



UNIVERSIDADE ESTADUAL PAULISTA
“JÚLIO DE MESQUITA FILHO”
Câmpus de São José do Rio Preto

Mateus Alves Bertoni

**APLICAÇÃO DO MÉTODO CONJUNTO
STACKING DO CLASSIFICADOR FLORESTA
DE CAMINHOS ÓTIMOS PARA O
PROBLEMA DE DETECÇÃO DE INTRUSÃO**

São José do Rio Preto
2021

Mateus Alves Bertoni

APLICAÇÃO DO MÉTODO CONJUNTO *STACKING* DO CLASSIFICADOR FLORESTA DE CAMINHOS ÓTIMOS PARA O PROBLEMA DE DETECÇÃO DE INTRUSÃO

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Financiadora: CAPES Nº - 88882.434393/2019-01

Orientador: Prof. Dr. José Remo Ferreira Brega

São José do Rio Preto

2021

B547a

Bertoni, Mateus Alves

Aplicação do método conjunto Stacking do classificador Floresta de Caminhos Ótimos para o problema de detecção de intrusão / Mateus Alves Bertoni. -- São José do Rio Preto, 2021

106 f. : il., tabs.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto

Orientador: José Remo Ferreira Brega

1. Aprendizado do computador. 2. Sistemas de detecção de intrusão (Medidas de segurança). 3. Floresta de caminhos ótimos. 4. Computadores Medidas de segurança. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Mateus Alves Bertoni

APLICAÇÃO DO MÉTODO CONJUNTO *STACKING* DO CLASSIFICADOR FLORESTA DE CAMINHOS ÓTIMOS PARA O PROBLEMA DE DETECÇÃO DE INTRUSÃO

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Financiadora: CAPES Nº - 88882.434393/2019-01

Comissão Examinadora

Prof. Dr. José Remo Ferreira Brega

UNESP - Câmpus de Bauru

Orientador

Prof. Dr. Alan Demétrius Baria Valejo

UFSCar - Departamento de Computação

Prof. Dr. João Paulo Papa

UNESP - Câmpus de Bauru

Bauru

02 de julho de 2021

Este trabalho é dedicado a todos aqueles que, mesmo na escuridão da noite, têm fé no amanhecer.

Agradecimentos

Primeiramente, agradeço meus pais, Anecy e Rogério, por todo suporte e apoio que me deram desde sempre, possibilitando que eu pudesse estar aqui, escrevendo uma dissertação de mestrado em uma das melhores universidades da América Latina. Sem vocês, nada disso seria possível, obrigado por me incentivarem e apoiarem minhas decisões.

Agradeço em especial meus falecidos avós, que embora fossem todos tão diferentes, me ensinaram muito sobre as sutilezas da vida. Se hoje estou aqui, é porque carrego partes deles comigo.

Agradeço também a todos meus amigos que me acompanharam nessa jornada. Foram diversas as vezes em que enfrentei dificuldades e pensei em desistir no meio do caminho, obrigado por terem aliviado as adversidades que confrontei e por me ajudarem a retomar meu trilha. Obrigado, BCC013, Paris 9, 4032, Surfers, Prins, Rep. Alabama e Rep. Tcheca.

Aos professores que tive durante minha graduação e pós-graduação na UNESP, também devo agradecimentos, pois são especiais aqueles que de fato buscam iluminar os alunos e realmente agregar na formação de um cidadão que contribui para um mundo melhor. Agradeço especialmente os professores Humberto, Marar, Andréia e Adriano.

Aproveito também para agradecer meu orientador, prof. Remo, que mesmo não tendo sido meu professor na graduação, e sequer me conhecendo, me deu um voto de confiança e a oportunidade de trabalhar junto a ele, acreditando na minha capacidade de entregar os resultados que esperávamos. Suas conversas foram imprescindíveis e esclarecedoras, obrigado por todo aprendizado desses anos.

Assim, também agradeço meu veterano, amigo e "co-orientador não-oficial", Gustavo, que me ajudou muito durante toda essa jornada, desde meu primeiro dia de UNESP até hoje. Sua capacidade e humildade para transmitir conhecimento é incomparável e posso dizer com toda certeza que, mesmo sem o título, já é um exímio professor. Obrigado pela ideia de utilizar o classificador OPF no trabalho, e todas demais contribuições para a execução da metodologia experimental do mesmo, seus *insights* foram importantíssimos.

Agradeço também ao prof. Papa pelas vezes que me atendeu fora da sala de aula para tirar dúvidas sobre Aprendizado de Máquina e sobre o classificador OPF; inclusive, agradeço-o pela criação deste classificador, que foi o foco principal do trabalho. Também agradeço a Bruna, ex-colega de graduação, por compartilhar comigo o conjunto de dados uneSPY que foi utilizado neste trabalho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, à qual agradeço.

*"No chains around my feet but I'm not free."
(Concrete Jungle, Bob Marley)*

Resumo

Com a expansão tecnológica, a base dos processos de negócios de inúmeras instituições tornou-se a informação, causando certa dependência das organizações perante a computação. Em um cenário onde existem praticamente infinitas ameaças às informações das organizações, a área de Segurança da Informação demonstra-se ser essencial. Para garantir a proteção do ambiente organizacional, diversas medidas de segurança são cabíveis, dentre elas a implementação de controles lógicos. Um dos controles lógicos mais conhecidos e amplamente utilizado é o Sistema de Detecção de Intrusão, que, basicamente, pode ter sua metodologia baseada em assinatura ou anomalia. Esta última é capaz de detectar ameaças completamente desconhecidas, e por isso tem sido extensivamente estudada pela comunidade científica. A fim de aprimorar a performance destes sistemas, técnicas de Aprendizado de Máquina são aplicadas, buscando melhorar as taxas de detecção e diminuir os erros dos mesmos. Dentre essas técnicas, ultimamente os métodos conjuntos têm se destacado na área, produzindo resultados promissores em suas implementações. Este trabalho apresenta a aplicação de um método conjunto de *Stacking* do classificador *Optimum-Path Forest* para o problema de detecção de intrusão, com o intuito de contribuir para a área de Segurança da Informação. Para tal, dois conjuntos de dados foram empregados, um amplamente utilizado pela comunidade científica, NSL-KDD, e outro novo, uneSPY; ambos foram avaliados através de três experimentos distintos, que implementaram diversos classificadores, como Regressor Logístico, *Decision Tree*, *Support Vector Machine*, *Random Forest*, *Optimum-Path Forest*, e foram comparados com os modelos conjuntos *stack* do classificador *Optimum-Path Forest*. Os resultados experimentais forneceram observações interessantes sobre o conjunto de dados uneSPY e demonstraram aptidão dos modelos conjuntos do *Optimum-Path Forest*.

Palavras-chave: Segurança da informação. Sistemas de Detecção de Intrusão. Aprendizado de Máquina. Métodos Conjuntos. *Stacking*. *Optimum-Path Forest*. Detecção de Intrusão. Floresta de Caminhos Ótimos.

Abstract

Machine Learning techniques have been extensively researched in the last years, mainly due to their effectiveness when dealing with recognition classification applications. Typically, one can comprehend using a Machine Learning system to autonomously delegate routines, save human efforts, and produce great insights regarding decision-making tasks. This work introduces and validates a stacking-based ensemble approach using Optimum-Path Forest classifiers to the intrusion detection task. Instead of using only the widely adopted NSL-KDD dataset, this work also employs a new one, called uneSPY; which provides valuable insights concerning new intrusion detection datasets. Both datasets were evaluated under three distinct experiments, which implemented several classifiers, including Logistic Regression, Decision Trees, Support Vector Machines, Random Forests, Optimum-Path Forests, and were compared against Optimum-Path Forest stacking-based ensembles. The experimental results showed an Optimum-Path Forest stacking-based ensemble classification suitability, particularly when considering its ability to generalize large volumes of data while sustaining its performance.

Keywords: Information Security. Intrusion Detection System. Machine Learning. Ensemble Learning. Stacking. Optimum-Path Forest. Intrusion Detection. Ensemble Methods.

Lista de ilustrações

Figura 1 – Geografia dos ataques maliciosos via <i>web</i> , Nov/2019-Out/2020.	19
Figura 2 – Danos financeiros sofridos por ataques às organizações.	20
Figura 3 – Tentativas de conexão por porta de rede sobre tempo, capturadas por um <i>Honeypot</i>	22
Figura 4 – Arquitetura genérica de um SDI baseado em assinatura.	31
Figura 5 – Arquitetura genérica de um SDI baseado em anomalia.	33
Figura 6 – <i>Dataset</i> fictício representado no gráfico	35
Figura 7 – Distribuição da utilização dos classificadores singulares em SDIs ao longo dos anos.	37
Figura 8 – Representação visual dos procedimentos do algoritmo OPF descritos.	41
Figura 9 – Arquitetura genérica de um método conjunto	43
Figura 10 – Algoritmo genérico do método conjunto de <i>Stacking</i>	46
Figura 11 – <i>Pipeline</i> experimental dos modelos <i>Stacking</i>	57
Figura 12 – Exemplo da aplicação de um dos tipos, OPF homogêneo, de modelos de <i>stacking</i> homogêneo referentes ao Experimento 2.	58
Figura 13 – Representação visual da pseudo-implementação de dois modelos, R_4 e T_7 , de <i>stacking</i> heterogêneo referentes ao Experimento 3.	60
Figura 14 – Exemplo de implementação da validação cruzada em 10-dobras no <i>dataset</i>	61
Figura 15 – Exemplos de implementação dos três experimentos realizados	62
Figura 16 – Representação da pseudo-implementação do Experimento 1	69
Figura 17 – Representação das três diferentes categorias implementadas no Experimento 2	70
Figura 18 – Representação da pseudo-implementação de um <i>Stacking</i> homogêneo da categoria 1 do classificador <i>Decision Tree</i> referente ao Experimento 2	71
Figura 19 – Representação da pseudo-implementação de um dos modelos de <i>Stacking</i> heterogêneo do classificador OPF referente ao Experimento 3.	72
Figura 20 – Representação gráfica dos resultados obtidos no Experimento 1 com o <i>dataset</i> <i>uneSPY</i>	74
Figura 21 – Representação gráfica dos resultados obtidos no Experimento 2 com o <i>dataset</i> <i>uneSPY</i>	75
Figura 22 – Representação gráfica dos resultados obtidos no Experimento 2 com o <i>dataset</i> <i>uneSPY</i>	76
Figura 23 – Representação gráfica dos resultados obtidos no Experimento 3 com o <i>dataset</i> <i>uneSPY</i>	77
Figura 24 – Representação gráfica dos resultados obtidos no Experimento 3 com o <i>dataset</i> <i>uneSPY</i>	78
Figura 25 – Representação gráfica dos resultados obtidos no Experimento 1 com o <i>dataset</i> <i>NSL-KDD</i>	79

Figura 26 – Representação gráfica dos resultados obtidos no Experimento 2 com o <i>dataset</i> NSL-KDD	80
Figura 27 – Representação gráfica dos resultados obtidos no Experimento 2 com o <i>dataset</i> NSL-KDD	81
Figura 28 – Representação gráfica dos resultados obtidos no Experimento 3 com o <i>dataset</i> NSL-KDD	82
Figura 29 – Representação gráfica dos resultados obtidos no Experimento 3 com o <i>dataset</i> NSL-KDD	83
Figura 30 – Análise estatística conduzida pelo teste de Friedman com Nemenyi post-hoc.	84

Lista de tabelas

Tabela 1 – Resumo dos artigos referenciados no capítulo.	53
Tabela 2 – Parâmetros utilizados pela ferramenta <i>tcpdump</i>	65
Tabela 3 – Assinaturas de ataques comuns e quantidade de amostras no <i>dataset</i> <i>uneSPY</i>	66
Tabela 4 – Descrição das características do conjunto de dados <i>uneSPY</i>	98
Tabela 5 – Descrição dos tipos IGMP empregados no conjunto de dados <i>uneSPY</i>	99
Tabela 6 – Descrição da codificação das mensagens DHCPv6 empregadas no conjunto de dados <i>uneSPY</i>	99
Tabela 7 – Descrição da codificação dos tipos de pacotes empregados no conjunto de dados <i>uneSPY</i>	100
Tabela 8 – Descrição da codificação dos protocolos empregados no conjunto de dados <i>uneSPY</i>	101
Tabela 9 – Média e desvio padrão dos classificadores singulares treinados no <i>dataset</i> <i>uneSPY</i>	102
Tabela 10 – Média e desvio padrão dos <i>Stackings</i> homogêneos treinados no <i>dataset</i> <i>uneSPY</i>	103
Tabela 11 – Média e desvio padrão dos <i>Stackings</i> heterogêneos treinados no <i>dataset</i> <i>uneSPY</i>	104
Tabela 12 – Média e desvio padrão dos classificadores singulares treinados no <i>dataset</i> NSL-KDD	104
Tabela 13 – Média e desvio padrão dos <i>Stackings</i> homogêneos treinados no <i>dataset</i> NSL-KDD	105
Tabela 14 – Média e desvio padrão dos <i>Stackings</i> heterogêneos treinados no <i>dataset</i> <i>uneSPY</i>	106

Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i>
CDI	Confidencialidade, Integridade e Disponibilidade
DT	<i>Decision Tree</i>
EUA	Estados Unidos da América
FN	<i>False Negative</i>
FP	<i>False Positive</i>
GPU	<i>Graphics Processing Unit</i>
IDS	<i>Intrusion Detection System</i>
IoT	<i>Internet of Things</i>
k-NN	<i>k-Nearest Neighbor</i>
LR	<i>Logistic Regression</i>
LSVM	<i>Linear Support Vector Machine</i>
ML	<i>Machine Learning</i>
MLP	<i>Multilayer Perceptron</i>
MST	<i>Minimum Spanning Tree</i>
OPF	<i>Optimum-Path Forest</i>
PCA	<i>Principal Component Analysis</i>
R2L	<i>Remote to Local</i>
RAM	<i>Random Access Memory</i>
RF	<i>Random Forest</i>
ROC	<i>Receiver Operating Characteristic</i>
SDI	Sistema de detecção de intrusão
SVM	<i>Support Vector Machine</i>

TN	<i>True Negative</i>
TP	<i>True Positive</i>
U2R	<i>User to Root</i>
VOIP	<i>Voice over IP</i>

Sumário

1	INTRODUÇÃO	16
1.1	Hipótese de pesquisa	23
1.2	Objetivos	24
1.2.1	Objetivo geral	24
1.2.2	Objetivos específicos	24
1.3	Estrutura da Dissertação	25
2	FUNDAMENTAÇÃO TEÓRICA	26
2.1	Segurança da Informação	26
2.2	Sistemas de Detecção de Intrusão	29
2.2.1	Baseado em assinaturas	30
2.2.2	Baseado em anomalia	32
2.2.3	Falsos positivos e negativos	33
2.3	Aprendizado de Máquina	34
2.3.1	Classificadores singulares	36
2.3.1.1	<i>Decision tree</i>	36
2.3.1.2	<i>Random Forest</i>	37
2.3.1.3	<i>Naive Bayes</i>	37
2.3.1.4	<i>Artificial Neural Network</i>	37
2.3.1.5	<i>Support Vector Machines</i>	38
2.3.1.6	<i>Logistic Regression</i>	38
2.3.2	Avaliação e comparação	38
2.4	Floresta de Caminhos Ótimos	39
2.4.1	Fase de treinamento	41
2.4.2	Fase de teste	42
2.5	Métodos conjuntos	42
2.5.1	Benefícios da combinação	44
2.6	<i>Stacking</i>	45
2.7	Aplicações dos métodos conjuntos	47
3	TRABALHOS CORRELATOS	50
4	METODOLOGIA	56
4.1	Métricas	61
4.2	Teste estatístico	63
4.3	Conjunto de dados	64

4.3.1	uneSPY	65
4.3.2	NSL-KDD	67
4.4	Experimento 1	68
4.5	Experimento 2	69
4.6	Experimento 3	71
5	RESULTADOS EXPERIMENTAIS	73
5.1	uneSPY	73
5.2	NSL-KDD	75
5.3	Análise geral	76
6	CONCLUSÃO	85
6.1	Direcionamentos para Trabalhos Futuros	86
6.2	Publicações Realizadas	87
	REFERÊNCIAS	88
	 APÊNDICES	 96
	APÊNDICE A – PROPRIEDADES DO DATASET UNESPY	97
	APÊNDICE B – TABELAS DOS RESULTADOS EXPERIMENTAIS	102

1 Introdução

Décadas atrás, eram raros os locais que possuíam computadores à livre disposição dos usuários, se reservando à este luxo apenas algumas universidades de respeito ou poucas organizações renomadas; e, aqueles que os tinham, normalmente dispunham de apenas uma máquina para processar sozinha o trabalho de diversos usuários.

Por haver apenas um computador em operação, seu uso era limitado exclusivamente para aqueles que realmente precisavam utilizá-lo para desempenhar suas funções de ofício e que possuíam o conhecimento técnico necessário para fazê-lo. Logo, não eram todos colaboradores que tinham acesso à máquina e isso fez com que o treinamento sobre como utilizá-las fosse restrito dentro do escopo da organização, por questões de tempo e recursos gastos, inviabilizando o contato da grande maioria dos mesmos com a computação. Atualmente, graças à evolução tecnológica e à queda de preço dos componentes e, consequentemente, dos dispositivos eletrônicos, este cenário pretérito se tornou apenas uma lembrança.

Hoje em dia, a maioria das organizações possui um número abundante de computadores realizando diversas tarefas simultâneas e distribuindo informações por todo ambiente de produção, substituindo o contexto antigo onde computadores trabalhavam isoladamente (TANENBAUM; FEAMSTER; WETHERALL, 2020). A partir do momento em que começaram a existir múltiplas máquinas em um mesmo local de trabalho, nasceu a necessidade de haver diálogo entre as mesmas, de maneira que elas pudessem conectar-se e se comunicar para compartilhar recursos e informações, formando redes de computadores. Logo, o desafio enfrentado pelas organizações de haver uma única máquina para processar as tarefas de *n* usuários passou a ser efetivar a intercomunicação e compartilhamento de recursos para diversos computadores. O foco então transformou-se em converter programas, dispositivos e, principalmente, dados para que estivessem disponíveis a todos que utilizavam a rede de computadores; esse novo objetivo possibilitou um modelo no qual a localização física da máquina, recurso ou usuário fosse indiferente para que a conexão e comunicação pudesse ser atingida com sucesso.

É possível observar que, mais do que qualquer outro recurso, o compartilhamento de informação tornou-se essencial atualmente, apresentando vantagens competitivas para as organizações que devidamente aplicam metodologias adequadas de Tecnologia da Informação (SCHUBERT; LEIMSTOLL, 2007). Hoje, as organizações possuem inúmeros dados sendo coletados, armazenados e processados diariamente, tanto *in loco* quanto *online*, e necessitam da comunicação dos computadores para garantir que as informações sejam devidamente transmitidas e que o ambiente permaneça em produção ininterrupta.

A informação tornou-se a base do modelo de negócio de diversas organizações, que utilizam-na para refinar seus processos e obter vantagens competitivas no mercado (KRAEMER;

DEDRICK; YAMASHIRO, 2000). Além da perspectiva do *business*, a integração da informação no meio corporativo, juntamente da adoção em massa da computação, promoveu melhorias e aumento na produtividade interna das organizações (BRYNJOLFSSON; HITT, 2003). Um exemplo prático da forte relação que as organizações hoje têm com a tecnologia, informação e comunicação é o cenário do trabalho remoto, definido como um arranjo de trabalho flexível onde colaboradores executam suas tarefas remotamente dos escritórios, não tendo contato físico com os colaboradores de lá, mas sendo capaz de comunicar-se através da tecnologia (MARTINO; WIRTH, 1990). Analisando o contexto da pandemia do *COVID-19*, este exemplo consegue caracterizar muito bem quão vitais são a informação, tecnologia e comunicação para as organizações, uma vez que não somente este modelo de trabalho seria completamente inviável sem os três elementos citados, mas eles também possibilitam manter os ambientes em constante produção e, em alguns casos, até melhorar a eficiência dos colaboradores (WANG et al., 2021).

Estas inovações dos modelos de negócio e de trabalho que são notórias atualmente somente foram praticáveis graças às redes de computadores, que possibilitam a conexão e comunicação das máquinas, proporcionando o tráfego da informação e o compartilhamento de recursos. No entanto, este mesmo sistema que propicia avanços e inovações também foi o responsável pela informação atingir patamares que jamais seriam imaginados décadas atrás; em 2012, estimava-se que aproximadamente 2.5 *Exabytes* de dados eram gerados diariamente (MCAFEE et al., 2012), e, seguindo essa mesma progressão, a estimativa era que o volume de dados alcançasse 40 *Zettabytes* em 2020 (KUNE et al., 2016). Isso significa que, todos os dias, uma quantidade massiva de informação é gerada, processada e transmitida entre inúmeros dispositivos eletrônicos; e essa quantia contempla desde dados completamente banais, como um usuário que acessa um *site* para consultar o placar de uma partida de futebol, até informações totalmente sigilosas, como dados internos confidenciais de uma organização ou credenciais de um usuário.

Havendo tamanha diversidade no que tange o conteúdo das informações que trafegam ininterruptamente pela rede, é evidente que existem informações que são extremamente sigilosas e que devem ter sua confidencialidade mantida por completo, sendo totalmente inacessíveis à qualquer um que não tenha autorização para acessá-las. Em um cenário utópico, as pessoas somente buscariam acesso às informações que lhes são pertinentes e, além disso, apenas os que fossem de fato autorizados a acessá-las teriam êxito em seus esforços. Porém, a realidade é bem diferente disso pois existem inúmeras pessoas mal-intencionadas procurando incessantemente maneiras de obter acesso à informações reservadas, que não lhe são de direito, e por diversas vezes conseguem concretizar suas investidas maliciosas.

Estas pessoas mal-intencionadas podem tanto ser internas à própria organização em questão como também externas à mesma. Em relação à taxonomia de tais indivíduos, existem diversas nuances que os categorizam em diferentes grupos; no momento, de maneira simplista,

cabe dizer que as tentativas de acesso indevido, provenientes de um ambiente externo, às informações sigilosas de uma organização são denominadas invasões e, conseqüentemente, seus agentes são chamados de invasores (STALLINGS, 2010). Estes invasores empregam várias técnicas, procedimentos e *softwares* específicos, que estão em constante evolução, para tentar adquirir acesso aos sistemas e suas informações.

Considerando a existência de ameaças às informações confidenciais das organizações, tornam-se necessários métodos de proteção desses dados. Assim, é feito um trabalho extenso por parte dos defensores para que as ameaças consigam ser prevenidas e as ferramentas para detectá-las sejam aperfeiçoadas. Há uma área da computação focada exclusivamente em lidar com este tipo de problema, denominada Segurança da Informação, que visa garantir que os riscos atrelados à informação e as medidas de segurança implementadas estão corretamente balanceados, alinhando as necessidades de proteção da informação com os objetivos de negócio da organização (WHITMAN; MATTORD, 2011). Este campo de pesquisa enfrenta muita volubilidade, pois a natureza das ameaças transforma-se constantemente, o que torna a defesa uma tarefa árdua, visto que os procedimentos e as ferramentas utilizadas para proteger os ambientes tornam-se obsoletos periodicamente. É um trabalho feito perante uma metamorfose permanente.

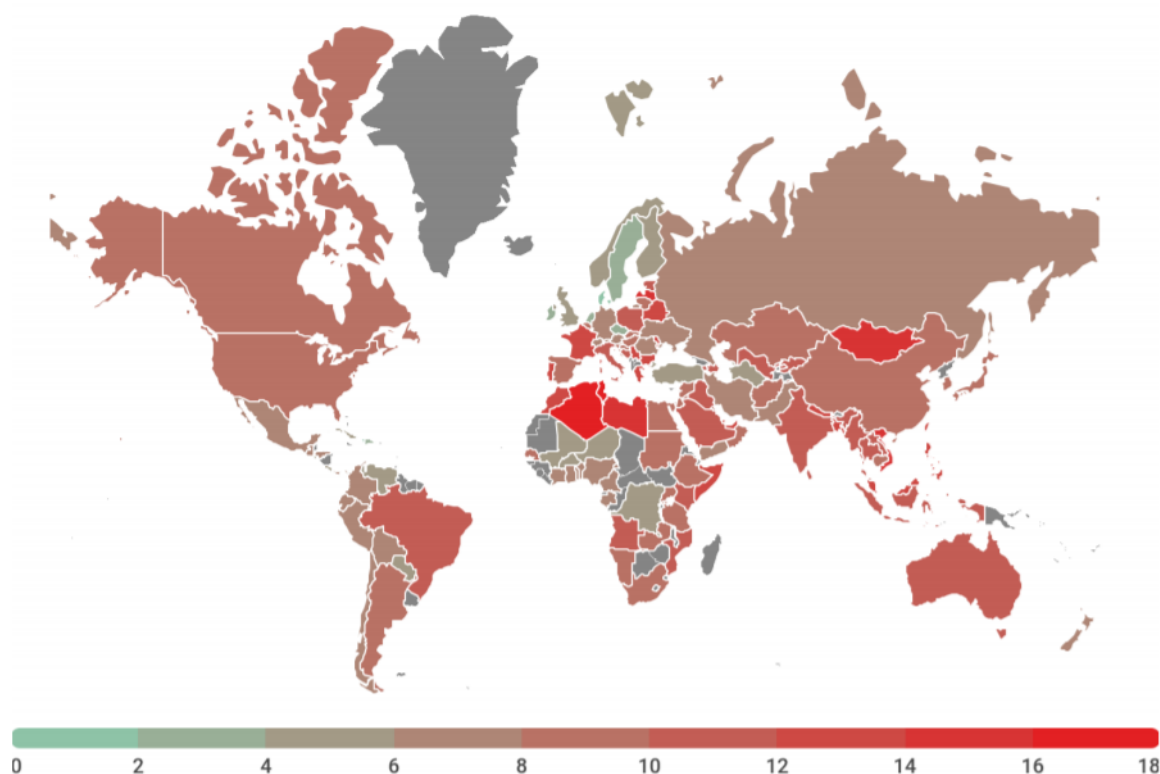
No entanto, mesmo com a existência desta área de pesquisa e a constante busca pelo seu aperfeiçoamento para enfrentar as ameaças, redes, sistemas e dispositivos continuam sofrendo incidentes de segurança frequentemente em escala global, vide Figura 1. Tais eventos indesejados causam danos diretos e indiretos às vítimas, desencadeando diversos efeitos negativos consequentes do incidente, e.g. vazamento de informações confidenciais, desvalorização da imagem da empresa, queda das ações da organização na bolsa de valores, desconfiança dos serviços prestados pela instituição, perda de espaço no mercado, entre muitos outros.

