for all analyzed classes of the CIC-IDS2017 dataset.

[100] apply a T-ensemble scheme to build a Bayesian CNN intrusion detection system in a way that predictions will only be made when the model reaches a determined threshold of certainty with the output. The system's evaluation is made on the NSL-KDD and UNSW-NB15 datasets, with and without dropping predictions. Both binary and multi-class detection are performed. For binary classification, average accuracies of 99.33% and 99.68% are reached for NSL-KDD and UNSW-NB15 datasets, respectively.

Proposing to improve intrusion detection, [101] apply correlation-based feature selection (CFS) together with ensemble classifiers (i.e., Bagging and adaboost). J48, RF, and Reptree were tested as base classifiers. The system was tested on the KDD99 and NSL-KDD datasets for binary and multi-class classification, resulting in 0 false alarm rate and 99.9% detection rate for the first and 0.5 FAR and 98.6% DR for the second set. Some attacks have 0 classification accuracy due to small sample sizes.

Aiming to identify both known and zero-day attacks, [102] propose a Hybrid IDS combining signature-based and anomaly-based detection. The SIDS is based on C5.0, and the AIDS uses One-class SVM, and both algorithms are combined through the stacking ensemble method for better performance. The proposed system

includes online and offline phases for real-time intrusion detection. NSL-KDD and ADFA datasets are used for training and obtained 83.24% and 97.40% accuracy, respectively.

[103] developed both a new dataset, GTCS, and an ensemble model addressing the issue of accuracy and FAR in IDS. The proposed GTCS is completely labeled and has 84 features. The model has an FS phase, where InfoGainAttributeEval is used. For classification, J48, IBK, and MLP classifiers' outputs are combined with majority voting. With 98.62% accuracy for detecting normal traffic, 98.87% for the botnet, 96.7% for brute force, 98.99% for DDoS, and 88.69% for infiltration, the model outperforms the algorithms used for comparison in the research.

Noticing a lack of comparisons between homogeneous and heterogeneous online ensemble methods, [104] present their research on the topic and propose three new heterogeneous models, comparing their performance for online training and how they managed concept drift. The proposed models are the combinations, in pairs, of three base classifiers: SVM, Hoeffding Adaptive Tree (HAT), and Adaptive random forest (ARF). For the final output, weighted majority voting was utilized. The dataset used is the KDD CUP99. HAT+ARF had the best performance (approximately 98%) but worst runtime (55.59 s), followed by SVM+ARF (97% accuracy and 44.39 s runtime) and then SVM+HAT, with worse accuracy but faster runtime (94% and 11.16 s).

[105] designed a two-layer integration model based on the stacking framework. The researchers tested different classifiers and ensemble methods to determine which had better performance and if the integration model improved the quality of the results. They also test how many base classifiers are ideal for the first layer of the proposed model. The best features of the KDDCup99 dataset are selected with principal component analysis and used for training. The authors observed improvement with the use of their framework for all the tested algorithms.

[106] propose the Least Square support vectors machine intrusion detection system (LSSVM-IDS). The main focus of the model is to identify the most important features through Principal Component Analysis feature selection. The data is then applied to the ensemble phase, which is composed of random forest, linear SVM, Naive Bayes, and logistic regression combined through a stacking classifier. USB-NB15 dataset is utilized, and the model reaches 95% accuracy.

Applying genetic programming, [107] propose a novel ensemble-based framework for online intrusion detection which takes into consideration the detection of concept drift (E2SC4ID). The framework is constantly updated as the base classifier, and ensemble models are incrementally discovered and replaced when concept drift is detected. An artificial dataset, named Hyperplane by the authors, is utilized along with the ISCX IDS dataset. The best results are obtained with Hyperplane, with a 0.928 AUC score. For the ISCX IDS, the highest score is 0.892.

[108] devised a stacking ensemble model for intrusion detection in wireless networks. The model has a gradient boosting machine and random forest as base classifiers, and the NSL-KDD dataset was used in its development. Feature selection was also made with RF. The overall accuracy of the proposed NIDS was 91.06%.

[109] use an extra tree classifier on an ensemble model for network intrusion detection. The extra tree is an ensemble of decision tree classifiers. KDDCup99 and NSL-KDD datasets are tested, and the model achieves 99.97% and 99.32% accuracy, respectively.

[110] conducted a comparative experiment with RF ensemble algorithm and autoencoder to define which has better performance for intrusion detection. The experiments are performed on the NSL-KDD dataset. The AE presents much better accuracy than the RF, with 90.30% and 77.38% scores, respectively.

[111] developed a support vector machine-based IDS which uses ensemble learning for performance optimization. The proposed model processes data with a data bag approach instead of a single flow representation. adaboost and bagging are combined with SVM for data processing, and the model reaches 90.58% recall, 88.95% precision, 11.24% FPR, and 0.8976 F-score on the Kioto 2006+ dataset.

Applying Deep Learning for intrusion detection, [112] implemented a Bidirectional Long Short-Term Memory neural network on the UNSW-NB15 dataset (although they also performed experiments on another dataset that goes beyond the scope of this dissertation). An Ensemble authored method called Deep Blockchain Framework was also implemented. The best accuracy obtained in the tests was 99.41%, with 60 nodes in the middle layer.

[113] performed a series of tests with various Ensemble algorithms on the NSL-KDD dataset. The authors implemented voting (between k-NN, decision tree, MLP, and logistic regression), Bagging, random forest, and adaboost (all using decision trees as classifiers). The mean accuracy, precision, and F1-Score results were 81.2%, 72.5%, and 82% for voting; 85.4%, 77.1%, and 84.7% for Bagging, and the best values of

accuracy, precision, and F1-Score for random forest and adaboost methods were 86.3%, 78.6% and 85.7% for the first and 86.5%, 78.5% and 85.8% for the latter.

[114] presented a framework for choosing ensemble methods based on accuracy to reduce the false-positive and false alarm rates. A complex combination of decision tree, JRIP Rule Learner, Naive Bayes Tree, Naive Bayes, One-R, Logistic Model Trees, logistic regression, Decision Stumps, and k-NN classifiers was presented in order to dynamically model Ensembles, that is, choosing the best combination among the above classifiers. The tests were performed on the CIC-IDS2017 dataset and the results presented by the authors document a decrease in computational cost in the data processing. The authors did not report results for intrusion detection.

[115] proposed a combination of XGboost with Particle Swarm Optimization. The authors implemented a multi-class intrusion detection system tested on the NSL-KDD dataset. The average precision (AP - averaged precision) obtained in the tests was 90%, 79%, 94%, 15%, and 49% to detect benign packages, Probe, DoS, U2R, and R2L, respectively. The work compares the results with other different Ensemble methods.

[116] performed experiments on the UNSW-NB15 dataset to obtain the best meta-classifier (second layer classifier) hypothesis for an intrusion detection model using Stacking. Performing the classifications of the first layer with k-NN, Naive Bayes, and decision tree, the authors tested some possibilities of meta-classifiers and compared the results. Using three different subsets, the best results were using decision trees in the second layer, pointing to an average accuracy of 98.56%.

[117] presented an intrusion detection system that specializes in detecting web attacks. An Ensemble through Stacking was implemented; however, instead of using conventional classifiers, the authors used other Ensemble methods such as random forest, Gradient Boost Machine, and XGboost to create the Stackings. Four different datasets were tested: CIC-IDS2017, HTTP-CSIC-2010, NSL-KDD, and UNSW- NB15. The average accuracy result reported by the authors was 96.07% for the proposed model, with the best result obtained in CIC-IDS2017 with 99.98%.

[118] compare two different ensemble methods to define which performs best with the C4.5 decision tree classifier for network intrusion detection. adaboost improved the model's performance more than the bagging ensemble, but both presented better results than C4.5 by itself. The used dataset is UNSW-NB15. The highest accuracy reached with bagging is 97.70%, and with boosting, it is 97.75%.

3.7. Works published in 2021

With the main focus on IoT networks, [119] use fog computing to propose a network IDS based on a distributed architecture. The approach uses xgboost, K-NN, and gaussian Naive Bayes as individual learners and RF for final classification. The system is tested on UNSW-NB15 and the IoT-based DS2OS dataset and performs better on the second one, reaching a detection rate of 99.99% for most of the attack types. The best features of each dataset are selected with mutual information-based feature selection.

[120] apply the Crow-Search algorithm for feature selection and then run the dataset through a Linear Regression, random forest, and xgboost ensemble for multi-class classification. To ensure the functioning of the model, the UNSW-NB15 dataset is utilized. The proposed algorithm presents results between 62.00 and 100.00 for recall, 85.00 and 100.00 for precision, and 73.00 and 100.00 for accuracy, depending on the class.

[121] present an IDS based on boosting ensemble model and using xgboost. Training and testing were done on the KDDCup99 dataset. The multiclass classification method detects five classes: DoS, Normal, Probe, R2L and U2R, presenting 99.98%, 99.95%, 99.01%, 94.60% and 53.84% detection rates, respectively.

[122] developed DBN-LSSVM, using a deep belief network for feature reduction and a least-square vector machine for intrusion detection. The KDDCup99 dataset is used for the development of the model. It reaches 99.12 accuracies, 98.54 detection rate, and 0.63 false alarm.

Combining decision tree, random forest, and gradient boosting decision tree, [123] developed an ensemble model for network intrusion detection. For the ensemble, the authors apply the stacking method. The dataset used was UNSW-NB15. Results showed that the stacking ensemble reached 0.9874 AUC score, 0.9874 F1-score, and 0.0015 FAR.

In their paper, [124] propose an ensemble machine learning assisted network intrusion detection (NIDS) model for anomalies detection in network flow data. The algorithm they present is enhanced with multiple stages of data preparation, such as data subsampling, normalization, and feature reduction, and for the intrusion detection phase, heterogeneous ensemble learning is applied. The heterogeneous ensemble classifier comprises the algorithms Naive Bayes, J48, k-NN, SVM, Boosting, Bagging, adaboost, and random forest, and it performs binary classification as well as multiclass classification. The utilized datasets were the NSL-KDD, KDD99, and UNSW-NB15, and the results pre-

sented a near 100% true positive rate (98.8%), with a 0.7% false-positive rate.

Analyzing the behavior of a network's users can be very useful for detecting malicious activity, [125] propose an algorithm TABIS to detect the risk level associated with each user and to perform intrusion detection. For selecting the best features of the KDD-cup dataset used for training the model, an Ensemble Service-Centric Feature Selection algorithm was used, and for the prediction step, a Sigmoid Recurrent neural network was applied. The model was tested for different numbers o services on the network, and the best results obtained were 93.8% accuracy, a false detection rate of 0.5%, and a time complexity of 14 seconds.

[126] utilize the CIDDS-002 dataset to evaluate the performance of an IDS system with two base classifiers (decision tree and gaussian Naive Bayes) and four ensemble classifiers (random forest, adaboost, Bagging, and Stacking). The performance evaluation was validated through the application of 10-fold cross-validation, and the results showed that the Bagging ensemble method, when applied in conjunction with the decision tree base classifier, produced the best results, with a mean accuracy of 99.71%, average f1 score of 100%. The worst performing ensemble was the combination of Bagging and GNB (accuracy of 67.57% and f1 score of 61.80%).

[127] work with what they consider the ten most important features of the NSL-KDD data set to compare and evaluate the performance of combinations of 5 distinct ensemble learning models in search for the most effective approach to maximize detection rates on an IDS. The classification algorithms (random forest, support vectors machine, logistic regression, K-nearest Neighbors, Naive Bayes, and multilayer perceptron) were organized in five different ensembles as follows: MLP-RF-NB, MLP-KNN-RF, MLP-KNN-SVM, MLP-SVM-LR, and LR-SVM-KNN. The first combination presented the best results, with 99.66% accuracy, 99.66% precision, 99.51% recall, and 99.59 f1 score.

[128] present their comparative research on the performance of ensemble learning (bagging-based, boosting-based, stacking-based) and traditional machine learning techniques (K nearest neighbor, decision tree, Naive Bayes). Their study is also the first to compare the performance of ensemble learning paired with different feature selection algorithms (i.e., Person's correlation coefficient, extra tree classifier) for anomaly-based intrusion detection. The dataset used was CICDDoS 2019, from which 37 features were selected. The stacking-based ensemble presented the best performance for both reflection-based and exploitation-based attacks.

With the purpose of decreasing the false alarm rate of IDS, [129] propose a NIDS that uses ensemble learning and four different classifiers: random forest, adaboost, xgboost, and Gradient Boosting decision tree, with a voting classifier. The proposed system is composed of 3 different units, one for packet capture (PCU), one for data processing (DPU), and one for classification (CE), and it can be fed with data from network analyzers such as Wireshark, but for the purposes of the paper, the NSL-KDD and UNSW-NB15 datasets were utilized.

[130] work with sustainable ensemble learning and propose a multi-class identification model able to adapt to different attacks. The base classifier's results are assembled using a modified voting mechanism, in which probability and classification confidence is multiplied to define the weight of each output. Knowledge is passed among models to maintain performance and stability. NSL-KDD dataset is used and the five classes are identified with high accuracy (DoS: 0.9718; Probe: 0.9388; Normal: 0.9017; R2L: 0.9923; U2R: 0.9803).

[131] work with the network packets from the Lemay* dataset to evaluate three ensemble techniques on their capabilities to handle imbalanced data for intrusion detection in Cyber-Physical systems: Bagging, with random forest, Extreme Gradient Boosting, and Stacking, with Generalized Linear Model. Two other Stacking models were studied: one with random forest and SGBoost and another with random forest and SGBoost and SVM. All models were analyzed with large (around 22,000 samples) and small (400:1 normal:attack ratio) test files and were compared to traditional decision tree and SVM algorithms. The results show that the ensemble models have better performance, and among them, the stacking methods present the best results.

Also, considering the problem of imbalanced datasets, [132] proposes a semi-supervised ensemble approach for intrusion detection by adapting the existing Semi-Boost algorithm. Besides updates to the sampling method, the author suggests that Poisson's distribution be used to divide samples among the classifiers of the Bagging model and that different penalty factors be applied to each base classifier. The base classifiers are decision tree algorithms, and the dataset used for training the model is the NSL-KDD. The results are compared to those of the adaboost and C4.5 algorithms and present results worse than the first but better than the former, with an accuracy of 63.88% and precision of 55.41%, among other evaluation indicators.

The problem [133] set out to solve is the inability

of IDS to detect intrusion in real-time. For that purpose, they trained Naive Bayes, Artificial neural network, K nearest Neighbor, support vectors machine, and C4.5 algorithms on the CIC-IDS2017 dataset. adaboost, bagging, and Stacking ensemble models were also trained with each of the mentioned methods, and all of them performed better than the individual classifiers. The best combination obtained was that of adaboost with C4.5 as the base classifier. After these results, the best performing ensemble was tuned for even better performance, achieving 99.06% accuracy, 98.9% Cohen's kappa value, and 100% F score.

To simplify the intrusion detection process, [134] suggests a system that utilizes the same classifiers for feature selection and for the main classification process and concludes that it resulted in high accuracy rates. The training and testing of the proposed system were done on the NSL-KDD dataset and on additional data captured by the author to simulate real-time attacks. The algorithms used are random forest, K-nearest neighbor, and SVM, applied with the Wrapper Methods for Correlation-based Feature Selection on the FS phase and with a voting classifier for the classification phase. The RF algorithm had a better performance on its own than the other ones, including the ensemble model, presenting a 99.825% accuracy, 99.906% precision, 99.745% recall, and 99.792% F score.

Trying to propose a unique IDS, [135] built an ensemble of classifiers and an F1-score-based ranking system to define which one performed better in detecting each kind of network attack. RF and Principal Components Analysis were used to select the 24 best features of the CIC-IDS2018 dataset. To reach the best algorithms combination, seven classifiers were tested on the accuracy, prediction time, attack detection rate, and time efficiency, and the best two were used in the proposed model: LightGBM and HBGB reached an accuracy of 97.5%, and recall rate of 96.7%.

Using the NSL-KDD dataset, [136] propose an IDS and compare the results obtained by using boosting and stacking ensemble techniques. Only 10 of the dataset's features are used, the best according to random forest classifier and Recursive Feature Elimination feature selection. For the stacking ensemble method, adaboost and Extreme Gradient are the base classifiers, and logistic regression is the meta classifier, while for the boosting model, the weak learner is decision tree, and the meta classifier is adaboost. The boosting ensemble performed better than the stacking, with 80.10% accuracy, 0.824 precision, 0.823 recall, and 0.819 F1 score.

Working with multiple FS algorithms, [137] propose HFS-KODE, an IDS system that relies on K-means, One-class SVM, DBSCAN, and Expectation-Maximization classifiers on an ensemble. The system presents 99.99%, 99.73%, and 99.997% accuracy for each of the used datasets, CIC-IDS2017, NSL-KDD, and UNSW-NB15. The authors present a lengthy and complete analysis of HFS-KODE and the comparisons between other existing IDS, showing that their Feature selection method can also enhance other algorithms.

[138] try to solve the problem of classification errors for minority classes by proposing the SMOTE technique for data balancing, paired with C4.5 feature selection, bagging ensemble learning with a random forest as base classifiers and a majority voting meta-classifier. Both binary and multi-class classification was tested using UNSW-NB15 and CSE-CIC-IDS2018 datasets. For binary classification, accuracy was 99.65% and 99.98% for each dataset, respectively, and for multi-class classification, it was 87.35% and 96.53%.

[139] observe that many times, in order to improve the adaptability of an ensemble intrusion detection model, the added complexity also increases training cost. Addressing the issue, they propose a snapshot ensemble based on group convolution (GCSE). Each snapshot is trained individually, and the final output is obtained through averaging. On both the NSL-KDD and UNSW-NB15 datasets, GCNSE outperforms the other models reproduced in the research, reaching 84.95% accuracy in the first and 79.59% accuracy in the second dataset.

[140] propose an IDS for a smart healthcare environment, using the Bagging ensemble method with a random forest classifier. The authors use the NSL-KDD dataset. The proposed method performs better than the other single and bagging ensembled classifiers, reaching an accuracy of 97.67%, 0.977 F-measure and AUC, and 0.921 kappa.

[141] work with Optimum-path forest as a base classifier for a stacking-based intrusion detection system. For testing and training, the authors use the NSL-KDD dataset and the uneSPY developed by them. Three experiments were done for the purpose of comparing results: one with single classifiers, one with a homogeneous ensemble, and a third with heterogeneous stacking. It was observed that OFP performs worse as a base classifier for the proposed ensemble methods than as a single classifier.

To deal with Concept Drift, [142] developed the Adaptive Extreme Gradient Booster (AXGB), an IDS classifier that uses ensemble learning. For the experiment, the KDD CUP99 dataset was used as streaming data. The proposed model reaches 99.078% accuracy.

[143] present a hybrid IDS using 2D CNN, RNN,

and MLP, ensembled with a One-Versus-All strategy named COREM. A recently published dataset is used in the research called Kitsune Network Attack. COREM performs well, reaching 92.66%, 90.64%, and 90.56% overall accuracy on training, validation, and test data, respectively, but it has great difficulties in detecting minority classes.

[144] turn their attention to collaborative attacks, devising an ensemble learning-based IDS that should be capable of dealing with this sort of network invasion, along with other single attacks. For the development of the model, the researchers used the UNSW-NB15 dataset, incremented with some collaborative attacks signatures created by them, and reduced with information gain feature selection. The system is based on DT, SVM, and XGboost classifiers, with a majority voting ensemble. The proposed approach has 97.96% accuracy, 0.07 false alarm rate, 0.85 recall, and 0.99 precision.

[145] point out the computational cost of implementing ensembles of autoencoders and propose and compare four different methods to decrease the complexity of these types of models, which are then applied to NIDS. The methods differ on how they deactivate the AEs (criteria-based and random) and when that is done (post-training and in-training). IoT network datasets Kitsune and NBaIoT are used for the experiments. The proposed approach speeds the processing of samples 4.2 times.

[146] propose a network-based intrusion detection system using boosted tree (adaptive boosting), bagged tree (bagging), subspace discriminant, and RUSBooted classifiers with a voting method. The CIDS 2017 dataset is used. The model outperformed many recent experiments and reduced convergence and execution time.

[147] utilize deep learning for feature selection, SVM (with RBF kernel) and k-NN for classification, and Dempster-Shafer method to combine outputs in their proposed IDS. Eight different combinations of the classifiers are tested, with different k and RBF values. The system's performance is evaluated on UNSW-NB15, CICISD2017, and NSL-KDD datasets. Sample selection from de datasets was made through the ensemble margin technique. 90.98%, 98.7%, and 99.80% were the detection accuracy of the model for each of the UNSW-NB15, CIC-IDS2017, and NSL-KDD datasets.

[148] propose an ensemble of neural networks, along with a high-performing architecture connecting to explore parallel processing. The researchers also generated their own dataset for the development of the model. Among the tested configurations for the NN, the highest accuracy reached was that of 99.98%, with a 0.84% false alarm rate.

4. Background

This chapter details the classifiers, datasets, and ensemble algorithms most used in related works, and presents an overview of intrusion detection systems.

4.1. Datasets

Many efforts have been put into creating intrusion detection models based on Data Mining. Some datasets have been used for this. This section briefly describes the most common datasets from the research documented in our work.

Table 2: Publications segregated by Base Datasets and Year.

ANO	KDD99	NSL-KDD	ISCX	UNSW	CICIDS
2015	3	4	1	0	0
2016	5	2	2	0	0
2017	5	10	1	5	0
2018	7	9	0	2	1
2019	6	11	1	6	1
2020	6	16	2	8	4
2021	4	12	0	8	7

4.1.1. KDD-Cup'99 and NSL-KDD

The KDD-Cup'99 dataset is a subset of the DARPA dataset (which contains data from simulated operations from the US Air Force's local area network) widely used for training intrusion detectors in computer networks and made available at the Massachusetts Institute of Technology (MIT) Lincoln Laboratory[1]. It was initially introduced as part of a challenge at the KDD-Cup competition held in 1999.

A problem with the KDD-Cup'99 dataset is that it has a lot of redundant and duplicated data, respectively 78% and 75%. To provide a more robust set, [149] presented the NSL-KDD dataset in 2009, as a derivation of KDD-Cup'99t. The most obvious benefits are eliminating redundant records in the training sample and deleting duplicate records in the testing sample.

Table 3 describes the package relationship for each class available in the aforementioned datasets.

Table 3: Packages by class in the KDD-Cup'99 and NSL-KDD datasets

Dataset	Total	Benign	DoS	Probe	U2R	R2L
KDD'99	76896	70217	5728	520	50	381
NSL-KDD	69483	53561	12517	2976	50	379

[1] https://www.ll.mit.edu/

The KDD-Cup'99 dataset contains 41 features that organize 4,898,431 simulated connections. The NSL-KDD contains 39 different types of attacks in the dataset.

4.1.2. CICIDS 2017

The set of CICIDS-2017 datasets [150], made available by the *Canadian Institute for Cybersecurity*[2] through the *University of New Brunswick*[3] is divided into eight files (tab. 4), according to [151] and [152].

Table 4: Subsets of CICIDS-2017 Dataset.

Dataset	Attack type	Benign packets	Attack packets
1	Bruteforce	432074	13835
2	Infiltration	288566	36
3	DDoS	97718	128027
4	Portscan	127537	158930
5	Botnet	189067	1966
6	Web	168186	2180
7	DoS	440031	252672
8	Benign packets	529918	0

Each dataset contains data for a specific attack type accompanied by benign traffic data (only one subset contains only benign traffic). All have 78 *features* with their labels in column 79.

4.1.3. ISCX 2012

The ISCX 2012 dataset [153], provided by the Information Security Center of Excellence at the University of New Brunswick[4] contains packages referring to DDoS attacks, HTTP DoS, Infiltration, and SSH brute force. The data is organized into seven subsets whose title is related to the day of the week when the traffic was captured (tab. 5). There are 20 features where about 2% of the traffic corresponds to malicious packets.

Table 5: Subsets of ISCX 2012 Dataset.

Day of traffic	Attack type	Amount of data
Friday	Benign	16.1 GB
Saturday	Benign	4.22 GB
Sunday	Infiltration + benign	3.95
Monday	HTTP DoS + benign	6.85 GB
Tuesday	DDoS	23.4 GB
Wednesday	Benign	17.6 GB
Thursday	SSH brute force + benign	12.3 GB

[2]https://www.unb.ca/cic/

[3]https://www.unb.ca/

[4]https://www.unb.ca/cic/about/hub.html

4.1.4. UNSW-NB15

The UNSW-NB15 [154] dataset was developed at the School of Engineering and Information Technology at the University of New South Wales at the Australian Defense Force Academy[5]. 100 GB of benign and malicious traffic was captured to deliver ten classes of data: Benign, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms.

The dataset was developed based on a critique of the scientific popularity of the KDD-Cup 1999 and NSL-KDD datasets. The authors [154] highlight that the datasets mentioned above contemplate old attacks in a way that the IDS generated from this data would not be robust enough to detect modern attacks.

4.2. Base Classifiers

Machine Learning is a line of artificial intelligence belonging to computer science that seeks, through mathematical and statistical methods and in conjunction with related areas, to analyze a dataset and learn patterns through classification, clustering, and regression. Many algorithms can be employed to solve such tasks. Classification, regression, and clustering are problems that, according to [155], the methods of machine learning, in general, solve very well. The authors of [156] define that "classification" consists of training an algorithm of learning in a sample of tests (a portion of the dataset with data in a variable) and testing it on the remaining sample of the data to verify the performance of the algorithm.

In classification problems, machine learning algorithms learn patterns in the data present in the training sample and must be able to identify the classes to which the test sample data belongs. Regression is about finding patterns in data distribution in a dataset and predict continuous values. Clustering is a way of grouping similar data according to specific characteristics [155]. Table 6 organizes the list of classifiers that have been most used in recent years by the criterion used in the systematic literature review. Next, the highlighted classifiers are briefly described.

[5]https://www.unsw.adfa.edu.au/seit

Table 6: Publications segregated by Base Classifier and Year.

Year	SVM	k-NN	MLP	Decision tree
2015	2	1	2	8
2016	1	1	1	6
2017	4	2	2	8
2018	4	0	6	7
2019	6	5	6	14
2020	7	6	6	25
2021	8	5	4	18

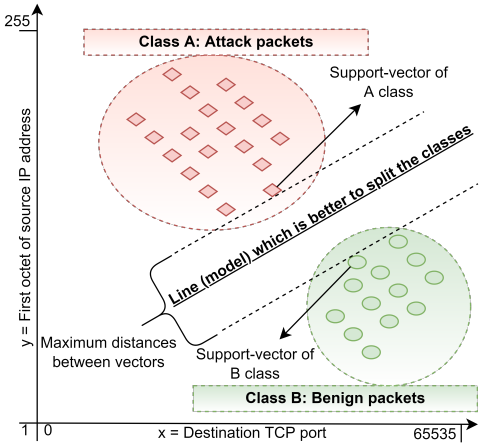
4.2.1. Support vectors machine

An SVM-based classifier seeks, given in the parameter space two linearly separable classes (*a* and *b*), to draw the best hyperplane so that the element of class *a* closest to the first element of class *b* is considered the limits. The limits are called "support vectors" because they represent the closest elements *a* and *b*. The hyperplane, therefore, will be constructed at exactly half of these limits.

The SVM strategy is to find the most significant margin of separation between classes, extracting a classification function for a sample of observations [157]. The idea of keeping as much margin as possible for tracing the hyperplane is that the test samples will be distributed differently in the parameter space, so the more significant the margin, the lower the error rate.

Figure 6 illustrates a situation of SVM application in intrusion detection in computer networks.

Figure 6: SVM hypothetical example



For the analysis of Figure 6, it must be taken into account that the author created a hypothetical situation of two-dimensional distribution to illustrate the purpose of the application of SVM and that, not necessarily, only the parameters "Source IP first octet" (*y* axis) and "Destination port" (*x* axis) help classify network traffic effectively.

Furthermore, the leftmost cluster represents "class A" and all its objects where the element closest to "class B" (located more to the right) is chosen as the support vector. The same applies, in reverse logic, to another class. Therefore, when finding the support vectors of the analyzed classes, the SVM establishes the margins representing the most significant possible distance between the classes. Finally, the hyperplane that best divides the classes is drawn, a midline between the margins.

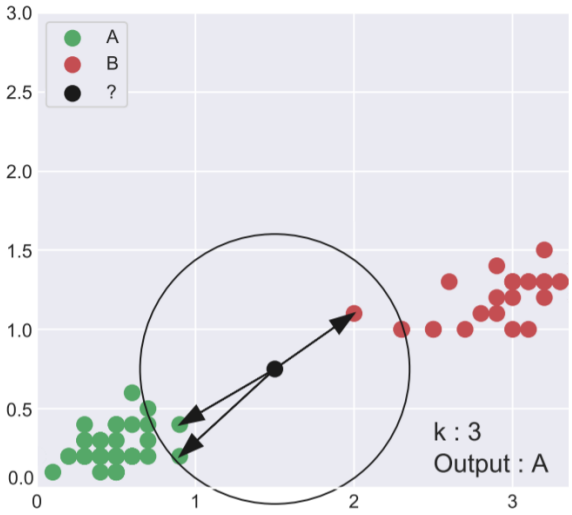
Moreover, the ability of SVM to deal with situations of class separation in *n*-dimensional spaces is important. In the example of Figure 6 (above to understand the basic functioning of the method), a simple straight line deals with dividing the classes. The authors of [158] emphasize that in the existence of more dimensions, the method can solve binary classification problems with equal efficiency. If in a two-dimensional situation, the division of classes is carried out by drawing a straight line, in a three-dimensional situation, for example, a plane would be drawn to complete the classification. In short, SVM can handle several dimensions, highlighting the computational cost required as more dimensions (and features) are analyzed.

4.2.2. K-Nearest Neighbors

K-Nearest Neighbors (*k*NN) is a simple machine learning algorithm used for classification and regression. Given a set of labeled data points, the algorithm can predict the label or value of a new data point by finding the k data points in the training set that are closest to it (i.e., its "neighbors") and returning the majority label or average value of those neighbors. The value of *k* is a hyperparameter that can be tuned for a particular problem. The algorithm is easy to implement and does not require any training phase, but it can be computationally expensive for large datasets and may not generalize well to unseen data. Because *k*NN algorithm needs to compute the distance to all training samples for all of the testing samples, its computational cost can be explained quite simply [159]. By establishing a value for *k*, the algorithm, in the parameter space, calculates the distance from the test sample to all training samples and orders them by the distance. Finally, for the closest *k* neighbors, the most frequent class is observed, which will finally be assigned to the analyzed object.

According to [160], although it is simple, *k*-NN can obtain excellent results in data classification. The best value of the *k* hyperparameter is usually obtained empirically. Figure 7 illustrates a hypothetical parameter space for a better understanding of how *k*-NN works.

Figure 7: Hypothetical parameter space for k-NN [160]



Analyzing Figure 7, it is possible to observe that, when the value of $k == 3$, the two nearest neighbors (most of 3) are in the cluster on the left. In this way, the analyzed central object should be assigned with the class on the left as a label.

*k*NN with the Euclidean distance metric is one of the classifiers used to create the *Stackings* presented in this work. For example, observing the elements A, B, the values referring to the axes x and y are obtained (in a two-dimensional analysis), and A has the values x_A and y_A and B has the values x_B and y_B. The distance D will therefore be calculated as $D = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$. When considering more dimensions n in the parameter space, the equation must be adjusted by adding the sum of the values $(x_n + y_n)^2$ to the square root. The tested k values were stipulated in a loop ranging from 3 to 9, where the best k in the range $\{3, 5, 7, 9\}$ was chosen for each test.

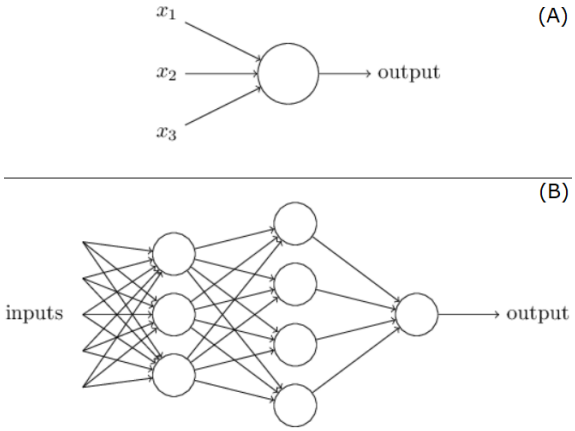
4.2.3. Multi Layer Perceptron

An artificial neural network is a structure imitating in a simple way the human brain in the sense of communication between neurons. The human brain responds to stimuli at the inputs of its neurons, and the cellular organization causes others to be activated (or not) in response to the processing of that stimulus. The structure of the other layers and the propagation of stimuli depend on the type of neural network created. The last layer of neurons corresponds to the classes of the analyzed data so that it is possible, given the input parameters and the propagation of the stimuli, to trigger the neuron corresponding to the estimated class [161]. Likewise, an artificial neural network typically has the number of inputs corresponding to the number of features of a dataset to be processed.

multilayer perceptron (MLP) is a type of artificial neural network used for supervised learning tasks such as classification and regression. It consists of an input layer, one or more hidden layers, and an output layer. Each layer is composed of nodes (also known as neurons) that apply a nonlinear activation function to the weighted sum of their inputs, and the outputs of one layer become the inputs to the next. The weights of the connections between the neurons are learned from the training data using a optimization algorithm such as stochastic gradient descent. During the training phase, the model iteratively adjusts the weights to minimize a loss function that measures the difference between the predicted and actual outputs. The resulting model can be used to make predictions for new, unseen data. MLP has been widely used in many applications and has been shown to be effective in solving complex problems.