Estes incidentes de segurança costumam resultar em grandes impactos financeiros e causam danos verdadeiros à imagem e reputação das organizações, que podem demorar meses ou até anos para se recuperarem. A Figura 2 mostra que, baseado em um estudo realizado (CISCO, 2018), 53% dos ataques sofridos pelas organizações resultam em danos financeiros de no mínimo quinhentos mil dólares americanos, incluindo a perda de clientes, oportunidades e receitas futuras. Além disso, sabendo dos problemas de segurança enfrentados pelas organizações, atualmente muitos países estão formulando leis que visam garantir a proteção dos dados pessoais dos usuários, como a Lei Geral de Proteção dos Dados brasileira ¹, que aplica multas e sanções rigorosas para as organizações que falharem em protegê-los, aumentando ainda mais os impactos decorrentes de incidentes que afetam a segurança da informação.

É importante salientar que as falhas de segurança, como uma possível violação de dados, não são um problema relevante apenas para os departamentos de segurança e tecnologia

¹ Disponível em <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/L13709.htm>.

Figura 1 – Geografia dos ataques maliciosos via *web*, Nov/2019-Out/2020.



Fonte: *Kaspersky Security Bulletin: Statistics 2020*. (KASPERSKY, 2020)

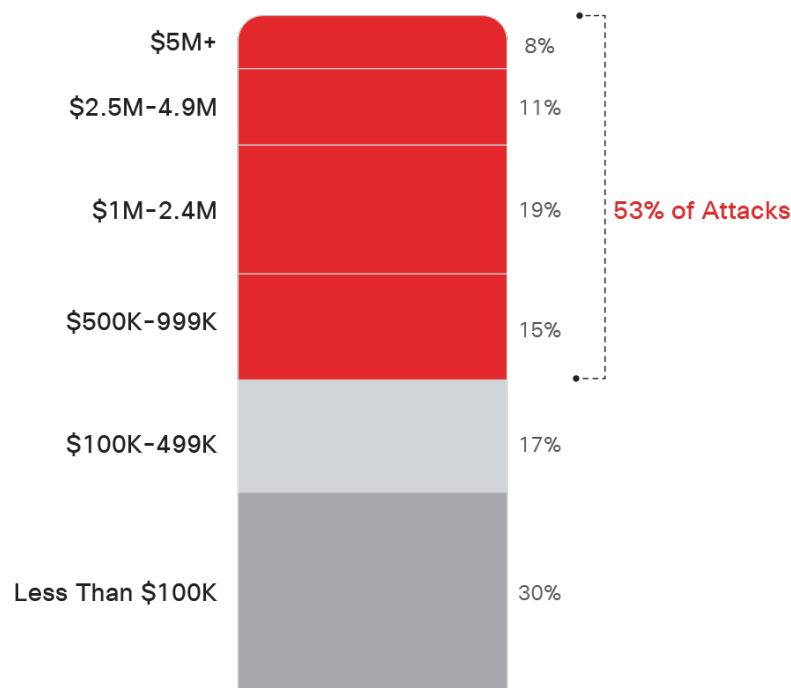
da informação de uma instituição. Pelo contrário, visto todo o impacto atrelado aos mesmos, esses eventos afetam o negócio integralmente e, por isso, a segurança da informação deve ser um assunto ponderado holisticamente pelas diversas áreas de uma organização, desde o nível estratégico até o nível operacional, levando em consideração toda a pirâmide hierárquica organizacional. Porém, por mais que haja a noção de que é vital conscientizar os colaboradores sobre a importância da segurança da informação e proteger-se, os invasores continuam tendo êxito em seus ataques e não é difícil encontrar ocorrências de incidentes de segurança gravíssimos. Nos últimos anos, vários eventos podem ser destacados:

- O prefeito da cidade de Nova Orleans (EUA) foi obrigado a declarar estado de emergência após massivos vazamentos de dados sensíveis e perda de serviços decorrentes de um ataque sofrido ²;
- Em questão de duas semanas, duas cidades no estado da Flórida (EUA) foram vítimas de ataques de *ransomware* e somaram um prejuízo financeiro de 1.1 milhão de dólares americanos ³;

² Disponível em <<https://www.forbes.com/sites/daveywinder/2019/12/14/new-orleans-declares-state-of-emergency-following-cyber-attack>>.

³ Disponível em <<https://www.bbc.com/news/technology-48770128>>.

Figura 2 – Danos financeiros sofridos por ataques às organizações.



Fonte: CISCO 2018, *Annual Cybersecurity Report*. (CISCO, 2018)

- Entre 2019 e 2020, globalmente, 27% das organizações foram impactadas por ataques que envolviam dispositivos móveis e também 67% experienciaram algum incidente de segurança em dispositivos IoT ⁴;
- Um dos maiores bancos americanos, *Capital One*, sofreu com o vazamento de dados de aproximadamente 1 milhão de clientes após uma invasão ⁵; e
- A rede social *Facebook* teve os dados de 533 milhões de usuários expostos, incluindo números de identificação, endereços de e-mail e números de celulares ⁶.

Foi-se o tempo em que investimento em segurança era sinônimo apenas de gastos em cerca-elétrica, câmeras de vídeo ou vigilantes fortemente armados. Hoje em dia, a implementação de soluções que englobam a Segurança da Informação integralmente são essenciais para proteger os ambientes e garantir que, no modelo de negócios atual onde a informação é o carro-chefe da maioria das organizações, riscos desnecessários sejam evitados. Focar apenas na implementação de medidas de controle físico, como aquelas citadas no início do parágrafo, não é o suficiente; é fundamental implementar também medidas de controles lógicos e organizacionais, juntamente às medidas de controles físicos. Os controles lógicos correspondem à instalação de ferramentas

⁴ Disponível em <<https://www.ntsc.org/assets/pdfs/cyber-security-report-2020.pdf>>.

⁵ Disponível em <<https://www.capitalone.com/about/newsroom/capital-one-announces-data-security-incident>>.

⁶ Disponível em <<https://www.forbes.com/sites/kateoflahertyuk/2021/04/06/facebook-data-breach-heres-what-to-do-now>>.

adequadas para lidar com o quesito tecnológico da proteção do ambiente, enquanto os controles organizacionais focam em sanções, políticas, treinamentos e conscientização dos colaboradores para assegurar a segurança da informação (SCHOU; TRIMMER, 2004).

Enfatizando especificamente os controles lógicos de proteção que são utilizados no âmbito da segurança, eles são inúmeros e possuem focos de atuação particulares, diferenciando-se em sua aplicação. Basicamente, alguns focam na prevenção de incidentes, outros especializam-se na detecção de eventos e há também aqueles que visam responder ocorrências, sendo acionados quando os demais falham em suas funções de prevenir e detectar. Assim, uma das ferramentas mais utilizadas na área de Segurança da Informação é o Sistema de Detecção de Intrusão (SDI), do inglês *Intrusion Detection System* (IDS), que é responsável por detectar e notificar quaisquer atividades suspeitas ou tentativas de intrusão na rede, trabalhando em conjunto com as demais ferramentas e compondo a arquitetura geral de segurança do ambiente (NEMATI, 2007); usualmente, o SDI pode ser dividido em duas categorias diferentes, onde a primeira denomina-se baseada em assinaturas e a outra é conhecida como baseada em anomalias, sendo cada uma única em seu propósito e especificidade (BHUYAN; BHATTACHARYYA; KALITA, 2014).

A metodologia baseada em assinatura reconhece ataques através da análise e comparação dos pacotes que trafegam na rede com assinaturas de ataques conhecidos, armazenados em um banco de dados; esta categoria de SDI possui baixo potencial de detectar ataques novos ou modificados (MUKHOPADHYAY; CHAKRABORTY; CHAKRABARTI, 2011). Por outro lado, o sistema baseado em anomalia foca em identificar desvios no comportamento padrão da rede enquanto analisa seus dados, sendo capaz de detectar ataques conhecidos e desconhecidos (CHANDOLA; BANERJEE; KUMAR, 2009).

Ao tratar dos SDIs e da análise dos pacotes que trafegam na rede, os dados são um fator crucial que envolve esse tópico. Por natureza, o ser humano encontra dificuldade quando necessita desempenhar funções mecânicas cognitivamente exaustivas, como o processamento numeroso de dados. No entanto, a integração do esforço humano com a capacidade computacional da máquina agrega muito valor à segurança, uma vez que o computador é capaz de processar dados incansavelmente e com uma eficiência muito maior que os humanos, restando aos indivíduos a responsabilidade de analisar a informação fornecida e processada pela máquina, matéria da qual as pessoas têm um desempenho superior. A grande força da união entre humanos e máquinas ocorre quando cada um fornece ao conjunto o que faz de melhor.

Um dos pontos que dificulta o trabalho dos defensores na batalha contra os infratores é justamente o abundante volume de tráfego malicioso em potencial, que torna complexo distinguir o que são ameaças reais ou não e reduz o poder de tomada de decisão dos responsáveis. A Figura 3 apresenta o número de tentativas de conexão por algumas portas de rede observadas por um *Honeypot* durante um período de tempo. Ao contemplar a figura, torna-se perceptível o volume massivo de dados que devem ser analisados pelos especialistas, e isso sem contar os

eventos legítimos que trafegam pela rede e devem ser distinguidos dos maliciosos.

Figura 3 – Tentativas de conexão por porta de rede sobre tempo, capturadas por um *Honeypot*



Fonte: Verizon 2020, *Data Breach Investigations Report*. (VERIZON, 2020)

Então, levando em consideração a enorme quantidade de dados com que os responsáveis pela rede têm de lidar, vide a Figura 3 como exemplo, e a dificuldade que humanos têm em processar tanta informação em curtos períodos de tempo, acrescido da constante mutação das ameaças, onde métodos e *softwares* estão sempre sendo modificados para conseguir burlar as defesas dos sistemas, nota-se a ascensão da aplicação de técnicas de Aprendizado de Máquina, do inglês *Machine Learning* (ML), no contexto de Segurança da Informação, uma vez que elas conseguem emular o aprendizado humano e habilmente desempenhar tais funções, que são laboriosas para os humanos, transformando dados brutos e abundantes em informações relevantes (NAQA; MURPHY, 2015), permitindo prontidão na tomada de decisão por parte dos administradores de rede e responsáveis pela segurança dos ambientes. Neste contexto, estas técnicas visam melhorar a eficiência de detecção de atividades suspeitas e diminuir a taxa de alarmes falsos, auxiliando na automação do processo de detecção de ameaças conhecidas e, principalmente, contribuindo com a capacidade de detectar ameaças parcialmente modificadas e também ameaças totalmente desconhecidas. Essa característica das técnicas de ML, de reconhecimento de padrões, têm sido de grande valia no desenvolvimento de SDIs mais robustos, sendo comumente aplicadas junto àqueles baseados em anomalias (MUSA et al., 2020).

No que diz respeito à aplicação das técnicas de ML nos SDIs, existe uma gama abundante de possibilidades de implementação das técnicas, sendo a mais tradicional a utilização de classificadores singulares, onde qualquer classificador pode ser implementado para trabalhar no problema de detecção de intrusão, como *Decision Tree*, *Logistic Regression* e *Support Vector Machine* (HAQ et al., 2015). No entanto, é relevante salientar que algumas técnicas podem exigir um ajuste correto dos seus parâmetros, requerendo esforços adicionais para que seu funcionamento seja eficiente. Uma abordagem que visa superar fardo da seleção de parâmetros

é o algoritmo da Floresta de Caminhos Ótimos, do inglês *Optimum-Path Forest* (OPF), que já demonstrou sua capacidade de ser superior às técnicas tradicionais de reconhecimento de padrão em alguns casos (PAPA; FALCAO; SUZUKI, 2009).

Ainda no mesmo tema, trabalhos recentes empregam uma estratégia onde classificadores singulares, citados no parágrafo anterior, são combinados através da implementação de um método conjunto para resolver um único problema (KRISHNAVENI et al., 2021; ZHOU; CHENG, 2019). Esta implementação conjunta pode ser vista como um aprimoramento da abordagem que utiliza somente o classificador singular, sendo capaz de atingir taxas de detecção melhores e também menos alarmes-falsos (AMARUDIN; FERDIANA; WIDYAWAN, 2020).

Sendo assim, levando em consideração toda a problemática de Segurança da Informação que envolve as organizações e os danos diretos e indiretos decorrentes de incidentes de segurança, juntamente com a implementação de SDIs que empregam técnicas de ML, tornando-se mais robustos, para proteger o ambiente organizacional e também o aprimoramento destas técnicas através dos métodos conjuntos, esta monografia apresenta a aplicação do método conjunto *Stacking* do classificador OPF para o problema de detecção de intrusão. As maiores contribuições deste trabalho são:

- Um modelo *stack* que emprega o classificador OPF, visando melhorar significativamente as taxas de detecção e reduzir as taxas de alarmes falsos, através da combinação do algoritmo *Stacking* de método conjunto e as características inerentes do classificador OPF. A revisão literária conduzida indica que, até o momento da escrita desta monografia, a implementação de um modelo conjunto *stack* do classificador OPF é inédita; e
- Utilização de um conjunto de dados novo, denominado uneSPY (RUBIO, 2016), contribuindo para a comunidade científica com dados concretos de experimentos reais, que foram submetidos à uma metodologia científica metódica, de um conjunto de dados completamente rotulado e fornecendo a base para a realização de pesquisas futuras.

1.1 Hipótese de pesquisa

Dado o problema de detecção de intrusão apresentado, a implementação do *Stacking* do classificador OPF visa atender a questão da eficiência dos modelos que são empregados para solucionar este problema. Como será demonstrado e discutido ao longo da monografia, o propósito da implementação do modelo *Stacking* do OPF é averiguar se ele será capaz de melhorar a capacidade de generalização desse algoritmo, assim aprimorando sua eficiência na solução do problema de detecção de intrusão.

1.2 Objetivos

Esta seção apresenta quais são os objetivos gerais e específicos, sendo o item 1.2.1 referente ao objetivo geral e o item 1.2.2 pertinente aos objetivos específicos.

1.2.1 Objetivo geral

Enfatizar a importância da Segurança da Informação, discutindo o problema da detecção de intrusão e soluções oportunas, e apresentar uma resolução eficaz, que apoia-se na detecção de anomalias, através da implementação de técnicas de *Machine Learning* para o problema de detecção de intrusão, contribuindo com a área da Segurança da Informação em suas tarefas de confronto à ameaças.

1.2.2 Objetivos específicos

Problemas relacionados à ataques cibernéticos estão cada vez mais frequentes, trazendo insegurança e preocupação às instituições, que temem pela confiabilidade de suas informações e pela saúde de seu negócio. Logo, é necessário o conhecimento de controles de segurança e métodos eficientes de prevenção, detecção e respostas de incidentes, todos englobados pela área de Segurança da Informação. Portanto, para apresentação de uma solução dos problemas relacionados aos ataques cibernéticos, esta monografia apresenta::

- a) Conceitos de Segurança da Informação relevantes ao problema;
- b) Análise dos tipos de Sistemas de Detecção de Intrusão mais utilizados, capazes de capturar, classificar e identificar pacotes que trafegam na rede;
- c) Classificação destes sistemas de acordo com suas características particulares;
- d) Análise de algoritmos de *Machine Learning* aptos à classificação e predição de dados;
- e) Classificação destes diferentes algoritmos de acordo com suas especialidades, vantagens e desvantagens;
- f) Realização de um estudo sobre o estado-da-arte na área que compreende a união dos Sistemas de Detecção de Intrusão e *Machine Learning*;
- g) Demonstração da utilização de um conjunto de dados novo através da implementação de experimentos reais que forneçam uma base para trabalhos futuros; e
- h) Apresentação de uma aplicação que valide os temas discutidos e seja capaz de contribuir cientificamente para a comunidade de Segurança da Informação, através da implementação de um algoritmo de *Machine Learning* para o problema de detecção de intrusão, no caso a implementação do método conjunto *Stacking* do classificador OPF;

1.3 Estrutura da Dissertação

O restante do documento está organizado como segue. A fundamentação teórica está embasada no Capítulo 2, que apresenta, essencialmente, conceitos sobre Segurança da Informação, Sistemas de Detecção de Intrusão e Aprendizado de Máquina. O Capítulo 3 expõe trabalhos correlatos, e recentes, à fundamentação que é retratada nesta monografia. A metodologia adotada para o desenvolvimento é apresentada no Capítulo 4, enquanto os experimentos propostos são exibidos no Capítulo ???. Por fim, o Capítulo 5 exhibe e discute os resultados experimentais, ao passo que o Capítulo 6 conclui o trabalho e identifica oportunidades de trabalhos futuros. Ao término do documento, encontram-se as referências utilizadas e apêndices contendo informações extras.

2 Fundamentação Teórica

Este capítulo apresenta toda a fundamentação teórica que sustenta o trabalho apresentado e sua divisão é descrita a seguir. A Seção 2.1 retrata conceitos importantes sobre Segurança da Informação, necessários para a compreensão total do trabalho. Em seguida, a Seção 2.2 discorre sobre os Sistemas de Detecção de Intrusão, identificando suas particularidades. Por fim, a Seção 2.3 apresenta definições essenciais sobre a área de Aprendizado de Máquina, que está diretamente ligada com os objetivos do trabalho.

2.1 Segurança da Informação

No Capítulo 1 foi dada ênfase na relevância da Segurança da Informação na atualidade e também a participação crescente que ela terá no futuro que aproxima-se. Essa é uma área que enfrenta grandes dificuldades devido ao constante confronto com ameaças que diariamente se atualizam, buscando maneiras de burlar as proteções vigentes em ambientes de produção. O maior objetivo deste trabalho apresentado é contribuir à esta área com uma solução eficiente e aprimorada para confrontar essa constante mutação das ameaças existentes.

Portanto, antes de prosseguir, é interessante contextualizar determinados conceitos da Segurança da Informação que, de maneira holística, comunicam-se com o restante do trabalho e agregam na percepção do propósito do mesmo.

Décadas atrás, pesquisadores já começavam a descobrir o valor e o impacto que a informação viria a ter nos processos de negócio das organizações (BRYNJOLFSSON, 1994). Antigamente, os fatores de produção que costumavam sustentar os processos de negócio da maioria das empresas podiam ser divididos, essencialmente, entre capital, trabalho e matéria prima.

Hoje, observa-se uma enorme adição àqueles três fatores de produção: a informação. Alguns exemplos práticos que estão presentes no cotidiano das pessoas são organizações com modelos de negócio que seriam impraticáveis antes. O *Uber* ⁷, que pode ser considerado uma das maiores empresas de “táxi” do mundo, não possui carros; o *Airbnb* ⁸, uma das maiores provedoras de hospedagem do mundo, não possui imóveis. A base dos processos de negócio que sustentam os modelos dessas organizações é a informação, tamanha importância sua na conjuntura atual.

Como foi abordado previamente, uma vez que há modelos de negócio que apoiam suas operações na informação, e existe uma quantidade praticamente infinita de informações

⁷ Disponível em <<https://www.uber.com/pt-pt/>>

⁸ Disponível em <<https://www.airbnb.com.br/>>

sensíveis sendo criadas, coletadas, trafegadas e armazenadas diariamente, observa-se também, em contrapartida, a presença de malfeitores que buscam acesso indevido à essas informações confidenciais. Fundamentalmente, a Segurança de Informação é vista como uma disciplina responsável por garantir a confidencialidade, integridade e disponibilidade deste ativo crucial, a informação (SOLMS, 2001).

Esta tríade da Segurança da Informação, como é popularmente apelidada, tem suas raízes profundamente conectadas à mentalidade de segurança militar dos EUA, que já investia em diversos estudos sobre segurança computacional há muitas décadas atrás (GOLLMANN, 2010). Um destes estudos, conhecido como *The Anderson Report* (ANDERSON, 1972), além de abordar a proteção de informações sigilosas governamentais e militares, identificou três macro categorias de riscos à segurança, que eventualmente tornaram-se a fundação da tríade CID:

- a) Divulgação não-autorizada de informação: um indivíduo não-autorizado é capaz de ler e beneficiar-se de informações armazenadas em computador. Essa categoria pode também ser compreendida como “análise de tráfego”, na qual o invasor apenas observa padrões de uso de dados; através desses padrões, o invasor tem a capacidade de inferir o conteúdo da informação. Além disso, esta categoria também inclui o uso não-autorizado de programas privados. (Confidencialidade);
- b) Modificação não-autorizada de informação: um indivíduo não-autorizado é capaz de fazer alterações em informações armazenadas. Não necessariamente o invasor consegue ver a informação que modificou. (Integridade); e
- c) Negação de serviço não-autorizada: um indivíduo é capaz de impedir um usuário legítimo de utilizar, ou modificar, informações. Ainda que o invasor consiga realizar essa ação, isso não significa que ele necessariamente pode acessar ou modificar informações por si próprio. (Disponibilidade).

Nesse mesmo período, as pesquisas da área começaram a mudar seu foco, pois até então a atenção estava voltada à proteção dos computadores em si. Saltzer e Schroeder (SALTZER; SCHROEDER, 1975) enfatizaram que a principal preocupação da segurança deveria ser em relação à informação armazenada, e não simplesmente os computadores. Considerado essencial por Saltzer e Schroeder, a tríade CID deveria estar sempre presente nos princípios básicos de proteção da informação.

Com o advento dessas pesquisas, uma grande mudança na abordagem de segurança começou, partindo da proteção de *hardware* e *software* dos computadores para a proteção da informação. No entanto, essa transformação foi um lento processo gradual que focou primeiro na confidencialidade, depois na integridade e somente então na disponibilidade. Sendo assim, os primeiros modelos de Segurança da Informação amplamente adotados concentravam-se em garantir a proteção contra esses tipos de ameaças à segurança dos dados, que são

contemplados pela tríade CID. Esses modelos primários sustentavam-se em abordagens técnicas, vulgo controles lógicos, que posteriormente seriam complementados por controles físicos e organizacionais.

Um destes primeiros modelos foi o Bell-La Padula (BELL; PADULA, 1976), que focava em estabelecer regras para fornecer confidencialidade e proteger as informações. Outro modelo notável da época foi o Biba (BIBA, 1977), cujo objetivo estava na integridade dos dados, ao contrário do modelo de Bel-La Padula, que focava na confidencialidade.

Conforme os anos foram passando e as pesquisas desenvolvendo-se, novos modelos surgiram, novas abordagens apareceram, diferentes técnicas foram testadas e, claro, ameaças inéditas também irromperam. Ainda com a atenção voltada aos controles lógicos, os pesquisadores aprofundaram-se cada vez mais, dividindo-os em três categorias essenciais para garantir a proteção dos ambientes: prevenção, detecção e resposta. Estes três conceitos, também chamados de fases, são implementados de maneira sucessiva e requerem estratégias específicas individualmente. Como mencionado anteriormente, para quem trabalha com a defesa, o lado do invasor é sempre uma incerteza pois ele está em constante mudança.

Quando o assunto é Segurança da Informação, é sempre mais eficaz prevenir do que remediar, mesmo que isso requeira análise extensa e muito planejamento. Medidas têm de ser tomadas para proteger a informação, que é um ativo que necessita de proteção proporcional ao seu valor, de qualquer modificação, destruição ou divulgação não-autorizada, seja ela intencional ou não.

No entanto, por maior que seja o esforço investido na prevenção de ataques, não há nenhuma garantia que o ambiente será completamente seguro. Muito pelo contrário, devido ao crescente aumento das ameaças, não importa quão sólida a defesa seja, sempre existe a chance de que ela eventualmente seja quebrada. Por isso que a detecção de invasões é extremamente importante, e melhor ainda se a defesa for implementada em camadas, já que quando uma camada falhar, é conhecido onde ocorreu o problema, facilitando sua detecção. Embora não exista nenhuma receita de construção para uma segurança perfeita, independente da estratégia adotada, é essencial que a detecção ocorra rapidamente e uma notificação sobre a falha seja divulgada, possibilitando uma resposta eficaz.

O modelo Denning de detecção de intrusão (DENNING, 1987) é um marco nesse quesito. Seu modelo foi desenvolvido como um Sistema de Detecção de Intrusão em tempo real para sistemas, cujo objetivo estava em detectar violações, penetrações, e outras formas de infrações computacionais. Esse modelo é baseado na hipótese de que violações de segurança podem ser detectadas através da monitoração de registros do sistema, buscando por padrões anormais de uso. Visando a proteção dos ambientes, a detecção é um fator inegavelmente crucial, cujos primórdios são consolidados com este modelo de Denning, e é a essência deste trabalho apresentado, onde o objetivo está em contribuir com sua melhoria. Os Sistemas de Detecção de Intrusão são apresentados detalhadamente a seguir.

2.2 Sistemas de Detecção de Intrusão

A habilidade de conectar qualquer computador, de qualquer lugar mundo, a qualquer outro computador, também independente do lugar, é uma faca de dois gumes. Para os usuários em casa, despretensiosamente navegando pela *Internet*, com certeza é um privilégio divertido e prazeroso. Já para os responsáveis pela segurança da informação de instituições é um pesadelo.

A maioria das organizações possuem enormes quantidades de informações confidenciais armazenadas *in loco* e também *online*. A divulgação dessas informações sigilosas para um competidor, usuários indevidos, ou simplesmente qualquer pessoa ou entidade que não dispõe de autorização para acessá-las, pode ter consequências trágicas para a vítima desse incidente de segurança (MOUSSAILEB et al., 2018).

Além do perigo atrelado à essas informações serem indevidamente expostas publicamente, há também o risco de informações manipuladas adentrarem o ambiente organizacional. Ou seja, além do risco de informações saírem, existem também o risco de informações entrarem, que é literalmente o caminho oposto dos dados que são vazados da rede interna para a externa. Isso acontece porque *malwares*, em geral, têm a capacidade de burlar mecanismos de defesa implementados e destruir ou modificar dados, causando estragos cujos reparos são custosos. Tais violações de segurança normalmente acontecem por erro humano oriundo dos colaboradores da instituição em questão, que podem, por exemplo, ter configurado erroneamente algo ou importado um *malware* involuntariamente ao ambiente interno. Também é comum casos onde ataques profundamente específicos são direcionados às organizações, realizados por agentes externos à elas, considerados cibercriminosos. Consequentemente, é necessária implementação de mecanismos adequados para assegurar a proteção do ambiente, mantendo os *bits* “bons” dentro dela, representados por toda e qualquer informação sensível, e os *bits* “maus” fora, aqueles que não são naturais ao ambiente e podem causar danos ao mesmo (TANENBAUM; FEAMSTER; WETHERALL, 2020).

Atentando-se à este controle dos *bits* “bons” e “maus” que se destaca o processo de detecção de intrusão. Esta técnica consiste em monitorar eventos que ocorrem em um sistema computacional ou em uma rede e analisá-los, buscando sinais de possíveis incidentes; que são violações, ou ameaças iminentes de violação, das políticas de segurança computacionais, políticas aceitáveis de uso ou práticas padrões de segurança (DENNING, 1987). Basicamente, é uma constante procura por qualquer atividade considerada não-habitual na rede, baseada em regras previamente estabelecidas.

Sistemas de detecção de intrusão focam especialmente em identificar eventuais incidentes, registrar informações sobre eles e reportá-los aos responsáveis pela segurança do ambiente (LANGIN, 2011). Adicionalmente, organizações empregam SDIs para outros propósitos, como identificar problemas relacionados às políticas de segurança, documentar ameaças existentes e descapacitar indivíduos de violarem as políticas de segurança.

Por causa de seu grande potencial de contribuição à preservação da confiabilidade da informação armazenada eletronicamente, atualmente os SDIs tornaram-se uma necessidade de praticamente todas organizações (BUL'AJOUL; JAMES; PANNU, 2015) (RASTEGARI; HINGSTON; LAM, 2015) (DASTANPOUR et al., 2014) (ENYINDAH; ONWUACHU, 2017). Eles registram dados associados às ações detectadas, alertam violações e produzem relatórios detalhados, além de estarem constantemente monitorando toda a rede em busca de atividades anormais (RAZA et al., 2016). Existem diferentes tipos de SDIs e eles são diferenciados principalmente pela forma com que lidam com os eventos observados e também pelo seu processo de implementação.

Há várias metodologias implementáveis de SDIs, devido à variância das configurações de rede e dos parâmetros da própria ferramenta em questão. Cada tipo distinto possui vantagens e desvantagens em relação à detecção, configuração, custo e desempenho em geral. Comumente, os SDIs podem ser divididos em duas categorias, que se embasam na abordagem adotada para detecção dos eventos monitorados: baseados em assinatura e baseados em anomalia. Ambas técnicas utilizam a seleção de características dos pacotes analisados da rede para construir um sistema eficiente, e serão abordadas nos itens seguintes.

2.2.1 Baseado em assinaturas

A metodologia dos SDIs baseados em assinatura funciona através da comparação de assinaturas examinadas em tempo real com assinaturas presentes em um banco de dados. Estas assinaturas nada mais são do que uma junção organizada de vários aspectos intrínsecos aos pacotes que trafegam na rede. O banco de dados associado ao SDI contém uma vasta lista com assinaturas de ataques conhecidos.

Se algum padrão de assinatura, construído através da extração das características dos pacotes observados no ambiente monitorado, corresponde às assinaturas existentes no banco de dados, é sinalizada uma violação da política de segurança ou um possível ataque. Este procedimento descrito não é custoso computacionalmente, o que torna essa abordagem interessante. Seu baixo custo computacional se dá pelo fato de que não são inspecionadas todas atividades ou todo o tráfego da rede do ambiente monitorado, apenas são buscadas assinaturas que possivelmente são conhecidas e que estão presentes no banco de dados.

Sua implementação é considerada fácil pois não é necessário que esse sistema aprenda sobre o ambiente que está sendo analisado (VALDES; SKINNER, 2001). Seu funcionamento consiste somente em procurar, inspecionar e comparar o conteúdo dos pacotes capturados na rede para encontrar ameaças com assinaturas reconhecidas. Em uma analogia rudimentar, seu funcionamento assemelha-se à um jogo da memória.

Além disso, também são analisadas as chamadas do sistema, buscando por *payloads*, ou corpo de dados, maliciosos. Sendo assim, a metodologia baseada em assinatura demonstra-se

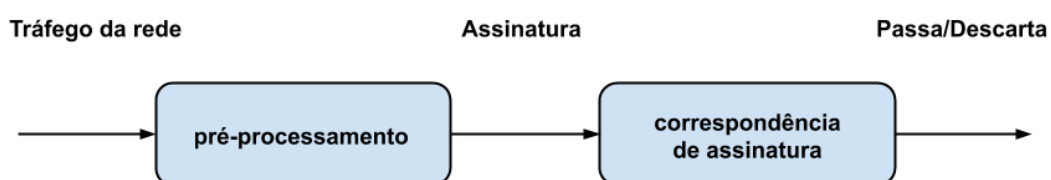
muito efetiva contra ataques e violações já conhecidas, porém incapaz de detectar ataques novos até que seu banco de dados seja atualizado com novas assinaturas.

Por serem dependentes da adição de novas assinaturas no banco de dados para conseguir detectar novos ataques, os SDIs baseados em assinatura são relativamente fáceis de serem burlados. Eles conseguem ser ignorados por invasores que simplesmente modifiquem ataques conhecidos e foquem suas investidas em sistemas que não foram atualizados com novas assinaturas, incapazes de detectar tal mutação do ataque.

Outro fator que pode ser considerado negativo é que esse SDI necessita de recursos expressivos para ser capaz de manter-se atualizado constantemente, levando em consideração o número praticamente infinito de possíveis alterações às ameaças existentes (LEE; MOSKOVICS; SILACCI, 1999). No entanto, é descomplicado modificá-lo e aprimorá-lo, uma vez que seu funcionamento é completamente fundado apenas na implementação de assinaturas.

A Figura 4 representa a arquitetura comum de um sistema de detecção de intrusão baseado em assinatura, que faz uso de um pré-processamento para encontrar a assinatura do pacote analisado e comparar esta atividade, encontrada no ambiente que está sendo monitorado, com suas assinaturas já conhecidas, presentes em seu banco de dados. Caso uma correspondência seja identificada, um alerta é emitido; caso contrário, o detector nada faz. Diversos SDIs comerciais empregam esta abordagem porque ataques conhecidos conseguem ser detectados de maneira econômica e confiável, com poucos erros de falso-positivo.

Figura 4 – Arquitetura genérica de um SDI baseado em assinatura.



Fonte: Elaborado pelo autor.

Apoiando-se na pesquisa literária, tanto os sistemas baseados em assinatura quanto aqueles baseados em anomalias possuem suas limitações. A metodologia de detecção baseada em assinatura falha na detecção de ataques desconhecidos ou modificados, além de ter a atualização contínua das assinaturas no banco de dados como a maior preocupação técnica enfrentada. Por outro lado, a técnica de detecção baseada em anomalia pode produzir altas taxas de alarmes falsos, que será abordada a seguir.

2.2.2 Baseado em anomalia

Essa abordagem pode ser descrita como um sistema de detecção que modela o comportamento da rede, usuários e sistemas computacionais e emite um alerta sempre que houver qualquer desvio de seu comportamento considerado como normal (DENNING, 1987). A metodologia de detecção de anomalia supostamente é capaz de detectar um ataque completamente novo ou novas ameaças potenciais à rede. Seu funcionamento baseia-se em construir perfis que representam o desempenho normal da rede e então compará-los ao comportamento atual dos dados que trafegam, buscando encontrar alguma incompatibilidade entre eles (NENE; SINGH, 2013). Diante disso, qualquer ação que desviar significativamente do comportamento considerado habitual é prontamente detectada.

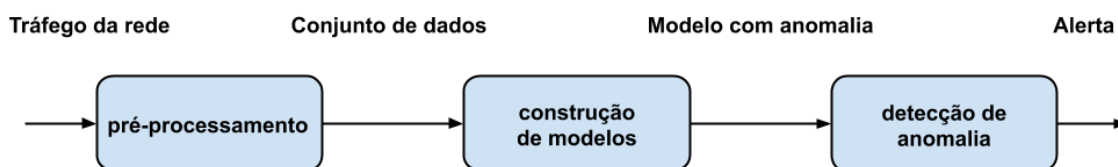
O mecanismo de detecção baseado em anomalia irá detectar qualquer tráfego que é novo ou incomum, sendo eficaz em identificar até varreduras e sondagens direcionadas ao *hardware* da rede em questão. É possível emitir avisos sobre potenciais invasões com antecedência, porque tais sondagens e pesquisas são os antecessores de praticamente todos ataques (FELLIN; HANEY, 2014). Isso aplica-se à qualquer serviço novo instalado em qualquer *hardware*; por exemplo, se *telnet* for implantado em algum roteador de rede, com fins de manutenção, e esquecido após o término do reparo, o SDI manterá sua vigilância sobre ele e alertará caso atividades incomuns ocorrerem.

Na literatura, a vasta maioria dos sistemas de detecção baseados em anomalia são desenvolvidos baseando-se em técnicas de Aprendizado de Máquina. Alguns estudos aplicam algoritmos de aprendizado singular, como *Logistic Regression*, *Decision Trees*, entre muitas outros. Por outro lado, alguns sistemas são baseados na combinação de diferentes técnicas de aprendizado, como métodos híbridos ou conjuntos (GAUTAM; DOEGAR, 2018). Tanto o aprendizado singular quanto o combinado são desenvolvidos através de classificadores, que são resultantes do processo de aprendizado de máquina. No contexto do SDI baseado em anomalia, esses classificadores são utilizados para identificar se o acesso que chega à rede é normal ou anômalo, ou até mesmo trabalham na classificação de dados que estão trafegando em tempo real na rede.

O monitoramento da rede e a identificação e classificação dos pacotes somente são viáveis porque os algoritmos de Aprendizado de Máquina possibilitam a implementação de um sistema capaz de aprender através de dados que lhe são apresentados. Por exemplo, um sistema que emprega técnicas de aprendizado de máquina pode ser treinado a partir de pacotes que chegam, ou até mesmo através de pacotes antigos armazenados, aprendendo a distinguir pacotes normais de anômalos, e então, após o aprendizado, ser utilizado para classificar novos pacotes de entrada entre pacotes normais ou anômalos. Porém, uma das dificuldades enfrentadas com essa abordagem é apresentada justamente no processo de aprendizado, onde, caso haja ruído nos dados de treinamento, pode haver um encadeamento de classificações erradas, que irão gerar muitos falsos-positivos ou falsos-negativos depois.

A Figura 5 representa uma arquitetura deste sistema descrito, que consiste no pré-processamento, para extração de informações a serem consumidas, e então foca em construir um padrão da conexão atual, que caso desviado do comportamento considerado como normal irá emitir um alerta.

Figura 5 – Arquitetura genérica de um SDI baseado em anomalia.



Fonte: Elaborado pelo autor.

2.2.3 Falsos positivos e negativos

É válido ressaltar a impossibilidade de um SDI ser um mecanismo completamente perfeito, especialmente pelo fato de que o monitoramento do tráfego de uma rede é um problema complexo e que envolve enormes quantidades de dados. Durante essa seção, por algumas vezes, foram citados os termos ‘falsos positivos’ e ‘falsos negativos’, que nada mais são do que os possíveis resultados imprecisos que o SDI pode gerar. Cabe frisar que este conceito de falso positivo e falso negativo não é exclusivo ao contexto de detecção de intrusão e é amplamente utilizado em diversas áreas científicas, e.g. medicina.

Falsos positivos ocorrem quando o SDI detecta erroneamente uma anomalia em um tráfego normal. Um evento habitual é apontado, por engano, como um incidente à ser analisado. Analogamente, é como um médico indicar que um paciente perfeitamente saudável está doente e encaminhá-lo para maiores exames. Normalmente, voltando ao contexto de detecção de intrusão, lidar com falsos positivos é uma adversidade aceitável pois, como trata-se de um alerta sobre um problema inexistente, suas consequências são mínimas. Porém, um grande número destas classificações incorretas significa trabalho extra e desnecessário àqueles incumbidos de checar estes avisos. Os SDIs baseados em anomalia geralmente possuem uma taxa maior de falsos positivos do que os baseados em assinatura (KAJA; SHAOUT; MA, 2019).

Já falsos negativos acontecem quando um tráfego indesejado não é detectado pelo SDI. Ao analisar o comportamento da rede, um evento anômalo é categorizado equivocadamente como normal. Em paralelo, é como, após uma busca por irregularidades, classificar um helicóptero como totalmente livre de falhas, quando o mesmo encontra-se com problema no motor. É exatamente por isso que os falsos negativos são muito mais perigosos e intoleráveis que os falsos positivos. Quando ocorre um falso positivo, no pior dos casos, após o alerta emitido, haverá uma checagem feita em vão. Porém, quando acontece um falso negativo, sequer existe

a possibilidade de problema algum ser checado, pois nenhum aviso prévio será realizado e isso pode provocar consequências severas.

Nenhum sistema de detecção de intrusão será capaz de fornecer predições totalmente corretas, todos estão sujeitos à erros, por mais que sejam mínimos. Ambos equívocos relacionados à falsas classificações criam problemas para a segurança e podem ser sinônimo da necessidade de calibração do sistema (VANERIO; CASAS, 2017). Como os falsos positivos tendem a ser menos prejudiciais ao ambiente, basicamente resultando somente em gastos de recursos e tempo, sua eventualidade é mais aceita. Por outro lado, os falsos negativos podem ter decorrências fatais e por isso sua ocorrência é praticamente intolerável. Sendo assim, continuamente buscam-se desenvolver técnicas que capacitem os sistemas à aumentarem suas taxas de detecção e diminuam suas taxas de falsos positivos e falsos negativos.

Nessa procura de aperfeiçoar o desempenho dos SDIs, observa-se que os sistemas baseados em anomalia têm mostrado-se pertinentes (RUBIO et al., 2019). Ao combinarem estes sistemas com técnicas de Aprendizado de Máquina, que serão abordadas a seguir, resultados promissores têm sido produzidos nesta área (KAJA; SHAOUT; MA, 2019) (PALMIERI, 2019) (DIVYASREE; SHERLY, 2018). Logo, como será tratado no restante do trabalho, no meio científico, existem boas expectativas quanto à implementação de SDIs baseados em anomalia utilizando técnicas de Aprendizado de Máquina para melhoria de performance e redução de erros.

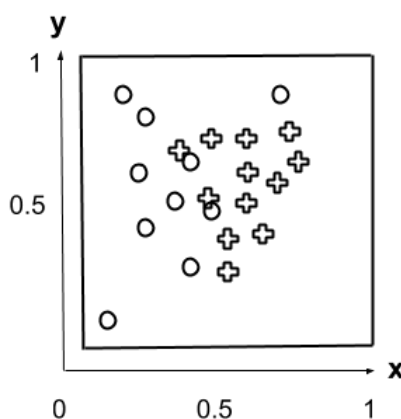
2.3 Aprendizado de Máquina

Machine Learning, do inglês, é um ramo específico de Inteligência Artificial que visa adquirir conhecimento através do treinamento de dados, que são baseados em fatos conhecidos. Seu objetivo é possibilitar que computadores desenvolvam cognição sem o auxílio de programação externa, focando especialmente na realização de predições a partir de um treinamento prévio com dados (HAQ et al., 2015). Como citado anteriormente, sistemas de detecção de intrusão baseados em anomalia normalmente utilizam técnicas de ML em sua arquitetura de implementação para aprimorar seu desempenho e oferecer uma solução ao problema de detecção de intrusão que seja menos dependente do fator humano (NENE; SINGH, 2013). Tópicos desse ramo serão abordados a seguir, enfatizando sua forte relação com o SDI mencionado no item 2.2.2.

Uma tarefa árdua enfrentada pelas áreas de ML, reconhecimento de padrões e mineração de dados é a construção de bons modelos a partir de conjuntos de dados. Geralmente, um conjunto de dados, do inglês *dataset*, possui vários vetores de características, no qual cada um destes vetores fornece a descrição de um objeto através da união de determinadas particularidades do mesmo (ZHOU, 2012). O número de características presentes em um *dataset* é chamado de dimensão. A Figura 6 apresenta um exemplo onde cada objeto de um

dataset fictício é representado por um ponto no gráfico e é detalhado por suas características expostas no mesmo; no caso, as características são: coordenada-x, coordenada-y e sua forma. Por exemplo, baseando-se no gráfico da Figura 6, é possível extrair vetores de características semelhantes à algo do tipo: (.5, .3, cruz) ou (.7, .9, círculo), cujas dimensões neste *dataset* fictício em específico são essas três já citadas.

Figura 6 – *Dataset* fictício representado no gráfico



Fonte: Elaborado pelo autor.

Nesta área de pesquisa, quando se fala em “modelo”, normalmente o que é referido é um modelo de predição ou um modelo da estrutura dos dados que deseja-se construir a partir de um *dataset*, e.g. *Logistic Regression*, *Decision Tree*, *Support Vector Machine*, entre outros. O processo de criar modelos a partir dos dados denomina-se aprendizado, ou treinamento, e é realizado através de um algoritmo de aprendizado. O modelo enfim treinado pode ser chamado de aprendiz, ou também de classificador (ZHOU, 2012).

Existem diferentes abordagens para o processo de aprendizagem, dos quais as mais comuns são as supervisionadas e não-supervisionadas. No aprendizado supervisionado, o mais relevante para este trabalho, o objetivo é ser capaz de prever o valor de uma propriedade específica de uma amostra desconhecida ao aprendiz. Por exemplo, tomando como base novamente a Figura 6; para prever a forma de algum dos objetos deste *dataset* representado, definem-se rótulos para as unidades conhecidas do problema, ou seja “círculo” e “cruz”, e então o aprendiz deve ser capaz de indicar qual é o rótulo de uma amostra cujo rótulo verdadeiro ele desconhece, apoiando-se somente nas demais características conhecidas da amostra, que neste cenário são as coordenadas *x* e *y*, e.g. (.3, .6). Quando rótulos são categóricos, como neste exemplo da Figura 6, onde eles são formas, a tarefa que acaba de ser descrita titula-se classificação e o aprendiz recebe o nome de classificador, como citado no parágrafo anterior. Para ambas metodologias de aprendizagem, o processo de treinamento é conduzido em *datasets* que contêm a informação do rótulo das amostras, e, no contexto onde a classificação a ser realizada é binária, normalmente os termos “positivos” e “negativos” são empregados para distinguir as duas diferentes classes de rótulos envolvidas.

Após concluído o processo de treinamento, é necessário observar se o modelo resultante cumpre ou não as exigências do usuário para conseguir defini-lo como um “bom” modelo (ou não). Existem diversas métricas para analisar o desempenho de um classificador e diferentes usuários podem ter expectativas divergentes perante elas, o que dificulta entender qual é a perspectiva “correta” a ser adotada diante do classificador em questão, até que o problema enfrentado seja compreendido holisticamente e abordado diretamente. Por isso é comum realizar o treinamento de múltiplos modelos, avaliando e estimando suas performances para somente então decidir qual deles é aceitável, ou até o melhor entre todos outros.

O objetivo essencial do aprendizado é a generalização, ou seja, ser capaz de generalizar o conhecimento adquirido a partir dos dados de treinamento para amostras desconhecidas. Logo, um bom classificador é aquele capaz de generalizar bem, produzindo poucos erros de predição. No entanto, é inviável estimar os erros de predição diretamente, porque é necessário, para tal, saber de antemão qual é o verdadeiro rótulo das amostras desconhecidas, que no caso é uma incógnita. Assim, um processo típico é fazer com que o classificador efetue predições em dados de teste, nos quais as informações verdadeiras dos rótulos são conhecidas, e então utilizar os erros provenientes do teste como uma estimativa do erro real de predição (BLUM; MITCHELL, 1998). Este processo descrito, de expor um classificador à dados desconhecidos, porém controlados, é chamado de teste. No entanto, para não gerar classificadores enviesados, é uma boa prática garantir que os dados de teste não sobreponham os dados de treino. A seguir, serão abordados alguns dos algoritmos mais populares utilizados no processo de aprendizagem.

2.3.1 Classificadores singulares

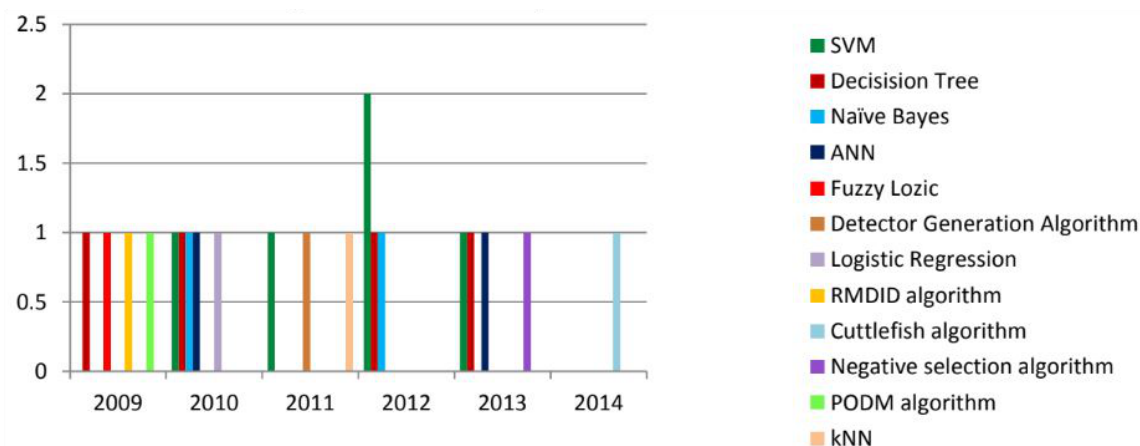
Resultado do processo de aprendizagem descrito, um classificador singular é uma técnica de ML que pode ser empregada diretamente no desenvolvimento de um sistema de detecção de intrusão baseado em anomalia. Note que, de maneira alguma, sua aplicabilidade se restringe aos SDIs, porém, como não convém ao estudo, suas outras utilizações não serão detalhadas.

O estudo de Haq et al. (HAQ et al., 2015) discute abrangentemente a aplicação de ML dentro do contexto dos Sistemas de Detecção de Intrusão. Em seu estudo, observa-se que classificadores singulares são utilizados com frequência nessa conjuntura, como pode ser observado na Figura 7. A seguir, são detalhados alguns destes classificadores singulares comumente utilizados pela comunidade.

2.3.1.1 *Decision tree*

Decision Tree (DT) é um modelo de classificação que utiliza uma estrutura semelhante à de uma árvore para realizar uma decisão. Normalmente, são empregados em operações de busca e detecção de intrusão, pois conseguem obter um bom desempenho (SALZBERG, 1994). Uma das tarefas deste algoritmo é a criação de um classificador para a prever valor de uma

Figura 7 – Distribuição da utilização dos classificadores singulares em SDIs ao longo dos anos.



Fonte: Haq et al. (HAQ et al., 2015)

classe a fim de testar instâncias desconhecidas. Este modelo é amplamente usado como um classificador singular por causa de sua simplicidade e fácil implementação (FARID et al., 2013).

2.3.1.2 *Random Forest*

O algoritmo *Random Forest* (RF) retrata uma combinação de classificadores de árvore onde cada árvore é dependente dos valores de um vetor aleatório amostrado de forma independente e com a mesma distribuição para todas árvores na floresta (BREIMAN, 2001). O erro de generalização da floresta de classificadores de árvores está relacionado à força individual das árvores presentes na floresta e à correlação entre as mesmas. Este algoritmo demonstra-se como uma ferramenta eficiente para predições.

2.3.1.3 *Naive Bayes*

Naive Bayes é um dos métodos probabilísticos baseados no teorema de Bayes (JENSEN, 1996) usado para determinar corretamente uma classe de dados desconhecidos, calculando a probabilidade posterior para cada classe. Depois de realizado o cálculo probabilístico referente às classes, é determinado um rótulo àquela que apresentar a maior probabilidade de correspondência para com o dado desconhecido. Este algoritmo normalmente produz bons resultados em classificações que têm relações mais simples. Além disso, como ele requer apenas uma varredura dos dados de treinamento, ele pode facilitar o trabalho de classificação.

2.3.1.4 *Artificial Neural Network*

Em português Redes Neurais Artificiais, é uma unidade de processamento de informações que teve seu desenvolvimento inspirado na funcionalidade do cérebro humano (HAYKIN, 1998). Normalmente, as redes neurais estão organizadas em camadas, que são compostas por nós interconectados contendo uma função de ativação. Padrões são apresentados à rede através de

uma camada de entrada, que se comunica com uma ou mais camadas ocultas onde, por meio de um sistema de conexões ponderadas, o processamento acontece. Essas camadas ocultas então unem-se à uma camada de saída para produzir o resultado da detecção.