The original Perceptron model was developed by [162], where the author presented an architecture that can be analyzed, according to [163], based on Figure 8 (A).

Figure 8: Perceptron models [163]. Simple architecture (A), and multilayer architecture (B).



For each entry x_1, x_2, x_3, a weight w_1, w_2, w_3 is introduced, valued according to the importance of the feature represented by each entry in the hypothetical dataset analyzed. From this point, a threshold t is defined that will be the basis for the neuron's output, being 0 if the weighted sum of the weights for each input is less than the value stipulated in the *threshold* ($0\ if\ \sum_j w_j x_j \leq t$) or 1 if the result is less ($1\ if\ \sum_j w_j x_j > t$).

In Figure 8 (B) it is possible to observe a model of *neural network* (or Perceptron with an intermediate layer - multilayer perceptron). Such a model can classify problems of more complex order since this layer will deal with the outputs of the neurons of the first processing layer, making the model able to separate data in a non-linear way. Each decision made by neurons in the second layer will weigh the decisions made in the previous layer.

Furthermore, it is a "feed-forward" network; that is, data propagation always takes place in the direction of the inputs to the output. The training process for these types of networks consists of learning the best weights (variables that specify the importance of each feature) and bias (a variable that will balance the threshold). [161] highlights that, although it is relatively simple, the multilayer perceptron has been successfully applied as a type of *neural network* in the solution of several complex problems.

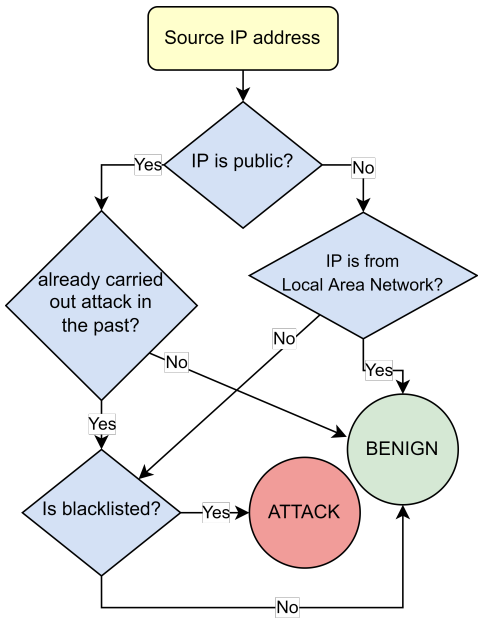
4.2.4. Decision trees

decision trees are a popular machine learning algorithm used for both classification and regression tasks. The algorithm builds a tree-like model of decisions and their possible consequences. Each node in the tree represents a test on an input feature, each branch represents the outcome of the test, and each leaf node represents a prediction. The tree is constructed by recursively splitting the data into subsets based on the feature that provides the most information gain until a stopping criterion is reached. The final prediction is made by traversing the tree from the root to a leaf node, following the branches corresponding to the feature values of a new sample. Decision trees are easy to interpret, fast to train, and can handle both categorical and numerical data. However, they are prone to overfitting and can result in complex trees that are difficult to interpret. To overcome these limitations, various decision tree ensembles such as random forests and Gradient Boosted Trees have been developed. According to [164], the methods that use decision trees belong to the family "Top Down Induction of decision trees" (TDIDT) which is a data structure that defines "nodes" and "decision nodes". The first node is responsible for representing a class, while the second node is responsible for representing a test on some attribute. For each example path to be a leaf, the attributes are conditioned by the "decision nodes" to make one up to a "node" class. This sequence of logical "*ifs*" will determine the path to be followed in the tree for each example.

Work [165] defines DT as a flowchart-like structure where each internal node represents a test for an attribute, each branch represents the result of a test, and each leaf represents a class. The authors emphasize that less time can be constructed, and fewer processing methods stand out as less DT.

Figure 9 illustrates a simple intrusion detection process starting from the analysis of attributes for a source IP address of a hypothetical packet.

Figure 9: Decision tree example for Intrusion Detection on computer networks.



Tests will be performed for each IP address sample to be processed by DT. The tests validate the class denoted for the analyzed object (IP). The "BENIGN" class will only be assigned in case the IP address is not public and belongs to the internal network or in cases the IP has not carried out attacks in the past or, even if it has already been carried out, is no longer included in a blacklist. In any other situations foreseen in the tree, the analyzed IP address will be assigned the "ATTACK" class.

4.3. Ensemble Methods

According to [166], Ensemble Machine Learning methods consist of the joint application of different or the same classification or regression algorithms to solve the same problem. While traditional techniques build learning based on a specific technique, ensemble techniques seek to aggregate, either in a parallel or sequential way, multiple learning techniques to achieve improvements in terms of variance, bias, or predictions [167]. The construction of ensemble methods can be

performed with classifiers of different types, called "heterogenous ensembles" or multiple classifiers that use the same learning, which is called "homogeneous ensembles" [166].

Different machine learning algorithms are applied to generate individual models with a hypothetical training sample. Generically, the learning or the way of classifying the methods is grouped in the item "Combination" depending on the ensemble technique used, which illustrates the possibility of sharing the power of classification between the methods of the previous time. As a consequence, we have a "Classifier" at the end (to the right) of the flow that is nothing more than a classification process that will add the intelligence of the four methods present in the ensemble.

Still, according to the author, there are three ways to organize an ensemble method concerning how the individual algorithms relate: "combining classifiers", "ensembles of weak" and "mixture of experts". The first form is the most used in the literature, especially for solving pattern recognition problems. It is based on the combination of classifiers with excellent accuracy and seeks to combine them to improve the final (joint) accuracy in the classification. The second form is the most used by the machine learning community and consists of the joint application of light classification algorithms (with reasonable accuracy), making the final classifier have excellent accuracy, extracting the best from each light classifier. Finally, the last method is more explored in neural network-based approaches and assumes that a better solution can be obtained by combining rules and mixing parameters.

Table 7: Ensemble-based publications segregated by base datasets and year.

ANO	Bagging	Boosting	Stacking	Voting
2015	5	5	1	1
2016	2	1	0	5
2017	4	6	1	7
2018	6	5	1	7
2019	2	4	2	7
2020	7	5	9	3
2021	4	8	7	6

4.3.1. Bagging

The authors of [168] emphasize that this is one of the first proposed ensemble algorithms. It uses the bootstrap[6] to obtain subsamples of the training dataset, which means sampling a large amount of data collected randomly and without repetition. Then each classifier is trained with a specific sample obtained using bootstrap. The algorithm's ranking process consists of a majority of votes. Given a test sample, each algorithm defines the class according to its previous learning. The ensemble will define the common classification method by choosing the most chosen class among the ensemble's algorithms. The above method can be understood as follows, considering in dataset D the training sample D_a and the test sample D_b:

Train

1. Choose the number of samples n and the base classifier C;
2. Create n training samples with bootstrap D_{a1}, D_{a2}, ..., D_{an};
3. Fit base classifier C for each sample D_{ai} to create n classifiers C_1, C_2, ..., C_n.

Test

1. For each sample x in D_b, test x per all classifiers C_1, C_2, ..., C_n;
2. Choose the final class based in voting - consider the label most chosen by the classifiers C_1, C_2, ..., C_n.
3. Repeat the process for each sample x in D_b.

The use of decision trees generates a method called a random forest, a classification algorithm widely used in the literature and a classic example of the application of the bagging method. According to [169], the random forest algorithm can be considered an example of bagging with the characteristic that the decision trees are similar to each other, which leads to decisions with a high correlation index.

4.3.2. Boosting

In 1990, Schapire showed that clustering several weak classifiers could generate a general strong classification method, except when the dataset is very small [170]. Years later, [171] presented adaboost, a technique capable of implementing the initial concept of [170] effectively. The algorithm consists of performing an initial classification with some learning method and, in the following classifications, modifying the weights to make the labelling errors more evident, thus forcing a new classification that is more biased towards success. The adaboost algorithm can be observed as follows, considering a dataset D with N instances:

[6]Data selection method that obtains a sample for analysis of the given dataset and, when necessary, search for new samples obtained from the set whole, considering the data that were selected in the previous sampling(s) - also known as sampling with replacement.

Train

1. Choose the start classifier C;
2. Choose starting weights w_{1i} whereas $w \in [0, 1]$ is the sum of all weights $\sum_{i=1}^{N} w_{1i} = 1$. Usually $w_{1i} = \frac{1}{N}$;
3. For $k = 1 \rightarrow N$, create a training sample D_k with samples of D;
4. Fit machine learning algorithm C in D_k to crate classifier C_k;
5. For each misclassification of C_k in D, calculate the general error e_k being the sum of the weights of all errors $\sum_{j=1}^{N} w_{ki}$;
6. If $e_k \in (0, 0.5)$ calculate new weight $\beta_k = \frac{e_k}{1-e_k}$ and update $w_{k+1,i} = w_{ki} \cdot \beta_k$ causing the weight of correctly classified elements to decrease;
7. Normalize $w_{k+1,i}$;
8. For elements classified incorrectly e_k assign $w_{ki} = \frac{1}{N}$;
9. Repeat the process for all $C_1, C_2, ..., C_n$ classifiers in order to update the weights based on $\beta_1, \beta_2, ..., \beta_n$.

Test

1. For each object x on test dataset, classify x based on $C_1, C_2, ..., C_n$;
2. For each y class attributed to x by C_k calculate $\mu_y(x) = \sum C_k(x) = y\ ln\left(\frac{1}{\beta_k}\right)$, so that the smaller the error β_k greater will be the value of $\mu_y(x)$;
3. The class with the maximum $\mu_y(x)$ is chosen as the label of x;
4. Repeat the process for each x element in the test dataset.

It is noticed that the adaboost penalizes the errors to keep their weight high while the hits have their weights reduced each round. While the bagging method selects the final label by the majority of votes, the boosting specified here by the adaboost algorithm uses the weighted majority of votes.

There are other boosting algorithms extensively explored in the literature, according to [172]: Gradient Boosting and xgboosting. While adaboost works by adjusting the weights for each new classification round, Gradient Boosting and xgboosting try to adjust the classification of new rounds based on the residual errors of the previous classification using gradient descent for weight adjustment. The difference between Gradient Boosting and xgboosting is that the former, by the method, is slow. The second is an evolution that, despite using the same methodology, has the capacity for parallelization (using multiple CPUs during training), support for distribution between processor nodes (clusters in a distributed systems environment), and optimization of the cache of way to improve the use of the hardware that will run it.

4.3.3. Stacking

The ensemble model called stacking was initially proposed by [173]. The technique consists of implementing two classification layers, the first composed of n classifiers and the last composed of a single final classifier. Unlike the bagging and boosting techniques, the classifiers do not necessarily need to be the same; the first layer can be composed of heterogeneous classifiers.

The authors of [174] define the stacking process as a combination of predictions made by multiple learning methods $(L_1, L_2, ..., L_n)$ generated by multiple classifiers $(l_1, l_2, ..., l_n)$ in an initial layer. Such classifiers are trained with the same training sample D_{Train} which contains samples in the format $s_i =< x_1, y_i >$ where x_i is the feature vector containing values for all *features* of D_{Train} and y_i is the label of the class to which the vector belongs.

In the first phase the classifiers $l_1, l_2, ..., l_n$ perform predictions for a vector x_q. In the second phase, a M meta-classifier makes the final prediction of the class, taking into account the prediction made in the previous layer. [174] highlights the importance of choosing a meta-classifier as fundamental for increasing classification performance and [175] compliments stating that the use of a meta-classifier is justified only when the ensemble's performance is superior to the performance of the singular classifier so that it is necessary to observe in the implementation process which individual classifier has the best performance for comparison with the performance of the ensemble.

Works [176] and [177] define the stacking algorithm as the following process:

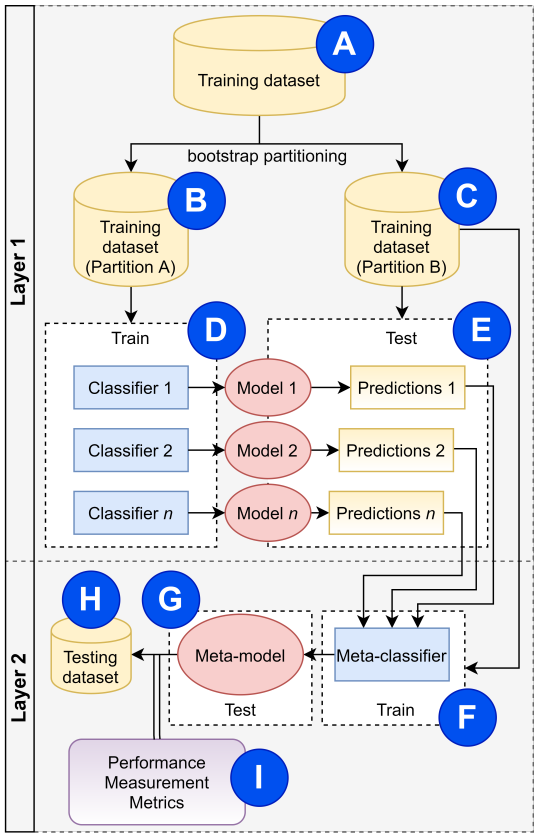
1. Split the training dataset D_{Train} in two folds $D_{Train}A$ and $D_{Train}B$;
2. Choose the individual classifiers $l_1, l_2, ..., l_n$ on first layer and fit us using $D_{Train}A$;
3. For each classifier $l_1, l_2, ..., l_n$ make predictions on fold $D_{Train}B$; and
4. Fir the meta-classifier on fold $D_{Train}B$ using as input parameters a new dataset which is $D_{Train}B$ concatenated with the predictions of the previous step made by $l_1, l_2, ..., l_n$.

Figure 10 illustrates stacking algorithmic process.

Consider letters A-I (Figure 10) as a way to analyse the process:

A – Illustrates the training sample, which is part of a hypothetical dataset;

Figure 10: Workflow of the stacking algorithm.



B – After the partitioning stage, a sample of A is found in B;

C – Likewise, a sample of A is found in C;

D – Represents the training process of individual classifiers $\{1, 2, ..., n\}$ aiming to create predictor models $\{1, 2, ..., n\}$;

E – Test stage, in which the predictor models $\{1, 2, ..., n\}$ make predictions using as test sample the second partition obtained in C (C1). The predictions are linked to partition B as new features (C2);

F – Represents the training process of a meta-classifier that is made by using the partition obtained in C (C3), linked to the predictions from C2;

G – Test process of meta-model, in which the predictions are made in the sample tests from a hypothetical dataset;

H – Represents the test sample from a hypothetical dataset; and

I – Process of analysing the stacking performance, from which factors such as accuracy measurements, precision, recall can be obtained.

Finally, quoting [173], the stacking method can be defined as a way of forwarding information from a group of classifiers to another before reaching the final classification. This process can be useful as it may reduce the error rate in a prediction model.

4.3.4. Voting

A simple way to implement classifiers together is through voting. Suppose a binary classification scenario with five classifiers where two choose class 0 while three choose class 1. By applying the majority vote, class 1 will be assigned as the label predicted by the ensemble because most classifiers chose that class.

A common type of voting is the weighted vote, where specific classifiers (usually those that produce a better performance) have their vote with greater weight, making their predictions have a multiplier factor allowing a more significant influence on the classification decision.

4.3.5. Intrusion Detection Systems

According to [178], actions related to maintaining the integrity, accessibility, or reliability of an electronic data source or a communication network define the objectives of an IDS. The authors make a historical reading of the emergence and evolution of IDS, attributing to James Anderson – at the time of the publication of the article "Computer Security Threat Monitoring and Surveillance", by [179], the first report of an IDS Two remarkable facts in the evolution of IDS occurred in 1986 and later in 1993 when publications scientifically documented intrusion detection models based on a statistical analysis of network traffic (both created by Dorothy Denning and Peter Neumann in [180]).

The authors of [181] define an IDS as software developed to detect attacks that have the potential to cause damage to the communication network or systems, whether they come from an insecure medium such as the Internet or the local network. Such software performs security countermeasures by detecting any anomaly in network traffic or in the behaviour of hosts that could characterize an attack.

IDS has two classification dimensions: network topology and packet approach. The first dimension, according to [182], defines whether the detector works by analysing the network flow (topologically, in this case, the IDS is usually placed in the communication network in the form of "bridge" (situation topological where the network device is the only means through which traffic takes place) or receiving a copy of all traffic packets through mirroring (a technique used to send copies of frames from other ports to a specific port where the IDS will be connected) from a port of the "switch" (network interconnection device acting in the data link layer

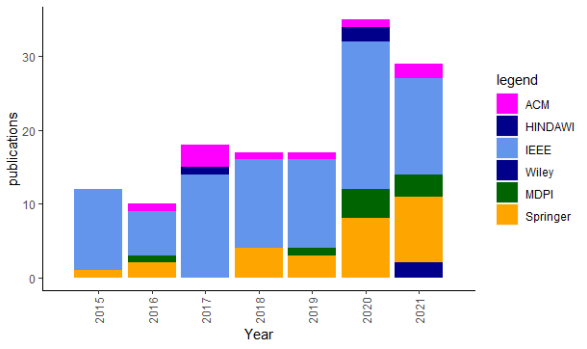
of the OSI reference model) or analysing the behaviour of the operating system. In this situation, the detector works by observing measurements of processing, disk space, memory usage, or auditing records made in the system logs. The second dimension of classification concerns the way the detector analyses the data. [183] state that in this categorization, there are two classes: signature detection and anomaly detection. The authors explain that in signature detection, the IDS has a database of known attacks, and basically, its job is to compare the packets that reach the network with its database to verify similarities. In anomaly detection, however, the system defines a model of what would be the "normal" behaviour of the network or system, causing packets that do not qualify as "normal" to be labelled malicious.

5. Discussion and Open Issues

In this section, we will describe the summary of the compilation of the analysis of the results obtained after the review.

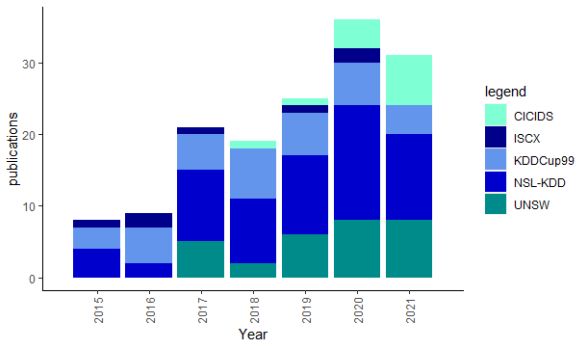
At first glance concerning the publishers, it is possible to observe in Figure 11 that in the analysed period, most of the articles were published in IEEE, concentrating 64% of the publications, followed by the share of less than a third of IEEE, Springer concentrates 20% of publications, that is, IEEE and Springer concentrate 84% of publications. In comparison, the others concentrate the remaining 16%. It is also possible to observe that, over the years, the prevalence of publications in IEEE has remained the same as the general proportion.

Figure 11: Publications by the publisher and byt the year of publication.



Concerning the datasets, as can be seen in Figure 12, it is possible to notice that 65% of the works concentrated their experiments on the NLS-KDD (34%), KDDcup99 (22%), UNSW-NB15 (16%) and CIC-IDS-2017 (5%) datasets in such a way that, at first glance, they imply that they are the most reliable Datasets. It is also possible to notice that over the years, the NLS-KDD dataset has shown up as the most used dataset in the experiments year by year.

Figure 12: Publications by the dataset used and the year of publication.



Concerning the basic classifiers, we can observe that 42% of the reports focused their experiments on making ensembles, as shown in Figure 13. The DT classifier concentrate 13%, RF 9%, NB 7%, SVM 7%, and k-NN 6%. Unlike the datasets used, the use of basic classifiers was not so prevalent. It is also evident that the Base Classifier DT is the most used algorithm year by year.

Figure 13: Publications by Classifier by Year.

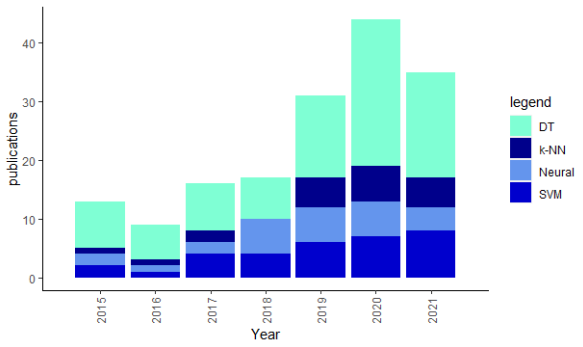
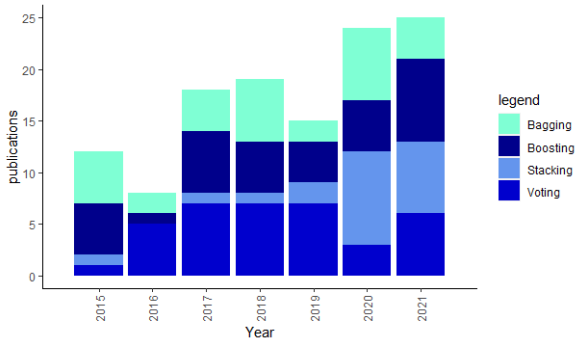


Figure 14: Publications by Ensemble algorithm by Year.



As for the Ensemble methods, as shown in Figure 14, the complete set of experiments used 56 different techniques. Of these, 47% of the experiments concentrated on the use of Bagging (16%), Boosting (12%), Voting (11%), and Stacking (9%). In the evolution of use per year in the last two years, it is possible to verify two facts, first, a decrease in the frequency of adoption of the Voting method, and second an increase in the frequency of use of the Stacking method.

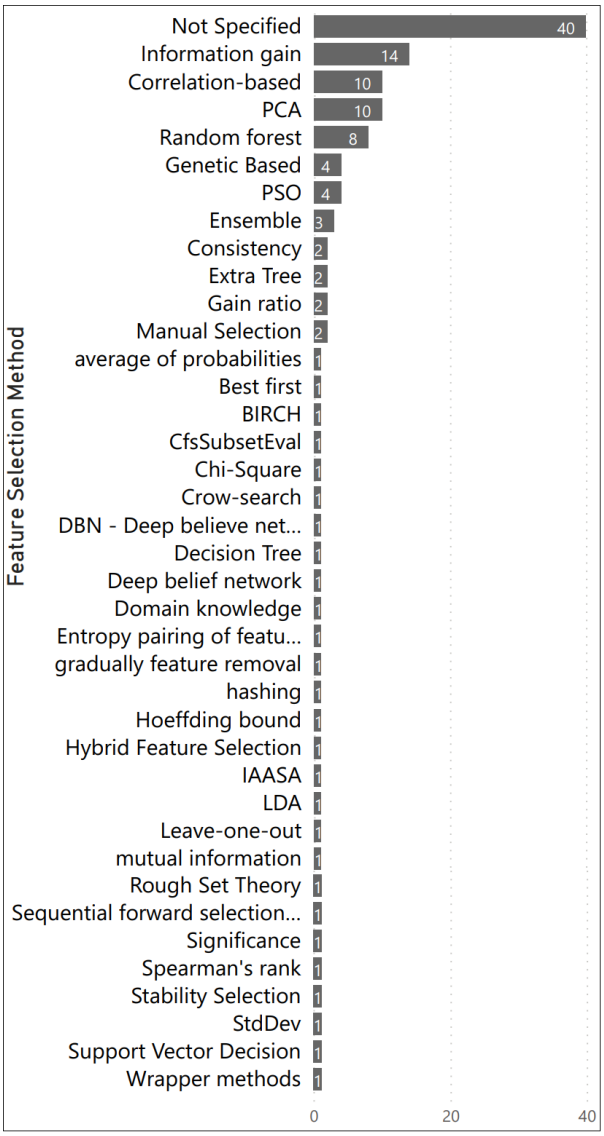


Figure 15: Feature Selection used on set of reports.

Concerning the methods of Feature Selection as showed in Figure 15, the first fact that deserved attention was to verify that of the experiments, 31% did not make use of Feature Selection. The most used methods concentrated 36% of the experiments distributed among Information Gain (11%), Correlation-based (8%), PCA (8%), and random forest (6%).

Overcoming the temporal analysis of the articles, one can enter the evaluation of the results obtained. As for the datasets, as can be seen in Table 8, of the four experiments with perfect results, three of them used the NLS-KDD dataset, and with one use, we have the KDDcup99 datasets, UNSW-NB15, uneSPY. As for the occurrence in the TOP-20 for the NLS-KDD, there are 12

occurrences. For KDD-cup99, there were four occurrences, and finally, the CIDDS-2007 with three occurrences. Regarding datasets, it is latent that among the best results, we have the NLS-KDD.

When evaluating the 20 best results, we can first take the results in terms of the basic classifiers. In this area, it is possible to observe the best results by using basic classifiers; so it is possible to observe in Table 8 that the first four best results reached perfection in terms of classification with accuracy of 1,000. The best results used random trees, linear regression, random forest, xgboost, clustering, RBM, and optimum-path. However, it is worth mentioning that the DT method is present in five works, random forest in forest works, SVM in five works, and *k*NN in four works; thus, it can be seen that these methods have produced good results.

Regarding the ensemble methods, as observed in Table 8, the best results are obtained from the extra tree method, bayesian combination, and stacking, with perfect results in terms of accuracy, precision, recall, and F-measure.

Also noteworthy in the top-20 are stacking methods with four occurrences, bagging with five occurrences, voting with five occurrences, and random forest with four occurrences.

As shown in Table 8 for the methods of feature selection among the four experiments with the perfect results, three of them did not specify or did not use the method; and as for the occurrence in the top-20, there is no specific method highlighted, except that six works did not make use of any feature selection methods.

Furthermore, Table 9 organizes a relationship between the datasets used by the year published and by article. Table 10 organizes, also by the year published and by article, a list of the ensemble methods used. Finally, Table 11 lists the classification algorithms used by the year published and article, and 12 lists methods of feature selection used organized by the year published and by article.

6. Conclusion

A Systematic Review of Literature carried out comprehensively, using several bases of scientific articles and a considerable period can bring works that reflect the state-of-the-art of a given area of study. In this research, 138 works were found that, as of 2014, document experiments related to the application of Ensemble Pruning techniques to create Intrusion Detection Systems. Works that connect by restricted search in the suggested line of research.

In order to deliver a comprehensive survey, we present details such as:

- Segregation by datasets and year (Table 2 and Figure 12) ;
- Segregation by dataset, year, and work (Table 9);
- Segregation by classifiers and year (Table 6 and Figure 13);
- Segregation by ensemble techniques and year (Table 7 and Figure 14);
- Relation by ensemble techniques, work and year (Table 10)
- Relation by classifiers, work and year (Table 11)
- Relation by feature selection methods, work and year (Table 12)
- Ranking of most used feature selection methods (Figure 15)
- Ranking of best accuracy works (Table 8);

The detailing of each research obtained allowed us to understand the techniques, algorithms, datasets, and results. This way, it was possible to document an overview of the subject. Furthermore, the k-nearest neighbors, decision tree, multilayer perceptron, and support vectors machine classifiers; the KDD-Cup'99, NSL-KDD, CICIDS-2017, UNSW-NB15, and ISCX-2012 datasets; and the Bagging, Boosting, Stacking and Voting ensemble algorithms were detailed for a better understanding of our research.

The decrease in the number of classification errors and, consequently, the increase in accuracy rates show that the use of ensemble algorithms brings considerable benefits to the cybersecurity context. Although it was not the scope of this survey, it is clear that for ensemble methods to act, training time tends to be longer, then we found a first bringing an important problem for future research: decrease computational time of training.

We observed a weakness: that only 3 of the 138 works applied or reported some technique for selecting the classifiers that would be part of the ensemble. There is an evident need to choose the most suitable classifiers for each ensemble technique, and this has not been a concern in research from what we have observed. Then, we found a second problem for future works (a possible path for solve the first future research problem) – we believe that developing research that studies ensemble pruning methods can be exciting in searching for better results and reducing the computational cost of detecting attacks.

Table 8: Top 20 works relative to datasets concerning accuracy.

Work	Dataset	Classifier	Ensemble Method	Feature Selection Method	Accuracy	Precision	Recall	F-Measure
[109]	KDDCUP'99, NSL-KDD	Random tree	Extra tree	Not used	1.000	1.000	1.000	1.000
[120]	UNSW-NB15	Linear regression, random forest, xgboost	Not Specified	Crow-search	1.000	1.000	1.000	—
[95]	NSL-KDD	Random forest, Clustering, RBM	Independent Bayesian combination classification	Not Specified	1.000	1.000	1.000	1.000
[141]	NSL-KDD, uneSPY	Optimum-path forest	Stacking	Not Used	1.000	1.000	1.000	1.000
[22]	ISTO Botnet	Decision tree, reptree, random tree	Bagging	Correlation-based	0.9997	—	—	—
[54]	KDDCUP'99	Naive bayes, boosting, PART	Bagging	Information Gain	0.9997	0.999	0.9998	—
[39]	Gure KDD	Decision tree, k-means, k-NN	Voting	Not Specified	0.9993	—	—	—
[73]	NSL-KDD	Not specified	Boosting, Gradient boost, random forest, Extra Tree	StdDev	0.999	0.999	0.999	—
[88]	HTTPCSIC-2010	SVM	Majority voting	Not Used	0.999	0.996	—	0.996
[137]	CICIDS-2017, NSL-KDD, UNSW-NB15	K-means, One class SVM, DBSCAN, Expectation maximization	KODE	CfsSubsetEval, genetic search rule-based engine, average of probabilities	0.999	0.909	—	0.903
[148]	Own Dataset	Neural network	Not Specified	Not Used	0.999	—	—	—
[21]	NSL-KDD	C4.5, random forest, CART	Bagging, adaboost, Multiboost, Rotation Forest, Majority voting, Average probability	Particle Swarm Optimization	0.998	—	—	—
[30]	NSL-KDD	Rule learner, decision tree	DAREnsemble, Voting rule	Not Used	0.998	—	—	—
[147]	UNSW-NB15, CICIDS 2017, NSL-KDD, KDD99	k-NN, SVM	Dempster-Shafer method	Ensemble margin	0.998	0.999	0.997	0.998
[76]	CICIDS-2017	Decision tree	Random forest	Ensemble between ReliefF, InfoGain, Consistency e Correlation	0.997	—	—	—
[85]	KDD-CUP-99', NSL-KDD	Decision tree	Stacking	Manual	0.997	—	—	—
[126]	CIDDS-002	decision tree, Naive Bayes	Eandom forest, Bagging, adaboost, Stacking	Not Used	0.997	0.99	1.000	1.000
[134]	NSL-KDD	Random forest, K-nn, SVM	Voting	Wrapper	0.997	0.984	0.986	0.986
[16]	NSL-KDD	Reptree	Bagging	BIRCH Hierarchical Clustering	0.996	—	—	—
[127]	NSL-KDD	Random forest, SVM, k-nn, Naïve Bayes, MLP	Stacking	Not Specified	0.996	0.996	0.995	0.989

Table 9: Relation of datasets used per article per year

Dataset	2015	2016	2017	2018	2019	2020	2021
ADFA						[102]	
AWID		[23]		[55]			
CAIDA	[19]						
CIC-IDS-2017				[58]	[76]	[92] [117]	
CIC-IDS-2018						[90]	
CIDDS-001					[75]		
DARPA				[63] [63]			
DoS				[63]			
DS2OS						[119]	
Euskalert	[20]						
GPRS		[25]	[38]				
GTCS						[103]	
Gure			[39]				
HTTPCSIC-2010			[45]			[88] [117]	
Hyperplane						[114]	
ISCX	[13]	[24] [31]			[84]	[107] [114]	
ISTO	[22]						
KDD-CUP'99	[11] [14]	[32] [31]	[49] [35]	[66] [51]	[80] [81]	[85] [101]	[124] [121]
	[15]	[26] [27]	[43] [44]	[52] [54]	[79] [69]	[104] [105]	[122]
		[29]	[45]	[56] [57]	[75] [76]	[109]	
				[58]			
KSL-KDD				[65]		[108]	
Kyoto				[56] [66]	[72]	[111]	
LLS-DDoS1.0	[19]						
NIMS					[68]		
NSL-KDD	[12] [16]	[32] [30]	[33] [34]	[53] [56]	[69] [70]	[85] [91]	[124]
	[17] [18]		[36] [37]	[58] [59]	[71] [73]	[92] [93]	
	[21]		[38] [43]	[60] [61]	[74] [76]	[94] [95]	
			[47] [48]	[64] [66]	[78] [82]	[96] [97]	
			[41] [50]	[67]	[83] [84]	[100] [101]	
					[77]	[102] [109]	
						[110] [113]	
						[115] [117]	
OWN		[28]					
UGR'16						[86]	
UMaT-OD-20						[87]	
UNB-ISCX			[45]				
UNSW			[34] [38]	[58] [62]	[79] [68]	[86] [89]	[118] [120]
			[42] [40]		[75] [76]	[100] [106]	[123] [124]
			[46]		[82] [83]	[112] [116]	
						[117] [119]	

Table 10: Relation of ensemble methods used per article per year.