2.3.1.5 *Support Vector Machines*

É um método de ML usado para resolver problemas de classificação baseando-se em um hiperplano ideal no espaço, onde são separadas as duas classes do problema enfrentado. O conceito por trás deste modelo para detecção de intrusão baseia-se em utilizar os dados de treinamento como uma representação somente do que são classes normais de objeto, ou o que é reconhecido como uma atividade não-maliciosa pelo SDI, considerando então todo o resto como anomalias (TAX; DUIN, 1999). O classificador construído pela metodologia *Support Vector Machine* distingue o espaço de entrada em uma região finita, onde estão os objetos normais, e assim considera que todo o resto do espaço contém as anomalias (CATANIA; BROMBERG; GARINO, 2012).

2.3.1.6 *Logistic Regression*

Logistic Regression (LR), é um dos modelos baseados em estatística mais comuns para predição de dados binários (positivo/negativo) (ABU-NIMEH et al., 2007). Sendo empregado em diversas áreas, além da detecção de intrusão, sua ampla aplicação deve-se muito à sua simplicidade. Seu desempenho destaca-se quando a relação entre os dados tende a ser linear. Ao mesmo tempo, sua performance é prejudicada se há uma relação não-linear complexa entre os dados.

2.3.2 Avaliação e comparação

Apresentados alguns algoritmos de classificadores singulares, é importante salientar, como foi abordado em suas respectivas descrições, que cada um possui suas particularidades e comporta-se diferentemente. Por exemplo, alguns modelos desempenham melhor perante um problema simples, outros destacam-se em cenários mais complexos. Por isso, a escolha do classificador 'perfeito' para solucionar um problema de classificação não é uma tarefa simples e deve levar em consideração suas características específicas.

Normalmente, existem múltiplas opções de algoritmos de aprendizagem a serem escolhidas, juntamente com diversos parâmetros a serem ajustados. A tarefa de escolher o melhor algoritmo e as configurações adequadas de seus parâmetros é conhecida como seleção de modelo, que depende da estimativa da performance do aprendiz (ZHOU, 2012). De maneira empírica, isto envolve o desenvolvimento de experimentos e testes para efetuar a comparação dos classificadores. Tendo isso em mente, não é aconselhável estimar o erro de predição de um aprendiz baseado em seu erro de treinamento, ou seja, o erro que o aprendiz obteve perante seus dados de treinamento. Uma forma adequada de avaliar sua performance é através de

um conjunto de validação. Isso ocorre porque ambos rótulos no conjunto de treinamento e no conjunto de validação são conhecidos antes do processo de treinamento e devem ser utilizados juntamente para aferir e ajustar o aprendiz final, uma vez que o modelo já tenha sido selecionado.

Na maioria dos casos, os conjuntos de treinamento e validação são obtidos através da divisão de um *dataset*, já conhecido, em duas partes. Durante a divisão, as propriedades do conjunto original de dados devem ser mantidas íntegras ao máximo possível; caso contrário, o conjunto de validação pode fornecer estimativas enganosas. Por exemplo, em um cenário onde os dados reservados ao conjunto de treinamento contém apenas instâncias positivas, restando ao conjunto de validação somente instâncias negativas, haverá um enviesamento do classificador resultante desse processo de aprendizado.

Portanto, na classificação, é uma boa prática dividir aleatoriamente o conjunto original de dados e garantir que a porcentagem de divisão das classes seja mantida para ambos conjuntos de treinamento e validação, processo que é denominado estratificação, evitando o cenário descrito anteriormente e mantendo a imparcialidade do modelo. Terminado esse processo, após obter os erros estimados, é possível comparar os diferentes classificadores resultantes e realizar a escolha de qual aparenta ser o mais adequado à ser empregado para a resolução problema.

2.4 Floresta de Caminhos Ótimos

Assim como os classificadores apresentados no item 2.3.1, a Floresta de Caminhos Ótimos, do inglês *Optimum-Path Forest (OPF)*, também é um classificador utilizado para predição de dados. Porém, ao contrário daqueles descritos no item mencionado, ele é considerado um *framework* e não simplesmente um classificador singular.

O OPF, desenvolvido por Papa et al. (PAPA; FALCAO; SUZUKI, 2009), é um classificador multiclasse eficiente na fase de treinamento e eficaz na etapa de classificação. Sua ideia principal baseia-se na segmentação do espaço de características. Como foi abordado anteriormente, os trabalhos que visam solucionar o problema da detecção de intrusão normalmente têm como objetivo melhorar a taxa de detecção e diminuir a taxa de alarmes falsos; sendo assim, acredita-se que o desempenho desse algoritmo pode destacar-se perante os demais devido às suas características particulares nas fases de treinamento e classificação, que são descritas detalhadamente ao longo desta seção.

Este classificador é considerado um *framework* pois pode ser aplicado em problemas de aprendizado semi-supervisionado, não-supervisionado e supervisionado. Basicamente, o OPF é um grafo e, considerando que se trate de um problema supervisionado, pode trabalhar com duas relações de adjacência diferentes: grafo completo ou grafo *knn*. As diferenças entre estes dois métodos são a relação de adjacência, o procedimento para estimar os protótipos - que são como nós-semente que representam uma classe específica e conquistam outros nós - e a função

de custo do caminho. Na metodologia deste classificador, o problema de reconhecimento de padrões é retratado como um problema de partição de um grafo.

No que diz respeito ao trabalho apresentado, foi implementada a versão do OPF supervisionado com a arquitetura de grafo completo, que é a versão original do algoritmo e também a mais utilizada (FERNANDES; PAPA, 2019; FILHO et al., 2019). O conceito primário da implementação do algoritmo de grafo completo como relação de adjacência é a construção de um grafo completo, onde quaisquer duas amostras necessariamente estão conectadas. Neste caso, os nós do grafo representam o vetor de características de uma amostra e suas arestas conectam todos os nós entre si; estas arestas são ponderadas por uma função de distância, que tradicionalmente é uma função de distância euclidiana.

Em relação aos protótipos citados, eles também são nós do grafo, mas possuem um papel especial no funcionamento do algoritmo pois são considerados como nós-semente do grafo em questão, representando uma classe específica e conquistando os demais nós. Os protótipos são definidos através das *Minimum Spanning Trees (MST)*⁹, que encontram, através da seleção de amostras localizadas nas fronteiras de classes¹⁰, aquelas amostras mais próximas das classes diferentes. Depois de estabelecer os protótipos, os mesmos competem para conquistar os nós adjacentes enquanto buscam pelo melhor caminho, aquele de menor custo, definido pela função de custo de caminho. Ao término desse procedimento, *Optimum-Path Trees (OPT)* são geradas. Por fim, durante a etapa de teste, o OPF calcula a acurácia de detecção, onde cada amostra nova (proveniente do conjunto de testes) é inserida no grafo completo finalizado da fase de treinamento, e o protótipo que oferecer o caminho de menor custo será aquele que irá conquistá-la; este é o processo que classifica as amostras com seu rótulo, no caso 'normal' ou 'anômalo'.

A Figura 8 ilustra estes procedimentos descritos, onde:

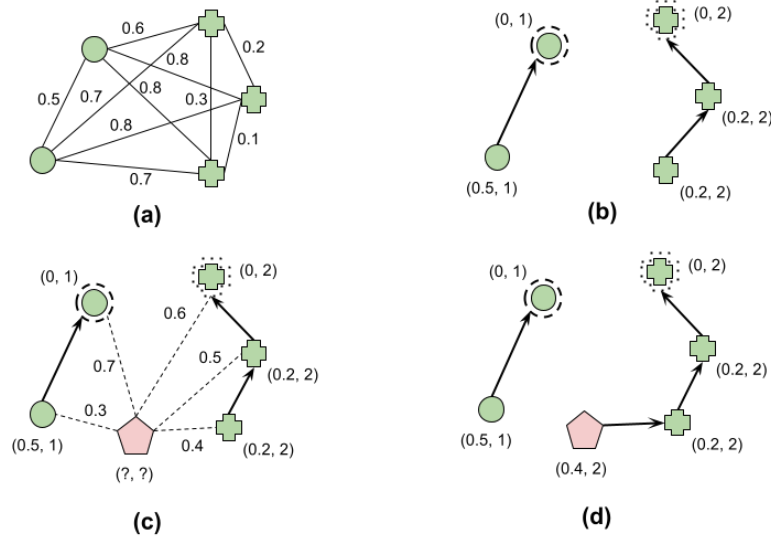
- a) Grafo completo finalizado, dado um conjunto de treinamento;
- b) Florestas de Caminhos Ótimos (*Optimum-Path Forests*) resultantes para $f_{\max}$ e dois protótipos definidos (nós destacados). As anotações (x, y) representam, respectivamente, o custo e o rótulo das amostras. As setas indicam os nós predecessores no caminho ótimo;
- c) Amostra de teste (pentágono vermelho) inserido no grafo completo e suas conexões (linhas pontilhadas) e;
- d) O caminho ótimo do protótipo mais fortemente conectado é o de rótulo 2, e o custo de classificação 0.4 é atribuído à amostra de teste. Essa amostra teste é então classificada como "cruz" (rótulo 2).

Considere $\mathcal{Z}$ um conjunto de dados, onde $\mathcal{Z} = \mathcal{Z}_1 \cup \mathcal{Z}_2$, e $\mathcal{Z}_1$ e $\mathcal{Z}_2$ representam os

⁹ MSTs são subgrafos que conectam todos os nós de um mesmo conjunto utilizando o mínimo custo possível.

¹⁰ Regiões mais prováveis de haver erros de classificação.

Figura 8 – Representação visual dos procedimentos do algoritmo OPF descritos.



Fonte: Elaborado pelo autor

conjuntos de treinamento e teste, respectivamente. Cada amostra $s \in \mathcal{Z}$ pode ser representada por seu vetor de característica $\vec{v}(s) \in \mathbb{R}^n$. O grafo OPF_{cpl} é representado por $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, onde $\mathcal{A}$ refere-se ao conjunto de arestas que conectam todos pares de nós e $\mathcal{V}$ é o conjunto de vetores de características $\vec{v}(s)$, $\forall s \in \mathcal{Z}$. Também, considere $\lambda(\cdot)$ uma função que atribui um rótulo real para cada amostra em $\mathcal{Z}$.

2.4.1 Fase de treinamento

Considere que o grafo $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{A})$ seja derivado do conjunto de treinamento, onde $\mathcal{V}_1$ detém todos vetores de características das amostras pertencentes ao conjunto de treinamento. O primeiro objetivo da fase de treino é obter um conjunto de protótipos $\mathcal{S}$, onde $\mathcal{S} \subset \mathcal{Z}_1$. Considere um caminho π_s , terminando em $\mathbf{s}$, estar em $\mathcal{G}$ e uma função $f(\pi_s)$ que associa um valor a esse caminho. Para que um protótipo conquiste amostras adjacentes, o propósito é minimizar $f(\pi_s)$ através de uma função de custo dos caminhos dada pela seguinte equação:

$$f_{max}(\langle \mathbf{s} \rangle) = \begin{cases} 0 & \text{se } \mathbf{s} \in \mathcal{S}, \\ +\infty & \text{caso contrário} \end{cases}$$

$$f_{max}(\pi \cdot \langle \mathbf{s}, \mathbf{t} \rangle) = \max\{f_{max}(\pi), d(\mathbf{s}, \mathbf{t})\}. \quad (1)$$

onde $f_{max}(\pi \cdot \langle \mathbf{s}, \mathbf{t} \rangle)$ computa a máxima distância entre amostras adjacentes $\mathbf{s}$ e $\mathbf{t}$ ao longo do caminho $\pi \cdot \langle \mathbf{s}, \mathbf{t} \rangle$, que representa um caminho direto sem quaisquer nós intermediários no trajeto. Um caminho π_s é referido como ótimo se $f(\pi_s) \leq f(\tau_s)$ para qualquer outro caminho τ_s .

A minimização de f_{max} atribui para cada amostra $\mathbf{t} \in \mathcal{Z}_1$ um caminho ótimo $P^*(\mathbf{t})$, cujo custo mínimo $C(\mathbf{t})$ é dado pela seguinte equação:

$$C(\mathbf{t}) = \min_{\forall \pi_t \in (\mathcal{Z}_1, \mathcal{A})} \{f_{max}(\pi_t)\}. \quad (2)$$

O processo de minimização de f_{max} atribui um caminho ótimo de S para cada amostra de treinamento $\mathbf{t}$ em uma ordem ascendente de custo mínimo, tal que o grafo é particionado em uma floresta de caminhos ótimos P . A raiz $R(\mathbf{t}) \in S$ de $P^*(\mathbf{t})$ pode ser obtida através de $P(\mathbf{t})$ ao seguir a relação dos predecessores ao longo do caminho.

Ao computar uma MST no grafo completo, é obtido um grafo acíclico conectado cujos nós são todos amostras de $\mathcal{Z}_1$ e as arestas são não-direcionais e com pesos das distâncias d entre os nós adjacentes. Dessas MSTs, uma árvore é considerada ótima se a soma do peso de suas arestas é mínima ao ser comparada com qualquer outra árvore no grafo completo. Como mencionado anteriormente, nas MSTs cada par de amostras é conectado através de um caminho único, que é ótimo de acordo com f_{max} , portanto a MST contém uma árvore de caminho ótimo para cada nó-semente selecionado.

Os protótipos ótimos são os elementos mais próximos das MSTs com diferentes rótulos em $\mathcal{Z}_1$. Ao remover as arestas entre classes diferentes, seus nós adjacentes tornam-se protótipos em S^* e o algoritmo pode computar uma floresta de caminhos ótimos em $\mathcal{Z}_1$, cuja complexidade computacional de encontrar uma MST é quadrática ($O(N^2)$). É importante ressaltar que uma classe pode ser representada por múltiplos protótipos e que deve existir pelo menos um protótipo por classe.

2.4.2 Fase de teste

O grafo do conjunto de teste $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{A})$ é composto por amostras $\mathbf{t} \in \mathcal{V}_2$. Cada amostra $\mathbf{t}$ é conectada à uma amostra $\mathbf{s} \in \mathcal{V}_1$, fazendo $\mathbf{t}$ parte do grafo original. O objetivo é encontrar um caminho ótimo $P^*(\mathbf{t})$ de S para $\mathbf{t}$ com classe $\lambda(R(\mathbf{t}))$ de seu protótipo $R(\mathbf{t}) \in S$. Consequentemente, a amostra $\mathbf{t}$ é removida do grafo. Este caminho pode ser identificado avaliando o valor de custo ótimo $C(\mathbf{t})$:

$$C(\mathbf{t}) = \min\{\max\{C(\mathbf{s}), d(\mathbf{s}, \mathbf{t})\}\}, \forall \mathbf{s} \in \mathcal{Z}_1. \quad (3)$$

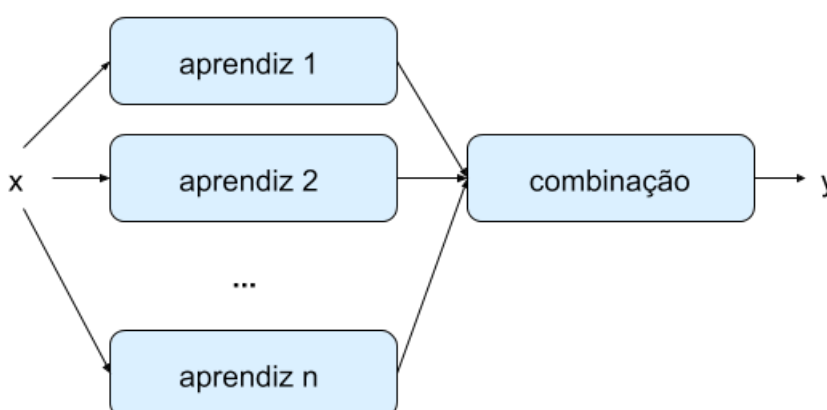
2.5 Métodos conjuntos

Métodos conjuntos treinam múltiplos aprendizes que visam solucionar um mesmo problema. O que os difere das abordagens singulares de aprendizagem, citadas no item 2.3.1, que tentam construir um único aprendiz a partir dados de treinamento, é que estes métodos

buscam a construção de um conjunto de n aprendizes e então combinam-os para obter um classificador conjunto.

Como foi abordado no item 2.3.2, avaliar e escolher o melhor entre diferentes classificadores singulares é uma tarefa complicada, e é exatamente aí que sustenta-se a idealização dos métodos conjuntos, no qual a implementação visa produzir, através da combinação de múltiplos aprendizes, resultados melhores do que os produzidos por classificadores singulares sozinhos (HAQ et al., 2015).

Figura 9 – Arquitetura genérica de um método conjunto



Fonte: Elaborado pelo autor

A Figura 9 apresenta a arquitetura genérica de um método conjunto, onde o conjunto contém um certo número de aprendizes, que são chamados de aprendiz base, ou também aprendiz fraco. Tais aprendizes normalmente são o resultado de um processo de treinamento, como descrito anteriormente, por um algoritmo de classificação, e.g. *Decision Tree*, *Logistic Regression* ou outros tipos de algoritmos de aprendizagem existentes, que vão além daqueles citados previamente. A maioria dos métodos conjuntos utiliza um único algoritmo de classificador singular para produzir aprendizes de base homogêneos, ou seja, aprendizes do mesmo tipo, o que resulta em conjuntos homogêneos. Há também alguns métodos que utilizam diferentes algoritmos de aprendizagem para produzir aprendizes heterogêneos, ou seja, aprendizes de diferentes tipos, levando à conjuntos heterogêneos.

Sendo assim, a habilidade de generalização de um classificador conjunto normalmente é maior do que a dos aprendizes de base. Na realidade, os métodos conjuntos são atraentes principalmente pelo fato de que eles são capazes de aprimorar os aprendizes fracos, que são apenas um pouco melhores do que simples deduções aleatórias, para aprendizes fortes, capazes de fazer previsões muito mais precisas.

Pensando em um contexto histórico, e de onde surgiram, existem três contribuições iniciais que levaram os métodos de conjunto ao seu ponto atual; sendo eles, combinação de classificadores, conjuntos de aprendizes fracos e mistura de especialistas. Especificamente,

aquele que está mais ligado à área de ML, e foi o mais estudado pela comunidade, é o conjunto de aprendizes fracos. Nessa linha de pesquisa, pesquisadores costumam trabalhar com aprendizes fracos e buscam desenvolver algoritmos robustos para impulsionar sua performance de fraco para forte. Portanto, foi nessa linha de trabalho que nasceram métodos conjuntos conceituados como *Boosting* (SCHAPIRE, 1990), *Bagging* (BREIMAN, 1996a), *Stacking* (WOLPERT, 1992), entre outros, e o entendimento teórico sobre como e porquê aprendizes fracos conseguem ser aprimorados para fortes.

Os métodos conjuntos tornaram-se um grande paradigma de aprendizado a partir dos anos noventa, com o fomento de dois trabalhos pioneiros. Um é empírico (HANSEN; SALAMON, 1990), onde foi descoberto que predições feitas pela combinação de um grupo de classificadores normalmente são mais precisas do que predições feitas pelo melhor classificador singular. Outro é teórico (SCHAPIRE, 1990), no qual foi provado que aprendizes fracos podem ser impulsionados para aprendizes fortes. Uma vez que aprendizes fortes são desejáveis porém difíceis de se obter, enquanto aprendizes fracos são de fácil obtenção na prática, estes resultados mostraram um caminho promissor em direção à geração de aprendizes fortes através de métodos conjuntos.

Para obter os classificadores conjuntos, geralmente necessita-se de duas etapas em sua construção, que são a criação dos aprendizes de base e a combinação dos mesmos. Logo, para conseguir um método conveniente, acredita-se que os aprendizes base devem ser o mais exatos possíveis, e até diversificados também. Além disso, vale ressaltar que, normalmente, o custo computacional da construção de um conjunto não é muito maior do que criar um único aprendiz. Isso acontece porque quando deseja-se utilizar só um aprendiz, normalmente é necessário gerar múltiplas versões dele para selecionar o modelo ou ajustar os parâmetros adequados; tal processo se compara à geração dos aprendizes de base em métodos conjuntos, enquanto o custo computacional para combiná-los costuma ser menor, devido ao fato de que a maioria das estratégias de combinação são simples.

2.5.1 Benefícios da combinação

Após gerar um conjunto de aprendizes de base, ao invés de ficar tentando encontrar o melhor aprendiz singular, os métodos conjuntos recorrem à combinação para alcançar uma sólida capacidade de generalização, na qual a combinação demonstra-se essencial. Dietterich atribuiu os benefícios provenientes da combinação à três questões fundamentais (DIETTERICH, 2000):

- Problema estatístico: é comum que o espaço da hipótese seja muito grande para ser explorado em dados limitados de treinamento, e também que haja várias hipóteses diferentes expondo a mesma precisão nos dados de treinamento. Se o algoritmo de treinamento escolhe alguma destas hipóteses, há o risco de que uma hipótese escolhida

imprecisamente não consiga prever os dados futuros corretamente. Sendo assim, ao combinar as hipóteses, o risco de escolher uma hipótese errada pode ser reduzido;

- Problema computacional: muitos algoritmos de aprendizagem realizam uma espécie de busca local que pode acabar emperrada. Mesmo que haja dados de treinamento suficientes, ainda pode ser que seja difícil encontrar a melhor hipótese. Ao executar a busca local a partir de vários pontos de partida diferentes, a combinação pode fornecer uma aproximação melhor à hipótese verdadeira desconhecida. Logo, ao combinar as hipóteses, o risco de escolher um local mínimo errado pode ser reduzido; e
- Problema representacional: em diversas tarefas de ML, a hipótese verdadeira desconhecida não pode ser representada através de nenhuma hipótese no campo de hipóteses. Então, ao combinar as hipóteses, pode ser possível expandir o espaço de funções representáveis, e assim o algoritmo de aprendizagem pode ser capaz de formar uma aproximação mais exata da hipótese verdadeira desconhecida.

Estes três problemas estão entre os fatores mais importantes pelos quais os métodos tradicionais de aprendizado falham. Sendo assim, através da combinação, o erro dos algoritmos pode ser reduzido; isto foi confirmado em diversos estudos empíricos (XU; KRZYZAK; SUEN, 1992) (BAUER; KOHAVI, 1999) (OPITZ; MACLIN, 1999).

2.6 *Stacking*

Essencialmente, os métodos conjuntos podem ser categorizados como uma seleção de classificadores combinados e focados em resolver um único problema. Sua maior diferença em relação aos classificadores singulares é a utilização de diversos classificadores unidos, ao invés de um sozinho (HANSEN; SALAMON, 1990). Um classificador conjunto é composto de vários aprendizes fracos, majoritariamente classificadores básicos, como *Decision Tree*, *Support Vector Machines*, entre outros. Logo, a ideia por trás da implementação de um método conjunto é combinar aprendizes fracos para criar um único aprendiz forte, robusto.

A capacidade de generalização de um classificador conjunto costuma ser melhor que a dos classificadores singulares, principalmente porque esses aprendizes fracos são simples, sendo apenas um pouco melhores que meras deduções aleatórias. Apesar de sozinhos serem considerados como classificadores "básicos", aprendizes fracos, ao serem combinados estes classificadores demonstram-se capazes de produzir ótimas previsões (SCHAPIRE, 1990), e é nesse conceito que se baseia a metodologia conjunta. No fim das contas, aprendizes fortes são os desejados, no entanto os mais difíceis de serem obtidos.

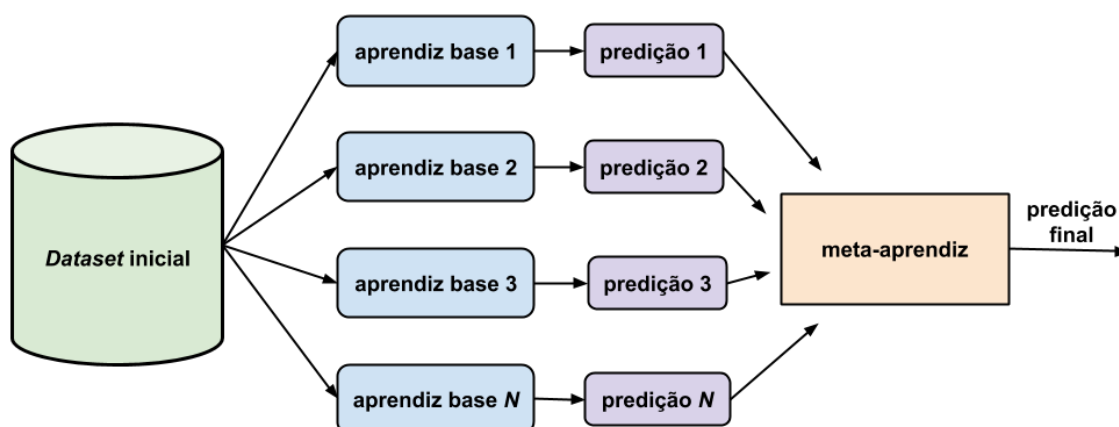
O algoritmo de *Stacking* (WOLPERT, 1992) é um método conjunto no qual um aprendiz é treinado para combinar diversos aprendizes individuais. Esta técnica tem como propósito aprimorar a capacidade de generalização de um modelo, enquanto outras técnicas

de métodos conjuntos buscam, por exemplo, diminuir a variância do modelo ou produzir o modelo menos enviesado possível. Seu funcionamento embasa-se no conceito de aprendizado por camadas, de forma sequencial, onde os *outputs* da primeira camada servem como *inputs* da segunda, a final. Há também uma variação deste algoritmo, onde é possível implementar n camadas, todas seguindo essa mesma relação entre *output* e *input* entre elas.

Nessa metodologia, os classificadores singulares são chamados de aprendizes de camada, ou aprendizes base, e estão localizados na primeira camada do conjunto, onde recebem como *input* os vetores de característica das amostras, provenientes do *dataset* adotado. O classificador responsável pela combinação dos aprendizes base é denominado meta-aprendiz, e ele recebe como *input* os *outputs* da primeira camada, que são, no caso, as previsões feitas pelos aprendizes base recém treinados. O processo de treinamento do meta-aprendiz atua como uma combinação do voto majoritário das previsões de todos aprendizes base, e seu resultado é a previsão final do modelo conjunto. Portanto, para construir o algoritmo *Stacking* é necessário definir dois elementos chave: os n aprendizes base à serem treinados e o meta-aprendiz, ou meta-classificador, que irá combiná-los.

Após escolher os aprendizes base, que são os classificadores simples mencionados anteriormente, é preciso particionar o *dataset* para o treinamento. Esse procedimento de divisão é essencial pois as previsões feitas em dados que forem utilizados para treinar estes primeiros aprendizes não são relevantes para realizar o treinamento do meta-aprendiz e portanto não podem ser iguais. Basicamente, os aprendizes base serão treinados usando parte do *dataset* original, porém reservando uma parte não-treinada do mesmo *dataset* para o treinamento do meta-aprendiz (BREIMAN, 1996b). Se os mesmos dados forem utilizados para treinar ambos aprendizes base e meta-aprendiz, há um grande risco de sobreajuste na classificação final, comprometendo todo o procedimento.

Figura 10 – Algoritmo genérico do método conjunto de *Stacking*



Fonte: Elaborado pelo autor

Na Figura 10 observa-se o esboço do funcionamento do algoritmo de *Stacking*, onde há o *dataset* inicial, a ser particionado conforme progridem as etapas do algoritmo; a primeira camada, composta de n aprendizes base que recebem, como *input*, dados oriundos do *dataset* para seu treinamento e produzem, como *output*, as previsões à serem consumidas na próxima camada; por fim, há o meta-aprendiz, recebendo como *input* a previsão dos aprendizes da primeira camada e realizando a previsão final do conjunto. Tipicamente, em casos de problemas de classificação, como é o enfrentado nesse projeto, o meta-aprendiz aplica um processo de voto majoritário com pesos para agregar as previsões dos aprendizes base.

A primeira camada de aprendizes base pode ser constituída de diferentes tipos de classificadores simples, o que caracteriza o conjunto como heterogêneo, ou de um único classificador, repetido n vezes, caracterizando o conjunto como homogêneo. Wolpert (WOLPERT, 1992) afirma que a implementação principal do algoritmo de *Stacking* é visando a combinação de diferentes classificadores (heterogêneo), embora possa também ser utilizada com apenas um tipo de classificador (homogêneo), funcionando como uma técnica de aprimoramento do mesmo. Ambos métodos são implementados nesse trabalho, sendo o primeiro alcançado através da junção de diferentes classificadores singulares, e.g., *Decision Tree*, *Random Forest* e OPF; e o último obtido por meio da combinação do mesmo tipo de classificador, e.g., um conjunto composto somente de classificadores OPF.

Como o *Stacking* é um modelo aplicado com o intuito de melhorar a capacidade de generalização, ele demonstra-se relevante ao trabalho, uma vez que o problema enfrentado é a detecção de intrusão. Como já foi abordado no Capítulo 2, essa área constantemente busca maneiras de aprimorar as taxas de detecção e diminuir os erros de previsão dos SDIs. Sendo assim, implementar um método que fundamenta-se em melhorar a capacidade de generalização dos modelos mostra-se adequado.

Para analisar a competência dessa metodologia na prática, o trabalho apresentado implementa três experimentos, todos apoiando-se no algoritmo de *Stacking* e no classificador OPF. Os experimentos serão descritos detalhadamente no próximo capítulo.

2.7 Aplicações dos métodos conjuntos

Com o passar dos anos, cada vez mais essa metodologia conjunta começou a ser empregada, e consequentemente testada, em inúmeros campos, apresentando resultados interessantes.

Existem diversas competições científicas que acontecem regularmente e, entre elas, a *KDD-Cup* é a mais famosa em termos de mineração de dados. Desde 1997, ela acontece todos os anos e atrai o interesse de equipes de mineração de dados do mundo todo. Os desafios da competição englobam uma grande variedade de tarefas práticas, como detecção de intrusão de redes (1999), detecção de embolias pulmonares (2006), recomendação de música (2011),

previsão de qualidade do índice do ar (2018), entre outras. Analisando o histórico da *KDD-Cup*, entre as inúmeras técnicas utilizadas nas soluções para enfrentar as tarefas anuais, métodos conjuntos chamaram a atenção várias vezes e também ganharam a disputa muitas vezes.

Em adição aos resultados notáveis em competições, os métodos conjuntos também têm sido aplicados em várias tarefas do mundo real. Na realidade, eles foram úteis em praticamente todos os lugares onde é possível explorar o uso de técnicas de ML.

Viola e Jones (VIOLA; JONES, 2001) (VIOLA; JONES, 2004) sugeriram um framework de detecção de objetos por meio da combinação do algoritmo *AdaBoost* com uma arquitetura *cascade*. Este *framework* foi reconhecido como um dos maiores avanços no campo de visão computacional, especialmente em detecção facial, da última década. Além disso, Li et al. (LI et al., 2001) também empregaram um método de classificadores conjuntos para detecção facial múltipla, superando técnicas convencionais.

Os classificadores conjuntos também têm sido muito úteis em diversas tarefas de diagnóstico médico auxiliado por computação, especialmente para aumentar a confiabilidade do diagnóstico. Zhou e outros (ZHOU et al., 2002) desenvolveram um arquitetura conjunta para identificação de células pulmonares cancerígenas; seu trabalho descreveu resultados com maior taxa de identificação e menor taxa de falsos-negativos. Polikar e outros (POLIKAR et al., 2008) propuseram um método de conjunto para detecção antecipada da doença de Alzheimer.

Além disso, essa metodologia também demonstrou-se muito apta na caracterização de problemas de segurança de computadores, pois cada atividade realizada em sistemas computacionais pode ser observada em níveis múltiplos de abstração, e a informação relevante pode ser coletada a partir de diversas fontes de informação (CORONA et al., 2009). Giacinto et al. (GIACINTO; ROLI; DIDACI, 2003) aplicaram métodos conjuntos para detecção de intrusão. Sua proposta, ao considerar que existem diferentes tipos de características que categorizam a conexão, construiu um conjunto para cada tipo de características e então combinou os resultados destes conjuntos para produzir uma decisão final. Seu estudo relatou que, para detecção de ataques conhecidos, classificadores conjuntos foram responsáveis por melhores performances que singulares. Anos depois, Giacinto et al. (GIACINTO et al., 2008) propuseram um método conjunto para detecção de intrusão baseada em anomalias. Schultz et al. (SCHULTZ et al., 2001) apresentaram um método conjunto para detectar *malwares* desconhecidos automaticamente.

Finalizando, é importante ressaltar que, além de visão computacional, segurança de computadores e diagnóstico médico auxiliado por computação, os métodos conjuntos também têm sido aplicado em muitas outras áreas e tarefas, como detecção de fraude de cartão de crédito (PANIGRAHI et al., 2009), previsão de falência (WEST, 2004), previsão de distribuições de espécies (ARAUJO; NEW, 2007), diagnóstico de falha de motor de avião (YAN; XUE, 2008), predição da temperatura superficial de estradas (LIU et al., 2017), reconhecimento de acentuação tônica na fala (PRASETIO; TAMURA; TANNNO, 2018), detecção de relatórios

financeiros fraudulentos (SEEMAKURTHI; ZHANG; QI, 2015), entre muitos outros. No que tange especificamente o problema de detecção de intrusão, o capítulo seguinte apresenta trabalhos que são relacionados à este.

3 Trabalhos correlatos

Este capítulo exhibe os trabalhos correlatos que foram analisados após uma extensa revisão literária sobre o problema da detecção de intrusão. Os trabalhos que são apresentados aqui tiveram suas respectivas metodologias examinadas minuciosamente, as quais serviram como suporte e auxiliaram as decisões tomadas para que este trabalho chegasse à sua metodologia experimental final, baseando-se no que as pesquisas atuais e a comunidade científica estabelece como referência.

Fica evidente que sistemas de detecção de intrusão não são controles lógicos obsoletos, pelo contrário, são empregados até hoje e compõem a arquitetura de segurança da informação da maioria das organizações. Também foi discutido que a área de Aprendizado de Máquina tem sido cada vez mais explorada, especialmente sua ramificação que contempla a aplicação de métodos conjuntos para alcançar resultados melhores que classificadores singulares simples. Uma vez que infere-se que não existem algoritmos de aprendizado perfeitos, pois todos possuem vantagens e desvantagens inerentes, a abordagem dos métodos conjuntos, que visa o aprimoramento através da combinação de robustez e diminuição de fraquezas, tem sido estudada, desenvolvida e implementada cada vez mais pelos cientistas. A seguir são exibidos diversos trabalhos com o propósito de expor que a implementação de sistemas de detecção de intrusão baseados em anomalia com a implementação de métodos conjuntos representam o estado-da-arte no contexto do problema de detecção de intrusão.

Timčenko e Gajin (TIMČENKO; GAJIN, 2017) analisaram diferentes tipos de algoritmos de métodos conjuntos, como os algoritmos *Bagged Tree*, *AdaBoost*, *GentleBoost*, *LogitBoost* e *RUSBoost*. Seu intuito foi obter informações sobre qual das metodologias de aprendizado conjunto observadas seria mais efetiva em relação ao problema de detecção de intrusão, considerando *datasets* desbalanceados. Mirza (MIRZA, 2018) implementou três classificadores (*Logistic Regression*, Redes Neurais e *Decision Tree*) que possuem características distintas e aperfeiçoou sua performance geral através da implementação de métodos conjuntos. Tama et al. (TAMA; PATIL; RHEE, 2017) apresentou um modelo de detecção de anomalia aprimorado utilizando um método conjunto de duas camadas e uma seleção de características baseada no algoritmo de *Particle Swarm Optimization*. Os experimentos revelaram que o modelo proposto superou os demais classificadores comparados.

No que diz respeito exclusivamente à implementação de métodos conjuntos com o algoritmo de *Stacking*, Chand et al. (CHAND et al., 2016) compararam o desempenho de nove conjuntos diferentes, os quais todos implementaram a metodologia *Stacking* e tinham em comum o classificador SVM como meta-aprendiz do modelo; neste experimento, alguns dos conjuntos *stack* obtiveram melhores resultados que o classificador SVM tradicional, singular.

Hsu et al. (HSU et al., 2019) trabalharam com a implementação de um conjunto *Stacking* composto de um *auto-encoder* e dois classificadores (SVM e RF); seu conjunto alcançou resultados com maior acurácia, precisão e *recall* ao ser comparado com os classificadores singulares.

Em relação a trabalhos mais recentes, Asrafi et al. (ASRAFI et al., 2020) propuseram um modelo *stack* que combina os algoritmos conjuntos de *Stacking* e *Boosting* para o problema da classificação de *malware*. Foram implementados sete algoritmos diferentes para aprendizado, incluindo k-NN, RF, *K-means*, *Perception*, *Multilayer Perceptron*, SVM e *Principal Component Analysis*, e três algoritmos de *Boosting*: *XGBoost*, *AdaBoost* e *Gradient Boosting* para testar os dados. Os resultados obtidos foram satisfatórios, demonstrando a maior acurácia (99.2%) com o modelo conjunto baseado em *stack* MLP-Adaboost.

Yang et al. (YANG et al., 2020) também apresentaram um modelo baseado no algoritmo *Stacking* para o problema de detecção em canais DNS; seu método conjunto combina três algoritmos diferentes como aprendizes-base: k-NN, SVM e RF. Em relação a avaliação do modelo, os resultados experimentais demonstraram em sua Curva Característica de Operação do Receptor (do inglês, *Receiver Operating Characteristic Curve*), ou simplesmente curva ROC, que o modelo *stack* proposto atingiu 0.9901, o que representa uma melhoria quando comparado aos demais métodos avaliados na pesquisa. O modelo proposto também exibiu menos alarmes falsos, maior habilidade para identificar o tráfego nos canais DNS e menor taxa de falsos-positivos. Consequentemente, a metodologia *Stacking* combinada com os aprendizes base obtiveram um desempenho excelente nessa tarefa, aprimorando a acurácia geral de detecção e reduzindo a taxa de falsos-positivos.

Outra abordagem para o problema de detecção de intrusão em redes foi proposta por Rajagopal et al. (RAJAGOPAL; KUNDAPUR; HAREESHA, 2020), onde foi implementado um *framework Stacking* composto pelos classificadores k-NN, LR e RF como aprendizes-base e o classificador SVM como meta-aprendiz do modelo. Os resultados empíricos indicaram que o modelo *stack* proposto foi capaz de realizar previsões melhores em um *dataset* em tempo real (97% acurácia) do que em um *dataset* emulado (94% acurácia); além disso, ele também obteve taxas consideravelmente baixas de alarmes falsos para todas categorias de ataque analisadas, o que é uma indicação de que o desempenho geral do modelo é satisfatório.

Quanto ao problema de detecção de intrusão em redes *wireless*, Rajadurai e Gandhi (RAJADURAI; GANDHI, 2020) propuseram um modelo conjunto baseado em *Stacking* apropriado para SDIs detectarem ataques e também classificá-los em cinco categorias diferentes. Este conjunto baseado em *Stacking* implementa os algoritmos *Gradient Boosting* e RF como seus aprendizes-base. O modelo proposto foi comparado com diversos algoritmos de aprendizado de máquina e modelos de redes neurais. Analisando os resultados, o modelo *stack* obteve melhor desempenho que os outros métodos conjuntos observados, como *Bagging* e *Boosting*, e também foi possível verificar que, quando comparado aos demais modelos de aprendizado de

máquina e redes neurais, o *stack* proposto performou melhor.

Musa et al. (MUSA et al., 2020) conduziram uma revisão literária sobre SDIs que empregam técnicas de Aprendizado de Máquina e concluiu que, entre os 20 artigos avaliados, todos dentro do período de 2015 a 2020, modelos conjuntos e híbridos têm sido capazes de superar suas versões tradicionais, de classificadores singulares, apresentando melhor acurácia preditiva e taxas de detecção, e portanto deveriam ser utilizados com mais frequência.

Além disso, Amarudin et al. (AMARUDIN; FERDIANA; WIDYAWAN, 2020) também administraram uma revisão literária sistemática sobre SDIs e Aprendizado de Máquina, na qual 62 artigos, publicados entre 2016 e 2020, foram examinados. Eles constataram que, dentre diversas pesquisas que propuseram técnicas e métodos para melhorar a acurácia dos modelos nos SDIs, muitas atingiram esse objetivo através da implementação de métodos conjuntos, tais como *Boosting*, *Bagging* e *Stacking*, e portanto trabalhos futuros deveriam aproveitar-se da metodologia conjunta para abordar os problemas de classificação e melhorar a acurácia na detecção de intrusão.

Na Tabela 1, são referenciados os artigos citados neste capítulo e informações sobre os mesmos; que são, o título e ano da publicação, o algoritmo proposto, o *dataset* empregado para treinamento e validação dos experimentos, o resultado principal obtido, e por fim, descobertas e desvantagens observadas.

Observando os trabalhos apresentados, é possível verificar que os Sistemas de Detecção de Intrusão baseados em anomalia são um objeto de contínuo estudo na área, estando em permanente busca pela melhoria de sua performance, almejando sempre aperfeiçoar a capacidade de detecção de atividades suspeitas e reduzir a quantidade de predições incorretas.

Também, técnicas de ML estão diretamente relacionadas à estes SDIs, uma vez que elas constituem parte essencial da arquitetura desses sistemas, tornando-os capazes de prever e detectar ataques desconhecidos, e também modificados. Observa-se que, atualmente, os métodos conjuntos estão presentes na vasta maioria das implementações de ML e que agregam muito valor, pois foram capazes de melhorar seu desempenho em várias das implementações, quando comparados aos classificadores singulares tradicionais. Nota-se também que não há apenas uma única maneira de empregar os métodos conjuntos: os trabalhos distinguem-se através dos classificadores singulares escolhidos para compor o modelo final, o algoritmo de aprendizado conjunto selecionado e também pelo ajuste de parâmetros. Outra coisa que é comum a todos os trabalhos citados é a utilização de um *dataset* para treinamento e validação dos experimentos; baseando-se no que foi apresentado, observa-se que os conjuntos de dados utilizados com maior frequência são *NSL-KDD*, *KDD Cup'99* e *UNSW-NB15*.

Por fim, analisando os trabalhos que foram apresentados ao longo desse capítulo e atentando-se às suas "desvantagens", representadas na última coluna da Tabela 1, é possível constatar que há espaço para melhora da implementação do algoritmo de *Stacking*, uma vez que

nem todos resultados alcançados foram satisfatórios. Portanto, o capítulo seguinte apresenta a metodologia adotada nesse trabalho, visando atender essa questão da eficiência dos algoritmos que são empregados para solucionar o problema de detecção de intrusão. Vale ressaltar que a metodologia experimental que será apresentada não é idêntica às que acabam de ser exibidas nesse capítulo.

Tabela 1 – Resumo dos artigos referenciados no capítulo.

Título	Algoritmo	Dataset	Resultado	Descoberta	Desvantagem
Ensemble classifiers for supervised anomaly based network intrusion detection (TIMČENKO; GAJIN, 2017)	Métodos conjuntos de <i>Bagged Tree</i> , <i>AdaBoost</i> , <i>LogitBoost</i> , <i>GentleBoost</i> e <i>RUSBoost</i> , todos empregando classificadores DT como aprendizes-base	UNSW-NB15	Acurácia <i>Bagged Tree</i> = 92.92%; Acurácia <i>GentleBoost</i> = 91.7%	Experimentos demonstraram que <i>Bagged Tree</i> funcionou como uma técnica de redução de variância, fornecendo resultados ótimos.	Algoritmo <i>RUSBoost</i> tem a vantagem de ser simples computacionalmente, mas apresentou o desempenho menos eficiente.
Computer Network Intrusion Detection using various Classifiers and Ensemble Learning (MIRZA, 2018)	Modelo baseado em voto majoritário aplicado aos algoritmos DT, LR e Redes Neurais.	KDD Cup 99	Acurácia do conjunto proposto = 96.14%	O desempenho geral foi aprimorado através da implementação de algoritmos de métodos conjuntos.	O <i>dataset</i> utilizado não possui uma boa distribuição entre amostras normais e anômalas.
An Improved Model of Anomaly Detection using Two-level Classifier Ensemble (TAMA; PATIL; RHEE, 2017)	Conjunto de dois níveis que emprega <i>Random Subspace Method</i> como aprendiz-base e <i>AdaBoost</i> como meta-aprendiz.	NSL-KDD	Acurácia do conjunto = 97.55%	Modelo proposto supera significativamente os demais comparados em termos de acurácia e taxa de alarmes falsos.	Ao comparar os resultados obtidos à três referências desse mesmo artigo, o modelo proposto é pior em termos de taxa de falsos-positivos.

continua na próxima página

A Comparative Analysis of SVM and its Stacking with other Classification Algorithm for Intrusion Detection (CHAND et al., 2016)	Modelo <i>stack</i> que combina os algoritmos <i>BayesNet</i> , <i>AdaBoost</i> , LR, <i>IBK</i> , <i>J48</i> , RF, <i>JRip</i> , <i>OneR</i> e <i>SimpleCart</i>	NSL-KDD	<i>Stacking</i> SVM-RF = 97.5% (maior acurácia)	<i>Stacking</i> dos algoritmos SVM e RF apresenta o melhor desempenho porque o classificador RF é bom para tarefas complexas de classificação	Nem todos classificadores foram capazes de aprimorar a performance do algoritmo SVM quando combinados através da metodologia <i>Stacking</i>
Toward an On-line Network Intrusion Detection System Based on Ensemble Learning (HSU et al., 2019)	Conjunto <i>stack</i> composto pelos algoritmos SVM, RF e <i>autoencoder</i>	1) NSL-KDD, 2) UNSW-NB15, e 3) Dados coletados privadamente	Acurácia do conjunto = 91.7%, 91.89% e 96.1% (respectivamente)	Conjunto proposto apresente alta acurácia para classificação e também é satisfatório para métricas de precisão e <i>recall</i> .	Em relação ao <i>dataset</i> privado, o classificador RF demonstra ser três vezes mais rápido que o conjunto proposto.
Comparing Performance of Malware Classification on Automated Stacking (ASRAFI et al., 2020)	Conjunto <i>stack</i> com os algoritmos k-NN, RF, <i>K-means</i> , <i>Perceptron</i> , MLP, SVM e PCA como aprendizes-base e <i>XGBoost</i> , <i>AdaBoost</i> e <i>GradientBoost</i> como meta-aprendizes	Malware API Call 2019	<i>Stack</i> Adaboost = 99.2% (maior acurácia)	Metodologia <i>Stacking</i> fornece maneiras de empregar diferentes algoritmos para treinamento e teste de dados em diferentes aspectos, e sua combinação melhora o desempenho.	Diferentes métodos de seleção de características poderiam ser explorados para descrever melhor as características do <i>malware</i> , além de <i>APIs</i>
Naruto: DNS Covert Channels Detection Based on Stacking Model (YANG et al., 2020)	Modelo <i>stack</i> que combina três diferentes algoritmos: k-NN, SVM e RF.	Dados coletados de maneira privada	Modelo <i>stack</i> de maior desempenho na curva ROC = 0.9901	Algoritmo <i>Stacking</i> combinado com os aprendizes-base apresentaram desempenho ótimo, melhorando a acurácia de detecção e diminuindo a taxa de falso-positivo.	Diferentes ambientes de rede precisavam de referências diferentes, e isso pode resultar em flutuações na performance das detecções.

continua na próxima página

A Stacking Ensemble for Network Intrusion Detection Using Heterogeneous Datasets (RAJAGOPAL; KUNDAPUR; HAREESHA, 2020)	Modelo <i>stack</i> composto pelos algoritmos RF, LR e k-NN como aprendizes-base e SVM como meta-aprendiz.	1) UNSW NB-15 e 2) UGR'16	UNSW NB-15 = 94% acurácia; UGR'16 = 97.19% acurácia	O modelo proposto fornece uma perspectiva competitiva interessante para detecção de intrusão em redes em tempo real.	Ainda é um desafio melhorar as taxas de detecção das categorias de ataques, como análise, negação de serviço, <i>worms</i> e <i>back-door</i> .
A stacked ensemble learning model for intrusion detection in wireless network (RAJADURAI; GANDHI, 2020)	Conjunto <i>stack</i> que implementa os algoritmos <i>Gradient Boosting Machine</i> e RF.	NSL-KDD	Acurácia do conjunto <i>stack</i> = 91.06%	O modelo proposto é adequado para detectar ataques e também para classificá-los dentre cinco classes diferentes.	As taxas de detecção e <i>recall</i> do modelo proposto para os ataques de <i>Probe</i> e negação de serviço foram piores que dos demais classificadores.

Fonte: Elaborado pelo autor

4 Metodologia

Apresentados todos conceitos que servem de fundamentação para o desenvolvimento desta solução e também os trabalhos relacionados, resta exibir a metodologia experimental que foi adotada para validar o que foi discutido; a metodologia é descrita a seguir. O restante do capítulo divide-se da seguinte maneira: a Seção 4.1 exibe as métricas que são empregadas na apuração dos experimentos; a Seção 4.2 discute sobre a importância da aplicação de um teste estatístico e como ele foi utilizado para validar a comensurabilidade dos modelos finais comparados. Os *datasets* que foram empregados, *uneSPY* e *NSL-KDD*, são descritos na Seção 4.3. Por fim, as últimas três Seções, 4.4, 4.5 e 4.6, descrevem detalhadamente os três experimentos conduzidos. Os resultados experimentais são apresentados no capítulo seguinte.

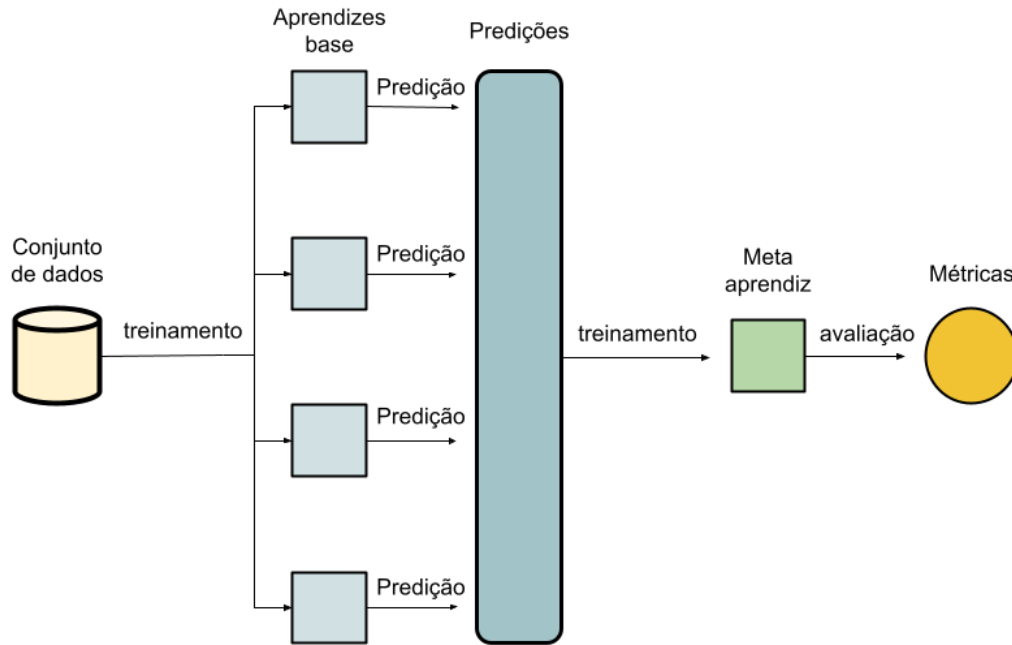
Como foi discutido, existe um problema eminente de segurança da informação, que, dentre n soluções cabíveis, a implementação de controles lógicos demonstra-se eficiente no combate às ameaças. Um controle lógico amplamente utilizado é o sistema de detecção de intrusão, que comumente divide-se entre aqueles baseados em assinatura ou anomalia. Dado que as ameaças enfrentadas pela área de Segurança da Informação estão em metamorfose constante, é necessário empregar uma defesa que consiga reagir às inovações dos invasores; os SDIs baseados em anomalia se destacam nesse quesito, pois, dentre os dois citados, são os únicos capazes de detectar ataques desconhecidos. Tal capacidade destes SDIs somente torna-se possível graças as técnicas de Aprendizado de Máquina, que permitem que o sistema seja treinado a partir de um conjunto de dados conhecido e consiga, então, classificar novos dados apresentados à ele entre 'normais' ou 'anômalos'. No meio de inúmeras técnicas de aprendizado de máquina aplicáveis, os métodos conjuntos vêm produzindo resultados promissores e ganhando evidência na comunidade científica, como foi possível observar no Capítulo 3.