Dataset	2015	2016	2017	2018	2019	2020	2021
Accuracy Updated Ensemble				[63]			
AdaBoost	[21]				[80]	[101] [113]	[118]
Average Probability	[20] [21]	[23] [25]	[33] [38] [33]	[55] [62]	[73] [76]	[104] [96] [110] [113] [119]	
Bagged Tree	[21]						
Bagging			[46]				
Boosting	[14] [16] [17] [19] [20] [21] [22]	[26]	[33] [42] [43]	[52] [54] [55] [56] [60] [62]	[78] [83]	[93] [101] [111] [113] [91] [97] [114] [88]	[118]
Clustering	[11] [14] [15] [19] [20] [21] [20]	[24]	[33] [42] [46] [47] [34]	[52] [60] [61] [62]	[68] [73] [78]	[89] [115]	[121]
DAREnsemble		[28]		[66]			
Decision Tree		[30]					
Deep Blockchain Framework						[88]	
Dynamic Deep Forest						[112]	
Ensemble Incremental Learning			[33]			[95] [95] [88]	
Extra Tree					[81]		
GBDecision Tree				[51]			
Genetic Based		[23]		[55]	[73]	[109]	
k-NN					[73]	[96]	
Logistic Regression			[35]				
Majority voting			[46]				
Naive Bayes	[21]		[33] [43] [37]	[56] [64] [67]	[84]	[88] [103]	
Neural Network		[27] [31]	[33]				[124]
One-vs-All		[32]	[40] [44] [45] [48] [49]			[90] [92] [94]	[120] [122]
Owner	[18]			[65]			
PSO						[107]	
Random committee			[35]				
Random Forest			[43]				
Random Subspace Method			[37]				
RandomCommittee		[29]					
Softmax			[35]		[70]	[88]	
Stacking					[70]		
SVM					[79]		
T-ensemble	[13]		[42]	[52]	[82] [77]	[85] [86] [87] [102] [105] [106] [108] [116] [117]	[123]
Voting						[100]	
Weighted Majority Voting	[12]	[29] [23] [30]	[36] [38] [39] [41]	[53] [57] [58] [59]	[69] [71] [74] [75] [78] [83]	[113]	
XGBoost			[55]		[96]		
Not Specified			[50]			[88]	

Table 11: Relation of base classifiers algorithm used per article per year.

Dataset	2015	2016	2017	2018	2019	2020	2021
Adaboost				[66]		[89] [105] [111]	[124]
Ant Colony					[80]		
Autoencoder			[48]		[82]		
Bagging						[105]	[124]
Boosting				[54] [67] [51]	[81]	[108] [90]	[124] [123]
Decision stump		[24]	[47]	[60] [62] [64]		[114]	
Decision tree	[21][11] [12] [14] [15] [18] [22]	[25] [23] [24] [26] [30]	[49] [38] [42] [44] [46] [39]	[53][66][63] [55] [57]	[71] [68] [74] [75] [76] [78] [80] [77]	[85] [87] [88] [89] [90] [91] [104] [94] [101] [103] [114] [96] [110] [113] [116]	[123] [124][118]
DNN		[32]	[48]	[65]	[71]		
ELM			[48]		[79]		
Gaussian models					[81]	[119] [89]	
Genetic-based				[84]		[107]	
K-means	[13]	[32] [28]	[36] [39] [40] [41]	[66] [59]	[70] [72]	[95]	
k-NN	[15]	[24]	[35] [39]		[71] [72] [74] [75] [78]	[86] [87] [92] [103] [114] [113] [116] [119]	[124]
Logistic regression		[24]	[49]	[57]	[71]	[113] [86] [87] [88] [89] [90] [106] [114]	
MLP	[15] [18]		[42]	[51] [53] [56] [57] [61]	[74] [75] [69] [68] [84]	[103] [113] [89] [112] [88]	
Naive bayes	[12] [13] [14] [15] [12]	[24] [24] [29]	[33] [42] [49]	[51] [54] [64] [66] [67]	[68] [77]	[87] [91] [106] [114] [116] [88] [114] [100]	[124]
PSO			[35]			[115]	
Random forest	[12] [14] [19] [21]	[23]	[34]	[60] [66]	[71] [72] [78] [82] [81] [77]	[86] [87] [91] [94] [95] [106] [108] [105] [101] [89]	[120] [123] [124]
Random tree	[22]	[29]	[43]	[60] [66]		[109]	
Reptree	[16] [22]		[42]	[60] [62]		[101]	
SVM	[15] [18]	[27]	[37] [45] [36] [35]	[52] [58] [59] [67]	[70] [71] [72] [74] [75] [82]	[86] [88] [89] [104] [111] [102] [106]	[124] [122]
XGBoost						[105] [119] [89]	[120] [121]
Not Specified	[17] [20]	[31]	[50] [48]		[73] [83]	[93] [117] [97]	

Table 12: Relation of feature selection method used per article per year.

Dataset	2015	2016	2017	2018	2019	2020	2021
Chi-Square					[68]		
Clustering		[32] [31] [30]	[49]	[66]	[81] [84]	[88] [92] [93] [100] [102] [104] [114] [109] [110] [111]	[118] [123] [126] [129] [130] [131] [138] [133] [139] [140] [141] [142] [143] [146] [148]
Consistency, information e correlation	[13]						
Correlation coefficient			[43] [47]	[54] [63] [60]	[72]	[91] [97] [94] [86] [103]	[144]
Correlation-based							[?]
Crow-search			[34]				
DBN				[65]			
Decision Tree					[80]		
Deterministic Selection	[11] [17] [18] [19] [20]	[23] [24] [25] [26] [28] [29]	[36] [38] [39] [42] [40] [46] [48] [41] [50] [33]	[51] [52] [56] [61] [62] [64]	[69] [70] [74] [78] [82]	[95] [113] [112] [107] [115] [117] [87]	[121] [127]
Domain knowledge					[73]		
Enhanced Support Vector Decision Function						[90]	
Ensemble						[85]	
Entropy						[116]	
Extra Tree			[45]				
Genetic Based				[58]			[122]
Gradually Feature Removal							[134]
Hoeffding bound			[35]				
Hybrid Feature Selection							[120]
IAASA					[83] [76] [75]		[147]
Information Gain					[79]		
Manual				[67]			
Mutual Information	[12]			[59]		[96]	[137]
Not Specified						[89]	
Not Used			[44]				
PCA						[108]	[136] [135]
PSO	[14]	[27]		[53] [57]	[71]	[106] [105]	[128]
Random Forest	[15] [22]		[37]	[55]		[101]	
Rough Set						[119]	
Sequential forward selection (modified)					[77]		
Spearman's Rank	[21]						[124]
StdDev							[132]
Wrapper methods	[16]						

References

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

[2] K. Schwab, A quarta revolução industrial, Edipro, 2019.

[3] Forbes, Cybersecurity in 2022 – a fresh look at some very alarming stats, `shorturl.at/bfpqS`, (Accessed on 07/05/2022) (Jan 2022).

[4] Allianz, Allianz risk barometer — agcs, `shorturl.at/gptw7`, (Accessed on 07/05/2022) (Jan 2022).

[5] S. Morgan, Cybercrime to cost the world $10.5 trillion annually by 2025, `shorturl.at/coUZ6`, (Accessed on 07/05/2022) (Nov 2020).

[6] Accenture, Cost of cybercrime study — 9th annual — accenture, `shorturl.at/bemsu`, (Accessed on 07/05/2022) (Mar 2019).

[7] K. R. F. Scannavino, E. Y. Nakagawa, S. C. P. F. Fabbri, F. C. Ferrari, Revisão sistemática da literatura em engenharia de software: teoria e prática.

[8] B. Kitchenham, P. Brereton, Z. Li, D. Budgen, A. Burn, Repeatability of systematic literature reviews, in: 15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011), IET, 2011, pp. 46–55.

[9] S. Keshav, How to read a paper, ACM SIGCOMM Computer Communication Review 37 (3) (2007) 83–84.

[10] C. S. Ishikiriyama, D. Miro, C. F. S. Gomes, Text mining business intelligence: a small sample of what words can say, Procedia Computer Science 55 (2015) 261–267.

[11] L. Mehra, M. K. Gupta, H. S. Gill, An effectual & secure approach for the detection and efficient searching of network intrusion detection system (nids), in: 2015 International Conference on Computer, Communication and Control (IC4), IEEE, 2015, pp. 1–5.

[12] N. F. Haq, A. R. Onik, F. M. Shah, An ensemble framework of anomaly detection using hybridized feature selection approach (hfsa), in: 2015 SAI Intelligent Systems Conference (IntelliSys), IEEE, 2015, pp. 989–995.

[13] M. Milliken, Y. Bi, L. Galway, G. Hawe, Ensemble learning utilising feature pairings for intrusion detection, in: 2015 World Congress on Internet Security (WorldCIS), IEEE, 2015, pp. 24–31.

[14] P. Amudha, S. Karthik, S. Sivakumari, Intrusion detection based on core vector machine and ensemble classification methods, in: 2015 International Conference on Soft-Computing and Networks Security (ICSNS), IEEE, 2015, pp. 1–5.

[15] P. Sornsuwit, S. Jaiyen, Intrusion detection model based on ensemble learning for u2r and r2l attacks, in: 2015 7th international conference on information technology and electrical engineering (ICITEE), IEEE, 2015, pp. 354–359.

[16] D. Gaikwad, R. C. Thool, Intrusion detection system using bagging ensemble method of machine learning, in: 2015 International Conference on Computing Communication Control and Automation, IEEE, 2015, pp. 291–295.

[17] M. Sreenath, J. Udhayan, Intrusion detection system using bagging ensemble selection, in: 2015 IEEE International Conference on Engineering and Technology (ICETECH), IEEE, 2015, pp. 1–4.

[18] Z. Ye, Y. Yu, Network intrusion classification based on extreme learning machine, in: 2015 IEEE International Conference on Information and Automation, IEEE, 2015, pp. 1642–1647.

[19] R. R. Robinson, C. Thomas, Ranking of machine learning algorithms based on the performance in classifying ddos attacks, in: 2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS), IEEE, 2015, pp. 185–190.

[20] S. Gonzalez, A. Herrero, J. Sedano, U. Zurutuza, E. Corchado, Different approaches for the detection of ssh anomalous connections.

[21] B. A. Tama, K. H. Rhee, A Combination of PSO-Based Feature Selection and Tree-Based Classifiers Ensemble for Intrusion Detection Systems, in: D.-S. Park, H.-C. Chao, Y.-S. Jeong, J. J. Park (Eds.), Advances in Computer Science and Ubiquitous Computing, Vol. 373, Springer Singapore, Singapore, 2015, pp. 489–495. doi:10.1007/978-981-10-0281-6$_7$1.

[22] O. Y. Al-Jarrah, O. Alhussein, P. D. Yoo, S. Muhaidat, K. Taha, K. Kim, Data randomization and cluster-based partitioning for botnet intrusion detection, IEEE transactions on cybernetics 46 (8) (2016) 1796–1806.

[23] B. Alotaibi, K. Elleithy, A majority voting technique for wireless intrusion detection systems, in: 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), IEEE, 2016, pp. 1–6.

[24] G. Folino, F. S. Pisani, P. Sabatino, An incremental ensemble evolved by using genetic programming to efficiently detect drifts in cyber security datasets, in: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, 2016, pp. 1103–1110.

[25] B. A. Tama, K.-H. Rhee, Classifier ensemble design with rotation forest to enhance attack detection of ids in wireless network, in: 2016 11th Asia Joint Conference on Information Security (AsiaJCIS), IEEE, 2016, pp. 87–91.

[26] P. Mehetrey, B. Shahriari, M. Moh, Collaborative ensemble-learning based intrusion detection systems for clouds, in: 2016 International Conference on Collaboration Technologies and Systems (CTS), IEEE, 2016, pp. 404–411.

[27] A. A. Aburomman, M. B. I. Reaz, Ensemble of binary svm classifiers based on pca and lda feature extraction for intrusion detection, in: 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), IEEE, 2016, pp. 636–640.

[28] B. Kiranmai, A. Damodaram, Extenuate ddos attacks in cloud, in: 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), IEEE, 2016, pp. 235–238.

[29] Y. Wang, Y. Shen, G. Zhang, Research on intrusion detection model using ensemble learning methods, in: 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), IEEE, 2016, pp. 422–425.

[30] D. Gaikwad, R. Thool, DAREnsemble: Decision Tree and Rule Learner Based Ensemble for Network Intrusion Detection System, in: S. C. Satapathy, S. Das (Eds.), Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems: Volume 1, Vol. 50, Springer International Publishing, Cham, 2016, pp. 185–193. doi:10.1007/978-3-319-30933-0$_2$0.

[31] G. Folino, F. S. Pisani, P. Sabatino, A Distributed Intrusion Detection Framework Based on Evolved Specialized Ensembles of Classifiers, in: G. Squillero, P. Burelli (Eds.), Applications of Evolutionary Computation, Vol. 9597, Springer International Publishing, Cham, 2016, pp. 315–331. doi:10.1007/978-3-319-31204-0$_2$1.

[32] T. Ma, F. Wang, J. Cheng, Y. Yu, X. Chen, A Hybrid Spectral Clustering and Deep Neural Network Ensemble Algorithm for Intrusion Detection in Sensor Networks, Sensors 16 (10) (2016) 1701. doi:10.3390/s16101701.

[33] S. T. Miller, C. Busby-Earle, Multi-perspective machine learning a classifier ensemble method for intrusion detection, in: Proceedings of the 2017 International Conference on Machine Learning and Soft Computing, 2017, pp. 7–12.

[34] M. H. Kamarudin, C. Maple, T. Watson, N. S. Safa, A logitboost-

based algorithm for detecting known and unknown web attacks, IEEE Access 5 (2017) 26190–26200.

[35] M. Rajasekaran, A. Ayyasamy, A novel ensemble approach for effective intrusion detection system, in: 2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM), IEEE, 2017, pp. 244–250.

[36] W. Chen, F. Kong, F. Mei, G. Yuan, B. Li, A novel unsupervised anomaly detection approach for intrusion detection system, in: 2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), IEEE international conference on high performance and smart computing (hpsc), and IEEE international conference on intelligent data and security (ids), IEEE, 2017, pp. 69–73.

[37] B. A. Tama, A. S. Patil, K.-H. Rhee, An improved model of anomaly detection using two-level classifier ensemble, in: 2017 12th Asia Joint Conference on Information Security (AsiaJCIS), IEEE, 2017, pp. 1–4.

[38] R. Primartha, B. A. Tama, Anomaly detection using random forest: A performance revisited, in: 2017 International conference on data and software engineering (ICoDSE), IEEE, 2017, pp. 1–6.

[39] M. Jabbar, R. Aluvalu, S. S. S. Reddy, Cluster based ensemble classification for intrusion detection system, in: Proceedings of the 9th International Conference on Machine Learning and Computing, 2017, pp. 253–257.

[40] M. M. Aravind, V. Kalaiselvi, Design of an intrusion detection system based on distance feature using ensemble classifier, in: 2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN), IEEE, 2017, pp. 1–6.

[41] S. Ruoti, S. Heidbrink, M. O'Neill, E. Gustafson, Y. R. Choe, Intrusion detection with unsupervised heterogeneous ensembles using cluster-based normalization, in: 2017 IEEE International Conference on Web Services (ICWS), IEEE, 2017, pp. 862–865.

[42] M. Belouch, S. E. hadaj, 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, 2017, pp. 1–4.

[43] A. Niranjan, A. Prakash, N. Veena, M. Geetha, P. D. Shenoy, K. Venugopal, Ebjrv: An ensemble of bagging, j48 and random committee by voting for efficient classification of intrusions, in: 2017 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), IEEE, 2017, pp. 51–54.

[44] S. Garg, A. Singh, S. Batra, N. Kumar, M. S. Obaidat, Enclass: Ensemble-based classification model for network anomaly detection in massive datasets, in: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, 2017, pp. 1–7.

[45] R. R. Reddy, Y. Ramadevi, K. Sunitha, Enhanced anomaly detection using ensemble support vector machine, in: 2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC), IEEE, 2017, pp. 107–111.

[46] V. Timčenko, S. Gajin, Ensemble classifiers for supervised anomaly based network intrusion detection, in: 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), IEEE, 2017, pp. 13–19.

[47] N. N. Mkuzangwe, F. Nelwamondo, Ensemble of classifiers based network intrusion detection system performance bound, in: 2017 4th International Conference on Systems and Informatics (ICSAI), IEEE, 2017, pp. 970–974.

[48] S. A. Ludwig, Intrusion detection of multiple attack classes using a deep neural net ensemble, in: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2017, pp. 1–7.

[49] U. R. Salunkhe, S. N. Mali, Security Enrichment in Intrusion Detection System Using Classifier Ensemble, Journal of Electrical and Computer Engineering 2017 (2017) 1–6. doi:10.1155/2017/1794849.

[50] S. Tengl, Z. Zhang, L. Teng, W. Zhang, H. Zhu, X. Fang, L. Fei, A collaborative intrusion detection model using a novel optimal weight strategy based on genetic algorithm for ensemble classifier, in: 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD)), IEEE, 2018, pp. 761–766.

[51] X. Yuan, R. Wang, Y. Zhuang, K. Zhu, J. Hao, A concept drift based ensemble incremental learning approach for intrusion detection, in: 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, 2018, pp. 350–357.

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

[53] Y. Gao, Y. Liu, Y. Jin, J. Chen, H. Wu, A novel semi-supervised learning approach for network intrusion detection on cloud-based robotic system, IEEE Access 6 (2018) 50927–50938.

[54] R. K. S. Gautam, E. A. Doegar, An ensemble approach for intrusion detection system using machine learning algorithms, in: 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), IEEE, 2018, pp. 14–15.

[55] F. D. Vaca, Q. Niyaz, An ensemble learning based wi-fi network intrusion detection system (wnids), in: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), IEEE, 2018, pp. 1–5.

[56] Y. Shen, K. Zheng, C. Wu, M. Zhang, X. Niu, Y. Yang, An ensemble method based on selection using bat algorithm for intrusion detection, The Computer Journal 61 (4) (2018) 526–538.

[57] A. H. Mirza, Computer network intrusion detection using various classifiers and ensemble learning, in: 2018 26th Signal Processing and Communications Applications Conference (SIU), IEEE, 2018, pp. 1–4.

[58] N. Marir, H. Wang, G. Feng, B. Li, M. Jia, Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark, IEEE Access 6 (2018) 59657–59671.

[59] A. Abdullah, R. Ponnan, D. Asirvatham, Improving multiclass classification in intrusion detection using clustered linear separator analytics, in: 2018 8th International Conference on Intelligent Systems, Modelling and Simulation (ISMS), IEEE, 2018, pp. 32–37.

[60] N. T. Pham, E. Foo, S. Suriadi, H. Jeffrey, H. F. M. Lahza, Improving performance of intrusion detection system using ensemble methods and feature selection, in: Proceedings of the Australasian Computer Science Week Multiconference, 2018, pp. 1–6.

[61] B. Zhang, Y. Yu, J. Li, Network intrusion detection based on stacked sparse autoencoder and binary tree ensemble method, in: 2018 IEEE International Conference on Communications Workshops (ICC Workshops), IEEE, 2018, pp. 1–6.

[62] S. Zwane, P. Tarwireyi, M. Adigun, Performance analysis of machine learning classifiers for intrusion detection, in: 2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC), IEEE, 2018, pp. 1–5.

[63] A. Muallem, S. Shetty, L. Hong, J. W. Pan, Tddeht: Threat detection using distributed ensembles of hoeffding trees on streaming cyber datasets, in: MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM), IEEE, 2018, pp. 1–6.

[64] M. A. Jabbar, K. Srinivas, S. Sai Satyanarayana Reddy, A Novel Intelligent Ensemble Classifier for Network Intrusion Detection System, in: A. Abraham, A. K. Cherukuri, A. M. Madureira, A. K. Muda (Eds.), Proceedings of the Eighth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2016), Vol. 614, Springer International Publishing, Cham, 2018, pp. 490–497. doi:10.1007/978-3-319-60618-$7_4$8.

[65] A. Parvat, S. Dev, S. Kadam, J. Chavan, Network Intrusion Detection System Using Ensemble of Binary Deep Learning Classifiers, in: A. Deshpande, A. Unal, K. Passi, D. Singh, M. Nayak, B. Patel, S. Pathan (Eds.), Smart Trends in Information Technology and Computer Communications, Vol. 876, Springer Singapore, Singapore, 2018, pp. 3–10. doi:10.1007/978-981-13-1423-0_1.

[66] S. Kaur, I. Garg, Ensemble Technique Based on Supervised and Unsupervised Learning Approach for Intrusion Detection, in: M. Singh, P. K. Gupta, V. Tyagi, J. Flusser, T. Ören (Eds.), Advances in Computing and Data Sciences, Vol. 905, Springer Singapore, Singapore, 2018, pp. 228–238. doi:10.1007/978-981-13-1810-8$_2$3.

[67] I. S. Thaseen, C. A. Kumar, A. Ahmad, Integrated Intrusion Detection Model Using Chi-Square Feature Selection and Ensemble of Classifiers, Arabian Journal for Science and Engineering 44 (4) (2018) 3357–3368. doi:10.1007/s13369-018-3507-5.

[68] N. Moustafa, B. Turnbull, K.-K. R. Choo, An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things, IEEE Internet of Things Journal 6 (3) (2018) 4815–4830.

[69] M. Labonne, A. Olivereau, B. Polvé, D. Zeghlache, A cascade-structured meta-specialists approach for neural network-based intrusion detection, in: 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2019, pp. 1–6.

[70] D. Liang, Q. Liu, B. Zhao, Z. Zhu, D. Liu, A clustering-svm ensemble method for intrusion detection system, in: 2019 8th International Symposium on Next Generation Electronics (ISNE), IEEE, 2019, pp. 1–3.

[71] X. Gao, C. Shan, C. Hu, Z. Niu, Z. Liu, An adaptive ensemble machine learning model for intrusion detection, IEEE Access 7 (2019) 82512–82521.

[72] F. Salo, M. Injadat, A. Moubayed, A. B. Nassif, A. Essex, Clustering enabled classification using ensemble feature selection for intrusion detection, in: 2019 International Conference on Computing, Networking and Communications (ICNC), IEEE, 2019, pp. 276–281.

[73] F. J. P. Montalbo, E. D. Festijo, Comparative analysis of ensemble learning methods in classifying network intrusions, in: 2019 IEEE 9th International Conference on System Engineering and Technology (ICSET), IEEE, 2019, pp. 431–436.

[74] S. Das, A. M. Mahfouz, D. Venugopal, S. Shiva, Ddos intrusion detection through machine learning ensemble, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), IEEE, 2019, pp. 471–477.

[75] W. He, H. Li, J. Li, Ensemble feature selection for improving intrusion detection classification accuracy, in: Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science, 2019, pp. 28–33.

[76] A. Binbusayyis, T. Vaiyapuri, Identifying and benchmarking key features for cyber intrusion detection: An ensemble approach, IEEE Access 7 (2019) 106495–106513.

[77] L. Lu, S. Teng, W. Zhang, Z. Zhang, D. Liu, X. Fang, 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), IEEE, 2019, pp. 4–9.

[78] P. Illy, G. Kaddoum, C. M. Moreira, K. Kaur, S. Garg, Securing fog-to-things environment using intrusion detection system based on ensemble learning, in: 2019 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2019, pp. 1–7.

[79] J. Sharma, C. Giri, O.-C. Granmo, M. Goodwin, Multi-layer intrusion detection system with ExtraTrees feature selection, extreme learning machine ensemble, and softmax aggregation, EURASIP Journal on Information Security 2019 (1) (2019) 15. doi:10.1186/s13635-019-0098-y.

[80] S. M. Mousavi, V. Majidnezhad, A. Naghipour, A new intelligent intrusion detector based on ensemble of decision trees, Journal of Ambient Intelligence and Humanized Computingdoi:10.1007/s12652-019-01596-5.

[81] B. Hu, J. Wang, Y. Zhu, T. Yang, Dynamic Deep Forest: An Ensemble Classification Method for Network Intrusion Detection, Electronics 8 (9) (2019) 968. doi:10.3390/electronics8090968.

[82] Y.-F. Hsu, Z. He, Y. Tarutani, M. Matsuoka, Toward an online network intrusion detection system based on ensemble learning, in: 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), IEEE, 2019, pp. 174–178.

[83] B. A. Tama, M. Comuzzi, K.-H. Rhee, Tse-ids: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system, IEEE Access 7 (2019) 94497–94507.

[84] G. Kumar, An improved ensemble approach for effective intrusion detection, The Journal of Supercomputing 76 (1) (2020) 275–291. doi:10.1007/s11227-019-03035-w.

[85] W. Lian, G. Nie, B. Jia, D. Shi, Q. Fan, Y. Liang, An Intrusion Detection Method Based on Decision Tree-Recursive Feature Elimination in Ensemble Learning, Mathematical Problems in Engineering 2020 (2020) 1–15. doi:10.1155/2020/2835023.

[86] S. Rajagopal, P. P. Kundapur, K. S. Hareesha, A Stacking Ensemble for Network Intrusion Detection Using Heterogeneous Datasets, Security and Communication Networks 2020 (2020) 1–9. doi:10.1155/2020/4586875.

[87] F. L. Aryeh, B. K. Alese, A Multi-layer Stack Ensemble Approach to Improve Intrusion Detection System's Prediction Accuracy, in: 2020 15th International Conference for Internet Technology and Secured Transactions (ICITST), IEEE, London, United Kingdom, 2020, pp. 1–6. doi:10.23919/ICITST51030.2020.9351316.

[88] S. Das, M. Ashrafuzzaman, F. T. Sheldon, S. Shiva, Network Intrusion Detection using Natural Language Processing and Ensemble Machine Learning, in: 2020 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, Canberra, ACT, Australia, 2020, pp. 829–835. doi:10.1109/SSCI47803.2020.9308268.

[89] S. Divakar, R. Priyadarshini, B. Kishore Mishra, A Robust Intrusion Detection System using Ensemble Machine Learning, in: 2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE), IEEE, Bhubaneswar, India, 2020, pp. 344–347. doi:10.1109/WIECON-ECE52138.2020.9397969.

[90] Q. R. S. Fitni, K. Ramli, Implementation of Ensemble Learning and Feature Selection for Performance Improvements in Anomaly-Based Intrusion Detection Systems, in: 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), IEEE, Bali, Indonesia, 2020, pp. 118–124. doi:10.1109/IAICT50021.2020.9172014.

[91] S. R. Khonde, V. Ulagamuthalvi, Ensemble and Feature Selection-based Intrusion Detection System for Multi-attack Environment, in: 2020 5th International Conference on Computing, Communication and Security (ICCCS), IEEE, Patna, India, 2020, pp. 1–8. doi:10.1109/ICCCS49678.2020.9276875.

[92] A. S. Kyatham, M. A. Nichal, B. S. Deore, A novel approach for network intrusion detection using probability parameter to ensemble machine learning models, in: 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), IEEE, 2020, pp. 608–613.

[93] Z. Lin, D. Hongle, Research on SDN intrusion detection based on online ensemble learning algorithm, in: 2020 International Conference on Networking and Network Applications (NaNA), IEEE, Haikou City, China, 2020, pp. 114–118. doi:10.1109/NaNA51271.2020.00027.

[94] S. Nandi, S. Maity, M. Das, NIDF: An Ensemble-inspired Feature Learning Framework for Network Intrusion Detection, in: 2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE), IEEE, Bhubaneswar, India, 2020, pp. 9–12. doi:10.1109/WIECON-ECE52138.2020.9397993.

[95] S. Otoum, B. Kantarci, H. T. Mouftah, A Novel Ensemble Method for Advanced Intrusion Detection in Wireless Sensor Networks, in: ICC 2020 - 2020 IEEE International Conference on Communications (ICC), IEEE, Dublin, Ireland, 2020, pp. 1–6. doi:10.1109/ICC40277.2020.9149413.

[96] A. Rai, Optimizing a New Intrusion Detection System Using Ensemble Methods and Deep Neural Network, in: 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), IEEE, Tirunelveli, India, 2020, pp. 527–532. doi:10.1109/ICOEI48184.2020.9143028.

[97] M. M. Rashid, J. Kamruzzaman, M. Ahmed, N. Islam, S. Wibowo, S. Gordon, Performance Enhancement of Intrusion detection System Using Bagging Ensemble Technique with Feature Selection, in: 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), IEEE, Gold Coast, Australia, 2020, pp. 1–5. doi:10.1109/CSDE50874.2020.9411608.

[98] X. Shi, Y. Cai, Y. Yang, Extreme trees network intrusion detection framework based on ensemble learning, in: 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA), IEEE, Dalian, China, 2020, pp. 91–95. doi:10.1109/AEECA49918.2020.9213695.

[99] J. Yang, Y. Sheng, J. Wang, A GBDT-Paralleled Quadratic Ensemble Learning for Intrusion Detection System, IEEE Access 8 (2020) 175467–175482. doi:10.1109/ACCESS.2020.3026044.

[100] J. Zhang, F. Li, F. Ye, An Ensemble-based Network Intrusion Detection Scheme with Bayesian Deep Learning, in: ICC 2020 - 2020 IEEE International Conference on Communications (ICC), IEEE, Dublin, Ireland, 2020, pp. 1–6. doi:10.1109/ICC40277.2020.9149402.

[101] C. Iwendi, S. Khan, J. H. Anajemba, M. Mittal, M. Alenezi, M. Alazab, The Use of Ensemble Models for Multiple Class and Binary Class Classification for Improving Intrusion Detection Systems, Sensors 20 (9) (2020) 2559. doi:10.3390/s20092559.

[102] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, A. Alazab, Hybrid Intrusion Detection System Based on the Stacking Ensemble of C5 Decision Tree Classifier and One Class Support Vector Machine, Electronics 9 (1) (2020) 173. doi:10.3390/electronics9010173.

[103] A. Mahfouz, A. Abuhussein, D. Venugopal, S. Shiva, Ensemble Classifiers for Network Intrusion Detection Using a Novel Network Attack Dataset, Future Internet 12 (11) (2020) 180. doi:10.3390/fi12110180.

[104] N. Martindale, M. Ismail, D. A. Talbert, Ensemble-Based Online Machine Learning Algorithms for Network Intrusion Detection Systems Using Streaming Data, Information 11 (6) (2020) 315. doi:10.3390/info11060315.

[105] T. Feng, M. Dou, P. Xie, J. Fang, Network intrusion detection based on data feature dynamic ensemble model, in: International Conference on Artificial Intelligence and Security, Springer, 2020, pp. 661–673.

[106] M. Abirami, U. Yash, S. Singh, Building an ensemble learning based algorithm for improving intrusion detection system, in: Artificial Intelligence and Evolutionary Computations in Engineering Systems, Springer, 2020, pp. 635–649.

[107] G. Folino, F. S. Pisani, L. Pontieri, A GP-based ensemble classification framework for time-changing streams of intrusion detection data, Soft Computing 24 (23) (2020) 17541–17560. doi:10.1007/s00500-020-05200-3.

[108] H. Rajadurai, U. D. Gandhi, A stacked ensemble learning model for intrusion detection in wireless network, Neural Computing and Applicationsdoi:10.1007/s00521-020-04986-5.

[109] B. S. Bhati, C. Rai, Ensemble based approach for intrusion detection using extra tree classifier, in: Intelligent computing in engineering, Springer, 2020, pp. 213–220.