Esta monografia apresenta a implementação de um método conjunto do algoritmo de *Stacking* do classificador OPF para o problema de detecção de intrusão, o que, através de uma extensa revisão literária, apontou-se como uma combinação ainda não apresentada, original. O intuito é provar, como foi argumentado ao longo da obra, que sua aplicação tende à ser distinta e produz resultados melhores que as abordagens clássicas, que empregam classificadores singulares para tratar esse mesmo problema.

Para avaliar precisamente o desempenho da solução apresentada, três experimentos são realizados. Os experimentos empregam os seguintes classificadores: (a) *Decision Tree*, (b) *Logistic Regression*, (c) *Support Vector Machine*, (d) *Linear Support Vector Machine*, (f) *Random Forest*; e (g) OPF. Adicionalmente, os experimentos utilizam o algoritmo de método conjunto *Stacking* com os classificadores OPF e LR como seus meta-aprendizes. A Figura 11 ilustra o *pipeline* experimental conduzido para os modelos conjuntos, onde o *dataset* fornece

os dados de treinamento para os aprendizes base da primeira camada; as previsões retornadas pelos aprendizes bases são então combinadas e servem como *input* para o meta-aprendiz do conjunto, que por sua vez retorna a predição final do modelo.

Figura 11 – *Pipeline* experimental dos modelos *Stacking*



Fonte: Elaborado pelo autor

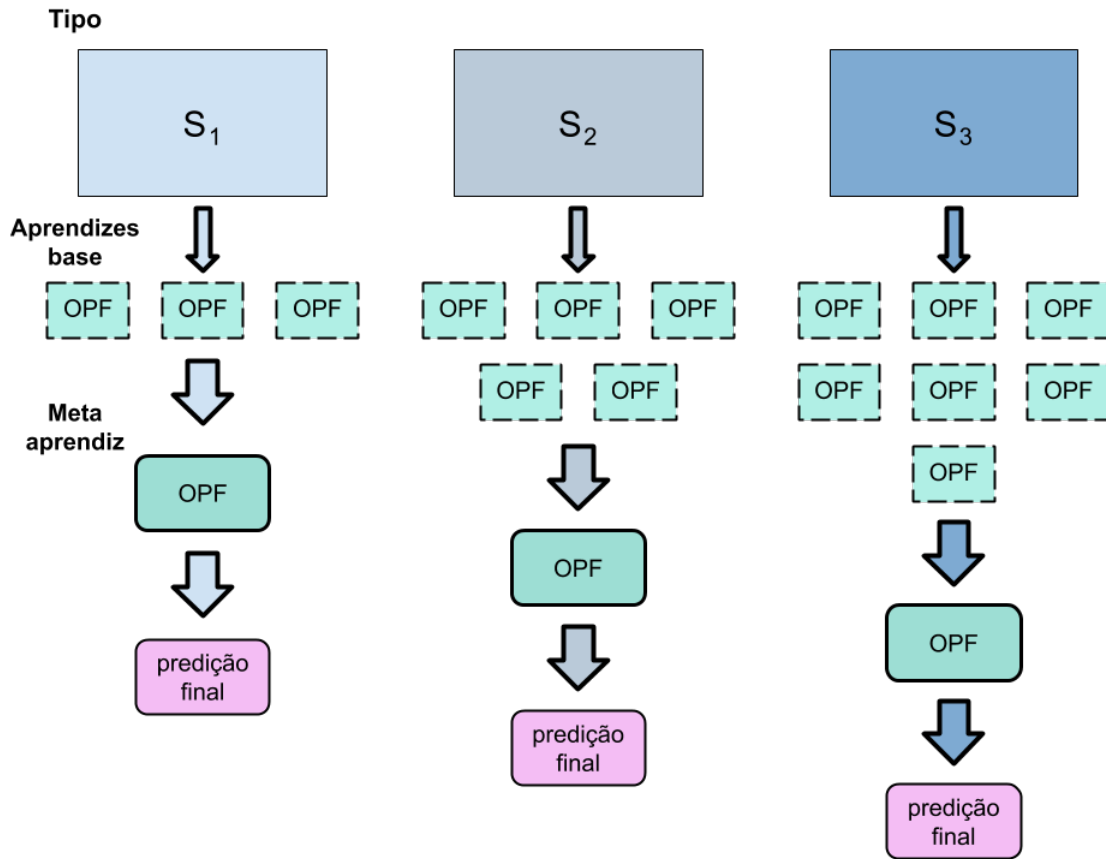
O primeiro experimento foca na implementação de classificadores singulares, onde cada um é treinado e testado separadamente nos *datasets* uneSPY e NSL-KDD. A biblioteca Scikit-Learn¹¹ suporta a implementação de todos classificadores singulares empregados neste experimento, enquanto o OPFython (ROSA; PAPA; FALCAO, 2020) implementa o classificador OPF.

O segundo experimento realiza a implementação do algoritmo de *Stacking* homogêneo para cada um dos classificadores singulares utilizados no experimento anterior, onde S_N representa o *stacking* composto de K classificadores homogêneos. Por exemplo, S_1 pode representar um *stacking* composto de três instâncias de *Decision Tree* que têm suas previsões combinadas por outro *Decision Tree* meta-aprendiz; ou S_3 pode ser um conjunto onde há sete modelos de OPF que são combinados por outro OPF, meta-aprendiz; vide Figura 12.

- S_1 (K): 3-K classificadores;
- S_2 (K): 5-K classificadores;
- S_3 (K): 7-K classificadores;

¹¹ Disponível em: <<https://scikit-learn.org>>

Figura 12 – Exemplo da aplicação de um dos tipos, OPF homogêneo, de modelos de *stacking* homogêneo referentes ao Experimento 2.



Fonte: Elaborado pelo autor

Por fim, o último experimento consiste na implementação de um *stacking* heterogêneo com o classificador OPF como meta-aprendiz. Neste experimento, os aprendizes-base que compõem a primeira camada são diferentes combinações dos classificadores singulares empregados no Experimento 1. R_N ilustra as combinações dos *stackings* heterogêneos de OPF, onde cada conjunto heterogêneo integra diversos classificadores como aprendizes-base, mas sempre mantém o OPF como meta-aprendiz. Ademais, o Experimento 3 também implementa um *stacking* heterogêneo do classificador LR como meta-aprendiz, onde, assim como descrito anteriormente, também emprega diversas combinações dos classificadores singulares utilizados no Experimento 1 para compôr sua camada de aprendizes-base. T_N representa os *stackings* heterogêneos de LR, nos quais cada conjunto heterogêneo incorpora classificadores distintos como aprendizes-base enquanto sustenta o LR como meta-aprendiz. A Figura 13 demonstra um exemplo deste experimento.

- R_1 (OPF): Linear SVC, SVC e RF;
- R_2 (OPF): Linear SVC, SVC e LR;

- R_3 (OPF): Linear SVC, SVC e DT;
- R_4 (OPF): SVC, RF e LR;
- R_5 (OPF): SVC, RF e DT;
- R_6 (OPF): RF, LR e DT;
- R_7 (OPF): Linear SVC, SVC, RF, LR e DT.

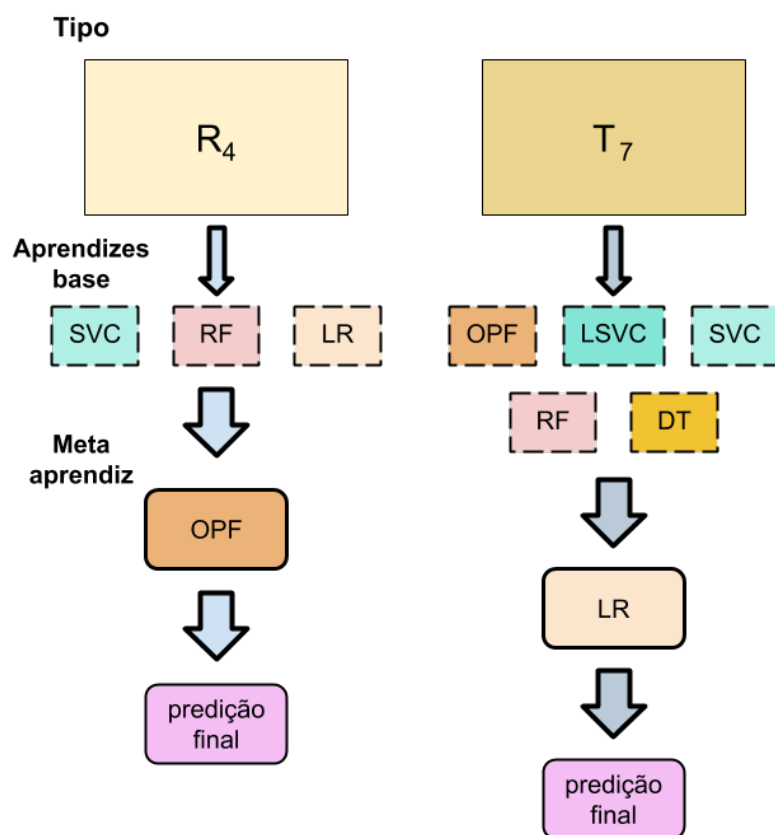
- T_1 (LR): OPF, Linear SVC e SVC;
- T_2 (LR): OPF, Linear SVC e RF;
- T_3 (LR): OPF, Linear SVC e DT;
- T_4 (LR): OPF, SVC e RF;
- T_5 (LR): OPF, SVC e DT;
- T_6 (LR): OPF, RF e DT;
- T_7 (LR): OPF, Linear SVC, SVC, RF e DT.

Em relação aos três experimentos descritos, o primeiro tem como função sustentar os outros dois, uma vez que ele estabelece como os classificadores singulares, aprendizes fracos, desempenham diante do problema de detecção de intrusão. Os resultados obtidos através desse experimento são comparados com os demais, contrastando seus resultados e aferindo a literatura (WOLPERT, 1992) que suporta essa solução. Além disso, como propósito principal deste trabalho apresentado, a experimentação avalia se os métodos conjuntos de *stacking* do classificador OPF são adequados ou não para lidar com o problema em questão.

Ainda no que diz respeito aos experimentos¹², todos *datasets* empregados foram redimensionados aleatoriamente para 10% do seu tamanho original e avaliados utilizando um procedimento de validação cruzada de 10-dobras, onde cada iteração emprega 90% do conjunto de dados para treinamento e 10% para testes; vide exemplo de sua aplicação na Figura 14. Essa abordagem é essencial para garantir que os modelos não sejam enviesados durante a etapa de treinamento, e que os resultados obtidos através dos testes sejam realistas e neutros. Além disso, para cada resultado das dobras (*folds*) a média é calculada e apresentada como resultado final. Por fim, uma análise estatística é realizada para comparar a performance de todos classificadores e modelos conjuntos e averiguar a comensurabilidade de todos eles. O teste estatístico escolhido foi o Teste de Friedman (FRIEDMAN, 1937) com o Teste Post-hoc de Nemenyi (NEMENYI, 1963), dado que a pesquisa realizada por Demsar (DEMSAR, 2006),

¹² O código fonte para a implementação do algoritmo de método conjunto *stacking* utilizado nesse trabalho está disponível em <https://github.com/malvesbertoni/ensemble_opf>.

Figura 13 – Representação visual da pseudo-implementação de dois modelos, R_4 e T_7 , de *stacking* heterogêneo referentes ao Experimento 3.

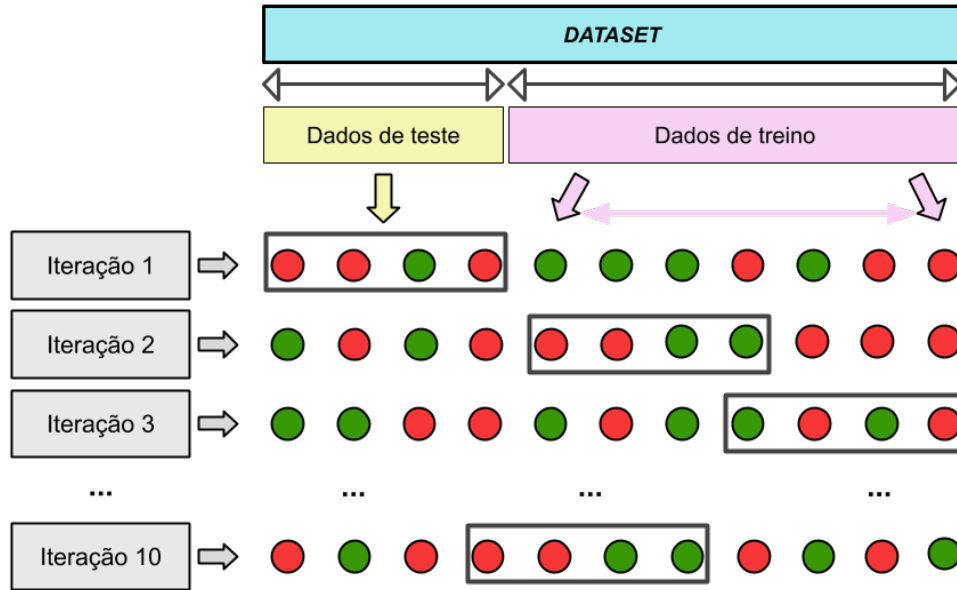


Fonte: Elaborado pelo autor

que avaliou vários testes estatísticos aplicados à diversas situações, chegou à conclusão que a implementação do Teste de Friedman com o Teste Post-hoc de Nemenyi é a abordagem mais confiável para qualificar cenários de comparação de múltiplos modelos sobre múltiplos *datasets*, contexto que corresponde com o do trabalho que é apresentado.

Vale ressaltar também que todos experimentos foram realizados em ambos conjuntos de dados, *uneSPY* e *NSL-KDD*, e também que todos passaram exatamente pelo mesmo procedimento descrito acima, a fim de seguir rigorosamente uma metodologia científica já consolidada pela comunidade, que já foi reproduzida em diversos outros trabalhos, e garantir que os resultados obtidos não têm variações aleatórias e não são frutos de alguma coincidência qualquer. A Figura 15 apresenta um exemplo de aplicação para cada um dos experimentos, possibilitando visualizar a diferença entre eles. Por fim, com o intuito de apoiar a replicabilidade dos experimentos, é interessante destacar as especificações técnicas da máquina empregada neste trabalho, que possui dois processadores *Intel Xeon Bronze 3106 @1.7GHz* e 64GB de memória RAM; o computador utilizado não possui GPU.

Figura 14 – Exemplo de implementação da validação cruzada em 10-dobras no *dataset*



Fonte: Elaborado pelo autor

4.1 Métricas

Para avaliar corretamente o desempenho dos classificadores e dos modelos conjuntos nos experimentos, são utilizadas quatro métricas reconhecidas e amplamente empregadas pela comunidade científica para avaliar os resultados obtidos. Note que verdadeiro positivo (TP) indica instâncias corretamente classificadas como normal, enquanto verdadeiro negativo (TN) demonstra instâncias devidamente categorizadas como anômalas. Falso positivo (FP) retrata anomalias detectadas incorretamente, que na realidade são eventos legítimos, à medida que falso negativo (FN) representa anomalias verdadeiras que foram erroneamente classificadas como eventos normais.

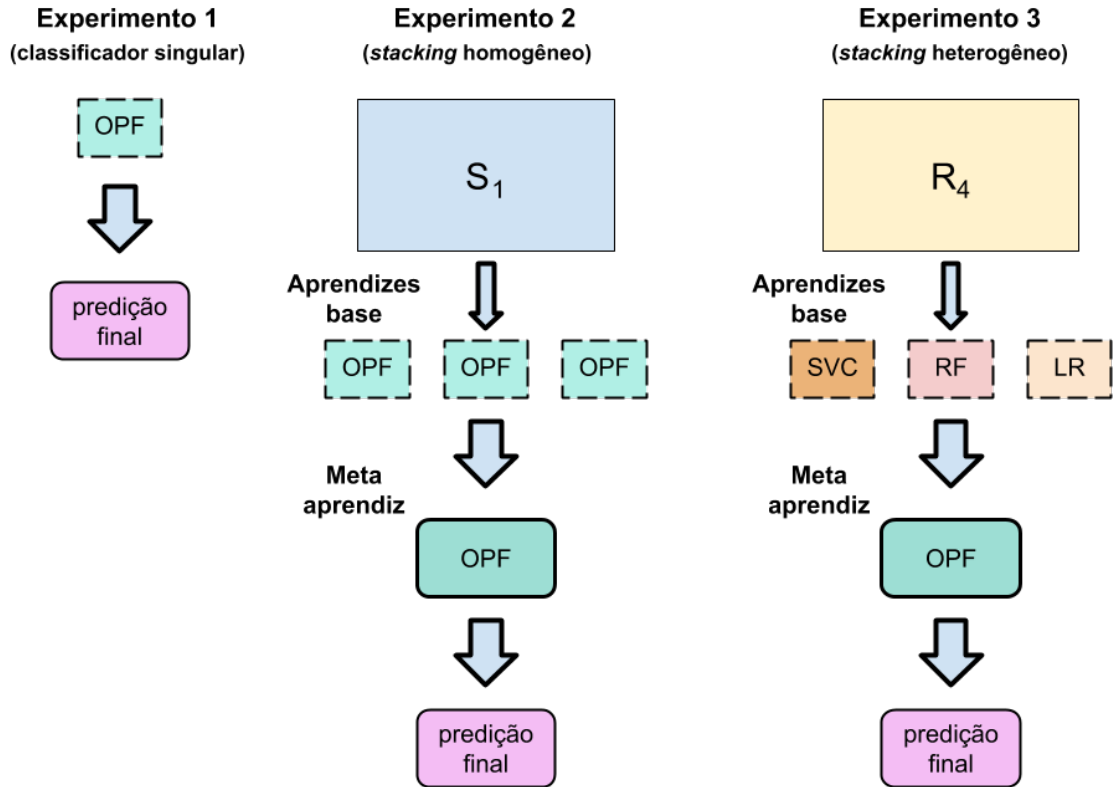
Acurácia é uma das métricas básicas para descrever a performance de qualquer classificador. Ela descreve, em porcentagem, o número de predições corretas, entre rótulos positivos e negativos, feitas pelo algoritmo e é calculada pela seguinte equação:

$$Acurácia = \frac{TN + TP}{TN + TP + FN + FP} \quad (4)$$

Sensitividade, ou *recall*, é descrito pela Equação 5 e mede a proporção de positivos que foram corretamente identificados pelo classificador, fornecendo uma indicação do quanto ele consegue prever instâncias positivas sem erros.

$$Recall = \frac{TP}{TP + FN} \quad (5)$$

Figura 15 – Exemplos de implementação dos três experimentos realizados



Fonte: Elaborado pelo autor

Precisão estima a probabilidade de uma predição positiva estar corretamente assinalada, sendo retratado pela Equação 6.

$$Precisão = \frac{TP}{TP + FP} \quad (6)$$

Por fim, *F1-score* é definida como a média harmônica entre sensibilidade e precisão e é representado pela Equação 7.

$$F1 = 2 * \frac{precisão * recall}{precisão + recall} \quad (7)$$

Todas instâncias classificadas corretamente ou incorretamente por um classificador podem ser representadas na forma de uma matriz de confusão; que é o molde para todas as métricas apresentadas e que é formada pelos quatro valores que aparecem em todas as equações: TP, TN, FP e FN. Para os experimentos propostos, decidiu-se adotar como padrão as seguintes métricas: acurácia, precisão, *recall* e *F1-score*.

4.2 Teste estatístico

Antes de prosseguir, é importante esclarecer a questão do teste estatístico, pois ele é vital para a validação estatística dos resultados obtidos e a comunidade científica está cada vez mais ciente da necessidade de realização destes testes para avaliar corretamente a conclusão de pesquisas. Durante certo tempo, trabalhos da área de ML simplesmente apresentavam a acurácia dos classificadores como a única métrica dos experimentos, o que abre inúmeras possibilidades quanto ao entendimento dos resultados, tornando a pesquisa inconclusiva. Com o avanço da maturidade da área, o aumento de aplicações reais e a disponibilidade de *frameworks* abertos de ML, a realização de testes estatísticos tornou-se essencial para a qualificação dos trabalhos.

Em uma pesquisa típica de ML, um novo algoritmo é proposto e a hipótese implícita é que tal algoritmo é capaz de aprimorar o desempenho de outros já existentes. Algum conjunto de dados é selecionado para treinos e testes, os algoritmos são implementados e a qualidade dos modelos resultantes é avaliada através de alguma métrica apropriada, que, como já mencionado, comumente é a acurácia de classificação.

O estudo de Demsar (DEMSAR, 2006) analisou diversas publicações de Conferências Internacionais de ML e identificou que, normalmente, não há nenhum padrão seguido na realização das comparações e também que os testes performados frequentemente têm fundamentos estatísticos questionáveis, o que leva à conclusões sem garantias e não-verificadas.

Ainda, sua pesquisa analisou teoricamente e empiricamente três famílias de testes estatísticos que podem ser utilizados para comparar dois ou mais classificadores sobre múltiplos conjuntos de dados, que é exatamente o caso deste trabalho: testes paramétricos (como o ANOVA (FISHER, 1959)), testes não-paramétricos (como Wilcoxon (WILCOXON, 1945) e Friedman (FRIEDMAN, 1937)) e o teste não-paramétrico que não assume comensurabilidade dos resultados (teste de sinal). Estes testes estatísticos foram observados em diversos classificadores e conjuntos de dados reais. A análise feita demonstrou que, para a maioria das publicações, a replicabilidade dos testes comumente pode ser um problema, e portanto experimentos seguros devem ser conduzidos em mais de um *dataset*, que é justamente o cenário desenvolvido para os experimentos deste trabalho apresentado: utilizar mais de um conjunto de dados para que os resultados obtidos sejam os mais confiáveis possíveis, e também replicáveis.

Demsar (DEMSAR, 2006) ainda identifica que os testes não-paramétricos, como o de Wilcoxon (WILCOXON, 1945) e Friedman (FRIEDMAN, 1937), são mais adequados para problemas como este que é apresentado. Eles são mais apropriados à esse cenário porque assumem comensurabilidade, embora limitada. E também são mais seguros que testes paramétricos, dado que não reconhecem distribuições normais ou homogeneidade de variância. Os resultados empíricos da pesquisa de Demsar também sugerem que os testes não-paramétricos são mais robustos que os demais. Além disso, seu estudo também propõe uma representação

visual da análise post-hoc quando múltiplos classificadores são comparados: diagramas “CD”, que são de fácil visualização e não ocupam muito espaço, sendo assim adequados para representar claramente a ordem dos algoritmos, a magnitude das diferenças entre eles e a significância das diferenças observadas, o que pode ser muito mais difícil de se apresentar somente em texto ou números.

Logo, devido às descobertas da análise realizada por Demsar, foi decidido empregar o Teste de Friedman (FRIEDMAN, 1937) com o Teste post-hoc de Nemenyi (NEMENYI, 1963) para este trabalho, e também foi escolhido utilizar o diagrama citado para facilitar o entendimento dos resultados obtidos, que será apresentado posteriormente.

4.3 Conjunto de dados

Como já foi citado ao longo do trabalho, o *dataset* é um elemento indispensável aos procedimentos do aprendizado de máquina. Ele é responsável pelo treinamento, teste e validação dos classificadores, logo, sem eles, trabalhos como este seriam impraticáveis.

Não somente é necessária a existência dos *datasets*, também é preciso que ele seja coerente ao problema analisado. *Datasets* são compostos por vetores de características, que por sua vez descrevem objetos através de um conjunto de atributos individuais. Isso significa que não contribui em nada um conjunto de dados que possui somente um tipo de classe, ou seja, remetendo ao exemplo da Figura 6, um *dataset* que apenas apresentasse de objetos da classe ‘círculo’ seria ineficaz pois não atenderia genuinamente ao problema de classificação, uma vez que, na ausência de outra classe, o treinamento seria enviesado e os testes ineficientes. Um *dataset* composto integralmente de objetos de uma mesma classe é um exemplo extremo de um caso de incoerência, porém é comum encontrar conjuntos de dados profundamente redundantes, o que pode comprometer a tarefa de classificação, produzindo classificadores enviesados, evento totalmente indesejado.

No que diz respeito à questão da detecção de intrusão, é observável na Tabela 1 que o *dataset KDD Cup’99* é amplamente utilizado. É válido enfatizar que a Tabela 1 representa somente uma pequena parcela da imensidão de trabalhos da área, no entanto inúmeras outras publicações também fazem uso deste mesmo conjunto de dados, o *KDD Cup’99* (ÖZGÜR; ERDEM, 2016). Sua aparição em abundância dá-se ao fato de que ele fornece uma base sólida ao problema de detecção de intrusão. Contudo, é razoável apontar que sua aplicação talvez já tenha sido saturada ao longo dos anos. Por isso, este trabalho emprega outros *datasets*: um próprio da Universidade Estadual Paulista, uneSPY, e outro também muito popular entre as pesquisas de detecção de intrusão, *NSL-KDD*.

4.3.1 uneSPY

O *dataset* uneSPY é constituído de mais de 1 milhão de amostras com 23 características distintas. Ele está disponível em sua versão crua, sem rótulos e incluindo todos valores de características originais, e em uma versão estendida, a qual já está rotulada (anomalia ou não) e pré-processada para valores categóricos¹³; o Apêndice A descreve detalhadamente mais propriedades deste conjunto de dados. No entanto, há uma adversidade a ser enfrentada ao utilizar esse conjunto de dados: há um número limitado de potenciais ameaças, que compõe apenas 10% das amostras totais do conjunto. Logo, ao trabalhar com a versão rotulada, é necessário proceder com cautela.

Os dados que compõem este *dataset* são referentes à análise do tráfego da antiga rede Wi-Fi da Universidade Estadual Paulista durante a semana, coletados pela ferramenta *tcpdump*. O projeto, de Rubio (RUBIO, 2016), foi realizado por meio da captura de pacotes de tráfego usando um adaptador Wi-Fi, do modelo TP-Link TL-WN725N V2. De acordo com o Serviço Técnico de Informática da universidade, a rede é responsável por fornecer acesso simultâneo para mais de mil usuários. Para obter informações mais qualificadas, a ferramenta *tcpdump* foi utilizada com diversos parâmetros, descritos na Tabela 2. Por fim, após coletar todos os dados, eles foram exportados para um banco de dados para rotulagem manual.

Tabela 2 – Parâmetros utilizados pela ferramenta *tcpdump*.

Parâmetro	Função
-G	Define o tempo de execução (segundos).
-W	Define a quantidade de arquivos gerados durante a captura.
-i <i>iface</i>	Define qual interface será usada.
-A	Mostra o <i>header</i> e <i>payload</i> em código ASCII.
-n	Omite o domínio <i>host</i> .
-vvv	Coleta informação extra sobre o <i>header</i> do pacote.
-w <i>arq</i>	Salva o <i>output</i> em um arquivo.

O processo de rotulagem foi realizado através da definição de uma assinatura de intrusão, utilizada para indicar se uma amostra é anômala ou não. Tratando especificamente deste *dataset*, a documentação de detecção de intrusão da DARPA foi empregada, conduzida pelo Lincoln Laboratory do MIT (KENDALL, 1999). A Tabela 3 apresenta algumas assinaturas comuns de ataques e a quantidade de amostras das mesmas presentes neste conjunto de dados.

¹³ Disponível em <<http://recogna.tech/files/datasets/unespy.rar>>

Tabela 3 – Assinaturas de ataques comuns e quantidade de amostras no *dataset* uneSPY.

Assinatura	Código	Quantidade
Normal	0	913,723
Flooding	1	12,206
Apache2	2	12,714
Land	3	12,628
Ping Of Death	4	11,334
Selfping	5	12,593
Mscan	6	12,219
Crashiis	7	14,778
Ipsweep	8	13,111

Uma maneira interessante de utilizar o uneSPY é ao lidar com estratégias de aprendizado não-supervisionadas, por ser possível extrair um padrão entre as amostras e identificar possíveis pontos fora da curva, anomalias, que podem representar ameaças. Ou seja, é possível realizar pesquisas de detecção de anomalias ao empregar esse *dataset*. Este trabalho é um dos primeiros a utilizar esse conjunto de dados para fornecer à comunidade científica ideias e experimentos esclarecedores e estabelecer direções para pesquisas futuras.

Por fim, é necessário discutir a questão da distribuição de dados deste *dataset*, que é enviesado devido à grande quantidade de dados normais. Para lidar com esse problema, e garantir que os experimentos não tivessem sua credibilidade comprometida, as seguintes medidas foram tomadas:

- Implementação da técnica de validação cruzada de k -dobras (do inglês, *k-fold cross-validation*) para assegurar que os experimentos contemplassem o maior número de variações possíveis dentro do contexto deste conjunto de dados; e
- Utilização de três métricas que são independentes da acurácia para a avaliação dos experimentos, uma vez que a acurácia pode ser influenciada pela distribuição dos dados e poderia comprometer a análise dos experimentos. As demais métricas são precisão, *recall* e *F1-score*, que são descritas detalhadamente no item ??.

4.3.2 *NSL-KDD*

Quando o assunto é detecção de intrusão, o *dataset KDD Cup'99*¹⁴ está entre os conjuntos de dados mais empregados em estudos de detecção de anomalias desde 1999. Sua idealização partiu dos dados coletados pelo programa de avaliação do sistema de detecção da DARPA de 1998 (LIPPMANN et al., 2000). Este *dataset* possui aproximadamente 4.900.000 amostras, no qual cada uma contém 41 características e está rotulada como normal ou ataque, determinando o tipo de ataque referente.

Apesar de sua abundante quantidade de amostras aparentar ser um ponto positivo, há um obstáculo neste conjunto de dados: o número de registros redundantes. Aproximadamente, 78% e 75% das amostras estão duplicadas nos conjuntos de treinamento e teste, respectivamente (TAVALLAEE et al., 2009). Essa grande quantia de registros redundantes no conjunto de treinamento pode ser nociva ao experimento, fazendo com que os classificadores fiquem tendenciosos às amostras mais repetidas. Da mesma forma, a presença de registros repetidos no conjunto de testes também pode afetar a avaliação dos resultados obtidos. Essa desvantagem inerente ao *dataset KDD Cup'99* foi revelada através de análises estatísticas que analisaram a acurácia de detecção de SDIs desenvolvidos por pesquisadores com este conjunto de dados (TAVALLAEE et al., 2009) (KAUSHIK; DESHMUKH, 2011).

Para resolver os problemas apontados, um *dataset* refinado, *NSL-KDD*, foi proposto (TAVALLAEE et al., 2009), com o objetivo de não sofrer com nenhum dos defeitos mencionados previamente e sendo composto somente de registros cuidadosamente selecionados do *dataset KDD Cup'99* completo. Sendo assim, no *NSL-KDD*, não há redundância no conjunto de treinamento, as amostras repetidas foram removidas para possibilitar que o classificador produza um resultado imparcial. Adicionalmente, não existem registros duplicados no conjunto de testes, então as taxas de detecção conseguem ser mais realistas (REVATHI; MALATHI, 2013). As classes de ataque definidas no *NSL-KDD* são divididas em quatro categorias (TAVALLAEE et al., 2009):

- a) *Denial of Service Attack (DoS)*: é uma categoria de ataque na qual os recursos computacionais da vítima são atacados, fazendo com que eles estejam muito ocupados e incapazes de atender requisições legítimas ou negando usuários legítimos de acessar a máquina;
- b) *Probing Attack*: é semelhante à uma busca, na qual o objetivo é colher informações sobre a rede de computadores para burlar seus controles de segurança;
- c) *User to Root Attack (U2R)*: é um tipo de *exploit* no qual o invasor faz uma escalada na hierarquia de acesso da máquina, começando com acesso à uma conta comum de usuário no sistema e procurando obter acesso ao *root* do sistema, para explorar alguma vulnerabilidade; e

¹⁴ Disponível em <<http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>>

- d) *Remote to Local Attack (R2L)*: ocorre quando um invasor, que consegue enviar pacotes à uma máquina através de uma rede, porém sem acesso autorizado à mesma, explora alguma vulnerabilidade para ganhar acesso local como um usuário legítimo desta máquina.

Considerando os pontos citados, este trabalho utilizará o *dataset NSL-KDD* para treinar e avaliar a performance dos experimentos propostos. A única diferença em relação ao *dataset* original é que, das 41 características iniciais, a primeira, que faz referência à duração do ataque, foi descartada e todos ataques foram agrupados e rotulados juntamente como ‘anomalia’, ao invés de especificar no rótulo suas diferentes classes de ataque.

4.4 Experimento 1

Neste experimento, diversos classificadores singulares são treinados e testados nos *datasets* *uneSPY* e *NSL-KDD*; o objetivo é avaliar suas performances no problema descrito e compará-los, a fim de encontrar aqueles que mais se destacam. Para avaliá-los, são utilizadas as métricas e o teste estatístico recém descritos.

Novamente, os classificadores empregados neste experimento são: (a) *Decision Tree*, (b) *Logistic Regression*, (c) *Support Vector Machine* ¹⁵, (d) *Linear Support Vector Machine*, (f) *Random Forest*; e (g) OPF. E, como já foi mencionado anteriormente, eleger o “melhor” classificador não é uma tarefa trivial e, muitas vezes, é subjetiva, pois cada problema traz consigo diferentes aspectos e numerosas soluções cabíveis. Logo, simplesmente observar apenas uma das métricas, como, por exemplo, ‘acurácia’, e apontar o algoritmo que obteve o melhor resultado nela como superior é uma abordagem incorreta. Todas métricas citadas, a matriz de confusão e o teste estatístico devem ser analisados para então decidir qual classificador demonstra-se mais propício à solução do problema em questão.

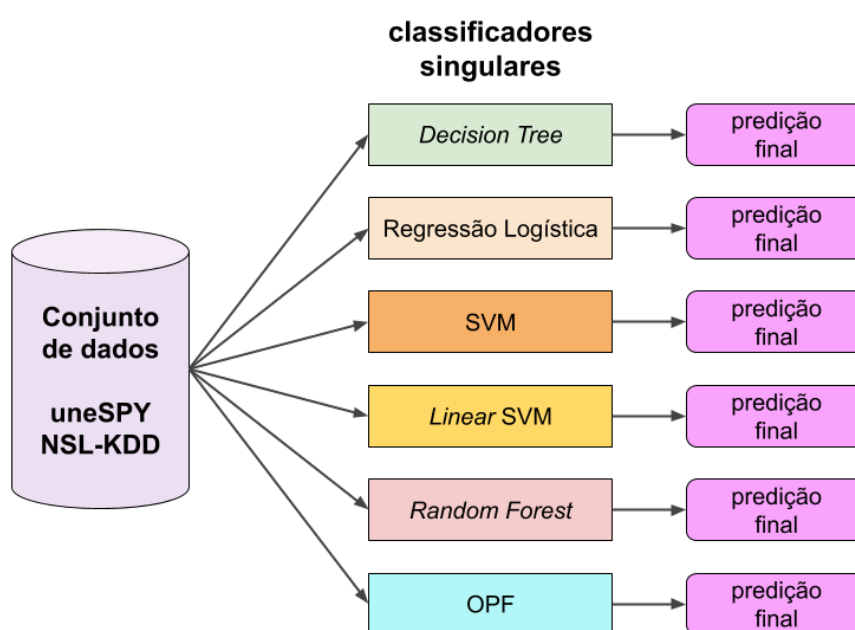
Treinar, testar e compará-los é necessário para verificar como cada classificador singular se comporta, e qual prevalece, no contexto observado. O objetivo por trás desse experimento é encontrar quais classificadores singulares são os mais robustos e quais são os mais fracos. A resposta dessa pergunta serve de base para os experimentos seguintes, uma vez que os métodos conjuntos baseiam-se, como diz a literatura, em transformar aprendizes fracos em aprendizes fortes. Sendo assim, sabendo distinguir, através deste primeiro experimento, quais são aqueles que, por natureza, demonstram-se fracos neste problema, será possível validar o conceito primordial dos métodos conjuntos nos experimentos seguintes: a capacidade de um algoritmo de método conjunto conseguir, ao combinar classificadores fracos, produzir resultados melhores que eles individualmente. E também, este experimento será capaz de validar a premissa do algoritmo de *Stacking* (WOLPERT, 1992), que diz ser capaz de melhorar a capacidade de

¹⁵ O classificador *Support Vector Machine* será representado nas figuras e nos gráficos através da sigla SVC, referente à seu nome na biblioteca *scikit-learn* (*Support Vector Classifier*)

generalização dos algoritmos através de sua implementação; tanto em sua forma homogênea quanto heterogênea.

A Figura 16 ilustra a pseudo-implementação deste experimento, onde cada um dos classificadores singulares citados é treinado e testado nos conjuntos de dados uneSPY e NSL-KDD, seguindo o procedimento descrito no início deste capítulo, e produzindo os resultados que serão apresentados no Capítulo 5.

Figura 16 – Representação da pseudo-implementação do Experimento 1



Fonte: Elaborado pelo autor

4.5 Experimento 2

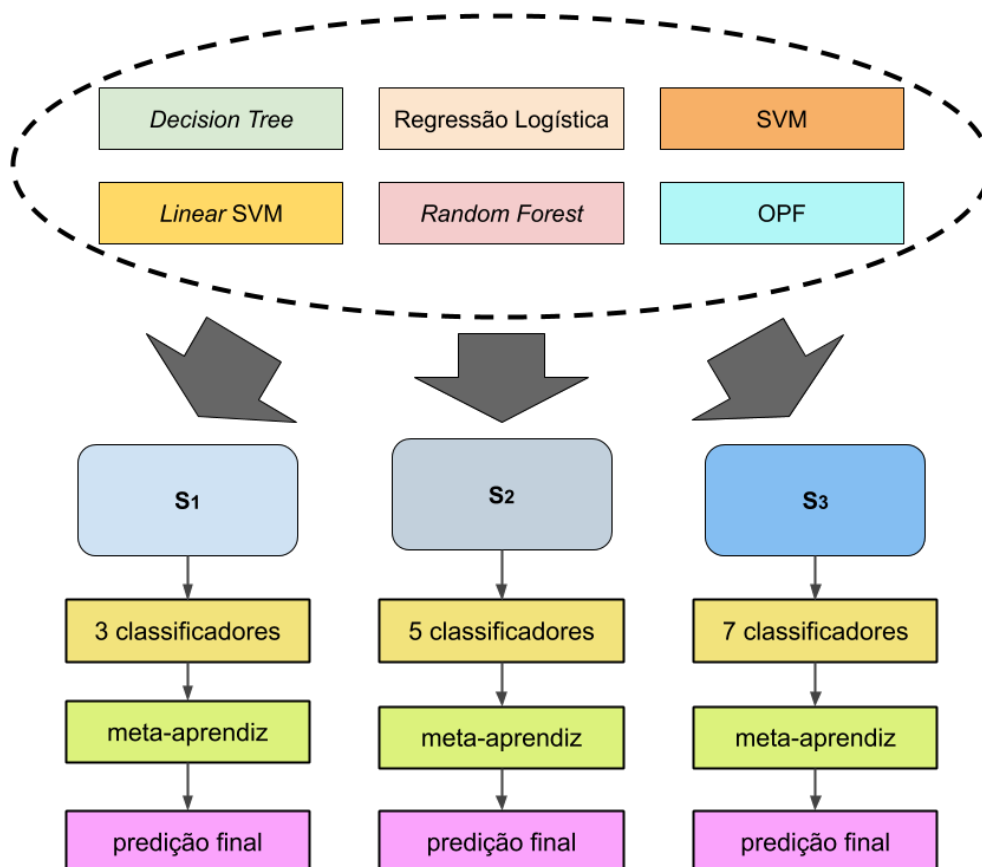
Este experimento implementa um algoritmo de *Stacking* homogêneo para cada um dos classificadores singulares empregados no experimento anterior. Mais do que isso, para refinar este experimento, foram derivadas diferentes versões dos classificadores singulares já mencionados. Isso significa que cada um daqueles classificadores teve seus parâmetros modificados, gerando modelos alterados de seu algoritmo original, que são combinados com o algoritmo primário no método conjunto, exercendo a função de meta-aprendiz, para a produção de seu *Stacking* homogêneo.

Essa modificação nos parâmetros dos classificadores originais é importante para garantir uma diferenciação, por menor que seja, entre as predições dos classificadores empregados no conjunto. Supondo que um *Stacking* homogêneo fosse implementado utilizando somente classificadores idênticos, literalmente, em sua camada de aprendizes base, o meta-aprendiz muito provavelmente receberia *inputs* também idênticos, uma vez que os classificadores utilizados

não teriam diferenciação nenhuma, e isso poderia prejudicar o estudo, enviesando os modelos. Portanto, por mais que se trate de um conjunto homogêneo, onde todos classificadores adotados são iguais, e por “iguais” entenda “do mesmo gênero”, os parâmetros dos classificadores da camada de aprendizes-base são levemente diferentes para garantir que os *inputs* recebidos pelo meta-aprendiz não sejam idênticos e assim a combinação das predições seja melhor definida.

Esclarecido isso, o experimento prossegue implementando os conjuntos homogêneos em três categorias distintas, nas quais o meta-aprendiz mantém-se o mesmo na segunda camada; todos conjuntos empregam os *datasets* uneSPY e NSL-KDD. As três categorias se diferenciam da seguinte maneira: três classificadores na primeira camada, cinco classificadores na primeira camada e sete classificadores na primeira camada. Logo, em cada uma dessas categorias, são empregados um determinado algoritmo de classificador singular com seus parâmetros modificados, combinado por seu algoritmo original como meta-aprendiz. Ou seja, são treinados três conjuntos diferentes para cada um dos seguintes classificadores: (a) *Decision Tree*, (b) *Logistic Regression*, (c) *Support Vector Machine*, (d) *Linear Support Vector Machine*, (f) *Random Forest*; e (g) OPF; como pode ser observado na representação da Figura 17.

Figura 17 – Representação das três diferentes categorias implementadas no Experimento 2

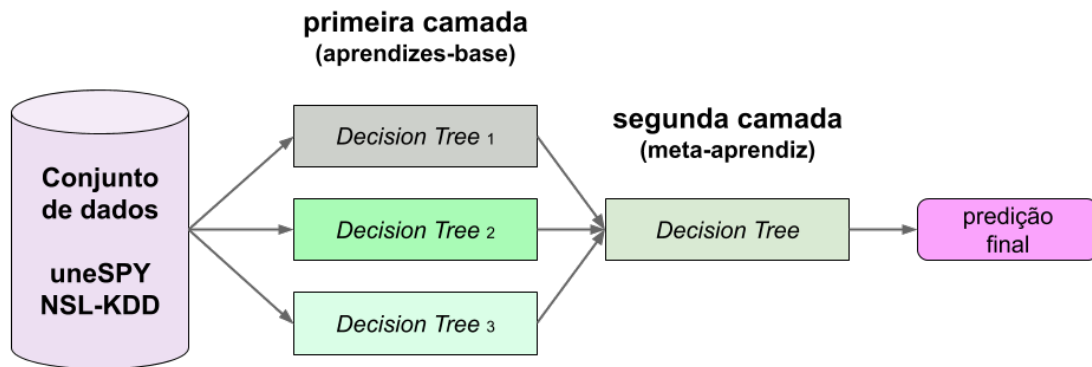


Fonte: Elaborado pelo autor

A Figura 12, apresentada anteriormente, ilustra a diferença entre as três categorias

que são implementadas neste experimento e a Figura 18 apresenta um exemplo específico da implementação de um *Stacking* homogêneo da categoria 1 (S_1) do classificador *Decision Tree*, onde há três classificadores na primeira camada, com parâmetros diferentes, e o meta-aprendiz, com o algoritmo original do classificador, na segunda camada.

Figura 18 – Representação da pseudo-implementação de um *Stacking* homogêneo da categoria 1 do classificador *Decision Tree* referente ao Experimento 2



Fonte: Elaborado pelo autor

Este experimento é importante porque, baseado nos resultados do primeiro experimento, ele valida a literatura dos métodos conjuntos, verificando se de fato os aprendizes fracos do primeiro experimento serão capazes de tornarem-se modelos robustos a partir de sua combinação, e também a premissa do algoritmo de *Stacking* (WOLPERT, 1992), confirmando se a implementação do *Stacking* homogêneo será capaz de aprimorar o desempenho do classificador empregado; além de também possibilitar uma análise da performance do OPF em sua versão conjunta comparada à sua versão singular.

4.6 Experimento 3

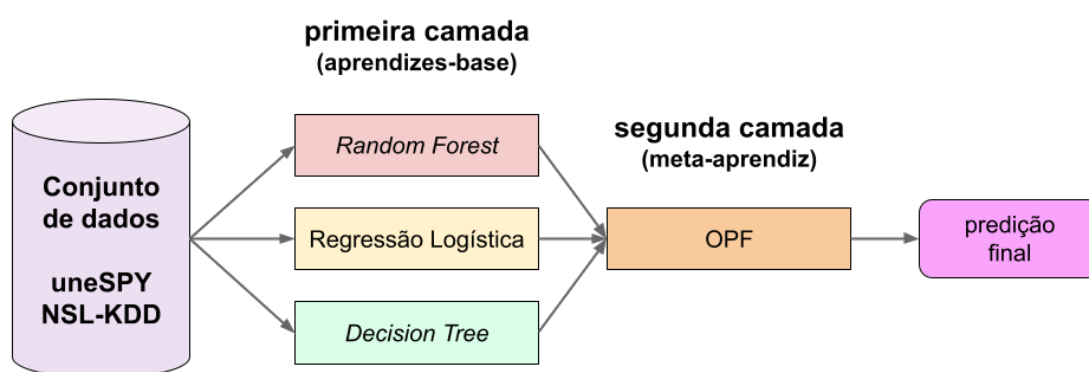
O último experimento destina-se à implementação dos *Stackings* heterogêneos dos classificadores OPF e LR. Nesta etapa final, também são utilizados todos os classificadores singulares do primeiro experimento, o que novamente destaca sua importância para a totalidade do trabalho.

Este experimento é responsável por produzir quatorze diferentes conjuntos, como demonstrado no início do capítulo, sendo cada meta-aprendiz responsável pela metade dos mesmos. Os únicos meta-aprendizes empregados são os classificadores já citados, OPF e LR. Em relação à primeira camada dos conjuntos, sua composição é a combinação do algoritmo original dos diferentes classificadores do primeiro experimento, sem nenhuma repetição dentro de um mesmo conjunto. Isso significa que não há nenhum modelo de *Stacking* heterogêneo contendo dois classificadores singulares do mesmo tipo na primeira camada e todos classificadores são

implementados em seu algoritmo original, sem as derivações citadas no Experimento 2; como ilustra a Figura 13.

A ideia que sustenta a realização deste experimento é possibilitar uma comparação holística do desempenho do classificador OPF em suas diversas aplicações e compará-lo aos demais. No Experimento 1, ele é implementado sozinho; no Experimento 2, ele é implementado em três *Stackings* homogêneos diferentes; e agora, ele é implementado em sete *Stackings* heterogêneos distintos. Além disso, esse último experimento também serve para validar as premissas referentes aos métodos conjuntos e ao *Stacking*, que afirmam serem capazes de alcançarem melhor desempenho e capacidade de generalização que os aprendizes fracos. A Figura 19 ilustra a pseudo-implementação de um dos modelos deste experimento, onde os *datasets* fornecem os dados para três classificadores distintos, RF, LR e DT, que compõem a primeira camada do *stack* sejam combinados pelo meta-aprendiz OPF; esse exemplo refere-se à implementação R_6 do Experimento 3.

Figura 19 – Representação da pseudo-implementação de um dos modelos de *Stacking* heterogêneo do classificador OPF referente ao Experimento 3.



Fonte: Elaborado pelo autor

Todos os três experimentos se complementam e servem de apoio para uma avaliação integral e confiável do método conjunto de *Stacking* do classificador OPF para o problema de detecção de intrusão. O primeiro experimento revela o desempenho singular de cada classificador ao enfrentar o problema de detecção de intrusão, o que funciona como uma referência de comparação para os demais experimentos que, ao implementar o algoritmo de método conjunto *Stacking*, devem obter resultados melhores que estes. Logo, o segundo experimento, ao implementar conjuntos homogêneos de todos classificadores empregados no experimento anterior, não só estabelece uma amostra para comparação do desempenho dos métodos conjuntos em relação aos classificadores singulares mas também da performance do *Stacking* homogêneo do OPF contrastando os demais. Por fim, o último experimento, que implementa tanto o conjunto heterogêneo do OPF quanto do LR, finaliza todas as possibilidades de comparação do funcionamento do *Stacking* do OPF, permitindo avaliá-lo de maneira fidedigna, além de validar a literatura que fundamenta todo esse trabalho.

5 Resultados experimentais

Este capítulo exhibe os resultados alcançados através dos Experimentos 1, 2 e 3, que foram descritos previamente. Os resultados serão apresentados separadamente, de acordo com o *dataset* empregado, e então serão novamente divididos entre os diferentes experimentos, exibindo-os um por vez.

No que diz respeito ao *dataset* uneSPY, a Figura 20 representa o primeiro experimento (classificadores singulares), as Figuras 21 e 22 correspondem ao segundo experimento (conjuntos homogêneos) e as Figuras 23 e 24 descrevem o último experimento (conjuntos heterogêneos).

Em relação ao conjunto de dados NSL-KDD, a Figura 25 representa o primeiro experimento (classificadores singulares), o segundo experimento (conjuntos homogêneos) é retratado através das Figuras 26 e 27 e, por fim, as Figuras 28 e 29 exibem os resultados do último experimento (conjuntos heterogêneos).

O Apêndice B, no final do trabalho, demonstra as tabelas destes experimentos na íntegra, das quais os gráficos aqui apresentados foram derivados. As tabelas Tabela 9, Tabela 10, Tabela 11 representam, respectivamente, os Experimentos 1, 2 e 3 para o *dataset* uneSPY. Enquanto as tabelas Tabela 12, Tabela 13 e Tabela 14, respectivamente, apresentam os resultados obtidos pelos Experimentos 1, 2 e 3 com o *dataset* NSL-KDD. Além disso, os modelos finais de todos classificadores singulares e *Stackings*, tanto homogêneos quanto heterogêneos, empregados neste trabalho estão disponíveis para *download*¹⁶.

5.1 uneSPY

Observando a Figura 20, é visível que, em termos de desempenho geral, os piores classificadores singulares empregados no Experimento 1 foram o *Logistic Regression* (LR) e *Linear SVC* (LSVC). Os outros classificadores atingiram bons resultados, especialmente o OPF, que foi praticamente tão bom quanto os dois melhores, *DecisionTree* (DT) e *Random Forest* (RF). Logo, tendo em vista seu desempenho perante os demais classificadores, a escolha de utilizar o classificador OPF como objeto de estudo deste trabalho demonstra-se apropriada até então, uma vez que seus resultados apresentam-se semelhantes aos que tiveram a melhor performance na resolução do problema.