[110] A. Sadiwala, K. Rathore, Y. Shah, H. Shah, K. Srivastava, Intrusion detection system against malign packets—a comparative study between autoencoder and ensemble model, in: Advanced Computing Technologies and Applications, Springer, 2020, pp. 165–175.

[111] J. Wei, C. Long, J. Li, J. Zhao, An intrusion detection algorithm based on bag representation with ensemble support vector machine in cloud computing, Concurrency and Computation: Practice and Experience 32 (24). doi:10.1002/cpe.5922.

[112] O. Alkadi, N. Moustafa, B. Turnbull, K.-K. R. Choo, A deep blockchain framework-enabled collaborative intrusion detection for protecting iot and cloud networks, IEEE Internet of Things Journal.

[113] N. Lower, F. Zhan, A study of ensemble methods for cyber security, in: 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, 2020, pp. 1001–1009.

[114] F. Folino, G. Folino, L. Pontieri, A p2p environment to validate ensemble-based approaches in the cybersecurity domain, in: 2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), IEEE, 2020, pp. 344–351.

[115] H. Jiang, Z. He, G. Ye, H. Zhang, Network intrusion detection based on pso-xgboost model, IEEE Access 8 (2020) 58392–58401.

[116] O. O. Olasehinde, O. V. Johnson, O. C. Olayemi, 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), IEEE, 2020, pp. 1–7.

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

[118] O. J. Mebawondu, O. D. Alowolodu, A. O. Adetunmbi, J. O. Mebawondu, Optimizing the classification of network intrusion detection using ensembles of decision trees algorithm, in: International Conference on Information and Communication Technology and Applications, Springer, 2020, pp. 286–300.

[119] P. Kumar, G. P. Gupta, R. Tripathi, A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks, Journal of Ambient Intelligence and Humanized Computing 12 (10) (2021) 9555–9572. doi:10.1007/s12652-020-02696-3.

[120] G. Srivastava, T. R. G, N. Deepa, B. Prabadevi, P. K. Reddy M, An ensemble model for intrusion detection in the Internet of Softwarized Things, in: Adjunct Proceedings of the 2021 International Conference on Distributed Computing and Networking, ACM, Nara Japan, 2021, pp. 25–30. doi:10.1145/3427477.3429987.

[121] B. S. Bhati, G. Chugh, F. Al-Turjman, N. S. Bhati, An improved ensemble based intrusion detection technique using XGBoost, Transactions on Emerging Telecommunications Technologies 32 (6). doi:10.1002/ett.4076.

[122] Z. Zhao, L. Ge, G. Zhang, A novel DBN-LSSVM ensemble method for intrusion detection system, in: 2021 9th International Conference on Communications and Broadband Networking, ACM, Shanghai China, 2021, pp. 101–107. doi:10.1145/3456415.3456431.

[123] Y. Zhao, G. Gan, Research on Intrusion Detection Technology Based on Ensemble Learning, in: International Conference on Frontiers of Electronics, Information and Computation Technologies, ACM, Changsha China, 2021, pp. 1–6. doi:10.1145/3474198.3478139.

[124] T. Acharya, I. Khatri, A. Annamalai, M. F. Chouikha, Efficacy of Heterogeneous Ensemble Assisted Machine Learning Model for Binary and Multi-Class Network Intrusion Detection, in: 2021 IEEE International Conference on Automatic Control & Intelligent Systems (I2CACIS), IEEE, Shah Alam, Malaysia, 2021, pp. 408–413. doi:10.1109/I2CACIS52118.2021.9495864.

[125] N. Ahmed, R. Durga, A Trust Aware Behavioral Based Intrusion Detection in Cloud Environment Using Ensemble Service Centric Featured Neural Network, in: 2021 4th International Conference on Computing and Communications Technologies (ICCCT), IEEE, Chennai, India, 2021, pp. 342–349. doi:10.1109/ICCCT53315.2021.9711827.

[126] Ainurrochman, A. Nugroho, R. Wahyuwidayat, S. T. Sianturi, M. Fauzi, M. F. Ramadhan, B. A. Pratomo, A. M. Shiddiqi, Ensemble Methods Classifier Comparison for Anomaly Based Intrusion Detection System on CIDDS-002 Dataset, in: 2021 13th International Conference on Information & Communication Technology and System (ICTS), IEEE, Surabaya, Indonesia, 2021, pp. 62–67. doi:10.1109/ICTS52701.2021.9608714.

[127] S. Ennaji, N. E. Akkad, K. Haddouch, A Powerful Ensemble Learning Approach for Improving Network Intrusion Detection System

(NIDS), in: 2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS), IEEE, Fez, Morocco, 2021, pp. 1–6. doi:10.1109/ICDS53782.2021.9626727.

[128] T. T. Khoei, G. Aissou, W. C. Hu, N. Kaabouch, Ensemble Learning Methods for Anomaly Intrusion Detection System in Smart Grid, in: 2021 IEEE International Conference on Electro Information Technology (EIT), IEEE, Mt. Pleasant, MI, USA, 2021, pp. 129–135. doi:10.1109/EIT51626.2021.9491891.

[129] A. Z. Kiflay, A. Tsokanos, R. Kirner, A Network Intrusion Detection System Using Ensemble Machine Learning, in: 2021 International Carnahan Conference on Security Technology (ICCST), IEEE, Hatfield, United Kingdom, 2021, pp. 1–6. doi:10.1109/ICCST49569.2021.9717397.

[130] X. Li, M. Zhu, L. T. Yang, M. Xu, Z. Ma, C. Zhong, H. Li, Y. Xiang, Sustainable Ensemble Learning Driving Intrusion Detection Model, IEEE Transactions on Dependable and Secure Computing (2021) 1–1doi:10.1109/TDSC.2021.3066202.

[131] H. Li, D. Chasaki, Ensemble Machine Learning for Intrusion Detection in Cyber-Physical Systems, in: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, Vancouver, BC, Canada, 2021, pp. 1–2. doi:10.1109/INFOCOMWKSHPS51825.2021.9484479.

[132] Z. Lin, Network intrusion detection based of semi-supervised ensemble learning algorithm for imbalanced data, in: 2021 International Conference on Networking and Network Applications (NaNA), IEEE, 2021, pp. 338–344.

[133] S. M. Nzuva, L. Nderu, T. Mwalili, Ensemble Model for Enhancing Classification Accuracy in Intrusion Detection Systems, in: 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET), IEEE, Cape Town, South Africa, 2021, pp. 1–7. doi:10.1109/ICECET52533.2021.9698798.

[134] P. Parkar, A Network Intrusion Detection System Based on Ensemble Machine Learning Techniques, in: 2021 IEEE 2nd International Conference on Applied Electromagnetics, Signal Processing, & Communication (AESPC), IEEE, Bhubaneswar, India, 2021, pp. 1–6. doi:10.1109/AESPC52704.2021.9708502.

[135] S. Seth, K. K. Chahal, G. Singh, A Novel Ensemble Framework for an Intelligent Intrusion Detection System, IEEE Access 9 (2021) 138451–138467. doi:10.1109/ACCESS.2021.3116219.

[136] V. Sidharth, C. R. Kavitha, Network Intrusion Detection System Using Stacking and Boosting Ensemble Methods, in: 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), IEEE, Coimbatore, India, 2021, pp. 357–363. doi:10.1109/ICIRCA51532.2021.9545022.

[137] E. Jaw, X. Wang, Feature Selection and Ensemble-Based Intrusion Detection System: An Efficient and Comprehensive Approach, Symmetry 13 (10) (2021) 1764. doi:10.3390/sym13101764.

[138] H.-C. Lin, P. Wang, K.-M. Chao, W.-H. Lin, Z.-Y. Yang, Ensemble Learning for Threat Classification in Network Intrusion Detection on a Security Monitoring System for Renewable Energy, Applied Sciences 11 (23) (2021) 11283. doi:10.3390/app112311283.

[139] A. Wang, W. Wang, H. Zhou, J. Zhang, Network Intrusion Detection Algorithm Combined with Group Convolution Network and Snapshot Ensemble, Symmetry 13 (10) (2021) 1814. doi:10.3390/sym13101814.

[140] A. Subasi, S. Algebsani, W. Alghamdi, E. Kremic, J. Almaasrani, N. Abdulaziz, Intrusion detection in smart healthcare using bagging ensemble classifier, in: International Conference on Medical and Biological Engineering, Springer, 2021, pp. 164–171.

[141] M. A. Bertoni, G. H. de Rosa, J. R. F. Brega, Optimum-path forest stacking-based ensemble for intrusion detection, Evolutionary Intelligencedoi:10.1007/s12065-021-00609-7.

[142] D. Mulimani, S. G. Totad, P. Patil, S. V. Seeri, Adaptive ensemble learning with concept drift detection for intrusion detection, in: Data Engineering and Intelligent Computing, Springer, 2021, pp. 331–339.

[143] A. P. Psathas, L. Iliadis, A. Papaleonidas, D. Bountas, A hybrid deep learning ensemble for cyber intrusion detection, in: International Conference on Engineering Applications of Neural Networks, Springer, 2021, pp. 27–41.

[144] S. R. Khonde, G. Kulanthaivel, V. Ulagamuthalvi, An ensemble approach for intrusion detection in collaborative attack environment, in: Smart Computing Techniques and Applications, Springer, 2021, pp. 137–146.

[145] A. J. Siddiqui, A. Boukerche, Adaptive ensembles of autoencoders for unsupervised IoT network intrusion detection, Computing 103 (6) (2021) 1209–1232. doi:10.1007/s00607-021-00912-2.

[146] P. Singh, V. Ranga, Attack and intrusion detection in cloud computing using an ensemble learning approach, International Journal of Information Technology 13 (2) (2021) 565–571. doi:10.1007/s41870-020-00583-w.

[147] M. Yousefnezhad, J. Hamidzadeh, M. Aliannejadi, Ensemble classification for intrusion detection via feature extraction based on deep Learning, Soft Computing 25 (20) (2021) 12667–12683. doi:10.1007/s00500-021-06067-8.

[148] T. N. Thinh, T. H. Q. Bao, D.-M. Ngo, C. Pham-Quoc, High-performance anomaly intrusion detection system with ensemble neural networks on reconfigurable hardware, Concurrency and Computation: Practice and Experiencedoi:10.1002/cpe.6370.

[149] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, A detailed analysis of the kdd cup 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, IEEE, 2009, pp. 1–6.

[150] I. Sharaf., A. Lashkari, Habibi, A. A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization., in: ICISSP, 2018, pp. 108–116.

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

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

[153] A. Shiravi, H. Shiravi, M. Tavallaee, A. A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, computers & security 31 (3) (2012) 357–374.

[154] N. Moustafa, J. Slay, others, et al., Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in: 2015 military communications and information systems conference (MilCIS), IEEE, 2015, pp. 1–6.

[155] W.-M. Lee, Python Machine Learning, John Wiley & Sons, 2019.

[156] K. Bakshi, K. Bakshi, Considerations for artificial intelligence and machine learning: Approaches and use cases, in: 2018 IEEE Aerospace Conference, IEEE, 2018, pp. 1–9.

[157] J. Mueller, L. Massaron, Aprendizado de Máquina Para Leigos, Para Leigos, Alta Books, 2019.
URL https://books.google.com.br/books?id=FyaWDwAAQBAJ

[158] R. Baeza-Yates, B. Ribeiro-Neto, Recuperação de Informação-: Conceitos e Tecnologia das Máquinas de Busca, Bookman Editora, 2013.

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

[160] J. Tchaye-Kondi, Y. Zhai, L. Zhu, A new hashing based nearest neighbors selection technique for big datasets, arXiv preprint arXiv:2004.02290.

[161] R. Tinós, Perceptron multicamadas, shorturl.at/bE138, (Acesso em 04/15/2020) (2020).

[162] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain., Psychological review 65 (6) (1958) 386.

[163] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.

[164] S. O. Rezende, Sistemas inteligentes: fundamentos e aplicações, Edi-

tora Manole Ltda, 2003.
[165] H. Sharma, S. Kumar, A survey on decision tree algorithms of classification in data mining, International Journal of Science and Research (IJSR) 5 (4) (2016) 2094–2097.
[166] Z.-H. Zhou, Ensemble methods: foundations and algorithms, Chapman and Hall/CRC, 2012.
[167] V. Smolyakov, Ensemble learning to improve machine learning results, https://blog.statsbot.co/ensemble-learning-d1dcd548e936, (Acesso em 30/09/2019) (2017).
[168] S. Khonde, V. Ulagamuthalvi, A machine learning approach for intrusion detection using ensemble technique-a survey.
[169] Y. Alsouda, S. Pllana, A. Kurti, Iot-based urban noise identification using machine learning: Performance of svm, knn, bagging, and random forest, in: Proceedings of the International Conference on Omni-Layer Intelligent Systems, ACM, 2019, pp. 62–67.
[170] R. E. Schapire, The strength of weak learnability, Machine learning 5 (2) (1990) 197–227.
[171] Y. Freund, R. E. Schapire, A decision-theoretic generalization of online learning and an application to boosting, Journal of computer and system sciences 55 (1) (1997) 119–139.
[172] J. D'Souza, A quick guide to boosting in ml - greyatom - medium, shorturl.at/GJM38, (Acesso em 01/10/2019) (2018).
[173] D. H. Wolpert, Stacked generalization, Neural networks 5 (2) (1992) 241–259.
[174] S. Agarwal, C. R. Chowdary, A-stacking and a-bagging: Adaptive versions of ensemble learning algorithms for spoof fingerprint detection, Expert Systems with Applications 146 (2020) 113160.
[175] S. Džeroski, B. Ženko, Is combining classifiers with stacking better than selecting the best one?, Machine learning 54 (3) (2004) 255–273.
[176] C. Aggarwal, Data classification: Algorithms and applications, ser, Frontiers in physics. Chapman and Hall/CRC.
[177] J. Rocca, Ensemble methods: bagging, boosting and stacking - towards data science, shorturl.at/itWX0, (Accesso en 23/04/2020) (2019).
[178] S. Osken, E. N. Yildirim, G. Karatas, L. Cuhaci, Intrusion detection systems with deep learning: A systematic mapping study, in: 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT), IEEE, 2019, pp. 1–4.
[179] J. P. Anderson, Computer security threat monitoring and surveillance, Technical Report, James P. Anderson Company.
[180] D. E. Denning, An intrusion-detection model, IEEE Transactions on software engineering (2) (1987) 222–232.
[181] G. Karatas, O. K. Sahingoz, Neural network based intrusion detection systems with different training functions, in: 2018 6th International Symposium on Digital Forensic and Security (ISDFS), IEEE, 2018, pp. 1–6.
[182] M. Kaouk, J.-M. Flaus, M.-L. Potet, R. Groz, A review of intrusion detection systems for industrial control systems, in: 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), IEEE, 2019, pp. 1699–1704.
[183] J. Ran, Y. Ji, B. Tang, A semi-supervised learning approach to ieee 802.11 network anomaly detection, in: 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), IEEE, 2019, pp. 1–5.

3 Stacking-based Committees for Detecting Attacks on Computer Networks - An Exhaustion Approach

This work was presented at the 39th Brazilian Symposium on Computer Networks and Distributed Systems in 2021[7] evaluated by Brazilian CAPES with A4 Qualis score. It aimed to demonstrate the application of the exhaustion method to choose the best composition of classifiers in an ensemble through stacking. The hypothesis was that the optimal choice obtained by testing all possibilities would create a robust intrusion detection system, effectively reducing errors but inefficient because of the computational cost.

Although it seems obvious that the hypothesis would come true, we needed to document this condition to reinforce the need and the benefits of implementing a pruning method that maintained the effectiveness in reducing errors but was also able to deliver a low computational cost. The main points highlighted in work regarding the method and results were:

- Use of the trends obtained by the Systematic Literature Review (Chapter 2) to choose the classifiers, dataset, and ensemble method;
- Application of the method by exhaustion where 44 stacking combinations involving the four most used classifiers were created;
- The results showed that the optimal combination had the lowest error rate in detecting five of the six attacks studied when compared to the performance of the best individual classifier.

The analysis of the optimal combinations allowed us to observe a logic that, later, in the works documented in Chapters 4 and 5, would prove efficient in reducing the computational cost with more comprehensive studies (in other datasets).

The next work remains in English (to maintain the standard of this thesis), but its originally published version was written in Portuguese.

[7] 39th Brazilian Symposium on Computer Networks and Distributed Systems - https://www.sbrc2021.facom.ufu.br/

Stacking-based Committees for Detecting Attacks on Computer Networks - An Exhaustion Approach

Abstract. *The application of Machine Learning techniques in detecting attacks on computer networks has obtained good results, emphasizing Ensemble's methods, which improve the performance of individual classifiers. The present study was carried out in Dataset CICIDS-2017 and considered classifiers' choice based on a systematic literature review, aiming to find the state-of-the-art 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%.*

1. Introduction

Information can be considered the greatest asset of a company or corporation in the modern economic scenario, according to [Fraimovich et al. 2020]. This fact highlights that owners should be concerned with protecting their data, especially confidentiality, integrity, and availability.

The heterogeneity of Internet accesses concerning the content and anatomy of packets transmitted over the world wide web makes identifying attacks (or malicious packets) a research problem to which the academic community has been dedicated. In this sense, several types of Intrusion Detection Systems (IDS) have been used to analyze "network behavior" to detect malicious flows that necessarily aim to cause damage to assets, especially information.

This study presents an exhaustive approach where several committees (grouping of classification algorithms) were created in order to unite the capacity of heterogeneous classifiers in order to propose the best combination to detect six different types of attack: Brute-force, Infiltration, DDoS, Portscan, attacks from Botnets and web attacks. Forty-four models were implemented that group 4 different classifiers (k-Nearest Neighbors - k-NN, Support-vector Machines - SVM, Decision Trees - DT and Multilayer Perceptron - MLP, which is a type of Neural Network - NN) considering all possibilities acceptable by the method of Ensemble "Stacking". The classifiers and the Ensemble technique were chosen after observing trends from a systematic literature review. A relevant contribution of this study is the presentation of the best model of combinations of classifiers for each attack type proposed to detect a visible drop in classification errors.

The present work is structured as follows: Section 2 illustrates the trends obtained through the systematic review of the literature, as well as brings a list of related works that used Stacking and the theoretical description of the fundamental elements of research; Section 3 presents the modeling of the committees, and an overview of the experimental process followed to obtain the results; Section 4 presents the numbers of Precision, Recall, F1-Score and Accuracy as well as an analysis regarding the reduction of classification errors in comparison with individual methods and a comparison of the results of this study with related works; Finally, Section 5 presents conclusions and proposals for future work.

2. Systematic Literature Review

A systematic review of the literature was carried out in order to obtain which classifiers and which Ensemble techniques are mostly used so that it was possible to observe some trends. Relevant scientific bases in the area of Computer Science and Computer Networks were searched for publications that contained the terms “Ensemble” and “Intrusion”, either in the title of the publication or in the Keywords and whose year of publication was between the years 2015 and 2020, that is, recent works that have used any Ensemble technique applied to intrusion detection in computer networks.

Because the development of this research took place in the second half of 2020, the publications of the year in question took into account only the first half. For the numbers to be adjusted, the values for the last year were multiplied by 2 (therefore, the numbers referring to 2020 are projections). In all, 68 works were found (considering the exclusion of the same works or those related to other areas).

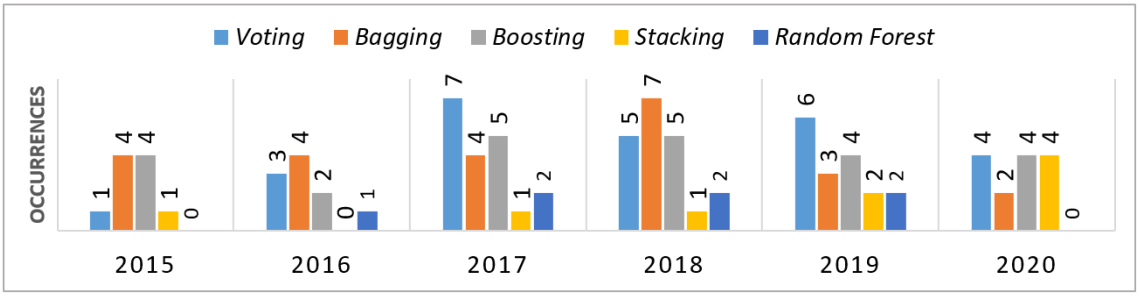


Figure 1. List of classifiers by number of occurrences in the analyzed interval. Source: The Author.

Section 2.5 provides more details about classification methods. It is possible to observe some trends, such as the growing use of k-NN and, although oscillating, the implementation of DT and NN, although it is recognized that a more careful methodology could bring more support for the observation of trends. Although there is no record in the last year for the application of SVM in the context of this work, it is important to observe the growth of the method in previous years. Figure 1 illustrates the number of publications per year between 2015 and 2020, grouped by the four most used classifiers. Other classifiers were also used in related works; however, these 4 stand out for the number of repetitions. Figure 2 illustrates the ratio of the number of publications per year in the interval between 2015 and 2020, grouped by the most used Ensemble methods:

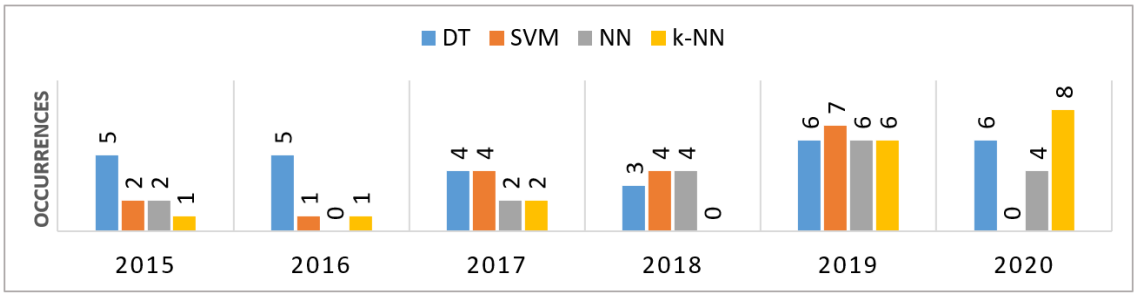


Figure 2. List of Ensemble techniques by number of occurrences in the analyzed range. Source: The Author.

It is visible that a large number of implementations using the techniques Voting, Bagging, or Boosting. However, the growth of the method called stacking stands out (more details in Section 2.4), which motivated its choice as the method for creating the committees of this work.

2.1. Correlated Work

This section lists a summary of related works that used Stacking for intrusion detection.

A Stacking Ensemble model was proposed by [Milliken et al. 2015]. The authors used the ISCX 2012 Dataset as a basis for the tests together with the OneR, Conjunctive, and Naive Bayes classifiers in a Stacking model to carry out the Ensemble—the experiments obtained as the best result an F-Score of 92%. The experiments demonstrated that entropy analysis between pairs of features from different subsets can contribute to the generation of a subset more suitable for attack classification.

[Belouch and hadaj 2017] performed a comparison between three different Ensemble techniques: Booting, Bagging and Stacking. The authors combined four different classifiers: Decision Tree, Naive Bayes, Multilayer Perceptron and REPTree. All experiments were performed on the Dataset UNSW-NB15. The publication documents all the results, and it is possible to observe that the best Ensemble in terms of accuracy was a Stacking of REPTree followed by a Stacking of Decision Trees .

[Sun et al. 2018] presented an intrusion detection model that combines three classic Ensemble techniques: Bagging, Boosting and Stacking. The base classifiers used were SVM and k-NN. With tests carried out on the Dataset NSL-KDD, after generating a subset through PCA, it was possible to measure the classification of normal packets with 99.41% and that of packets with Portscan with 93.13%.

The intrusion detection system presented by [Lu et al. 2019] separates the packets by layers to be classified according to the TCP, UDP, and ICMP protocols. For each group of packages, binary classifiers arranged in cascade are applied so that, at the end of the process, it is possible to perform a complete multiclass classification. The authors performed tests with the NSL-KDD Dataset using the SVM, DT, Bayes Network, REPTree, k-NN, BFTree, SimpleCart, and Naive Bayes classifiers. The results were superior to those compared to Bagging, Boosting and Stacking, reaching an accuracy of 95.33% for detecting DoS attacks, 91.14% for Portscan attacks, 55.21% for R2L and 1.42 for U2R, in addition to an accuracy of 98.02% in detecting benign packets.

In experiments with Datasets NSL-KDD and UNSW-NB15, [Hsu et al. 2019] presented intrusion detection models using Stacking either in sorting or selecting features, clustered using Stacking also in the detection process with the SVM classifiers, Autoencoder and Random Forest. Accuracy results indicate 91.7% with NSL-KDD and 91.8% with UNSW-NB15.

[Olasehinde et al. 2020] performed experiments on the Dataset UNSW-NB15 in order to obtain the best meta-classifier hypothesis (second layer classifier) for an intrusion detection model using Stacking. Performing the classifications of the first layer with k-NN, Naive Bayes, and DT, the authors tested some possibilities of meta-classifiers and compared the results. Using three different subsets, the best results were using DT in the second layer, pointing to an average accuracy of 98.56%.

An intrusion detection system specialized in detecting web attacks was presented by [Tama et al. 2020]. An Ensemble through Stacking was implemented; however, instead of using conventional classifiers, the authors used other Ensemble methods such as Random Forest, Gradient Boost Machine and XG boost to create the committess. Four different Datasets were tested: CICIDS-2017, HTTP-CSIC-2010, NSL-KDD and UNSW-NB15. The average result of Accuracy reported by the authors was 96.07% for the proposed model, with the best result obtained in CICIDS-2017 at 99.98%.

Concerning the related works mentioned above, this work presents as a differential a detailed observation through exhaustion where the method (documented in Section 3) allows the analysis of all possible combinations considering the classifiers used. The best committee implemented via Stacking will be presented for the different types of attacks analyzed.

2.2. Intrusion Detection Systems

[Karatas and Sahingoz 2018] define an IDS as software developed to detect attacks that have the potential to cause damage to the communication network or systems, whether they come from an insecure medium such as the Internet or the local network. Such software performs security countermeasures when detecting any anomaly in network traffic or in the behavior of hosts that could characterize an attack. The authors point out that many approaches have been applied to increase the detection rate, generally involving Machine Learning techniques (situations where the detector learns through training to detect anomalies), Rule-based (in which case the detector has characteristic signatures of attacks and only compares real traffic with signatures of already known attacks) and classic statistical methods (use of average calculations, standard deviation, median, among others).

2.3. Ensemble Learning

Ensemble Machine Learning methods, according to [Zhou 2012], consist of the joint application of different classification or regression algorithms in order to solve the same problem. While traditional techniques build learning based on a specific technique, Ensemble techniques seek to aggregate, either in parallel or sequentially, multiple learning techniques, making, in the words of [Smolyakov 2017], there are improvements regarding variance, bias or predictions.

According to [Zhou 2012], there are three ways of organizing a method of Ensemble concerning how the individual algorithms are related, namely: “combining classifiers”, “Ensembles of weak” and “mixture of experts”. The first form is the most used in the literature, especially for solving pattern recognition problems. It is based on combining classifiers with excellent Accuracy and seeks to combine them to improve the final (joint) Accuracy in the classification. The second form is the most used by the Machine Learning community and consists of the joint application of lightweight classification algorithms (with good Accuracy) so that the final classifier has excellent Accuracy by extracting the best of each lightweight classifier. Ly, the last method, is explored more in Artificial Neural Networks and assumes a better solution can be found by combining rules and mixing parameters.

By observing the trends extracted in the Systematic Literature Review, the Stacking method, the core of the methodology of this research, will be highlighted.

2.4. Stacking - Stacked Generalization

A model of Ensemble called Stacking was originally proposed by [Wolpert 1992]. The technique consists of implementing two layers of classification, the first consisting of n classifiers and the last composed of a single final classifier. Unlike bagging and boosting techniques, the classifiers do not necessarily need to be the same; the first layer can be composed of heterogeneous classifiers.

[Agarwal and Chowdary 2020] define the Stacking process as a combination of predictions performed by multiple learning methods ($L_1, L_2, ..., L_n$) generated by multiple classifiers ($l_1, l_2, ..., l_n$) in an initial layer. Such classifiers are trained with the same training sample D_{Train} that contains samples in the format $s_i =< x_1, y_i >$ where x_i is the feature vector containing values for all features of D_{Train} and y_i is the label of the class to which the array belongs.

In the first phase, the classifiers $l_1, l_2, ..., l_n$ make predictions for a vector x_q. In the second phase, a meta-classifier M makes the final prediction of the class, taking into account the prediction made in the previous layer. [Agarwal and Chowdary 2020] highlight the importance of choosing the meta-classifier as fundamental to increase the classification performance and [Džeroski and Ženko 2004] compliment by stating that the use of a meta-classifier is only justified when the performance of the Ensemble is superior to the performance of the single classifier so that it must be observed in the implementation process which individual classifier has the best performance for comparison with the performance of Ensemble.

[Aggarwal 2014] and [Rocca 2019] define the Stacking algorithm as the following process:

1. Split the training sample D_{Train} into two partitions $D_{Train}A$ and $D_{Train}B$;
2. Define the individual classifiers $l_1, l_2, ..., l_n$ of the first layer and train them using the partition $D_{Train}A$;
3. For each classifier $l_1, l_2, ..., l_n$ make predictions on partition $D_{Train}B$; It is
4. Train the metaclassifier on the partition $D_{Train}B$ using as input parameters a new dataset that is $D_{Train}B$ concatenated with the predictions from the previous step made by $l_1, l_2, ..., l_n$.

To analyze the process, consider the letters (from A to I) in Figure 3:

A - Represents the training sample that is part of a hypothetical Dataset;
B - After the partitioning step, we have a sample of A in B;
C - In the same way, we have a sample of A in C;
D - Represents the training process of the individual classifiers $\{1, 2, ..., n\}$ in order to generate predictive models $\{1, 2, ..., n\}$;
E - Testing stage, where predictor models $\{1, 2, ..., n\}$ make predictions using the second partition obtained in process C (C1) as test sampling. Predictions are concatenated in partition B as new features (C2);
F - Represents the meta-classifier training process that is performed with the partition obtained in process C (C3) already concatenated with the predictions of step C2;
G - Meta-model testing process where predictions are performed on the testing sample of a hypothetical Dataset;
H - Represents the sampling of tests from a hypothetical dataset; It is

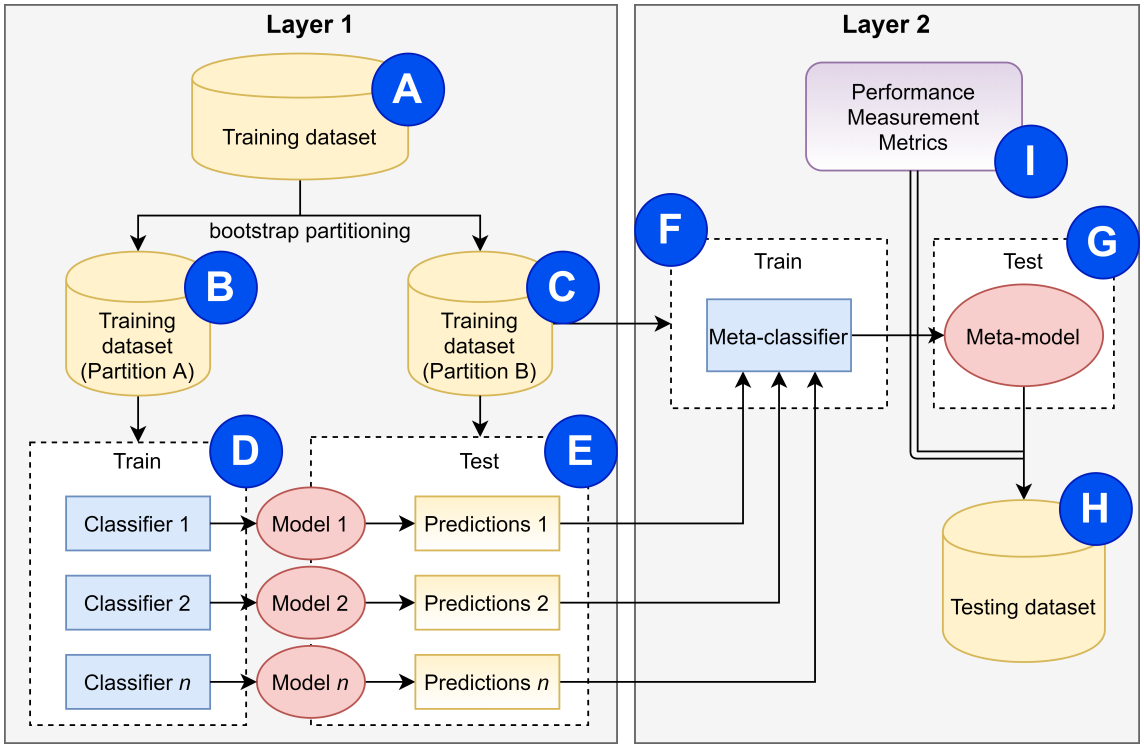


Figure 3. Stacking workflow. Source: The Author.

I - Stacking performance analysis process where you get, for example, metrics of accuracy, precision, and recall, among others.

Finally, in the words of [Wolpert 1992], the Stacking method is defined as a scheme for forwarding information from one set of classifiers to another before obtaining the final classification, which can be useful as a way to reduce the error rate in a prediction model.

2.5. Classification

The choice of algorithms for implementing the Stackings considered the trends obtained in the Systematic Literature Review. It can be seen in Figure 1 that there are four algorithms with highlighted use representing the state-of-the-art for Ensemble techniques in the creation of intrusion detection systems. This section describes the operation of the DT, k-NN, MLP, and SVM methods.

2.5.1. Decision Trees

According to [Rezende 2003], the methods that use Decision Trees belong to the Top Down Induction of Decision Trees (TDIDT) family, which is a data structure that defines "leaf nodes " and "decision nodes". The first node is responsible for representing a class, while the second is responsible for representing a test on some attribute. For each example to be classified, the attributes are conditioned by the "decision nodes" to obtain a path to a "leaf node" class. A sequence of logical "ifs" will determine the path to be traversed in the tree for each example.

[Sharma and Kumar 2016] define DT as a flowchart-like structure where each internal node represents a test for an attribute, each branch represents the test result, and each leaf represents a class. The authors point out that DT can be constructed with relatively less time and processing cost than other classification methods.

2.5.2. k-Nearest Neighbors

The k-Nearest Neighbors method works in a very simple way. By establishing a value for k, the algorithm, in the parameter space, calculates the test sample distance for all training samples and orders them from the smallest distance to the largest. Finally, for the k nearest neighbors, the most repeated class is observed, which will finally be attributed to the analyzed object.

The best value of the k hyperparameter is obtained by testing. According to [Tchaye-Kondi et al. 2020], although it is simple, k-NN can obtain excellent data classification results. [Deng et al. 2016] claim that because k-NN needs to compute the distance of all training samples for each test sample, the computational cost must be considered in the implementation process.

2.5.3. Multilayer Perceptron

An Artificial Neural Network is a structure similar to the human brain regarding communication between neurons. The human brain responds to stimuli at the entrances of its neurons, and cellular organization causes the others to be activated (or not) in response to the processing of that stimulus. Likewise, an Artificial Neural Network usually has the number of entries corresponding to the number of features of a Dataset to be processed. The structure of the other layers and how the propagation of stimuli happens depends on the type of Neural Network created. The last layer of neurons corresponds to the classes of the analyzed data so that, given the input parameters and the propagation of stimuli, it is possible to trigger the neuron corresponding to the estimated class.

A perceptron, a Neural Network type, produces a binary output when given multiple inputs. It is a mathematical model that, by using a single layer of neurons, can solve only linearly separable problems. Due to this limitation of the single-layer Perceptron, according to [Tinós 2020], it is common to use the Multilayer Perceptron since the second layer can combine the outputs of the first layer in order to produce a final classification solving linearly non-separable problems.

2.5.4. Support Vectors Machine

A classifier based on SVM seeks, presented in the parameter space, two linearly separable classes (a and b), to draw the best hyperplane so that the element of class a is closest to the first element of class b are considered the limits. The limits are called “support vectors” because they represent the closest a and b elements. The hyperplane will therefore be constructed exactly at half of these limits.

Also important is SVM's ability to deal with situations of class separation in n-dimensional spaces. [Baeza-Yates and Ribeiro-Neto 2013] highlight that the method can solve binary classification problems with equal efficiency in more dimensions. If, in a two-dimensional situation, the division of classes is carried out by drawing a straight line, in a three-dimensional situation, for example, a plane would be drawn to carry out the classification. In short, the SVM can handle several dimensions, highlighting the computational cost required as more dimensions (and features) are analyzed.

2.6. CICIDS-2017 Dataset

In order to validate the proposal of this study, six different Datasets were provided by the Canadian Institute for Cybersecurity[1] through University of New Brunswick[2] were used. All six datasets are part of a project named CICIDS-2017 published by [Sharaf. et al. 2018].

Each dataset contains data for a specific attack type and benign traffic data. All have 78 features with the respective labels in column 79. Although the data are available containing different types of the same attack, they were transformed into binary data to allow the creation of an intrusion detection system that does not have the objective to characterize the detected attacks but to differentiate legitimate traffic from malicious traffic. Table 1 presents more details about the structure of the datasets used:

Table 1. Details of Dataset CICIDS-2017. Source: The Author.

ID	Attack type	Packets number
1	Brute-force	432074 benigns; 13835 malicious.
2	Infiltration	288566 benigns; 36 malicious.
3	DDoS	97718 benigns; 128027 malicious.
4	Portscan	127537 benigns; 158930 malicious.
5	Botnet	189067 benigns; 1966 malicious.
6	Web	168186 benigns; 2180 malicious.

The CICIDS-2017 dataset is divided into eight files, according to [Panigrahi and Borah 2018] and [Stiawan et al. 2020]. Two files were discarded because the first had only benign traffic, and the second had many records, making it impossible to process it using the exhaustion method.

3. Methodology

This Section lists the list of materials and methods used in this study. The hardware used in the tests had the following characteristics: 8 Intel(R) Core(TM) i7 CPU 960 @3.20GHz processors and 16 GB of RAM. Coding was performed in Python using the MLXtend, Pandas, and Sci-kit Learn libraries.

3.1. Stacking Development

In order to implement the optimized models, the k-NN, MLP, DT, and SVM classifiers were defined according to the best hyperparameters obtained through grid search tuning.

[1]Canadian Institute for Cybersecurity - https://www.unb.ca/cic/

[2]University of New Brunswick - https://www.unb.ca/

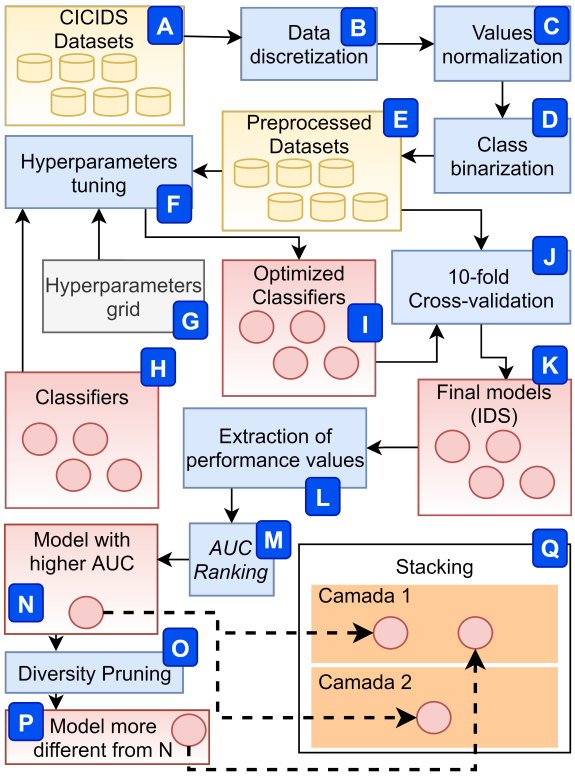


Figure 4. Methodology overview. Source: The Author.

The Table 2 organizes the list of the best Ensembles created through Stacking. Combining two or more classifiers in layer one and a meta-classifier in layer two, 44 models were created. The exhaustion approach method was quite laborious and brought a high computational cost. The models listed in Table 2 reflect the best committees in terms of accuracy for each type of attack:

Table 2. Architecture of the best Stackings created by the exhaustion method. Source: The Author.

Attack	Committee ID	Layer 1	Layer 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

In order to summarize the methodological process adopted in this work for the design of the IDS models, Figure 4 illustrates all the steps taken to obtain the Stackings and compile the results. Therefore, the letters (from A to N) in Figure 4 represent each process described in the sequence:

A - It was obtaining data for training and testing considering the raw data, still without discretization, normalization, and binarization of the classes.

B, C, D, and E - Relevant processes for adapting the data to the necessary data input standards for creating classifiers by machine learning algorithms.
These steps consisted of ("B") transforming all data into numbers maintaining the proportionality of features; ("C") adjusting the numerical data in the same range in order to guarantee the non-negative influence of certain specific feature(s); ("D") creation of two classes to guarantee the main objective of an intrusion detector (which is more to detect the attacks than to detect their type) and ("E") representing the pre-processed data already suitable for creating classifier models and tests for extracting performance metrics.

F, G, H, and I - The use of methods to obtain the best hyperparameters allow the creation of more adequate classification models in terms of performance in detection rates. It was a repetition process where several hyperparameters were tested for the different classifiers.

J, K and L - Training and testing process using 10-fold cross validation ("J") and, consequently, creation of models ("K") for future extraction of performance metrics ("L").

M and N - In order to suggest the best committee of classifiers for each of the six types of attack ("N"), the Stacking that presented the best performance ("M").

4. Results

In order to organize the results obtained, the Accuracy indices, Precision, Recall, and F1-Score was calculated. The absolute values of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) obtained by using a Confusion Matrix were used for the calculations.

Table 3 presents the best results obtained by the exhaustion method. The IDs of the Stackings are based on the Table 2 therefore, they represent the performance of each committee concerning each type of attack present in the Dataset object of this study:

Table 3. Compilation of results for the best Stackings created. Source: The Author.

Stacking #ID	Accuracy	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

As a way of observing the improvement in performance in the sense of reducing the number of incorrect classifications, the Table 4 lists, in percentage, for all individual classifiers and for each Stacking created the error rates of classification:

Table 4. Compiled with all individual error rates and the Stacking chosen by applying the combination proposed by the author organized according to the attacks. Source: Elaborated by the author.

Classifier	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%

It is clear in Table 4 that the Stackings created obtained, for all the attacks that they proposed to detect, lower error rates compared to the individual classifiers. The difference is greater when compared to the MLP and SVM classifiers, being minimal concerning k-NN and DT. In addition, Table 5 lists a comparison of the results obtained by applying the present study with related works cited in Section 2.1:

Table 5. Comparison between the results of the present study and related works. Source: The Author.

Approach	Accuracy	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			
Our work	**0.9992**	**0.999**	**0.9995**	**0.9995**

For studies that proposed to detect various types of attacks, such as the present study, the average of the results obtained was calculated for comparison purposes. Not all works brought information about Accuracy, Precision, Recall and F1-Score simultaneously; therefore, Table 5 considered only the values reported in related works. It is noteworthy that, as mentioned above in the Related Works Section, only [Tama et al. 2020] used the same dataset used in this research. However, for a more comprehensive comparison, it was decided not to use only a single study as a comparison.

5. Conclusion and Future Works

The results presented in this work show the performance of the Stacking method. Choosing the best classifiers that will compose a committee exhaustively made it possible to obtain the best combination of algorithms for each type of attack present in the analyzed dataset. The error rate numbers show that applying committees as suggested significantly reduces the number of incorrectly performed classifications. It is noteworthy that, in general terms, opting for the stackings presented concerning the individual implementation of k-NN and DT brings better results but with little significant difference. The most evident gain is in the choice of stackings concerning the individual implementation of the SVM and NN classifiers, evident by the greater difference in the classification error rates.

As a proposal for future work, we intend to implement Ensemble Pruning methods so that it is possible to find a combination of classifiers that is also adequate in terms of

computational cost. In some environments, the exhaustive method may be impracticable, as it depends on the execution of all possibilities. So, theoretically, applying a form of Pruning can be advantageous.

References

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.

4 An Ensemble Pruning Approach to Optimize Intrusion Detection Systems Performance

This work was presented at the IEEE International Conference on Systems, Man, and Cybernetics in 2022[8] evaluated by Brazilian CAPES with A2 Qualis score. It is an experiment that solves the research problem presented in the previous work (Chapter 3). By applying Diversity Pruning, we arrive at the optimal combination of classifiers without using the exhaustion method. This allows the creation of a strong ensemble that is effective in reducing classification errors and in the concern with the computational cost.

We arrived at the logic of creating a stacking where the classifiers of the first layer are a, which obtained the best result for the area under curve precision/recall, and b, which is most different from a. We allocate the classifier a as a meta-classifier in the second layer. All details regarding the method are available in Chapter IV of the work below. The results obtained suggest that:

- method is the best for detecting attacks on web servers and for portscan attacks (compared to individual classifiers);
- the method is superior to 3/4 of the individual classifiers and ties with k-NN in detecting brute force, infiltration, and attacks coming from botnets;
- the method was not effective for DDoS detection, where decision trees were superior in reducing classification errors;
- the computational cost (training and testing) is much lower when the composition of stacks is governed by diversity rather than exhaustion.

Although this experiment solves the problem of obtaining stackings by exhaustion, it leaves the need to expand the studies to test the method in more datasets, which we present in work documented in Chapter 5. The dataset used included six subsets, the six types of attacks discussed in work. The work follows in its original format made available in the sequence to maintain the characteristics of the conference.

[8] IEEE International Conference on Systems, Man, and Cybernetics - https://ieeesmc2022.org/

An Ensemble Pruning Approach to Optimize Intrusion Detection Systems Performance

***Abstract*—Machine learning techniques have achieved promising results in detecting attacks in computer networks, particularly ensemble learning methods, improving individual classifier's performance. This work focuses on building an ensemble of classifiers to minimize the computational cost to some extent. A diversity-driven pruning method was applied to create stackings using a combination of k-Nearest Neighbors, Decision Trees, Support Vector Machines, and Neural Networks, and validated on six differents datasets. An average accuracy of 99.94% and a reduction in the processing time of 97.34% are reported with heterogeneous ensembles, highlighting the robustness of the proposed approach.**

***Index Terms*—stacking, ensemble learning, ensemble pruning, intrusion detection**

I. Introduction

Information can be considered the most outstanding asset in companies in the current economic scenario [7], emphasizing how important it is for business owners to keep their data safe, particularly regarding matters such as confidentiality, integrity, and availability.

The heterogeneity of Internet access concerning the content and anatomy of packets obtainable on the global network makes identifying malicious packets a vital research problem in the academic community. In this regard, various natures of Intrusion Detection Systems (IDS) have been applied to mitigate malicious content by analyzing the "net behavior", which, in turn, aims to cause damage (steal) to information that is not supposed to be publicly available.

Creating committees by ensemble learning algorithms involves applying a set of classification or regression algorithms to solve a specific problem. Their use in the security of computer networks has brought the possibility of automation in detecting malicious traffic or improper behavior on an operating system. While traditional techniques use specific tools for learning, ensemble techniques aim to group several learning techniques simultaneously or sequentially, resulting in significant improvements regarding variance, bias, or prediction. However, for the computational cost is not high, there must be a method for selecting the classifiers that will compose the committee so that the results are more satisfactory than individual algorithms.

This manuscript bestows an approach to creating heterogeneous classifiers committees so that their best combination can be learned. The proposed methodology is validated in the context of network intrusion detection concerning six varieties of attack: Brute-force, Infiltration, DDoS, Portscan, Botnet, and Web. We carried ensemble learning by using stacks of classifiers so that their selection is based on diversity pruning. The approach introduced here provides a relevant contribution by presenting a mode of optimization in the classifiers' selection to compose heterogeneous committees with performance gains in computational cost and classification error reduction solving the problems presented by [13] at the 39th Brazilian Symposium on Computer Networks and Distributed Systems by applying the exhaustion method.

The main contributions of the manuscript are summarized below:

- to foster the scientific community in ensemble learning applied to network intrusion detection; and
- propose an optimized methodology that allows choosing the classifiers that will be part of the committee to reduce the computational cost and the attacks classification errors.

The paper is organized as follows: Section II delivers a systematic review of the literature, and Section III presents a theoretical background concerning Intrusion Detection Systems, ensemble pruning, and stacking of classifiers. Section IV describes the committee modeling and an overview of the experimental methodology. Section V presents the results concerning the accuracy, F1-score, precision, and recall measures, and who compares the results to other similar studies. Finally, Section VI states conclusions and ideas for further investigation.

II. Literature review

This section details a literature review to identify which classifiers and ensemble techniques are more commonly used in intrusion detection. Some tendencies are also reported. The searching process was carried out on relevant computer science and computer networks databases (i.e., IEEE, ACM, MDPI, Springer, and Wiley), seeking terms as "ensemble" and "intrusion" found either in the title or used as keywords. The authors focused on literature published in the period of 2015-2021.

A. Correlated work

A stacking ensemble model was proposed in [19]. The experiments showed that entropy analysis between pairs of features from different subsets could create a more suitable subset for attack classification. The authors used the ISCX 2012 dataset as a basis for the tests and OneR, Conjunctive, and Naive Bayes in a stacking model to create the ensemble. The best result obtained was an F-Score of 92%.

A comparison with three different ensemble techniques (boosting, bagging, and stacking) and four classifiers (Decision tree, Naive Bayes, Multilayer Perceptron, and REPTree) was performed by [4]. The experiments were carried out on the UNSW-NB15 dataset. According to the authors, the best ensemble regarding accuracy was a REPTree stacking followed by a Decision Tree stacking.

An intrusion detection model that combines three classic ensemble techniques was presented by [24]. Approaches such

as bagging, boosting, and stacking are used together with Support Vector Machines (SVM) and a k-nearest neighbors (k-NN) classifier. Tests performed on the NSL-KDD dataset right after generating a subset through Principal Component Analysis have shown an accuracy of 99.41% for regular packets and 93.13% for probe packet identification.

The IDS presented by [12] consists of layering the packets to be classified according to the TCP protocol. For each packet group, binary cascaded classifiers were applied to achieve multiclass classification at the end of the process. The authors performed experiments in the NSL-KDD dataset using SVM, Decision Trees, Bayesian Networks, REPTree, k-NN, BFTree, SimpleCart, and Naive Bayes. The results were superior to bagging, boosting, and stacking, pointing accuracy of 95.33% for DoS attacks detections, 91.14% for probe attacks detection, 55.21% for R2L, and 1.42% for U2R; in addition to an accuracy of 98.02% for benign packet detection.

Intrusion detection models using stackings either in the classification or in the selection of features concerning the NSL-KDD and UNSW-NB15 datasets are reported by [9]. Stacks are composed of SVMs, autoencoders, and Random Forest, with accuracy results 91.7% for NSL-KDD and 91.8% for the UNSW-NB15 dataset. A security-focused Hybrid Intrusion Detection System for IoT was proposed by [11]. The authors implemented a stacking model with a C5 decision tree in the first layer and an SVM with an RBF kernel as a meta-classifier. The results for the two layers (accuracy in individual implementations) were 93.30% and 92.50% for the C5 decision tree and SVM RBF, respectively, while detection with stacking obtained an average accuracy of 99.97%.

Experiments over the UNSW-NB15 dataset to obtain the best meta-classifiers for intrusion detection using stacking were performed by [20]. The authors conducted first-layer classifications using k-NN, Naive Bayes, and Decision Trees. A Meta Decision Tree, a Multi Response Linear Regression, and Multiple Model Trees are used as meta-classifiers for the second layer. Three different subsets were used, and Multiple Model Trees obtained the best results in the second layer with an accuracy of 98.56%.

An IDS specialized in detecting web attacks was presented by [25]. A stacking ensemble was implemented with Random Forest, Gradient Boost Machine, and XGboost instead of conventional classifiers. Four different datasets were tested: CICIDS-2017, HTTP-CSIC-2010, NSL-KDD e UNSW-NB15. According to the authors, the accuracy results were 99.99%, 98.82%, 92.17%, and 92.45% for the aforementioned datasets, respectively.

A comparison between machine learning and deep learning models for network intrusion detection with tests performed in Coburg intrusion detection datasets (CIDDSs) are presented by [26]. The authors also modeled a stacking that combined the best machine learning and the best deep learning methods. The authors highlighted the high computational cost of training deep learning models (with or without stacking). Experiments on the different subsets of CIDDSs (CIDDS-001 External Dataset, CIDDS-001 Internal, CIDDS-002 Dataset, and the combination of Internals CIDDS-001 and CIDDS-002) shoed promising results. Considering the low processing time required by CART, the authors highlight it as the most suitable choice.

A stacking model that combined, in the first layer, a k-NN, Random Forest, C4.5 decision tree, Logistic Regression, and a naive Bayes classifier, while using a combination of k-NN and Random Forest as meta-classifiers are presented by [3]. The proposed ensemble obtained an accuracy of 99.40%.

III. Theoretical backgroung

A. Intrusion Detection Systems

According to [21], an IDS focuses on maintaining the integrity, accessibility, or reliability of electronic data sources or communication networks. The authors carried out a historical overview of the IDS's origin and evolution, naming James Anderson as the first researcher to describe an IDS in the article "Computer Security Threat Monitoring and Surveillance" [2]. Two milestones in IDS history were reached in 1986 and 1993 when Dorothy Denning and Peter Neumann published instruction detection models based on a statistical analysis of network traffic [5].

According to [10], IDS is a software developed to detect attacks originated both from the Internet or the local network, which are potentially able to cause damage to communication networks or systems. The IDS software implement safety countermeasures when they detect anomalies in network traffic or host behavior. The authors emphasize that different approaches have been adopted to enhance the detection rate. Those approaches, in general, encompass machine learning techniques (situations in which the detector learn how to spot anomalies through training), Rule-Based (cases in which the detector has specific attack signatures and compares the real traffic with the attack signatures known), and classic statistical methods (average value calculation, standard deviation, and median)

B. Ensemble Pruning

According to [28], ensemble learning methods consist of applying a group of different classification or regression algorithms to solve a specific problem. Whereas traditional techniques use particular tools to enhance learning, ensemble techniques aim to group several methods simultaneously or sequentially, resulting in significant improvement regarding variance, bias, or prevision.

An ensemble method can be organized in three ways, according to [28], regarding the relationship among individual algorithms: "combining classifiers", "ensembles of weak", and "mixture of experts". The first, "combining classifiers", is the most frequent approach described in the literature, and it is based on the combination of classifiers to enhance the final classification accuracy. The "ensembles of weak" methodology consists of applying soft classification algorithms so that the final classifier reaches better accuracies by extracting the best from each weak classifier. Finally, the latter method, "mixture of experts", is usually applied in neural networks and assumes that a better solution is reached by combining different rules and parameters.

Based on the tendencies obtained from the comprehensive literature review, stacking was chosen as the methodology for the present work. The appropriate choice of the classifiers that will be part of the ensemble is a determining factor

for increasing the model's performance. In [29], the authors emphasized that the choice must take into account the diversity between the methods, which, according to [8], can be obtained through the analysis of an agreement matrix built on pair-wise comparisons.

Different models can be generated from the same data sampling or even from different subsets to create n classifiers that will be part of the ensemble depending on the details of each specific technique. The committee that comprises the pool of classifiers is further pruned, ending up in a more appropriate selection of models.

Pruning can be performed differently, as pointed out by [18]. Reduce-Error Pruning and Kappa Pruning [14], Complementarity Measure and Margin Distance Minimization [15], Orientation Ordering [16] and Boosting-based Ordering [17] are a few examples.

C. Diversity Pruning

In [8] the authors highlight the possibility of obtaining a Q value indicating how different a classifier c_1 is from another classifier c_2 to deal with a specific classification problem. For this purpose, a pair-wise analysis is made where the Equation 1 must be applied:

$$Q_{c_1,c_2} = \frac{b+c}{a+b+c+d} \tag{1}$$

Where:

- Q - Difference value between the c_1 and c_2 classifiers. The bigger, the more different. If $c_1 = c_2$ then $Q = 0$;
- c_1 e c_2 - The two classifiers chosen for the pair-wise analysis;
- a - Represents the absolute number of times that both c_1 and c_2 performed a ranking correctly;
- b - Represents the absolute number of times that c_1 performed a classification correctly and c_2, for the same data, misclassified;
- c - Represents the absolute number of times that c_2 performed a classification correctly and c_1, for the same data, misclassified;
- d - Represents the absolute number of times that both c_1 and c_2 misclassified;

D. Stacking

The ensemble model, known as stacking, was originally proposed by [27]. The technique consists of implementing two layers of classification, the first consisting of n classifiers and only the final classifier. Unlike the bagging and boosting techniques, the classifiers do not necessarily need to be the same. That is, heterogeneous classifiers can form the first layer.

Stacking process as a combination of predictions made by multiple learning methods $(L_1, L_2, ..., L_n)$ generated by multiple classifiers $(l_1, l_2, ..., l_n)$ forming an initial layer [1]. Such classifiers are trained under the same training sample D_{Train}, which contains samples in the format $s_i =< x_1, y_i >$, x_i being the characteristics vector containing values for all D_{Train} features and y_i being the class label to which the vector belongs.

In the first phase, classifiers $l_1, l_2, ..., l_n$ made predictions for a vector x_q. In the second phase, a meta-classifier M makes the final class prediction considering the previous layer's prediction. According to [1], the right classifier's choice is fundamental to enhance the classification performance. Also, using a meta-classifier is only justified when the ensemble performance is superior to the performance of a singular classifier [6]. Thus, during the implementation process, it is necessary to observe which individual classifier performs better when compared to the ensemble performance.

Figure 1 illustrates the stacking algorithmic process.

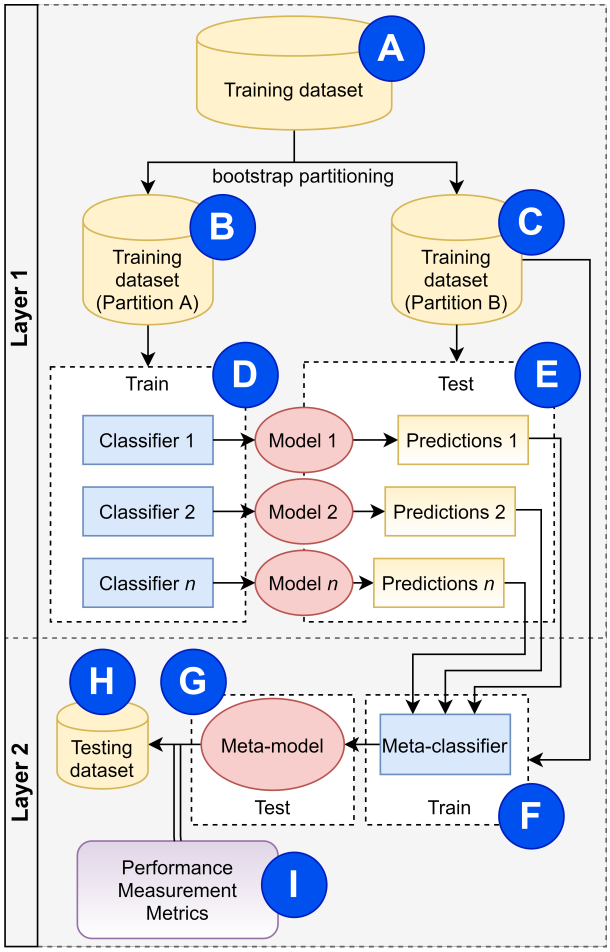


Fig. 1. Stacking working flow.

Consider letters A-I (Figure 1) as a way to analyse the process:

A – Illustrates the training sample, which is part of a hypothetical dataset;

B – After the partitioning stage, a sample of A is found in B;

C – Likewise, a sample of A is found in C;

D – Represents the training process of individual classifiers $\{1, 2, ..., n\}$ aiming to create predictor models $\{1, 2, ..., n\}$;

E – Test stage, in which the predictor models $\{1, 2, ..., n\}$ make predictions using as test sample the second partition obtained in C (C1). The predictions are linked to partition B as new features (C2);

F – Represents the training process of a meta-classifier that is made by using the partition obtained in C (C3), linked to the predictions from C2;

G – Test process of meta-model, in which the predictions are made in the sample tests from a hypothetical dataset;

H – Represents the test sample from a hypothetical dataset; and

I – - Process of analysing the stacking performance, from which factors such as accuracy measurements, precision, recall can be obtained.

Finally, quoting [27], the stacking method can be defined as a way of forwarding information from a group of classifiers to another before reaching the final classification. This process can be useful as it may reduce the error rate in a prediction model.

IV. Metodology

To validate this study's purpose, six different datasets were made available by the Canadian Institute for Cybersecurity via the University of New Brunswick were used. All the datasets are part of a project called CICIDS-2017, published by [22].

The experiments were carried out using the Scikit Learn and MLXtend libraries in the Python language and the training and tests were performed on a Debian Linux server with an octa-core processor (3.2 GHz) and 16 GB of RAM.

According to [23], the group of CICIDS-2017 datasets is divided into eight files. Two files were discarded because the first one presented only benign traffic. The second presented many records, which made it impossible to process by applying the method of exhaustion. Nevertheless, the work with the other six files was able to provide results for comparison. Each dataset contains data for a specific attack, followed by benign traffic data. All data present 78 features with their respective labels on column 79. Although the data have been displayed showing different types of the same attack, they were transformed into binary data to create an Intrusion Detection System that distinguishes legitimate traffic from malignant traffic instead of characterizing the detected attacks.

To summarize the methodological process adopted in this work for the design of the IDS models, Figure 2 illustrates all the steps taken to obtain the stackings and compile the results:

A – Data acquisition for training and testing considering raw data, yet without discretization, normalization, and binarization of classes.

B, C, D, and E – Relevant processes for adapting the data to the necessary data entry standards for creating the classifiers by machine learning algorithms. Such steps consisted of "B" transforming all data into numbers while maintaining the proportionality of the features; "C" to adjust the numerical data in the same range in order to guarantee the non-negative influence of a specific feature(s) in specific; "D" creation of two classes to guarantee the main objective of an intrusion detector (which is more to detect attacks than to detect their type) and "E" representing the pre-processed data already suitable for the creation of classifying models and tests for extracting performance metrics.

F, G, H, and I – The use of methods to obtain the best hyperparameters allows creating more adequate classification models in terms of performance in the detection rates. For the aforementioned values to be found, a list of possible

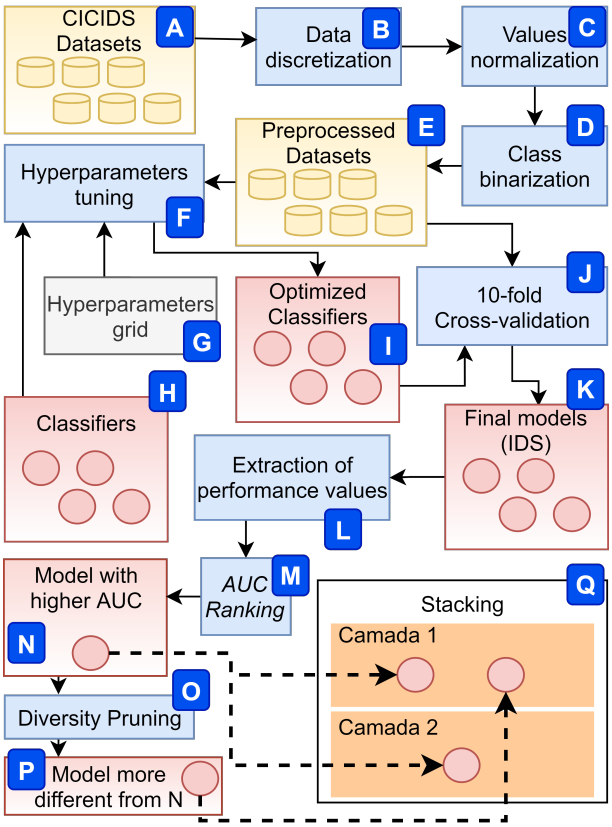


Fig. 2. Overview of the methodological process.

hyperparameters was created "G" and the classifiers "H" were trained/tested employing cross-validation "J" in the "E" datasets. The best performances were adopted for the relationship of best hyperparameters among those tested in order to generate the optimized classifiers "I".

K and L – Each of the four "I" optimized models was applied to each of the six "E" datasets to generate "K" predictive models. It was possible to extract several performance metrics for analysis after testing "L". The metrics generated were: accuracy, area under the precision-recall curve (AUC), precision, recall, F1-score, and the absolute numbers of true positive, false positive, false negative, and true negative.

M, N, O, P, and Q – Application of diversity pruning to create a logic so that the composition of the ensembles was automated, always seeking to create them, observing the reduction of processing time, and increasing performance in the classification. For this, in "M", all the classifiers were ordered, from highest to lowest AUC; in "N" we obtained the best classifier c in terms of AUC; in "O" it was observed in the disagreement measure which the difference scores in the comparison between the classifiers; in "P", when comparing the classifiers c, the highest score was obtained, which indicates the most different combination of classifiers, that is, the process at this point was able to provide the combination of classifiers c, d most different from each other, with c being the model with the highest AUC precision-recall obtained in the steps "M" and "N" and d the most different model from c according to the process "O" observing the Agreement Matrix; Finally, in "Q" stackings were created, following the diversity pruning methodology. In the first

layer, the classifiers c and d were allocated, and in the second layer, the classifier c. The list of created ensembles is detailed in Table I:

TABLE I
ANATOMY OF THE BEST STACKING COMMITTEES CREATED BY THE APPLICATION OF DIVERSITY PRUNING FOR EACH DATASET.

Attack	Layer 1 classifiers	Meta-classifier
Brute-force	k-NN, SVM	k-NN
Infiltration	k-NN, DT	k-NN
DDoS	DT, SVM	DT
Portscan	DT, SVM	DT
Botnet	DT, SVM	DT
Web	k-NN, SVM	k-NN

Following are the results obtained in the tests performed. The ensembles created were compared with other related works, with individual methods and the best possible combinations obtained by an exhaustion method.

V. RESULTS

To be able to attest to the performance of the proposed method based on diversity pruning, some comparisons were made. The first benefit obtained is related to the computational cost. Table II compiles the ratio of time spent to obtain the optimal models (which were obtained through the creation of all possible committees considering the existence of up to four classifiers in the first layer and a single meta- classifier in the second layer) and the models obtained through the proposed pruning methodology.

TABLE II
TIME RATIO (COMPUTATIONAL COST IN SECONDS AND HOURS) TO OBTAIN THE BEST COMMITTEES THROUGH EXHAUSTION, THROUGH DIVERSITY PRUNING AND THE DIFFERENCE (GAIN) BY OUR APPROACH.

Attack (dataset)	Exaustion (seconds)	Exaustion (hours)	Diversity (seconds)	Diversity (hours)	Difference (hours)
Brute-force	265145	73.6	8889	2.4	**–71.18**
Infiltration	62527	17.3	291	0.08	**–17,28**
DDoS	361117	100.3	15761	4.3	**–95.93**
Portscan	415095	115.3	19912	5.5	**–109.77**
Botnet	67429	18.3	684	0.19	**–18.54**
Web	104576	29.04	2272	0.6	**–28.41**

The time difference is noticeable, which, in the best scenarios, showed an improvement of 99.53%, 98.96%, and 97.93% (for Infiltration, Botnet and, Web, respectively), and in the cases with the least difference, they still managed to obtain cost gains. relevant computational data with 96.73%, 95.71% and 95.22% (for Brute-force, DDoS and Portscan).

Table III compiles a comparison of accuracy (Acc.), F1-Score (F1), Precision (Prec.), and Recall (Rec.) for the proposed method with the works cited in the "Correlated Work" section:

It is noteworthy that the work [19] presented only F1-Score results while the works [4], [24], [12] and [20] presented only accuracy data. The work [9] presented data on accuracy, precision and recall.

To observe the performance of our and individual approaches, Table IV was compiled, organizing the calculated accuracy results:

The results of the AUC precision-recall were organized in Table V. The superiority of the proposed method remained at the same levels as the accuracy data:

TABLE III
COMPARISON BETWEEN CORRELATED WORKS AND OUR APPROACH BY PERFORMANCE METRICS THAT WERE PUBLISHED BY THE AUTHORS.

Approach	Accuracy	F1-Score	Precision	Recall
Milliken et al. (2015) [19]	-	0.98	-	-
Belouch et al. (2017) [4]	0.8572	-	-	-
Sun et al. (2018) [24]	0.5727	-	-	-
Lu et al. (2019) [12]	0.7076	-	-	-
Hsu et al. (2019) [9]	0.9175	-	**0.931**	**0.921**
Olasehinde et al. (2020) [20]	0.9856	-	-	-
Tama et al. (2020) [25]	0.9604	-	-	-
Our approach	**0.9994**	**0.999**	**0.9995**	**0.9995**

TABLE IV
PERFORMANCE COMPARISON BETWEEN INDIVIDUAL CLASSIFIERS AND OUR APPROACH BY ACCURACY RESULTS.

Approach	Brute-force	Infiltration	DDoS	Portscan	Botnet	Web
Our	**0.99982**	**0.99991**	0.99937	**0.99970**	**0.99797**	**0.99968**
k-NN	**0.99982**	**0.99991**	0.99914	0.99956	0.99718	**0.99968**
DT	0.99970	0.99988	**0.99946**	0.99969	0.99790	0.99950
SVM	0.99471	0.99986	0.98741	0.99535	0.99522	0.98744
MLP	0.99765	0.99986	0.99675	0.99591	0.99551	0.99839

TABLE V
PERFORMANCE COMPARISON BETWEEN INDIVIDUAL CLASSIFIERS AND OUR APPROACH BY PRECISION x RECALL AUC RESULTS.

Approach	Brute-force	Infiltration	DDoS	Portscan	Botnet	Web
Our	**0.99887**	**0.71052**	0.99935	**0.99971**	**0.95254**	**0.99348**
k-NN	**0.99887**	0.65789	0.99907	0.99954	0.89023	**0.99348**
DT	0.99804	0.65788	**0.99946**	0.99970	0.95151	0.98931
SVM	0.99360	0.5	0.98579	0.99512	0.78091	0.50918
MLP	0.99450	0.5	0.99653	0.99573	0.79603	0.95518

Table VI list the classification error rates between our approach and individual classifiers:

TABLE VI
ERROR RATE COMPARISON BETWEEN INDIVIDUAL CLASSIFIERS AND OUR APPROACH

Approach	Brute-force	Infiltration	DDoS	Portscan	Botnet	Web
Our	**0.018**%	**0.008**%	0.629%	**0.029**%	**0.202**%	**0.032**%
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%

Figures 4 and 5 illustrate the performance relationship by type of attack detected and applied approach:

Afterward, the results obtained are analyzed to present some conclusions about our approach, besides bringing proposals for future research.

VI. CONCLUSIONS

The application of ensemble learning methods has as main objective the improvement of performance in comparison with the use of individual classifiers. However, choosing the classifiers that will compose the committees is not a trivial task. Depending on the choice, the results may be lower and, therefore, would not justify the implementation of stacking methods.

To gauge the best possible combination, all the possibilities of combining the k-NN, SVM, MLP, and DT algorithms were implemented in an exhaustive manner considering the chosen ensemble methodology's limits. However, the exhaustive method presented a high computational cost justifying

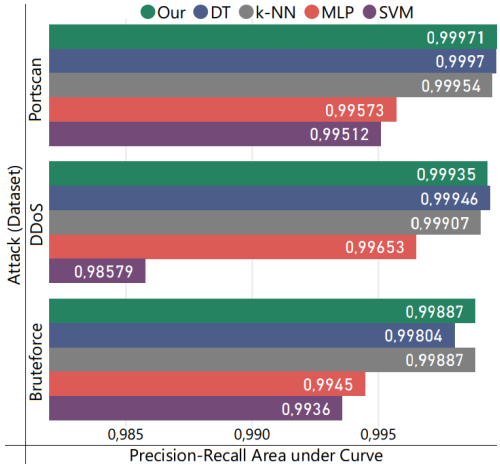


Fig. 3. Bar graph for Brute-force, DDoS and Portscan attacks.

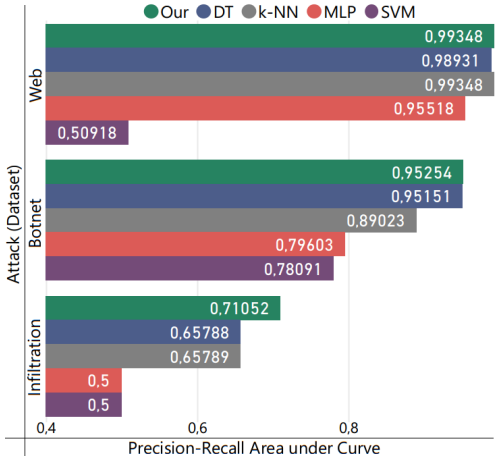


Fig. 4. Bar graph for Infiltration, Botnet and Web attacks.

using a more robust method for choosing classifiers. Given this need to reduce the computational cost and maintain the performance gain characteristic of good ensemble implementations, it was necessary to apply the diversity pruning where the difference between the classifiers was calculated using the disagreement measure.

With the diversity data between the classifiers, it was possible, empirically, to present a pruning method that was detailed in Figure IV. The results obtained by the presented method application demonstrate that in the detection of Infiltration, Portscan, and Botnets attacks, the overall performance was superior to any classifier applied individually. In contrast, for Brute-force and Web attacks, the individual application of k-NN or stacking brings equal results. The proposed method was not effective in detecting DDoS attacks.

Future work is intended to expand the scope of classifiers and datasets, seeking to obtain results in different scenarios that may contribute to the affirmation of the diversity pruning method as a robust possibility for application in detecting attacks on computer networks.

References

[1] 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.

[2] Anderson, J. P. (1980). Computer security threat monitoring and surveillance. Technical Report, James P. Anderson Company.

[3] Aryeh, F. L. and Alese, B. K. (2020). A multi-layer stack ensemble approach to improve intrusion detection system's prediction accuracy. In 2020 15th International Conference for Internet Technology and Secured Transactions (ICITST), pages 1–6. IEEE.

[4] 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.

[5] Denning, D. E. (1987). An intrusion-detection model. IEEE Transactions on software engineering, (2):222–232.

[6] Dzeroski, S. and Zenko, B. (2004). Is combining classifiers with stacking better than selecting the best one? Machine learning, 54(3):255–273.

[7] 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.

[8] Herrera, W. G., Pereira, M., Bento, M., Lapa, A. T., Appenzeller, S., and Rittner, L. (2020). A framework for quality control of corpus callosum segmentation in large-scale studies. Journal of neuroscience methods, 334:108593.

[9] 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.

[10] 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.

[11] Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., and Alazab, A. (2019). A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks. Electronics, 8(11):1210.

[12] 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.

[13] Lucas, T., da Costa, K., Moraes, E., Júnior, P. and das Neves, M. (2021). Stacking-based Committees para Detecção de Ataques em Redes de Computadores - Uma abordagem por exaustão. In XXXIX Brazilian Symposium on Computer Networks and Distributed Systems (pp. 644-657). SBC.

[14] Margineantu, D. D. and Dietterich, T. G. (1997). Pruning adaptive boosting. In ICML, volume 97, pages 211–218. Citeseer.

[15] Martınez-Munoz, G. and Suarez, A. (2004). Aggregation ordering in bagging. In Proc. of the IASTED International Conference on Artificial Intelligence and Applications, pages 258–263. Citeseer.

[16] Martinez-Munoz, G. and Suarez, A. (2006). Pruning in ordered bagging ensembles. In Proceedings of the 23rd international conference on Machine learning, pages 609–616.

[17] Martinez-Munoz, G. and Suarez, A. (2007). Using boosting to prune bagging ensembles. Pattern Recognition Letters, 28(1):156–165.

[18] Martinez-Munoz, G., Hernandez-Lobato, D., and Suarez, A. (2008). An analysis of ensemble pruning techniques based on ordered aggregation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(2):245–259.

[19] 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.

[20] 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.

[21] Osken, S., Yildirim, E. N., Karatas, G., and Cuhaci, L. (2019). Intrusion detection systems with deep learning: A systematic mapping study. In 2019 Scientific Meeting on Electrical-Electronics Biomedical Engineering and Computer Science (EBBT), pages 1–4. IEEE.

[22] 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.

[23] Stiawan, D., Idris, M. Y. B., Bamhdi, A. M. and Budiarto, R. (2020). CICIDS-2017 dataset feature analysis with information gain for anomaly detection. IEEE Access, 8, 132911-132921.

[24] 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.

[25] 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.
[26] Thapa, N., Liu, Z., Kc, D. B., Gokaraju, B., Roy, K., et al. (2020). Comparison of machine learning and deep learning models for network intrusion detection systems. Future Internet, 12(10):167.
[27] Wolpert, D. H. (1992). Stacked generalization. Neural networks, 5(2):241–259.
[28] Zhou, Z.-H. (2012). Ensemble methods: foundations and algorithms. Chapman and Hall/CRC.
[29] Zyblewski, P. and Wozniak, M. (2020). Novel clustering-based pruning algorithms. Pattern Analysis and Applications, pages 1–10.

5 Ensemble Diversity Pruning on Cybersecurity: Optimizing Computational Cost to Build IDS

This work was submitted to the 15th Asian Conference on Intelligent Information and Database Systems in 2023[9] for publication in the Springer journal "Lecture Notes in Computer Science" evaluated by Brazilian CAPES with A4 Qualis score. This comprehensive experiment sought to test diversity pruning in several datasets.

Our parallel work that found the state-of-the-art (Chapter 2) showed that the most used datasets in the area object of this thesis were: CIC-IDS-2017, KDD-CUP'99, NSL-KDD, UNSW-NB15, and ISCX-IDS-2012. As the CIC-IDS-2017 had already been used in a previous experiment (Chapter 4), including being the support for the expansion of the experiments, the present work addressed the remaining datasets (KDD-CUP'99, NSL-KDD, UNSW-NB15, and ISCX-IDS-2012).

The method presented proved to be robust in most of the studied datasets. In the UNSW-NB15 dataset, the decision tree algorithm made fewer errors for false-positive values, and the k-nearest neighbors made fewer errors for false-negative values. However, in the KDD-Cup'99, NSL-KDD, and ISCX-IDS-2012 datasets, the stackings created based on diversity, following the logic presented in Chapter 4, proved to be superior, obtaining reduced error values in comparison with individual classifiers.

The highlight of this work was the extension of the experiment, which had previously been limited to a single dataset, to a broader scenario. Although one of the four datasets studied did not present results that justify the adoption of pruning for diversity, another three reinforced the hypothesis that creating stackings according to this logic brings considerable benefits.

Therefore, in addition to the previous work (Chapter 4), five different datasets were tested, each with its characteristics and network traffic anatomy, and in four of them, the method proved to be satisfactory. This reinforces the need for customizing the intrusion detection system for each network, as the traffic anatomy tends to have its characteristics. The creation of expert systems is a need reinforced by the results of this work.

In order to maintain the characteristics of the conference, the paper is presented below in its original format.

[9] 15th Asian Conference on Intelligent Information and Database Systems- https://aciids.pwr.edu.pl/2023/

Ensemble Diversity Pruning on Cybersecurity: Optimizing Computational Cost to Build IDS

Abstract. Building Intrusion Detection Systems based on Ensemble Learning reduces the misclassification of malicious traffic on computer networks. However, finding a suitable combination of classifiers often requires a high computational cost. This work presents the most comprehensive application of Diversity Pruning that had already been shown to be effective in another work [11], expanding the tests to four other datasets: KDD-CUP'99, NSL-KDD, UNSW-NB15, and ISCX- IDS-2012. The results show a significant computational cost reduction and considerable increases in attack detection rates. Our proposal reduced the classification errors by 18.82%, 26.58%, 22.93%, and 52.34% and the training time by 98.69%, 98.97%, 98.41%, and 98.41% for the datasets mentioned above, respectively.

Keywords: Stacking · Ensemble Pruning · Intrusion Detection Systems

1 Introduction

A cybersecurity report by [4] compiles projections about the global information security scenario, highlighting the main threats and estimates, and suggesting priorities for risk mitigation. According to [4], ahead of risks of legal uncertainty (19%), pandemics (22%), natural disasters (25%) and interruption of services (42%), the risks with information security incidents represent in 2022 a total of 44% for businesses globally.

Projections of [6] estimate a loss of 10.5 trillion dollars in 2025. For comparative purposes, in 2015, it was 3 trillion dollars, and in 2021, 6 trillion. Investment in cybersecurity will maintain an annual level of US$ 1.75 trillion annually until 2025.

The statistics justify the efforts employed in the elaboration of this research. Also, according to [4], it is worth noting that prioritizing protection from attacks based on people (social engineering), investing in limiting data loss and business interruption. In particular, applying advanced analytics and intelligence technologies for security is essential for the survival of companies in an environment as critical as information security.

Given the context presented, it is essential to apply intelligent methods to detect attacks to prevent damage. Intrusion Detection and Prevention Systems (IDS/IPS) based on Machine Learning can analyze large amounts of data coming from network traffic and, through the training step, create robust models capable of recognizing attack patterns.

Nevertheless, creating an IDS model is a challenge already known and debated in cybersecurity. The challenge is increasingly related to reducing the

computational cost of training and increasing detection results by reducing false-positive and false-negative alarms.

At this point, an area directly related to the final performance in attack detection emerges. Ensemble algorithms stand out for their ability to improve the performance of a classifier-based on Machine Learning; however, they bring with them the need to choose the classifiers that will be used. They are pruned for choosing classifiers used as the final model.

The main idea of this work is the implementation of Diversity Pruning to choose the classifiers that would be part of the Ensemble, with a focus on reducing the computational cost (mainly the training time) and reducing classification errors. The approach introduced provides a relevant contribution by presenting an optimization in the classifiers' selection to compose heterogeneous committees with gains in computational cost and classification error reduction, solving the problems presented by [10].

The main contributions of the manuscript are: (I) to foster the scientific community in Ensemble pruning applied to network intrusion detection with machine learning; and (II) to propose an optimized methodology validated on four datasets that allows choosing the classifiers that will be part of the committee to reduce attacks classification errors and the computational cost.

The paper is organized as follows: Section 2 delivers a systematic review of the literature, and Section 3 presents a theoretical background concerning Diversity Pruning, and Stacking of classifiers. Section 4 describes the committee modeling and an overview of the experimental methodology. Section 5 presents the results concerning the error rates, accuracy, F1-score, precision, and recall measures. Finally, Section 6 states conclusions and proposals for further investigation.

2 Literature Review

This section details a literature review to identify which classifiers and Ensemble techniques are more commonly used in intrusion detection. Some tendencies are also reported. The search was carried out on relevant computer science and computer networks databases (i.e., IEEE, ACM, MDPI, Springer, and Wiley), seeking terms such as "Ensemble" and "intrusion" found either in the title or used as keywords. We select only works that use Stacking as an Ensemble model. The authors focused on the literature published from 2017 to 2021.

2.1 Correlated work

A Stacking model that combined, in the first layer, a k-NN, Random Forest, C4.5 decision tree, Logistic Regression, and a naive Bayes classifier while using a combination of k-NN and Random Forest as meta-classifiers was presented by [2]. The proposed Ensemble obtained an accuracy of 99.40%.

A comparison between machine learning and deep learning models for network intrusion detection with tests performed in the CIDDS datasets was presented by [23]. The authors also modelled a stacking that combined the best

machine learning and the best deep learning methods. The authors highlighted the high computational cost of training deep learning models, with or without Stacking. Considering the low processing time required by Decision Trees, the authors highlight it as the most suitable choice.

An IDS specialized in detecting web attacks was presented by [21]. A Stacking Ensemble was implemented with Random Forest, Gradient Boost Machine, and XGboost instead of conventional classifiers. Four different datasets were tested: CICIDS-2017, HTTP-CSIC-2010, NSL-KDD, e UNSW-NB15. According to the authors, the accuracy results were 99.99%, 98.82%, 92.17%, and 92.45% for the datasets mentioned above, respectively.

Experiments over the UNSW-NB15 dataset to obtain the best meta-classifiers for intrusion detection using Stacking were performed by [18]. The authors used k-NN, Naive Bayes, and Decision Trees to conduct first-layer classifications. A Meta Decision Tree, a Multi Response Linear Regression, and Multiple Model Trees are meta-classifiers for the second layer. Three different subsets were used, and Multiple Model Trees obtained the best results in the second layer with an accuracy of 98.56%.

A security-focused Hybrid Intrusion Detection System for IoT was proposed by [8]. The authors implemented a Stacking model with a C5 decision tree in the first layer and an SVM with an RBF kernel as a meta-classifier. The results for the two layers (accuracy in individual implementations) were 93.30% and 92.50% for the C5 decision tree and SVM RBF, respectively, while detection with Stacking obtained an average accuracy of 99.97%.

The IDS presented by [9] consists of layering the packets to be classified according to the TCP protocol. The authors performed experiments in the NSL-KDD dataset using SVM, Decision Trees, Bayesian Networks, REPTree, k-NN, BFTree, SimpleCart, and Naive Bayes. For each packet group, binary cascaded classifiers were applied to achieve multiclass classification at the end of the process. The results were superior to bagging, boosting, and Stacking, with pointing accuracy of 95.33% for DoS attacks detections, 91.14% for probe attacks detection, 55.21% for R2L, and 1.42% for U2R; in addition to an accuracy of 98.02% for benign packet detection.

An intrusion detection model that combines three Classic Ensemble Techniques was presented by [20]. Approaches such as bagging, boosting, and Stacking are used together with Support Vector Machines (SVM) and a k-nearest neighbors (k-NN) classifier. Tests performed on the NSL-KDD dataset right after generating a subset through Principal Component Analysis have shown an accuracy of 99.41% for regular packets and 93.13% for probe packet identification.

A comparison with three different Ensemble techniques (boosting, bagging, and Stacking) and four classifiers (Decision tree, Naive Bayes, Multilayer Perceptron, and REPTree) was performed by [3]. The experiments were carried out on the UNSW-NB15 dataset. According to the authors, the best Ensemble regarding accuracy was a REPTree Stacking followed by a Decision Tree Stacking.

3 Theoretical Backgroung

3.1 Stacking

The Ensemble model, known as Stacking, was originally proposed by [24]. The technique consists of implementing two layers of classification, the first consisting of n classifiers and only the final classifier. Unlike the bagging and boosting techniques, the classifiers do not necessarily need to be the same. That is, heterogeneous classifiers can form the first layer.

Stacking process as a combination of predictions made by multiple learning methods $(L_1, L_2, ..., L_n)$ generated by multiple classifiers $(l_1, l_2, ..., l_n)$ forming an initial layer [1]. Such classifiers are trained under the same training sample D_{Train}, which contains samples in the format $s_i =< x_1, y_i >$, x_i being the characteristics vector containing values for all D_{Train} features and y_i being the class label to which the vector belongs.

In the first phase, classifiers $l_1, l_2, ..., l_n$ made predictions for a vector x_q. In the second phase, a meta-classifier M makes the final class prediction considering the previous layer's prediction. According to [1], the right classifier's choice is fundamental to enhancing the classification performance. Also, using a meta-classifier is only justified when the Ensemble performance is superior to the performance of a singular classifier [5]. Thus, during the implementation process, it is necessary to observe which individual classifier performs better when compared to the Ensemble performance.

3.2 Ensemble Pruning by Diversity

According to [25], Ensemble learning methods consist of applying a group of different classification or regression algorithms to solve a specific problem. Whereas traditional techniques use particular tools to enhance learning, Ensemble techniques aim to group several methods simultaneously or sequentially, resulting in significant improvement regarding variance, bias, or prevision.

Different models can be generated from the same data sampling or even from different subsets to create n classifiers that will be part of the Ensemble depending on the details of each specific technique. The committee that comprises the pool of classifiers is further pruned, ending up in a more appropriate selection of models.

Pruning can be performed differently, as pointed out by [16]. Reduce-Error Pruning and Kappa Pruning [12], Complementarity Measure and Margin Distance Minimization [13], Orientation Ordering [14] and Boosting-based Ordering [15] are a few examples.

In [7] the authors highlight the possibility of obtaining a Q value indicating how different a classifier c_1 is from another classifier c_2 to deal with a specific classification problem. For this purpose, a pairwise analysis is made where the Equation 1 must be applied

$$Q_{c_1,c_2} = \frac{b+c}{a+b+c+d} \tag{1}$$

where:

- Q – a difference value between c_1 and c_2 classifiers. The bigger, the more different. If $c_1 = c_2$ then $Q = 0$;
- c_1 e c_2 – two classifiers chosen for the pairwise analysis;
- a – represents the absolute number of times that both c_1 and c_2 performed a ranking correctly;
- b – represents the absolute number of times that c_1 performed a classification correctly and c_2, for the same data, misclassified;
- c – represents the absolute number of times that c_2 performed a classification correctly and c_1, for the same data, misclassified;
- d – represents the absolute number of times that both c_1 and c_2 misclassified;

3.3 Datasets

The KDD-Cup'99 dataset is a subset of the DARPA dataset (which contains data from simulated operations from the US Air Force's local area network) widely used for training intrusion detectors in computer networks and made available at the Massachusetts Institute of Technology (MIT) Lincoln Laboratory. It was initially introduced as part of a challenge at the KDD-Cup competition held in 1999.

A problem with the KDD-Cup'99 dataset is that it has a lot of redundant and duplicated data, respectively 78% and 75%. To provide a more robust set, [22] presented the NSL-KDD dataset in 2009, as a derivation of KDD-Cup'99t. The most obvious benefits are eliminating redundant records in the training sample and deleting duplicate records in the testing sample.

The KDD-Cup'99 dataset contains 41 features that organize 4,898,431 simulated connections. The NSL-KDD dataset contains 21 different types of attacks in the training sample while having 37 attacks in the testing sample.

The ISCX-IDS-2012 dataset [19], provided by the Information Security Center of Excellence at the University of New Brunswick contains packages referring to DDoS attacks, HTTP DoS, Infiltration, and SSH brute force. The data is organized into seven subsets whose title is related to the day of the week when the traffic was captured. There are 20 features where about 2% of the traffic corresponds to malicious packets.

The UNSW-NB15 dataset, published by [17], was developed at the School of Engineering and Information Technology at the University of New South Wales at the Australian Defense Force Academy. 100 GB of benign and malicious traffic was captured to deliver ten classes of data: Benign, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms.

The dataset was developed based on a critique of the scientific popularity of the KDD-Cup 1999 and NSL-KDD datasets. The authors [17] highlight that the datasets mentioned above contemplate old attacks in a way that the IDS generated from this data would not be robust enough to detect modern attacks.

4 Methodology

For the application of Diversity Pruning, the experiments were conducted using hardware with the following characteristics: 16 cores of Intel(R) Xeon(R) Bronze 3106 CPU @1.70GHz, 64 GB of RAM, and 2 TB of SSD. The entire process was written in Python using the Scikit-Learn and MLXTend libraries. The methodological process adopted in this work is summarized in Figure 1. The letters present in Figure 1 have the following meaning: The KDD-Cup'99, NSL-KDD, UNSW-NB15, and ISCX-IDS-2012 datasets (A) were discretized (B), normalized in the range $\{0, ..., 1\}$ (C) and its labels were binarized (D). Using the 10-fold cross-validation (F) training and testing method, the classifiers (G) k-NN, MLP, DT, and SVM were trained in order to create the final models (H).

The performance metrics (I) accuracy, Area under the curve Precision/Recall, Precision, Recall, and F1-Score were calculated based on the confusion matrix and then ranked by the AUC value (J). The model with the highest AUC (K) made up the first layer of Stacking and was also added as a Meta-Classifier in the second layer. By applying Diversity Pruning (L), the most different model from the one with the highest AUC was added as a second classifier in the first layer of Stacking and it is tuned to obtain the best hyperparameters (N). Figure 2 illustrates all the obtained diversity values.

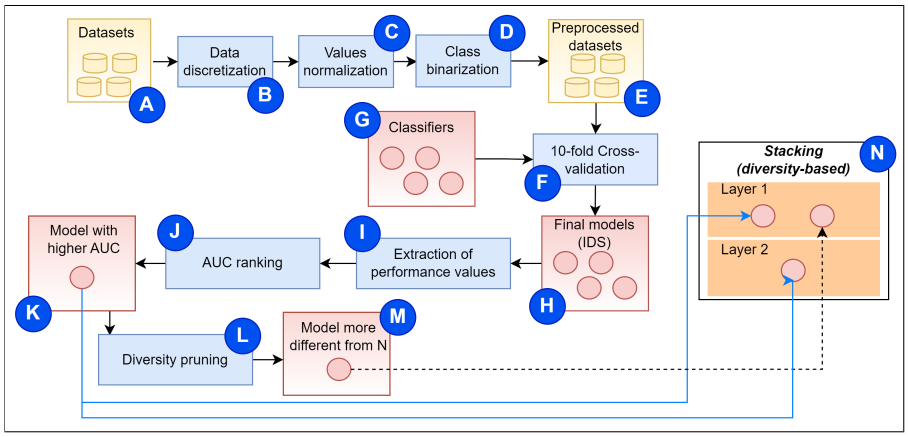


Fig. 1: Workflow of the methodological process.

For the KDD-Cup'99 dataset, the classifier that obtained the highest AUC was k-NN (for which the highest diversity index was SVM with 0.0094); For the NSL-KDD and UNSW-NB15 datasets, the classifier with the highest AUC was DT (for which the highest diversity index was SVM with 0.0077 and 0.2234, respectively); and for the ISCX-IDS-2012 dataset, the classifier with the highest AUC was k-NN (for which the highest diversity index was DT with 0.0376).

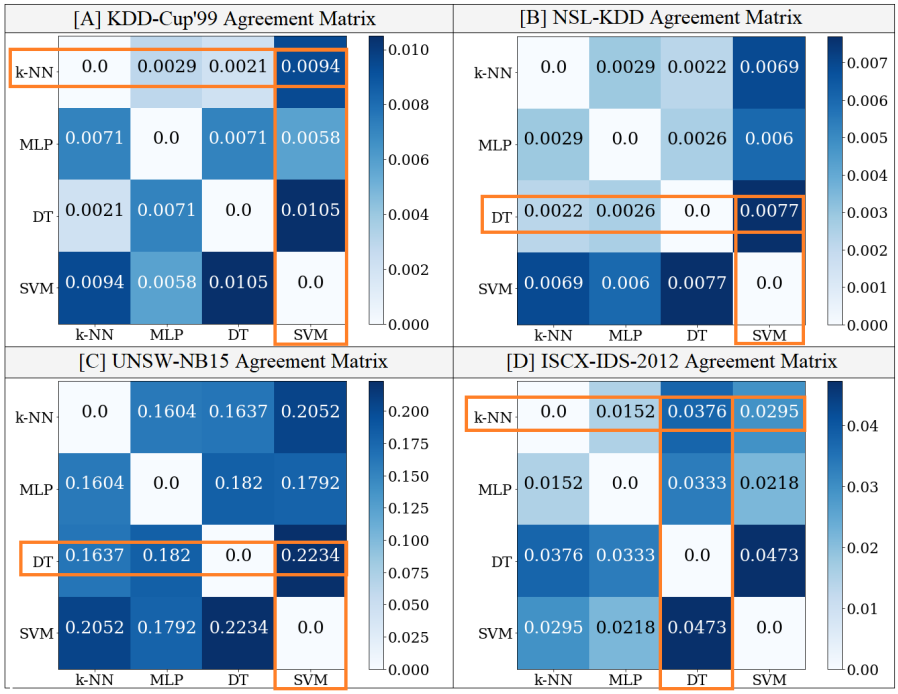


Fig. 2: Agreement matrices of diversity measures per tested dataset for all individual classifiers in a pairwise comparison.

The exhaustion method, described in our previous work [10], presented the problem of high computational cost, although it found the optimal combination of classifiers. Forty-four combinations of classifiers were created, representing all possible hypotheses.

Next, the detailed results for all tested hypotheses are presented and comparisons with the model by exhaustion.

5 Results and discussion

The analysis of the results of this research should be done in two ways: (I) the comparison between the models in the reduction of the classification error rate and (II) the comparison between the hypotheses of an exhaustive approach and the choice of the best Ensemble by the application of the diversity pruning. Figures 3, 4, 5, and 6 show, per each dataset, a comparison between evaluation metrics shown in Figures 3-6.

Table 1 organizes the results of Accuracy, Precision-recall Area under Curve (AUC), Precision, Recall, and F1-Score obtained values for all approaches on all datasets. Except in the UNSW-NB15 dataset (where the recall value was

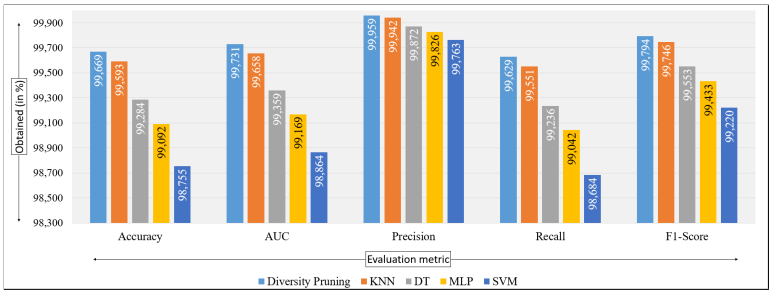


Fig. 3: KDD-Cup'99 – Evaluation metrics obtained values on dataset tests.

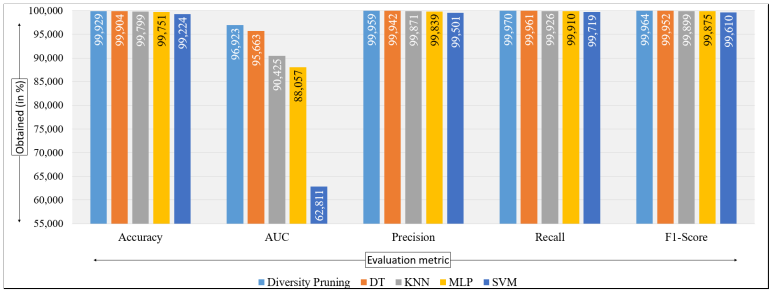


Fig. 4: NSL-KDD – Evaluation metrics obtained values on dataset tests.

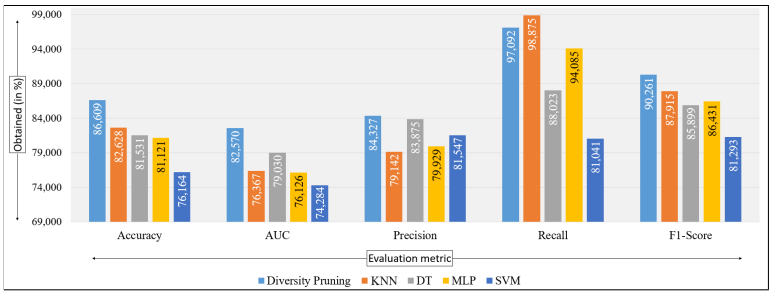


Fig. 5: UNSW-NB15 – Evaluation metrics obtained values on dataset tests.

higher in the k-NN classifier), the classification via Stacking (diversity based) was superior in all other approaches.

Figures 7, 8, 9, and 10 ilustrates the absolute number of False-positive, False-negative, and sum of errors per tested approach.

Table 2 organizes the absolute values of True Negative (TN), False Positive (FP), False Negative (FN), True Positive (TP), and a total of misclassified packets (Sum of errors) obtained on all dataset per approach:

Table 3 organizes the details of computational cost for creating the models by exhaustion compared to Stacking created via diversity pruning.

Table 1: Evaluation metrics obtained values for all approaches on all datasets

Approach	**Accuracy ↓**	**AUC**	**Precision**	**Recall**	**F1-Score**
KDD-Cup'99					
Diversity	**99.669%**	**99.731%**	**99.959%**	**99.629%**	**99.794%**
KNN	99.593%	99.658%	99.942%	99.551%	99.746%
DT	99.284%	99.359%	99.872%	99.236%	99.553%
MLP	99.092%	99.169%	99.826%	99.042%	99.433%
SVM	98.755%	98.864%	99.763%	98.684%	99.220%
NSL-KDD					
Diversity	**99.929%**	**96.923%**	**99.959%**	**99.970%**	**99.964%**
DT	99.904%	95.663%	99.942%	99.961%	99.952%
KNN	99.799%	90.425%	99.871%	99.926%	99.899%
MLP	99.751%	88.057%	99.839%	99.910%	99.875%
SVM	99.224%	62.811%	99.501%	99.719%	99.610%
UNSW-NB15					
Diversity	**86.609%**	**82.570%**	**84.327%**	97.092%	**90.261%**
k-NN	82.628%	76.367%	79.142%	**98.875%**	87.915%
DT	81.531%	79.030%	83.875%	88.023%	85.899%
MLP	81.121%	76.126%	79.929%	94.085%	86.431%
SVM	76.164%	74.284%	81.547%	81.041%	81.293%
ISCX-IDS-2012					
Diversity	**99.483%**	**92.613%**	**99.533%**	**99.935%**	**99.734%**
MLP	98.915%	89.157%	99.325%	99.558%	99.441%
KNN	98.073%	91.170%	99.479%	98.527%	99.001%
SVM	97.039%	81.057%	98.845%	98.091%	98.466%
DT	96.562%	90.627%	99.487%	96.953%	98.203%

Table 2: Absolute evaluation values obtained on all datasets per approach

Approach	**TN**	**FP**	**FN**	**TP**	**Sum of errors ↑**
KDD-Cup'99					
Diversity	**97114**	**162**	**1473**	**395270**	**1635**
KNN	97047	229	1783	394960	2012
DT	96772	504	3032	393711	3536
MLP	96591	685	3800	392943	4485
SVM	96346	930	5222	391521	6152
NSL-KDD					
Diversity	**935**	**61**	**44**	**147476**	**105**
DT	910	86	57	147463	143
KNN	806	190	109	147411	299
MLP	759	237	133	147387	370
SVM	258	738	414	147106	1152
UNSW-NB15					
Diversity	63284	29716	4788	159885	**34504**
k-NN	50089	42911	**1853**	**162820**	44764
DT	**65134**	**27866**	19723	144950	47589
MLP	54096	38904	9741	154932	48645
SVM	62801	30199	31221	133452	61420
ISCX-IDS-2012					
Diversity Pruning	**20585**	**3550**	**491**	**756857**	**4041**
MLP	19008	5127	3351	753997	8478
KNN	20228	3907	11154	746194	15061
SVM	15452	8683	14458	742890	23141
DT	20346	3789	23080	734268	26869

Table 3: Comparison between train and test times for all approaches

Approach	**Train time**	**Test time**	**Train gain**	**Test gain**
KDD-Cup'99				
Exaustion	267,435s	3m21s	–	–
Diversity	3,508s	231 ms	−98.69%	−99.89%
NSL-KDD				
Exaustion	245,908s	2m33s	–	–
Diversity	2,542s	157 ms	−98.97%	−99.91%
UNSW-NB15				
Exaustion	377,119s	6m07s	–	–
Diversity	6,004s	499 ms	−98.41%	−99.87%
ISCX-IDS-2012				
Exaustion	658,546s	7m36s	–	–
Diversity	9,774s	576 ms	−98.41%	−99.86%

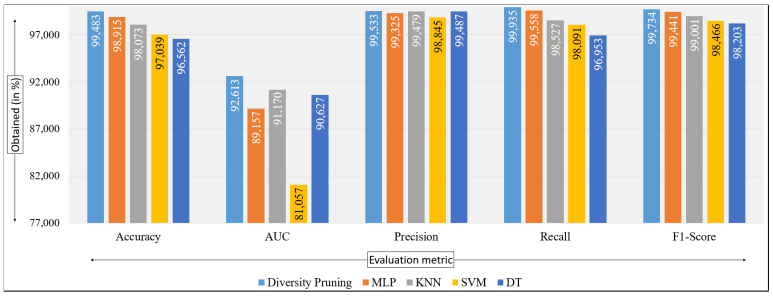


Fig. 6: ISCX-IDS'2012 – Evaluation metrics obtained values on dataset tests.

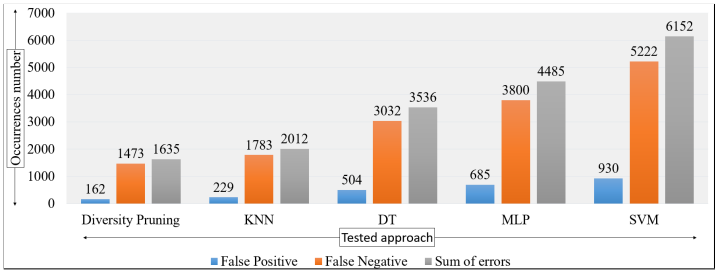


Fig. 7: KDD-Cup'99 – FP and FN values obtained on dataset tests per approach.

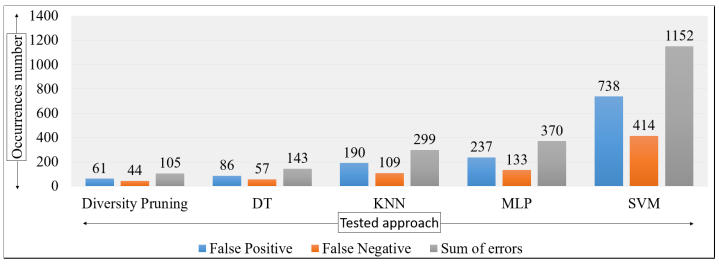


Fig. 8: NSL-KDD – FP and FN values obtained on dataset tests per approach.

The reduction in processing time to find a suitable combination of classifiers is evident. The exhaustion method, which had already been shown to be effective in reducing [10] classification errors, has a very high computational cost that makes its application in computer networks impossible due to the need for low latency in network traffic. More considerations and visions for future research are discussed below.

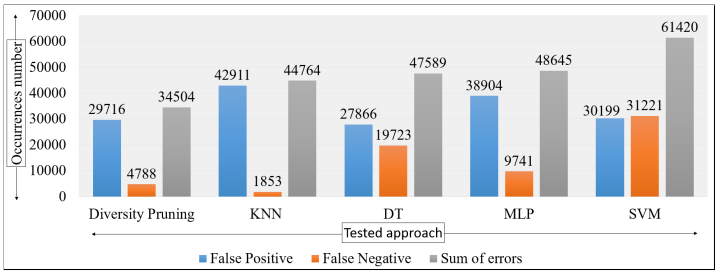


Fig. 9: UNSW-NB15 – FP and FN values obtained on dataset tests per approach.

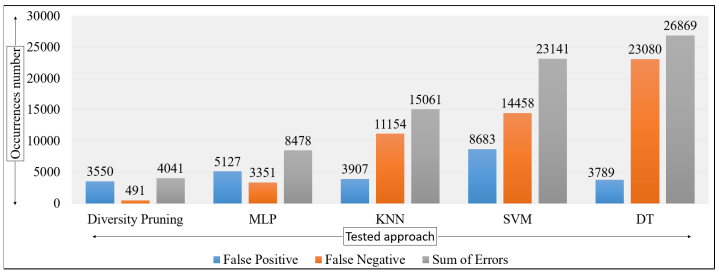


Fig. 10: ISCX-IDS'2012 – FP and FN values obtained per approach.

6 Conclusions

The application of Ensemble techniques tends to deliver meta-classifiers that error less. However, the need to correctly choose the classifiers that will be part of the Ensemble is evident. The wrong choice can lead to an increase in training time and not significantly improve the classification.

An analysis that can be done is the weight given to errors of the False-negative type: in intrusion detection, such errors represent an attack classified as benign, that is, an attacker gained access to the network. In the context of cybersecurity, this represents a more significant problem compared to the reverse situation (False-positives: who are legitimate users labelled as attackers).

False-negative ratings were lower in all scenarios except for the UNSW-NB15 dataset. However, in the same dataset, comparing the model obtained by Diversity with the one that obtained fewer False-negatives (k-NN), there is a reduction of $\approx 30\%$ in False-positive errors. In this sense, the proposed model found a promising path.

Complementing a previous work of ours [11] that was promising in detecting attacks with Stacking obtained via diversity pruning in six subsets of the CICIDS-2017 dataset, this present experiment shows the ability to expand the method in other cybersecurity datasets. In future research, we intend to expand the tests by applying the method in related areas, such as detecting attacks on the Internet of Things and Cloud Computing.

References

1. 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.
2. Aryeh, F. L. and Alese, B. K. (2020). A multi-layer stack Ensemble approach to improve intrusion detection system's prediction accuracy. In 2020 15th International Conference for Internet Technology and Secured Transactions (ICITST), pages 1–6. IEEE.
3. 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.
4. Brooks, C. (2021). Cybersecurity in 2022 – A Fresh Look at Some Very Alarming Stats. Forbes. Retrieved October 11, 2022, from https://cutt.ly/ABvtvvh
5. Dzeroski, S. and Zenko, B. (2004). Is combining classifiers with Stacking better than selecting the best one? Machine learning, 54(3):255–273.
6. Freeze, D. (2021). Cybercrime To Cost The World $10.5 Trillion Annually By 2025. Cybercrime Magazine. Retrieved October 11, 2022, from https://cutt.ly/MBvt3mC
7. Herrera, W. G., Pereira, M., Bento, M., Lapa, A. T., Appenzeller, S., and Rittner, L. (2020). A framework for quality control of corpus callosum segmentation in large-scale studies. Journal of neuroscience methods, 334:108593.
8. Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., and Alazab, A. (2019). A novel Ensemble of hybrid intrusion detection system for detecting internet of things attacks. Electronics, 8(11):1210.
9. 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.
10. Lucas, T. J. and da Costa, K. A. and Moraes, E. A. and Hernandes Júnior, P. R. G., and das Neves, M. J. (2021, August). Stacking-based Committees para Detecção de Ataques em Redes de Computadores-Uma abordagem por exaustão. In Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (pp. 644-657). SBC.
11. Lucas, T. J. and Pontara, K. A. P. and Scherer, R. and Papa, J. P. (2022). An Ensemble Pruning Approach to Optimize Intrusion Detection Systems Performance. IEEE Transactions on Cybernetics.
12. Margineantu, D. D. and Dietterich, T. G. (1997). Pruning adaptive boosting. In ICML, volume 97, pages 211–218. Citeseer.
13. Martınez-Munoz, G. and Suarez, A. (2004). Aggregation ordering in bagging. In Proc. of the IASTED International Conference on Artificial Intelligence and Applications, pages 258–263. Citeseer.
14. Martinez-Munoz, G. and Suarez, A. (2006). Pruning in ordered bagging Ensembles. In Proceedings of the 23rd international conference on Machine learning, pages 609–616.
15. Martinez-Munoz, G. and Suarez, A. (2007). Using boosting to prune bagging Ensembles. Pattern Recognition Letters, 28(1):156–165.
16. Martinez-Munoz, G., Hernandez-Lobato, D., and Suarez, A. (2008). An analysis of Ensemble pruning techniques based on ordered aggregation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(2):245–259.

17. Moustafa, N., Slay, J. (2015, November). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 military communications and information systems conference (MilCIS) (pp. 1-6). IEEE.
18. 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.
19. Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. computers security, 31(3), 357-374.
20. 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.
21. 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.
22. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A. A. (2009, July). A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE symposium on computational intelligence for security and defense applications (pp. 1-6). Ieee.
23. Thapa, N., Liu, Z., Kc, D. B., Gokaraju, B., Roy, K., et al. (2020). Comparison of machine learning and deep learning models for network intrusion detection systems. Future Internet, 12(10):167.
24. Wolpert, D. H. (1992). Stacked generalization. Neural networks, 5(2):241–259.
25. Zhou, Z.-H. (2012). Ensemble methods: foundations and algorithms. Chapman and Hall/CRC.

6 OPFsemble: An Ensemble Pruning Approach via Optimum-Path Forest

This work was accepted in 30th International Conference on Systems, Signals, and Image Processing in 2023[10] evaluated by Brazilian CAPES with A3 Qualis score. The search field is similar, but the method is different. In addition, the tests conducted here did not include cybersecurity datasets.

Here, OPFsemble is presented as an ensemble pruning alternative based on the unsupervised Optimum-Path Forest classifier (PAPA; FALCAO; SUZUKI, 2009), which can select the most representative classifiers, maintaining diversity among them. Five pruning variations are also presented to select the most representative classifiers and combine the final predictions.

Several aggregation models are compared with the proposed approach. The results demonstrate that OPFsemble provides the best scores and even statistical similarity with the baseline set approaches in experiments performed with twelve different datasets.

[10] 30th International Conference on Systems, Signals, and Image Processing - https://www.iwssip.org/

OPFsemble: An Ensemble Pruning Approach via Optimum-Path Forest

***Abstract*—One of the main drawbacks of classification and machine learning algorithms is selecting the learning models that best fit the problem domain. A common approach to tackle this issue comprises ensemble learning, i.e., several different models are employed to solve a given task, and the output consists of a pool of these models' outcomes. Nevertheless, such an approach is computationally costly and demands a strategy to prune similar models and keep the variability in the results. A general solution comprises clustering algorithms, which, on the other hand, usually require prior knowledge of the problem to estimate the number of clusters. This paper proposes the OPFsemble, an Optimum-Path Forest (OPF) ensemble pruning approach that uses the unsupervised OPF to select the most representative classifiers while maintaining diversity. It also proposes five variants of pruning to select the most representative classifiers and combine the final predictions. The proposed approach is compared against several aggregation methods for the ensemble process. Experiments conducted over twelve datasets show the OPFsemble provides the best scores and even statistical similarity with the baseline ensemble approaches.**

***Index Terms*—Ensemble Pruning, Optimum-Path Forest, Ensemble Model, Machine Learning**

I. Introduction

Artificial intelligence and machine learning-based solutions arise as an ever-growing topic of research in different areas of human knowledge that require considerable treatment of massive amounts of data, namely in medical applications [1], remote sensing [2], autonomous vehicles [3], and urban forest monitoring [4]–[7], to cite a few. Machine learning methods are usually governed by the underlying aspects of the examined data, which may imply satisfactory results over specific circumstances. Still, they present an unstable behavior over the same problem considering a different distribution of training/testing instances.

Combining several machine learning models to attack this problem in the so-called ensemble learning approach has been proved to reduce the variance and bias in the context of classification tasks and improve the prediction results [8]. Stacking [9] is one of the most popular ensemble learning strategies, which improves the classification by implementing two classification layers such that the first consists of n classifiers and the second produces the model outcome. The latter is especially interesting since it comprises a meta-learner that aggregates heterogeneous classifiers.

Even though this aggregation promotes diversity and probable better solutions, the method's complexity relies on the complexities of the individual classifiers. In this context, finding the trade-off between the smallest number of classifiers and maximizing the diversity is a challenging task. Recent works [10], [11] propose clustering-based approaches for ensemble pruning to reduce the number of classifiers by gathering the ones that behave similarly and picking a single representant from each group. The main drawback in these solutions regards an intrinsic property from most clustering approaches that usually requires knowing the number of clusters beforehand, which is not the case for this problem.

On the other hand, a graph-based clustering algorithm from the family of the Optimum-Path Forest (OPF) [12], namely Unsupervised Optimum-Path Forest [13], presents itself as a suitable alternative to this problem due to its capability of finding the best number of clusters on-the-fly. Further, the method has been conquering an increasing relevance in the field due to a successful application in several domains, like optimization [14], [15], data imbalance [16], [17], incremental learning [18], and medicine [19], to cite a few.

This paper proposes the OPFsemble, an ensemble pruning strategy that employs the unsupervised OPF to cluster similar models and provide the best solution per cluster, improving the diversity and reducing the modeling complexity. To increase divergence and gather the best-performing models from the clusters, we propose five strategies for the output voting process for further predictions.

Therefore, this paper offers three main contributions:

- To propose a novel classifier ensemble pruning method, the OPFsemble;
- To introduce five variants of voting to select the best-performing classifiers from the clusters;
- To foment the literature regarding the Optimum-Path Forest framework and introduce the technique to the context of ensemble pruning.

The remainder of this paper is presented as follows: Section II provides a theoretical background regarding the Unsupervised Optimum-Path Forest and ensemble learning, while Section III describes the proposed method. Further, the methodologies and results are discussed in Sections IV and V, respectively. Finally, Section VI states conclusions and future works.

II. Background

This section provides the theoretical background regarding supervised and unsupervised Optimum-Path Forest, as well as the main concepts related to classifiers ensemble.

A. Optimum-Path Forest Classifier

The Optimum-Path Forest classifier [12], [20] is a graph-based supervised algorithm that models each dataset training

instance as a node in a graph where each pair of samples is connected through edges weighted by the distance between them, and the Minimum Spanning Tree is performed to optimize the graph pruning nonessential edges. Further, the nodes positioned in the class frontier, i.e., connected to another label's instance, are inserted in the prototype set $\mathcal{P}$. Such prototypes then compete in a conquering-like process to provide each sample with an optimum-path cost. The classification step comprises a search for the training instances capable of offering the optimum-path cost to each test sample.

The unsupervised OPF is a clustering algorithm that, similarly to its supervised version, represents each dataset sample as a node in a graph. Each node is connected to its k-nearest neighbors, whose distance between them weights the linking edge. Besides, each node is also weighted according to a probability density function (pdf), as follows:

$$\rho(\boldsymbol{s}) = \frac{1}{\sqrt{2\pi\theta^2}k} \sum_{\forall \boldsymbol{t} \in \mathcal{A}_k(\boldsymbol{s})} \exp\left(\frac{-d^2(\boldsymbol{s},\boldsymbol{t})}{2\theta^2}\right), \quad (1)$$

where $\mathcal{A}_k(\boldsymbol{s})$ stand for the k-neighborhood of sample $\boldsymbol{s}$, $\theta = \frac{m_d}{3}$, and m_d denotes the maximum distance between any pair of nodes. Notice the value of k is optimized using brute force, such that the optimum value $k^* \in \{1 \leq k_{\max} \leq |\mathcal{T}|\}$, where k_{max} is a hyperparameter, is computed by minimizing the graph cut over the training set $\mathcal{T}$.

Finally, a sample $\boldsymbol{t} \in \mathcal{T}$ is assigned to the path rooted in $\mathcal{P}$ and whose minimum density value along it is maximum, i.e.:

$$\begin{aligned} f_{min}(\langle \boldsymbol{t} \rangle) &= \begin{cases} \rho(\boldsymbol{t}) & \text{if } \boldsymbol{t} \in \mathcal{P} \\ \rho(\boldsymbol{t}) - \delta & \text{otherwise,} \end{cases} \\ f_{min}(\langle \pi_{\boldsymbol{s}} \cdot \langle \boldsymbol{s}, \boldsymbol{t} \rangle \rangle) &= \min\{f_{min}(\pi_{\boldsymbol{s}}), \rho(\boldsymbol{t})\}, \end{aligned} \quad (2)$$

where $\pi_{\boldsymbol{s}}$ is the optimum-path rooted in $\mathcal{P}$ and ending in $\boldsymbol{s}$. Notice that δ denotes a meager constant.

B. Classifier Ensemble

Ensemble machine learning involves applying different classification or regression algorithms to solve a given problem [21]. While traditional techniques build learning based on a specific method, ensemble techniques explore aggregating multiple learning techniques, parallel or sequentially, to improve variance and bias and, consequently, the output results [8]. Such methods can be constructed with classifiers of different types, the so-called "heterogeneous ensembles" or multiple similar classifiers, i.e., the "homogenous ensembles" [22], [23]. The model may execute several machine learning algorithms over a hypothetical set of training samples and evaluate which one obtained the best classification performance. Further, such classifiers are introduced into the "Combination" set, which combines their outcome to provide a final decision. Figure 1 depicts an overview of the concept.

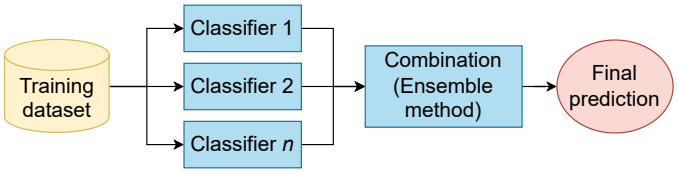


Fig. 1. Overview of an ensemble machine learning pipeline.

III. Proposed Method

The proposed strategy conveys the notion of grouping a blend of different classifiers according to their similar performance in the feature space constructed from predictions in a cross-validation fashion. In this sense, this study aims to use the Unsupervised OPF for grouping and selecting the most relevant classifiers by exploiting the clustering space and the most representative cluster's prototypes.

At first glance, we construct a set of n-diverse classifiers, with n being the number of baseline classifiers to compose the ensemble model. Afterward, the method employs the k-fold cross-validation procedure on the training set $\mathcal{T}$ to get predictions from each ensemble classifier. At each cross-validation fold, the set $\mathcal{T}$ is split into two subsets $\mathcal{T}_1$ and $\mathcal{T}_2$ for training and validation of the baseline classifiers, respectively. The predictions from all sets $\mathcal{T}_2$ are combined and stored in a set $\mathcal{S}$ of n samples and c features, where $c = |\mathcal{T}|$, i.e., the number of samples in $\mathcal{T}$. Hence, each feature stands for the hits and misses over each instance of the validation sets, where 0 stands for a misclassified sample, whereas 1 indicates it is correctly classified. Notice the samples in $\mathcal{T}_2$ are different in each cross-validation fold, meaning there are no repeated samples across all the folds. Also, the average F1 score computed from predictions of the cross-validation folds is assigned to each baseline classifier. The next step involves clustering the set $\mathcal{S}$ via the Unsupervised OPF. Each cluster stands for a group of classifiers derived from similar predictions; the prototype stands for the classifier at the cluster's center.

The test sample prediction assumes the best combination of predictions the selected prototypes perform, i.e., the most representative classifiers. We propose the following strategies to combine the final predictions:

- **Mode:** the most common predicted class among the prototypes;
- **Best mode:** the most common predicted class among predictions from the best-scored classifiers of each cluster;
- **Average:** compute the prediction from the best-scored prototype for each test sample;
- **Mode intercluster:** the most common predicted class among the forecasts from each cluster;
- **Mode intracluster:** the most common predicted class among predictions from the highest-scored cluster, i.e., the cluster with the best-performing baseline classifiers.

Since the size of the predictions' set may increase the computational cost of the unsupervised OPF, we propose reducing the prediction's set dimensionality based on Kullback-Lieber divergences of the baseline classifiers' predictions. This strategy reduces the forecast set to $n \times n$ in size, where n is the number of baseline classifiers defined for the ensemble

method, resulting in the Kullback-Lieber divergence between the pairs of classifiers.

IV. Methodology

A. Datasets

The experiments were performed over 12 datasets from the UCI Machine Learning Repository [24]: Cervical Cancer [25], Diabetic Retinopathy Debrecen (DRD) [26], Forest Type Mapping [27], Glass Identification, Indian Liver, Iris, Mammographic Mass [28], Seismic Bumps [29], Speaker Accent Recognition, Vertebral Column, Wisconsin Diagnostic and Wisconsin Prognostic. All data were processed to fill the missing data with the average value of each feature. Afterward, we scaled the data assuming the Gaussian distribution for standardization.

B. Experimental setup

OPFsemble was compared to similar clustering ensemble pruning approaches presented in Zyblewski and Woźniak [10]. Moreover, we compared the proposal's results against the Super Learner ensemble strategy [30]. Each dataset was split into 20 folds of training, validation, and test sets with proportions of 70%, 15%, and 15%, respectively.

To create the set $\mathcal{S}$, we set $k = 10$ to get predictions from each training set of the 20-fold splits. For the Unsupervised OPF, the k-max value was specified according to the best value obtained after applying the Grid Search optimization method on the set $k_{max} \in [5, 10, 20, 30, 40, 50]$. In this case, we assumed the value that provides the highest F1 score over the validation set.

The OPFsemble is created using 10, 50, and 100 instances of Extra Trees, Gradient Boosting, k-Nearest Neighbors, Multilayer Perceptron Artificial Neural Networks, Naive Bayes, OPF for supervised classification, Random Forest, and Support Vector Machines initialized with different randomized hyperparameters. The same strategy is adopted for the Super Learner ensemble method.

Finally, we employed the Wilcoxon signed-rank test [31] with 5% of significance to assess the statistical similarity among the proposed OPFsemble and the baseline ensemble methods over each dataset. OPFsemble was developed using Python 3.6, and the source code is publicly available in the GitHub repository [1].

V. Results

This section presents the results obtained from the proposed method and the comparison with the baseline algorithms previously described. Notice that the underscored bold values stand for the best-scored results attained by the best-performing models. In contrast, the scores highlighted in bold are the average F1-score that achieved statistical similarity with the best-performing method.

[1] Available at https://github.com/danilojodas/OpfSemble

A. Aggregation methods

Table I presents the average F1-score values obtained by OPFsemble and the aggregation baseline methods presented in Zyblewski and Woźniak [10]. The OPF ensemble achieved the best and similar results in most tested datasets. Considering the cases where only the OPFsemble attained the best performance, we can notice the following aspects of the obtained results: i) Cervical Cancer presented the best scores for 10 and 50 classifiers; ii) in the Glass dataset, the OPFsemble was the best-performing method considering the usage of 10 baseline classifiers and; iii) Indian Liver presented the best results for all OPFsemble approaches when 100 baseline classifiers were used. Even in cases where the proposed method did not achieve the highest average score, the statistical analysis indicates similar results as for the Diagnostic, Mammograph, DDR, and Prognostic datasets. Moreover, the average scores are prone to increase as we increase the number of baseline classifiers inside the ensemble method. This behavior is noticeable for the Cervical Cancer, Glass, and Indian Liver datasets.

B. Super Learner

This section presents the comparison between the OPFsemble and the Super Learner results. Table II shows the average scores obtained by both ensemble methods. In most cases, OPFsemble also attained the highest scores and statistical similarity with the Super Learner ensemble approach. The results may vary depending on the number of baseline classifiers employed in each ensemble technique. Notice the highest scores attained by OPFsemble in the Cervical Cancer dataset when 10 and 100 baseline classifiers are used. The same behavior can be found in other datasets, like Diagnostic, Iris, Mammographic, DDR, Prognostic, Seismic Bumps, and Vertebral Column. However, in most cases, OPFsemble attained statistical similarity with the Super Learner ensemble.

VI. Conclusions

This paper proposed the OPFsemble, an ensemble pruning approach based on the Unsupervised Optimum-Path Forest algorithm. Moreover, five variants were proposed for voting the final prediction from the clusters of classifiers. In most cases, the method attained better results than the Super Learner ensemble and similar techniques for prediction combination. Furthermore, the Wilcoxon signed-rank test revealed that the proposed ensemble approach achieved statistical similarities with the best-performing methods even when it showed smaller average scores. At last, the OPFsemble takes the ability and benefits of the OPF customization capacity in several machine learning tasks with fewer hyperparameters for modeling the data distribution. Regarding future works, we intend to exploit new functionalities for the voting process by using different divergence metrics to optimize the computational cost and increase the prediction's accuracy. Moreover, further experiments in other application domains, like urban tree monitoring and prediction of several urban tree aspects, are also expected in future directions.

TABLE I

AVERAGE F1 SCORE VALUES CONSIDERING THE COMPARISON WITH THE AGGREGATION ENSEMBLE.

Algorithm	Accent	Cervical Cancer	Diagnostic	Forest Types	Glass	Indian Liver	Iris	Mammographic	DRD	Prognostic	Seismic Bumps	Vertebral Column
10 classifiers												
OPFsemble Mode	**0.6599 ± 0.1324**	**0.9423 ± 0.0208**	0.9562 ± 0.0192	0.8630 ± 0.0318	**0.7121 ± 0.0899**	**0.6939 ± 0.0580**	**0.9540 ± 0.0325**	0.7874 ± 0.0457	0.6830 ± 0.0396	**0.7264 ± 0.1249**	**0.9091 ± 0.0198**	**0.8175 ± 0.0510**
OPFsemble Average	**0.6687 ± 0.1614**	0.9431 ± 0.0232	**0.9603 ± 0.0207**	0.8781 ± 0.0372	**0.7353 ± 0.0912**	**0.6777 ± 0.0680**	**0.9608 ± 0.0303**	0.7931 ± 0.0385	0.6787 ± 0.0401	**0.7249 ± 0.1363**	0.9071 ± 0.0210	**0.8219 ± 0.0456**
OPFsemble Intercluster	0.6421 ± 0.1724	**0.9501 ± 0.0202**	0.9613 ± 0.0190	0.8740 ± 0.0350	**0.7258 ± 0.0791**	**0.6829 ± 0.0645**	**0.9540 ± 0.0325**	0.7954 ± 0.0309	0.6945 ± 0.0418	**0.7265 ± 0.1345**	**0.9113 ± 0.0196**	**0.8263 ± 0.0502**
OPFsemble Mode Best	**0.6666 ± 0.1603**	**0.9463 ± 0.0205**	**0.9614 ± 0.0193**	**0.8825 ± 0.0384**	**0.7400 ± 0.0784**	**0.6896 ± 0.0586**	**0.9542 ± 0.0350**	0.7814 ± 0.0333	0.6826 ± 0.0369	0.7172 ± 0.1410	0.9082 ± 0.0207	**0.8291 ± 0.0447**
OPFsemble Intracluster	**0.6717 ± 0.1765**	**0.9489 ± 0.0201**	**0.9637 ± 0.0166**	0.8801 ± 0.0386	**0.7297 ± 0.0720**	**0.6816 ± 0.0649**	**0.9540 ± 0.0325**	0.7984 ± 0.0284	0.6907 ± 0.0436	**0.7165 ± 0.1406**	**0.9092 ± 0.0220**	**0.8267 ± 0.0470**
AggrCART	0.0396 ± 0.0571	0.0171 ± 0.0163	0.0232 ± 0.0151	0.0446 ± 0.0278	0.0468 ± 0.0311	0.1533 ± 0.0432	0.0222 ± 0.0297	0.1111 ± 0.0253	0.1458 ± 0.0289	0.0796 ± 0.0783	0.0180 ± 0.0111	0.1226 ± 0.0558
AggrKNN	0.0359 ± 0.0466	0.0188 ± 0.0170	0.0075 ± 0.0101	0.0602 ± 0.0278	0.0605 ± 0.0279	0.0966 ± 0.0485	0.0171 ± 0.0284	0.1054 ± 0.0237	0.1427 ± 0.0252	0.0874 ± 0.0581	0.0018 ± 0.0029	0.1361 ± 0.0791
AggrMLP	0.0208 ± 0.0323	0.0259 ± 0.0202	0.0131 ± 0.0133	0.0342 ± 0.0219	0.0728 ± 0.0379	0.0281 ± 0.0346	0.0495 ± 0.0584	0.1039 ± 0.0249	0.1146 ± 0.0287	0.0519 ± 0.0454	0.0015 ± 0.0046	0.1024 ± 0.0658
AggrNB	0.0759 ± 0.0710	0.4925 ± 0.2670	0.0262 ± 0.0165	0.0676 ± 0.0290	0.0840 ± 0.0873	0.4140 ± 0.0463	0.0364 ± 0.0415	0.1299 ± 0.0263	0.0478 ± 0.0168	0.2031 ± 0.0823	0.3403 ± 0.2957	0.2299 ± 0.0810
MVCART	**0.6673 ± 0.1413**	0.9355 ± 0.0175	0.9492 ± 0.0217	0.8657 ± 0.0454	0.6711 ± 0.0982	**0.6706 ± 0.0550**	0.9299 ± 0.0507	**0.8138 ± 0.0304**	0.6474 ± 0.0421	0.7063 ± 0.1137	0.9069 ± 0.0213	0.7766 ± 0.0616
MVKNN	0.5072 ± 0.1476	0.9239 ± 0.0276	0.9600 ± 0.0200	0.8598 ± 0.0412	0.6155 ± 0.0750	0.6521 ± 0.0607	**0.9475 ± 0.0452**	**0.8177 ± 0.0336**	0.6598 ± 0.0314	**0.7144 ± 0.1221**	0.9079 ± 0.0205	**0.8207 ± 0.0486**
MVMLP	**0.6568 ± 0.1300**	0.9413 ± 0.0271	**0.9708 ± 0.0172**	**0.8940 ± 0.0326**	0.5940 ± 0.0806	0.6063 ± 0.0483	0.9278 ± 0.0629	**0.8124 ± 0.0327**	**0.7130 ± 0.0354**	**0.7460 ± 0.1149**	0.9073 ± 0.0200	**0.8322 ± 0.0511**
MVNB	0.4157 ± 0.1181	0.6936 ± 0.2688	0.9357 ± 0.0218	0.8539 ± 0.0277	0.3235 ± 0.0875	0.5809 ± 0.0467	0.9363 ± 0.0527	0.7884 ± 0.0300	0.5359 ± 0.0379	**0.7206 ± 0.0812**	0.8846 ± 0.0285	0.7886 ± 0.0533
50 classifiers												
OPFsemble Mode	0.6358 ± 0.1203	**0.9538 ± 0.0165**	0.9654 ± 0.0198	**0.8770 ± 0.0397**	**0.7263 ± 0.0820**	**0.6737 ± 0.0543**	**0.9498 ± 0.0371**	0.7833 ± 0.0375	0.6939 ± 0.0340	**0.7120 ± 0.1301**	**0.9071 ± 0.0209**	0.8103 ± 0.0424
OPFsemble Average	**0.6551 ± 0.1209**	**0.9511 ± 0.0184**	**0.9673 ± 0.0146**	**0.8785 ± 0.0407**	**0.7454 ± 0.0674**	**0.6656 ± 0.0629**	**0.9650 ± 0.0330**	0.8094 ± 0.0383	0.6933 ± 0.0339	0.7135 ± 0.1301	**0.9081 ± 0.0210**	**0.8228 ± 0.0555**
OPFsemble Intercluster	**0.6609 ± 0.1823**	**0.9541 ± 0.0210**	0.9654 ± 0.0165	**0.8805 ± 0.0395**	0.7228 ± 0.0637	**0.6823 ± 0.0443**	**0.9542 ± 0.0351**	0.8030 ± 0.0350	**0.6986 ± 0.0338**	**0.7339 ± 0.1310**	**0.9076 ± 0.0206**	**0.8285 ± 0.0349**
OPFsemble Mode Best	**0.6865 ± 0.1417**	0.9475 ± 0.0207	0.9591 ± 0.0190	**0.8779 ± 0.0376**	**0.7340 ± 0.0748**	**0.6842 ± 0.0641**	**0.9518 ± 0.0458**	0.7816 ± 0.0358	0.6857 ± 0.0317	**0.7415 ± 0.1286**	**0.9075 ± 0.0220**	**0.8272 ± 0.0414**
OPFsemble Intracluster	**0.6652 ± 0.1534**	**0.9486 ± 0.0202**	**0.9702 ± 0.0140**	**0.8831 ± 0.0338**	**0.7339 ± 0.0600**	**0.6686 ± 0.0477**	**0.9649 ± 0.0330**	0.8181 ± 0.0398	0.6974 ± 0.0359	**0.7290 ± 0.1253**	**0.9072 ± 0.0199**	**0.8277 ± 0.0486**
AggrCART	0.0285 ± 0.0429	0.0178 ± 0.0152	0.0226 ± 0.0157	0.0453 ± 0.0226	0.0444 ± 0.0371	0.1498 ± 0.0440	0.0297 ± 0.0361	0.0931 ± 0.0280	0.1541 ± 0.0260	0.0877 ± 0.0917	0.0112 ± 0.0073	0.1482 ± 0.0531
AggrKNN	0.0414 ± 0.0539	0.0174 ± 0.0167	0.0082 ± 0.0111	0.0596 ± 0.0259	0.0515 ± 0.0253	0.0878 ± 0.0407	0.0262 ± 0.0356	0.0898 ± 0.0278	0.1391 ± 0.0236	0.0720 ± 0.0568	0.0010 ± 0.0026	0.1397 ± 0.0870
AggrMLP	0.0233 ± 0.0389	0.0200 ± 0.0171	0.0118 ± 0.0147	0.0348 ± 0.0220	0.0646 ± 0.0371	0.0262 ± 0.0381	0.0549 ± 0.0585	0.0915 ± 0.0263	0.1111 ± 0.0319	0.0461 ± 0.0499	0.0010 ± 0.0035	0.0887 ± 0.0606
AggrNB	0.0817 ± 0.0735	0.4097 ± 0.2546	0.0276 ± 0.0173	0.0697 ± 0.0280	0.0866 ± 0.0835	0.4021 ± 0.0474	0.0375 ± 0.0440	0.1037 ± 0.0238	0.0714 ± 0.0186	0.2106 ± 0.0752	0.3952 ± 0.3606	0.2203 ± 0.0889
MVCART	**0.6752 ± 0.1248**	0.9404 ± 0.0205	0.9545 ± 0.0206	0.8695 ± 0.0425	**0.7268 ± 0.0810**	**0.6778 ± 0.0539**	0.9304 ± 0.0504	**0.8313 ± 0.0345**	0.6621 ± 0.0418	0.7031 ± 0.1306	**0.9064 ± 0.0198**	0.7726 ± 0.0595
MVKNN	0.5073 ± 0.1647	0.9217 ± 0.0270	0.9625 ± 0.0185	0.8595 ± 0.0337	0.6472 ± 0.0891	**0.6610 ± 0.0600**	**0.9395 ± 0.0519**	**0.8243 ± 0.0371**	0.6697 ± 0.0400	0.7126 ± 0.1207	**0.9071 ± 0.0197**	**0.8080 ± 0.0489**
MVMLP	**0.6790 ± 0.1350**	0.9445 ± 0.0252	**0.9726 ± 0.0177**	**0.8895 ± 0.0353**	0.5863 ± 0.0752	0.6015 ± 0.0516	0.9054 ± 0.0645	**0.8297 ± 0.0389**	**0.7149 ± 0.0409**	**0.7486 ± 0.1213**	**0.9072 ± 0.0202**	**0.8174 ± 0.0660**
MVNB	0.4344 ± 0.1189	0.7144 ± 0.2673	0.9380 ± 0.0214	0.8553 ± 0.0285	0.3771 ± 0.0909	0.5934 ± 0.0486	0.9198 ± 0.0534	0.7948 ± 0.0432	0.5596 ± 0.0415	**0.7180 ± 0.0817**	0.6032 ± 0.3981	0.7577 ± 0.0507
100 classifiers												
OPFsemble Mode	**0.6628 ± 0.1638**	**0.9499 ± 0.0236**	0.9631 ± 0.0159	0.8779 ± 0.0362	0.6991 ± 0.0904	**0.6905 ± 0.0584**	**0.9475 ± 0.0329**	0.8057 ± 0.0216	0.6952 ± 0.0355	0.7264 ± 0.1444	**0.9084 ± 0.0210**	**0.8157 ± 0.0542**
OPFsemble Average	**0.6490 ± 0.1232**	0.9254 ± 0.0343	0.9672 ± 0.0172	**0.8914 ± 0.0373**	**0.7498 ± 0.0753**	0.6253 ± 0.0624	**0.9627 ± 0.0320**	**0.8251 ± 0.0387**	0.6949 ± 0.0277	0.7117 ± 0.1376	**0.9071 ± 0.0201**	**0.8178 ± 0.0544**
OPFsemble Intercluster	**0.6654 ± 0.1576**	**0.9483 ± 0.0218**	0.9636 ± 0.0171	0.8788 ± 0.0401	0.7216 ± 0.0716	0.6669 ± 0.0582	**0.9476 ± 0.0454**	0.8056 ± 0.0315	**0.6985 ± 0.0422**	0.7309 ± 0.1391	**0.9096 ± 0.0219**	**0.8232 ± 0.0496**
OPFsemble Mode Best	**0.6493 ± 0.1493**	**0.9555 ± 0.0128**	0.9585 ± 0.0160	0.8690 ± 0.0372	**0.7312 ± 0.0812**	**0.6986 ± 0.0385**	**0.9541 ± 0.0377**	0.7874 ± 0.0349	0.6918 ± 0.0335	0.7263 ± 0.1166	0.8991 ± 0.0194	**0.8188 ± 0.0537**
OPFsemble Intracluster	**0.6493 ± 0.1290**	0.9457 ± 0.0187	0.9666 ± 0.0171	**0.8856 ± 0.0413**	**0.7377 ± 0.0662**	0.6520 ± 0.0677	**0.9627 ± 0.0320**	**0.8264 ± 0.0334**	**0.6984 ± 0.0367**	**0.7216 ± 0.1468**	**0.9076 ± 0.0197**	**0.8272 ± 0.0461**
AggrCART	0.0290 ± 0.0442	0.0206 ± 0.0193	0.0210 ± 0.0162	0.0446 ± 0.0225	0.0440 ± 0.0337	0.1566 ± 0.0470	0.0271 ± 0.0363	0.0925 ± 0.0280	0.1546 ± 0.0296	0.0743 ± 0.0899	0.0099 ± 0.0076	0.1456 ± 0.0486
AggrKNN	0.0390 ± 0.0546	0.0188 ± 0.0171	0.0054 ± 0.0074	0.0592 ± 0.0270	0.0491 ± 0.0215	0.0828 ± 0.0406	0.0194 ± 0.0289	0.0870 ± 0.0284	0.1373 ± 0.0233	0.0751 ± 0.0569	0.0015 ± 0.0033	0.1465 ± 0.0820
AggrMLP	0.0333 ± 0.0406	0.0221 ± 0.0195	0.0111 ± 0.0149	0.0341 ± 0.0208	0.0638 ± 0.0350	0.0260 ± 0.0384	0.0523 ± 0.0572	0.0874 ± 0.0263	0.1133 ± 0.0317	0.0404 ± 0.0443	0.0010 ± 0.0035	0.0955 ± 0.0648
AggrNB	0.0849 ± 0.0714	0.3784 ± 0.2618	0.0256 ± 0.0172	0.0710 ± 0.0295	0.0852 ± 0.0811	0.4021 ± 0.0467	0.0376 ± 0.0440	0.1020 ± 0.0257	0.1059 ± 0.0245	0.2146 ± 0.0824	0.3925 ± 0.3600	0.2252 ± 0.0838
MVCART	**0.6854 ± 0.1384**	0.9455 ± 0.0164	0.9551 ± 0.0197	0.8709 ± 0.0421	**0.7196 ± 0.0760**	0.6752 ± 0.0499	0.9346 ± 0.0507	**0.8348 ± 0.0308**	0.6626 ± 0.0460	0.7103 ± 0.1251	**0.9067 ± 0.0205**	0.7761 ± 0.0539
MVKNN	0.5071 ± 0.1470	0.9236 ± 0.0279	0.9648 ± 0.0175	0.8583 ± 0.0336	0.6601 ± 0.0890	0.6568 ± 0.0585	**0.9395 ± 0.0519**	**0.8289 ± 0.0366**	0.6700 ± 0.0328	0.7033 ± 0.1279	0.9068 ± 0.0200	**0.8039 ± 0.0570**
MVMLP	**0.6793 ± 0.1378**	**0.9458 ± 0.0261**	**0.9749 ± 0.0157**	**0.8874 ± 0.0367**	0.5892 ± 0.0820	0.6070 ± 0.0558	0.9075 ± 0.0627	**0.8283 ± 0.0371**	**0.7108 ± 0.0378**	**0.7610 ± 0.1052**	**0.9074 ± 0.0204**	**0.8269 ± 0.0615**
MVNB	0.4299 ± 0.1243	0.7291 ± 0.2776	0.9380 ± 0.0214	0.8540 ± 0.0297	0.3834 ± 0.0890	0.5953 ± 0.0492	0.9198 ± 0.0534	0.8040 ± 0.0427	0.6083 ± 0.0343	0.7199 ± 0.0754	0.6068 ± 0.3933	0.7622 ± 0.0477

*Aggr stands for aggregation; CART=Classification and Regression Trees; KNN=*k*-Nearest Neighbors; MLP=Multilayer Perceptron; NB=Naive Bayes; DRD=Diabetic Retinopathy Debrecen.

TABLE II

AVERAGE F1 SCORE VALUES CONSIDERING THE COMPARISON WITH THE SUPER LEARNER ENSEMBLE.

Algorithm	Accent	Cervical Cancer	Diagnostic	Forest Types	Glass	Indian Liver	Iris	Mammographic	DRD	Prognostic	Seismic Bumps	Vertebral Column
10 classifiers												
OPFsemble Mode	**0.6599 ± 0.1324**	**0.9423 ± 0.0208**	**0.9562 ± 0.0192**	0.8630 ± 0.0318	**0.7121 ± 0.0899**	**0.6939 ± 0.0580**	**0.9540 ± 0.0325**	0.7874 ± 0.0457	0.6830 ± 0.0396	**0.7264 ± 0.1249**	**0.9091 ± 0.0198**	**0.8175 ± 0.0510**
OPFsemble Average	**0.6687 ± 0.1614**	0.9431 ± 0.0232	**0.9603 ± 0.0207**	0.8781 ± 0.0372	**0.7353 ± 0.0912**	**0.6777 ± 0.0680**	**0.9608 ± 0.0303**	0.7931 ± 0.0385	0.6787 ± 0.0401	**0.7249 ± 0.1363**	0.9071 ± 0.0210	**0.8219 ± 0.0456**
OPFsemble Intercluster	0.6421 ± 0.1724	**0.9501 ± 0.0202**	**0.9613 ± 0.0190**	0.8740 ± 0.0350	**0.7258 ± 0.0791**	**0.6829 ± 0.0645**	**0.9540 ± 0.0325**	0.7954 ± 0.0309	**0.6945 ± 0.0418**	**0.7265 ± 0.1345**	**0.9113 ± 0.0196**	**0.8263 ± 0.0502**
OPFsemble Mode Best	**0.6666 ± 0.1603**	**0.9463 ± 0.0205**	**0.9614 ± 0.0193**	**0.8825 ± 0.0384**	**0.7400 ± 0.0784**	**0.6896 ± 0.0586**	**0.9542 ± 0.0350**	0.7814 ± 0.0333	0.6826 ± 0.0369	**0.7172 ± 0.1410**	0.9082 ± 0.0207	**0.8291 ± 0.0447**
OPFsemble Intracluster	**0.6717 ± 0.1765**	**0.9489 ± 0.0201**	**0.9637 ± 0.0166**	**0.8801 ± 0.0386**	**0.7297 ± 0.0720**	**0.6816 ± 0.0649**	**0.9540 ± 0.0325**	0.7984 ± 0.0284	0.6907 ± 0.0436	**0.7165 ± 0.1406**	**0.9092 ± 0.0220**	**0.8267 ± 0.0470**
Super Learner	0.4580 ± 0.1382	**0.9445 ± 0.0191**	0.9592 ± 0.0186	0.8049 ± 0.0636	0.6317 ± 0.0705	0.6550 ± 0.0708	**0.9584 ± 0.0295**	0.8102 ± 0.0276	**0.6964 ± 0.0357**	**0.7196 ± 0.1415**	0.9073 ± 0.0207	**0.8178 ± 0.0419**
50 classifiers												
OPFsemble Mode	0.6358 ± 0.1203	**0.9538 ± 0.0165**	**0.9654 ± 0.0198**	**0.8770 ± 0.0397**	**0.7263 ± 0.0820**	**0.6737 ± 0.0543**	0.9498 ± 0.0371	0.7833 ± 0.0375	**0.6939 ± 0.0340**	0.7120 ± 0.1301	**0.9071 ± 0.0209**	0.8103 ± 0.0424
OPFsemble Average	**0.6551 ± 0.1209**	**0.9511 ± 0.0184**	**0.9673 ± 0.0146**	**0.8785 ± 0.0407**	**0.7454 ± 0.0674**	**0.6656 ± 0.0629**	**0.9650 ± 0.0330**	**0.8094 ± 0.0383**	**0.6933 ± 0.0339**	0.7135 ± 0.1301	**0.9081 ± 0.0210**	**0.8228 ± 0.0555**
OPFsemble Intercluster	**0.6609 ± 0.1823**	**0.9541 ± 0.0210**	0.9654 ± 0.0165	**0.8805 ± 0.0395**	0.7228 ± 0.0637	**0.6823 ± 0.0443**	0.9542 ± 0.0351	0.8030 ± 0.0350	**0.6986 ± 0.0338**	**0.7339 ± 0.1310**	**0.9076 ± 0.0206**	**0.8285 ± 0.0349**
OPFsemble Mode Best	**0.6865 ± 0.1417**	**0.9475 ± 0.0207**	0.9591 ± 0.0190	**0.8779 ± 0.0376**	**0.7340 ± 0.0748**	**0.6842 ± 0.0641**	**0.9518 ± 0.0458**	0.7816 ± 0.0358	0.6857 ± 0.0317	**0.7415 ± 0.1286**	**0.9075 ± 0.0220**	**0.8272 ± 0.0414**
OPFsemble Intracluster	**0.6652 ± 0.1534**	**0.9486 ± 0.0202**	**0.9702 ± 0.0140**	**0.8831 ± 0.0338**	**0.7339 ± 0.0600**	**0.6686 ± 0.0477**	**0.9649 ± 0.0330**	**0.8181 ± 0.0398**	**0.6974 ± 0.0359**	**0.7290 ± 0.1253**	**0.9072 ± 0.0199**	**0.8277 ± 0.0486**
Super Learner	0.4888 ± 0.1718	**0.9574 ± 0.0154**	**0.9684 ± 0.0174**	0.7978 ± 0.0399	0.6009 ± 0.0826	**0.6601 ± 0.0762**	**0.9670 ± 0.0278**	**0.8156 ± 0.0376**	**0.7005 ± 0.0390**	**0.7572 ± 0.1094**	**0.9075 ± 0.0197**	**0.8215 ± 0.0406**
100 classifiers												
OPFsemble Mode	**0.6628 ± 0.1638**	**0.9499 ± 0.0236**	0.9631 ± 0.0159	0.8779 ± 0.0362	0.6991 ± 0.0904	**0.6905 ± 0.0584**	0.9475 ± 0.0329	0.8057 ± 0.0216	**0.6952 ± 0.0355**	**0.7264 ± 0.1444**	**0.9084 ± 0.0210**	**0.8157 ± 0.0542**
OPFsemble Average	**0.6490 ± 0.1232**	0.9254 ± 0.0343	**0.9672 ± 0.0172**	**0.8914 ± 0.0373**	**0.7498 ± 0.0753**	0.6253 ± 0.0624	**0.9627 ± 0.0320**	**0.8251 ± 0.0387**	**0.6949 ± 0.0277**	**0.7117 ± 0.1376**	**0.9071 ± 0.0201**	**0.8178 ± 0.0544**
OPFsemble Intercluster	**0.6654 ± 0.1576**	**0.9483 ± 0.0218**	0.9636 ± 0.0171	0.8788 ± 0.0401	0.7216 ± 0.0716	0.6669 ± 0.0582	**0.9476 ± 0.0454**	0.8056 ± 0.0315	**0.6985 ± 0.0422**	**0.7309 ± 0.1391**	**0.9096 ± 0.0219**	**0.8232 ± 0.0496**
OPFsemble Mode Best	**0.6493 ± 0.1493**	**0.9555 ± 0.0128**	0.9585 ± 0.0160	0.8690 ± 0.0372	**0.7312 ± 0.0812**	**0.6986 ± 0.0385**	**0.9541 ± 0.0377**	0.7874 ± 0.0349	**0.6918 ± 0.0335**	**0.7263 ± 0.1166**	0.8991 ± 0.0194	**0.8188 ± 0.0537**
OPFsemble Intracluster	**0.6493 ± 0.1290**	0.9457 ± 0.0187	**0.9666 ± 0.0171**	**0.8856 ± 0.0413**	**0.7377 ± 0.0662**	0.6520 ± 0.0677	**0.9627 ± 0.0320**	**0.8264 ± 0.0334**	**0.6984 ± 0.0367**	**0.7216 ± 0.1468**	**0.9076 ± 0.0197**	**0.8272 ± 0.0461**
Super Learner	0.4822 ± 0.1199	**0.9483 ± 0.0249**	**0.9726 ± 0.0197**	0.7968 ± 0.0280	0.5801 ± 0.0980	0.6579 ± 0.0518	**0.9628 ± 0.0289**	**0.8242 ± 0.0425**	**0.6951 ± 0.0319**	**0.7441 ± 0.1155**	**0.9100 ± 0.0199**	**0.8269 ± 0.0404**

REFERENCES

[1] M. Shehab, L. Abualigah, Q. Shambour, M. A. Abu-Hashem, M. K. Y. Shambour, A. I. Alsalibi, and A. H. Gandomi, "Machine learning in medical applications: A review of state-of-the-art methods," *Computers in Biology and Medicine*, vol. 145, p. 105458, 2022.

[2] Z. Li, Y. Wang, N. Zhang, Y. Zhang, Z. Zhao, D. Xu, G. Ben, and Y. Gao, "Deep learning-based object detection techniques for remote sensing images: A survey," *Remote Sensing*, vol. 14, no. 10, p. 2385, 2022.

[3] L.-H. Wen and K.-H. Jo, "Deep learning-based perception systems for autonomous driving: A comprehensive survey," *Neurocomputing*, 2022.

[4] D. S. Jodas, G. d. N. Velasco, R. Araujo de Lima, A. Machado, and J. a. P. Papa, "Deep Learning Semantic Segmentation Models for Detecting the Tree Crown Foliage [Paper presentation]," in *18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP 2023)*. Lisbon, Portugal: SCITEPRESS, February 2023, pp. 1–8.

[5] D. S. Jodas, L. A. Passos, G. D. N. Velasco, M. H. C. Longo, A. R.

Machado, and J. P. Papa, "Multiclass Oversampling via Optimum-Path Forest for Tree Species Classification from Street-view Perspectives," in *2022 35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, vol. 1. IEEE, 2022, pp. 121–126.

[6] D. S. Jodas, T. Yojo, S. Brazolin, G. D. N. Velasco, and J. P. Papa, "Detection of Trees on Street-View Images Using a Convolutional Neural Network," *International Journal of Neural Systems*, vol. 32, no. 01, p. 2150042, 2022.

[7] D. S. Jodas, S. Brazolin, T. Yojo, R. A. de Lima, G. D. N. Velasco, A. R. Machado, and J. P. Papa, "A Deep Learning-based Approach for Tree Trunk Segmentation," in *2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2021, pp. 370–377.

[8] V. Smolyakov, "Ensemble Learning to Improve Machine Learning Results," 2017.

[9] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[10] P. Zyblewski and M. Woźniak, "Novel clustering-based pruning algorithms," *Pattern Analysis and Applications*, vol. 23, no. 3, pp. 1049–1058, 2020.

[11] W. G. Herrera, M. Pereira, M. Bento, A. T. Lapa, S. Appenzeller, and L. Rittner, "A framework for quality control of corpus callosum segmentation in large-scale studies," *Journal of Neuroscience Methods*, vol. 334, p. 108593, 2020.

[12] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki, "Supervised pattern classification based on optimum-path forest," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 120–131, May 2009.

[13] L. M. Rocha, F. A. M. Cappabianco, and A. X. Falcão, "Data clustering as an optimum-path forest problem with applications in image analysis," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 50–68, May 2009.

[14] L. C. S. Afonso, D. Rodrigues, and J. P. Papa, "Nature-inspired optimum-path forest," *Evolutionary Intelligence*, vol. 16, no. 1, pp. 317–328, 2023.

[15] L. C. S. Afonso, L. A. Passos, Passos, and J. P. Papa, "Enhancing Brain Storm Optimization Through Optimum-Path Forest," in *IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, 2018, pp. 000 183–000 188.

[16] L. A. Passos, D. S. Jodas, L. C. Ribeiro, M. Akio, A. N. De Souza, and J. P. Papa, "Handling imbalanced datasets through Optimum-Path Forest," *Knowledge-Based Systems*, vol. 242, p. 108445, 2022.

[17] L. A. Passos, D. S. Jodas, L. C. F. Ribeiro, T. Pinheiro, and J. P. Papa, "O^2PF: Oversampling via Optimum-Path Forest for Breast Cancer Detection," in *IEEE 33th International Symposium on Computer-Based Medical Systems*. IEEE, 2020.

[18] A. S. Iwashita, D. Rodrigues, D. S. Gastaldello, A. N. de Souza, and J. P. Papa, "An incremental Optimum-Path Forest classifier and its application to non-technical losses identification," *Computers and Electrical Engineering*, vol. 95, p. 107389, 2021.

[19] P. B. Ribeiro, L. A. Passos, L. A. Da Silva, K. A. da Costa, J. P. Papa, and R. A. Romero, "Unsupervised breast masses classification through optimum-path forest," in *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*. IEEE, 2015, pp. 238–243.

[20] J. P. Papa, A. X. Falcão, V. H. C. Albuquerque, and J. M. R. S. Tavares, "Efficient supervised optimum-path forest classification for large datasets," *Pattern Recognition*, vol. 45, no. 1, pp. 512–520, Jan 2012.

[21] Z. Zhou, "Ensemble methods: foundations and algorithms." Chapman and Hall/CRC, 2012.

[22] J. C. Harshitha and K. R. Bharathi, "An Advanced Ensemble Directionality Pattern (EDP) based Block Ensemble Neural Network (BENN) Classification Model for Face Recognition System," in *IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, 2022, pp. 1–8.

[23] Y. Wu and L. Liu, "Boosting Deep Ensemble Performance with Hierarchical Pruning," in *IEEE International Conference on Data Mining (ICDM)*, 2021, pp. 1433–1438.

[24] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[25] K. Fernandes, J. S. Cardoso, and J. Fernandes, "Transfer learning with partial observability applied to cervical cancer screening," in *Iberian conference on pattern recognition and image analysis*. Springer, 2017, pp. 243–250.

[26] B. Antal and A. Hajdu, "An ensemble-based system for automatic screening of diabetic retinopathy," *Knowledge-Based Systems*, vol. 60, pp. 20–27, 2014.

[27] B. Johnson, R. Tateishi, and Z. Xie, "Using geographically weighted variables for image classification," *Remote Sensing Letters*, vol. 3, no. 6, pp. 491–499, 2012.

[28] M. Elter, R. Schulz-Wendtland, and T. Wittenberg, "The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process," *Medical Physics*, vol. 34, no. 11, pp. 4164–4172, 2007.

[29] M. Sikora *et al.*, "Application of rule induction algorithms for analysis of data collected by seismic hazard monitoring systems in coal mines," *Archives of Mining Sciences*, vol. 55, no. 1, pp. 91–114, 2010.

[30] J. Van Der Laan Mark *et al.*, "Super learner," *Statistical Applications in Genetics and Molecular Biology*, vol. 6, no. 1, pp. 1–23, 2007.

[31] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

7 Conclusions

Meeting data protection requirements in the fully connected world means maintaining acceptable levels of confidentiality, integrity, availability, and non-repudiation of information, whether it belongs to an individual, a corporation, or a government. This doctoral thesis compiled experiments that dealt with detecting attacks on computer networks and computational systems.

Separating legitimate network traffic from a malicious one is a task that has become increasingly complex due to the growth of devices connected to the Internet, as more devices generate more data and data from different devices becomes complex due to the different characteristics and device goals.

Classification algorithms based on machine learning play a key role in this scenario because they can make a few mistakes in intrusion detection compared to traditional methods of protecting networks and systems (such as firewalls and proxies). At this point, we found the problem that would guide our experiments: combining the ability of machine learning algorithms to make a few mistakes with some method that would make the classification process more agile without significantly interfering with processing time.

In such a vast universe of classifiers, datasets, and ensemble methods, the work was initiated by observing the trends documented in the state-of-the-art that we present in Chapter 2 – here we could observe which are the main classifiers to compose committee creation experiments; the main datasets to test our hypotheses and the main ensemble method to test our pruning proposal.

Our experiments initially demonstrated the difficulty of estimating an optimal composition for a committee by the exhaustive method, documented in Chapter 3. Although it is possible to arrive at an ideal combination for an ensemble, the method is unfeasible because it has a very high computational cost. However, this experiment became relevant because we could observe a logic in the optimal composition: the classifier with the largest area under the curve (precision/recall) combined with its most diverse classifier in the first stacking layer and also added as a meta-classifier (last layer).

Although this logic was tested in six different subsets, all of them were produced by the same laboratory and were part of the same dataset (CIC-IDS-2017). There was then the possibility that this optimal combination was biased for that dataset. In an overview, therefore, we tested the method on ten datasets (six in Chapter 4 and four more in Chapter 5); in nine of the ten datasets, we obtained results that justify the adoption of pruning for diversity as an alternative. Then we extended the tests to the most relevant datasets in the area, obtained by the trends of Chapter 2. The results were satisfactory in three of the four tested datasets.

The benefit of adopting diversity pruning for the composition of a stacking must be evaluated for each network, as packet characteristics tend to vary from scenario to scenario. There are scenarios where making a few more mistakes is more feasible to keep a reduced training time (use classifiers individually). It is up to the sysadmin to decide whether increasing (even very little) the computational cost to reduce classification errors is feasible.

Another point that makes our work relevant is that surveying related works showed us that only three of the one hundred and thirty-eight related works were concerned with adopting some pruning technique.

As future work within this field of research we will seek:

- Perform pruning experiments using unsupervised methods;
- Extend the tests using datasets from related areas such as IoT and Cloud Computing;
- Expand the possibilities of combining classifiers to try to reduce classification errors even more;

The code base for all experiments conducted in this doctoral thesis is available on the author's personal GitHub, which can be accessed at https://github.com/thiagoFatecOurinhos and found in the "python" directory under the title "diversity_pruning_doutorado".

References

ARISTOTLE. *Politics*. [S.l.: s.n.], n.d.

CERT.BR. *Estatísticas do CERT.br – Incidentes*. 2020. <https://www.cert.br/stats/incidentes/>. (Acesso em 23/04/2020).

FORBES. *Cybersecurity in 2022 – A Fresh Look at Some Very Alarming Stats*. 2022. <https://www.forbes.com/sites/chuckbrooks/2022/01/21/cybersecurity-in-2022--a-fresh-look-at-some-very-alarming-stats/?sh=1a08a6246b61>. (Accessed on 07/05/2022).

GALTON, F. Vox populi. Nature Publishing Group, 1907.

KUNCHEVA, L. I.; WHITAKER, C. J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, Springer, v. 51, n. 2, p. 181–207, 2003.

MORGAN, S. *Cybercrime To Cost The World $10.5 Trillion Annually By 2025*. 2020. <https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/>. (Accessed on 07/05/2022).

PAPA, J. P.; FALCAO, A. X.; SUZUKI, C. T. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, Wiley Online Library, v. 19, n. 2, p. 120–131, 2009.

SUROWIECKI, J. *The wisdom of crowds*. [S.l.]: Anchor, 2005.