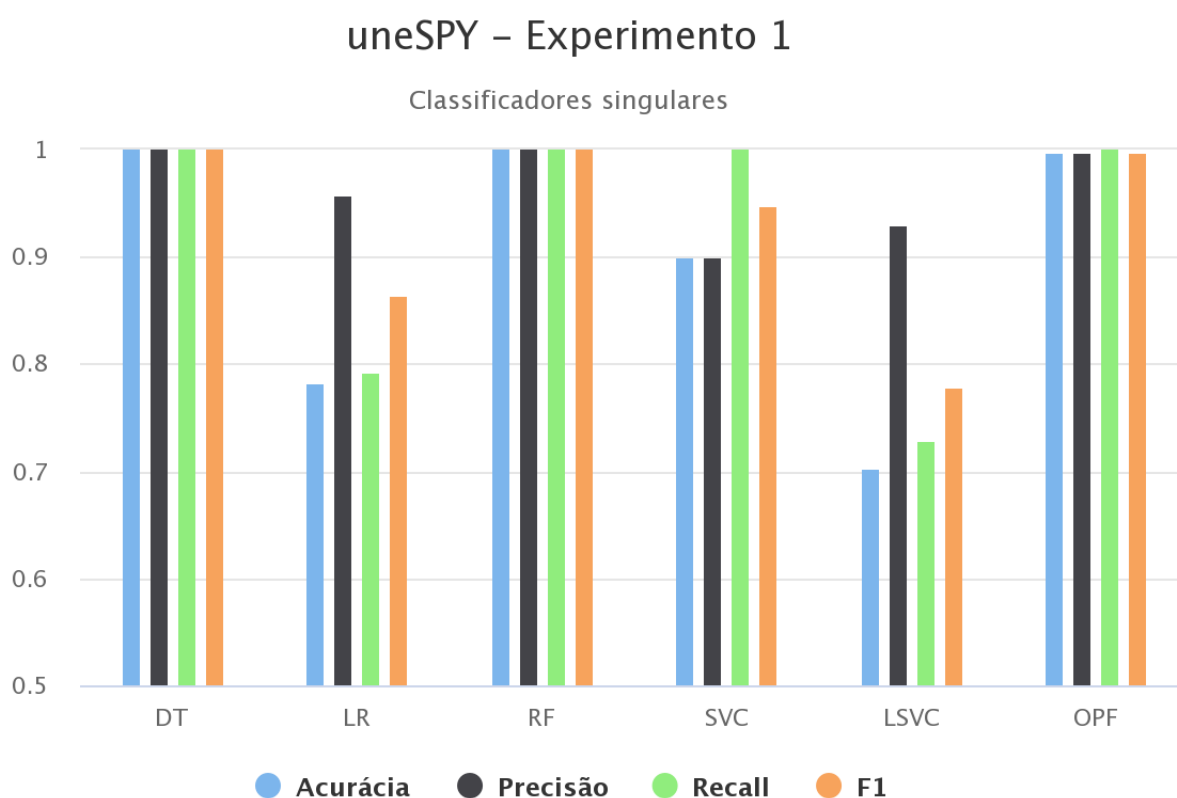
No que diz respeito ao segundo experimento, as Figuras 21 e 22 apresentam seus resultados; onde a primeira figura retrata os conjuntos dos classificadores DT, LR e RF, enquanto a segunda refere-se aos conjuntos do SVC, LSVC e OPF. É observável que Wolpert (WOLPERT, 1992) estava correto, pois todos modelos demonstraram melhorias evidentes, especialmente os

¹⁶ Disponível para acesso em <https://github.com/malvesbertoni/ensemble_opf/tree/master/output>.

aprendizes fracos, que tiveram o pior desempenho em sua implementação singular: *Logistic Regression* (LR) e *Linear SVC* (LSVC), como mencionado no parágrafo anterior; ambos conjuntos homogêneos destes algoritmos exibiram uma notável evolução na capacidade de generalização. No entanto, os modelos que implementam o OPF não melhoraram, ao contrário, tiveram um desempenho pior; seus resultados serão discutidos posteriormente.

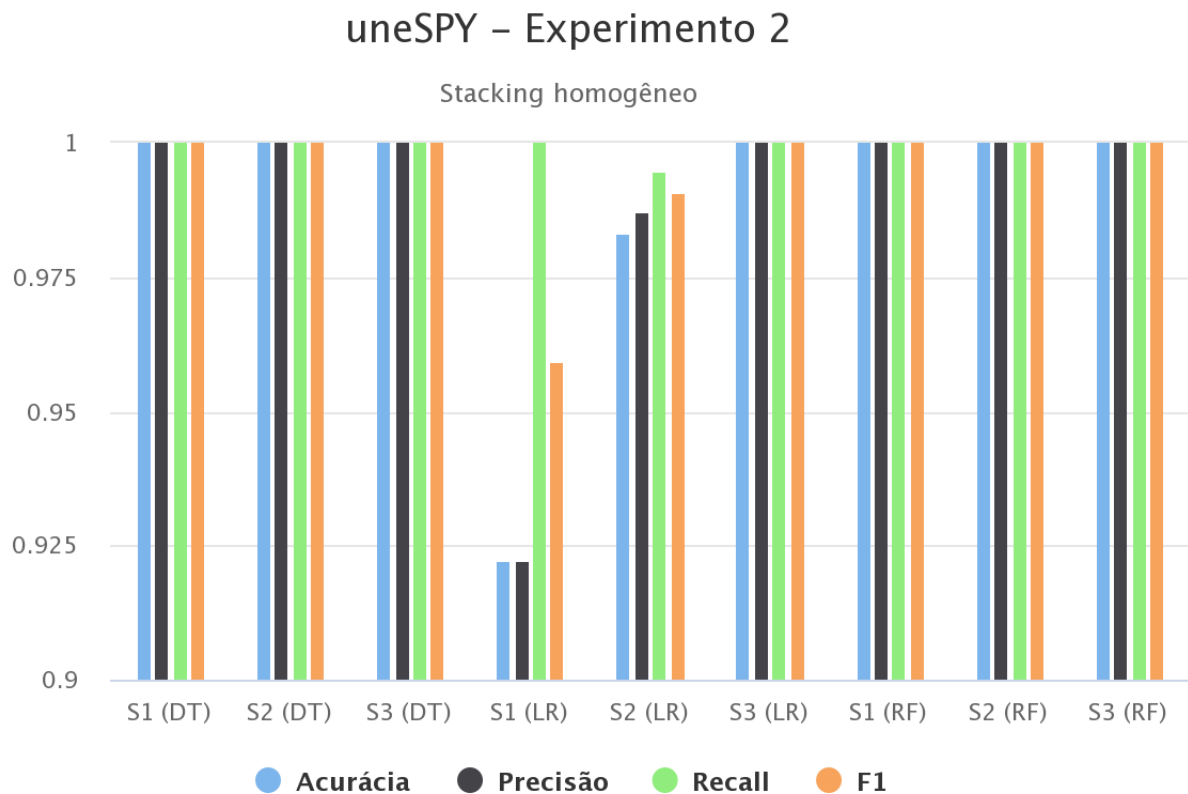
Por fim, as Figuras 23 e 24 exibem os resultados obtidos no Experimento 3, onde a primeira apresenta os resultados dos conjuntos heterogêneos no qual o meta-aprendiz é o classificador LR, enquanto a outra refere-se àqueles cujo meta-aprendiz é o classificador OPF. Ao comparar os resultados dos *Stacking* heterogêneos do classificador LR com sua implementação singular, é nítido que os conjuntos heterogêneos alcançaram resultados melhores em todas as métricas. Quanto aos modelos do OPF, a maioria dos resultados foi positivo, apresentando melhorias em relação à sua versão singular e aos *Stackings* homogêneos em todas as métricas, porém dois conjuntos heterogêneos foram piores. Novamente, a literatura (WOLPERT, 1992) aparenta estar correta. A situação do OPF será discutida no fim do capítulo.

Figura 20 – Representação gráfica dos resultados obtidos no Experimento 1 com o *dataset* uneSPY



Fonte: Elaborado pelo autor

Figura 21 – Representação gráfica dos resultados obtidos no Experimento 2 com o *dataset* uneSPY

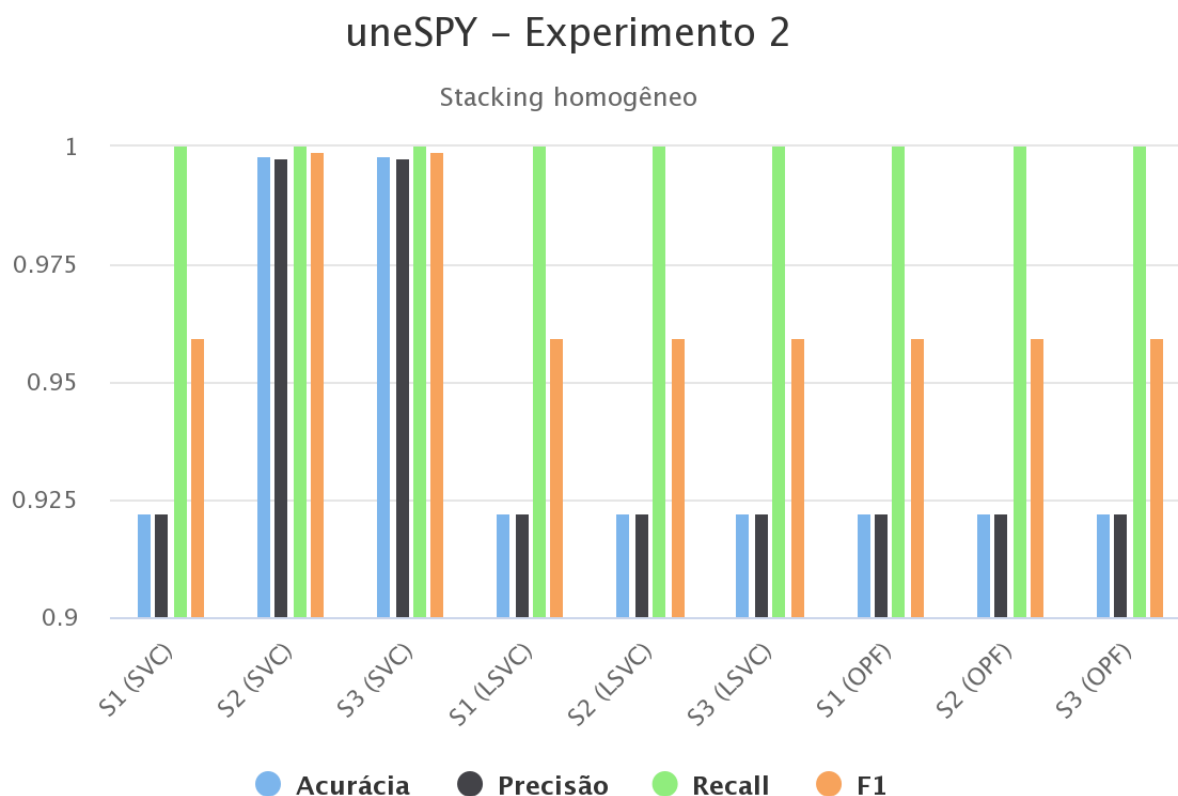


Fonte: Elaborado pelo autor

5.2 NSL-KDD

Ao analisar a Figura 25, comparando todas as métricas dos diferentes classificadores e considerando seu desempenho geral, fica evidente que àqueles que tiveram a pior performance foram o *Support Vector Machine* (SVC) e o *Linear SVC* (LSVC). Novamente, a implementação singular do OPF alcançou resultados excelentes, quase tão bons quantos os dois melhores, *Decision Tree* (DT) e *Random Forest* (RF).

Em relação aos *Stackings* homogêneos do Experimento 2, os resultados são apresentados pelas Figuras 26 e 27, onde a primeira representa os conjuntos dos classificadores DT, LR e RF e a outra descreve os conjuntos do SVC, LSVC e OPF. Similarmente ao que foi retratado no segundo experimento conduzido no *dataset* uneSPY, aqui, no conjunto de dados NSL-KDD, todos *Stackings* homogêneos também produziram resultados melhores que suas versões singulares, mais uma vez ratificando a teoria dos métodos conjuntos (SCHAPIRE, 1990) e do Wolpert (WOLPERT, 1992). No entanto, assim como no outro Experimento 2, os modelos do classificador OPF não obtiveram resultados melhores, tendo até piorado, juntamente com os modelos do classificador *Support Vector Machine*; essa situação será discutida ao final do capítulo.

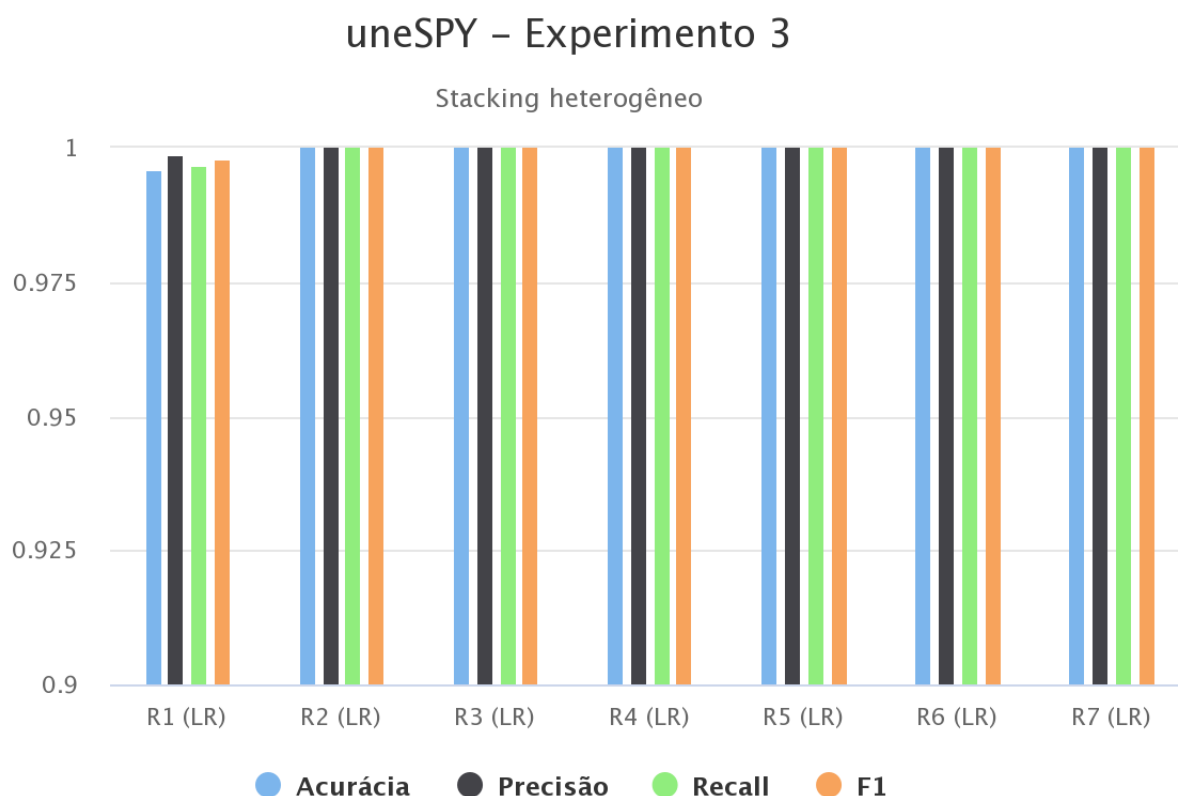
Figura 22 – Representação gráfica dos resultados obtidos no Experimento 2 com o *dataset* uneSPY

Fonte: Elaborado pelo autor

Por último, as Figuras 28 e 29 retratam os resultados provenientes dos *Stackings* heterogêneos do Experimento 3, sendo a primeira responsável por descrever os conjuntos formados com o classificador LR como meta-aprendiz e a segunda àqueles cujo meta-aprendiz é o classificador OPF. De maneira semelhante ao que foi observado com o outro conjunto de dados, ao analisar as métricas, os conjuntos heterogêneos do classificador LR como meta-aprendiz exibem resultados melhores que sua versão singular. Porém, diferente do que aconteceu com o *dataset* uneSPY, ao comparar os resultados, é perceptível que os *Stackings* heterogêneos compostos do classificador OPF como meta-aprendiz apresentaram desempenho pior do que sua abordagem singular, o que é interessante.

5.3 Análise geral

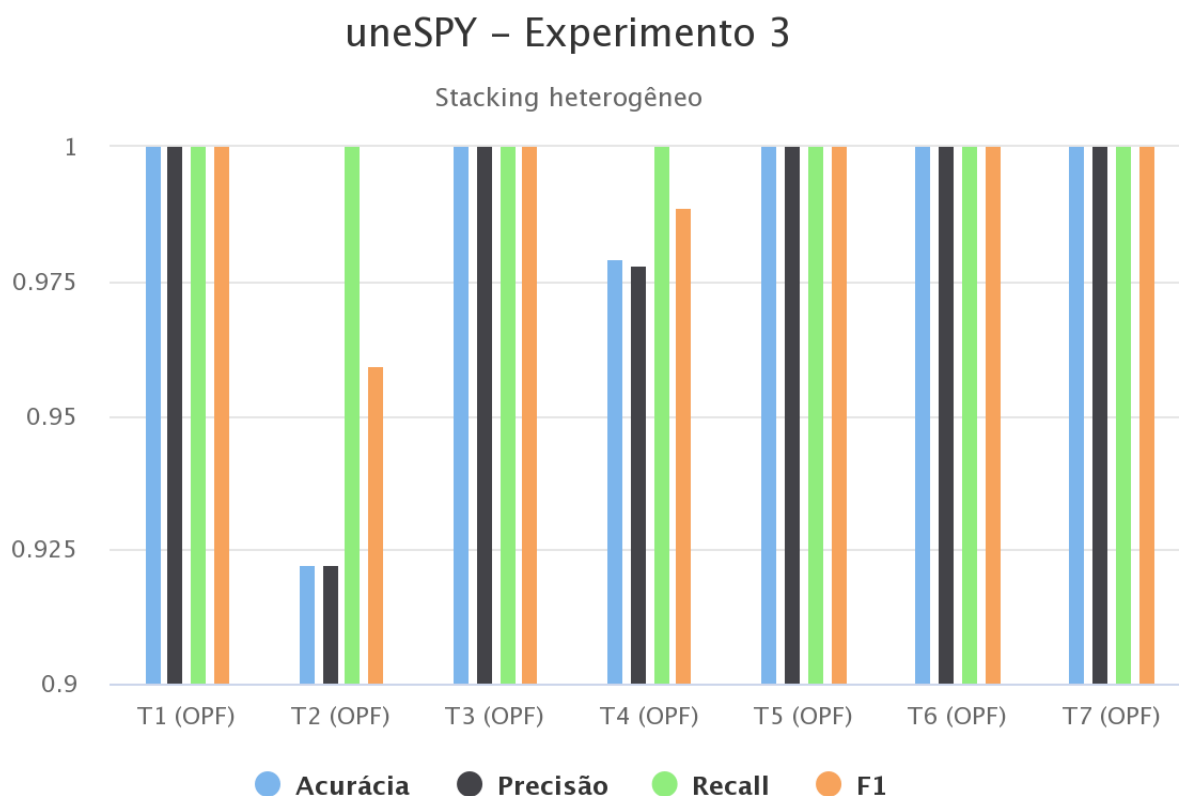
Como é possível observar nas figuras apresentadas e nas tabelas, disponíveis no Apêndice do trabalho, exceto para alguns casos específicos dos *Stackings* homogêneos e heterogêneos que têm o classificador OPF como meta-aprendiz, a literatura (WOLPERT, 1992) aparenta estar certa: o algoritmo de método conjunto *Stacking* realmente melhora a capacidade de generalização de um modelo. No entanto, em relação ao desempenho do OPF, é plausível

Figura 23 – Representação gráfica dos resultados obtidos no Experimento 3 com o *dataset* uneSPY

Fonte: Elaborado pelo autor

que os resultados insatisfatórios estejam atrelados a algumas características intrínsecas deste classificador não combinarem com a metodologia do algoritmo *Stacking*, sendo assim um arranjo inadequado. Como foi mencionado na Seção 4.1, este classificador utiliza uma função de custo de caminho para calcular seus protótipos e a etapa de conquista dos nós adjacentes. Essa técnica do OPF pode não estar funcionando corretamente com a estrutura do *Stacking*, fazendo com que todo o processo da conquista de nós adjacentes do classificador, que é crucial para o desempenho do OPF, esteja imprecisa e incapaz de determinar a amostra correta a ser conquistada, o que resulta em um cenário caótico para o mesmo.

Por exemplo, imagine o seguinte cenário: existe uma amostra $[0,0]$ com o rótulo “0” e outra amostra $[1,1]$ também com o rótulo “0”. Como acabou de ser descrito, o processo de conquista destas amostras provavelmente está sendo calculado erroneamente por causa da sua imperfeita união com a metodologia *Stacking*, o que faz com o que o algoritmo do OPF fique desorientado ao abordar o processo de conquista da classe “0”. Supondo que o protótipo da classe “1” é em torno de $[1,1]$, com certeza a distância dele até a amostra $[1,1]$ é menor; logo, ele irá conquistar a essa amostra, errada, ao invés da amostra correta do protótipo da classe “0”. Uma vez que o classificador OPF somente fornece os rótulos finais de suas previsões, e não informa também as probabilidades, é impossível afirmar que seja exatamente isso que está

Figura 24 – Representação gráfica dos resultados obtidos no Experimento 3 com o *dataset* uneSPY

Fonte: Elaborado pelo autor

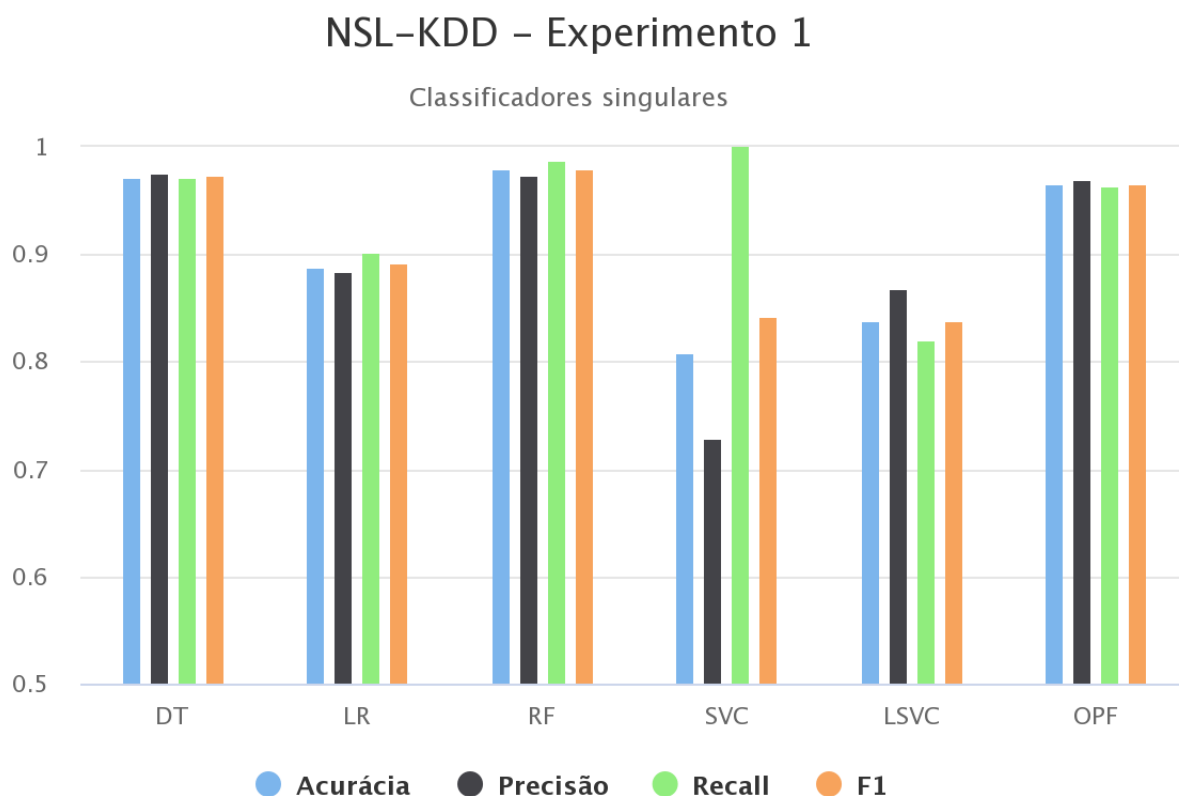
acontecendo em sua implementação com o *Stacking*, embora seja a análise mais racional.

Finalmente, como foi mencionado anteriormente na Seção ??, o teste estatístico de Friedman com post-hoc de Nemenyi também foi aplicado para verificar a comensurabilidade dos modelos, que é representado pela Figura 30. Esta figura apresenta visualmente o teste estatístico feito, onde as linhas horizontais coloridas conectam todos classificadores que são estatisticamente comensuráveis.

Ao analisar a Figura 30, é possível verificar que alguns *Stackings* homogêneos e heterogêneos do OPF que não atingiram ótimos resultados são semelhantes à conjuntos que obtiveram excelentes resultados. Ainda assim, outros destes modelos conjuntos do OPF, que podem ser vistos no canto esquerdo extremo da Figura 30, não são comensuráveis aos melhores conjuntos.

Portanto, isso significa que, mesmo executando uma metodologia rígida para os experimentos, onde os conjuntos de dados foram particionados para as fases de treino e teste, aplicando 10-dobras no processo de implementação dos modelos, empregando quatro métricas distintas para representação dos resultados finais e performando um teste estatístico para a análise dos resultados, ainda assim os resultados podem estar enviesados. Essa distorção dos

Figura 25 – Representação gráfica dos resultados obtidos no Experimento 1 com o *dataset* NSL-KDD

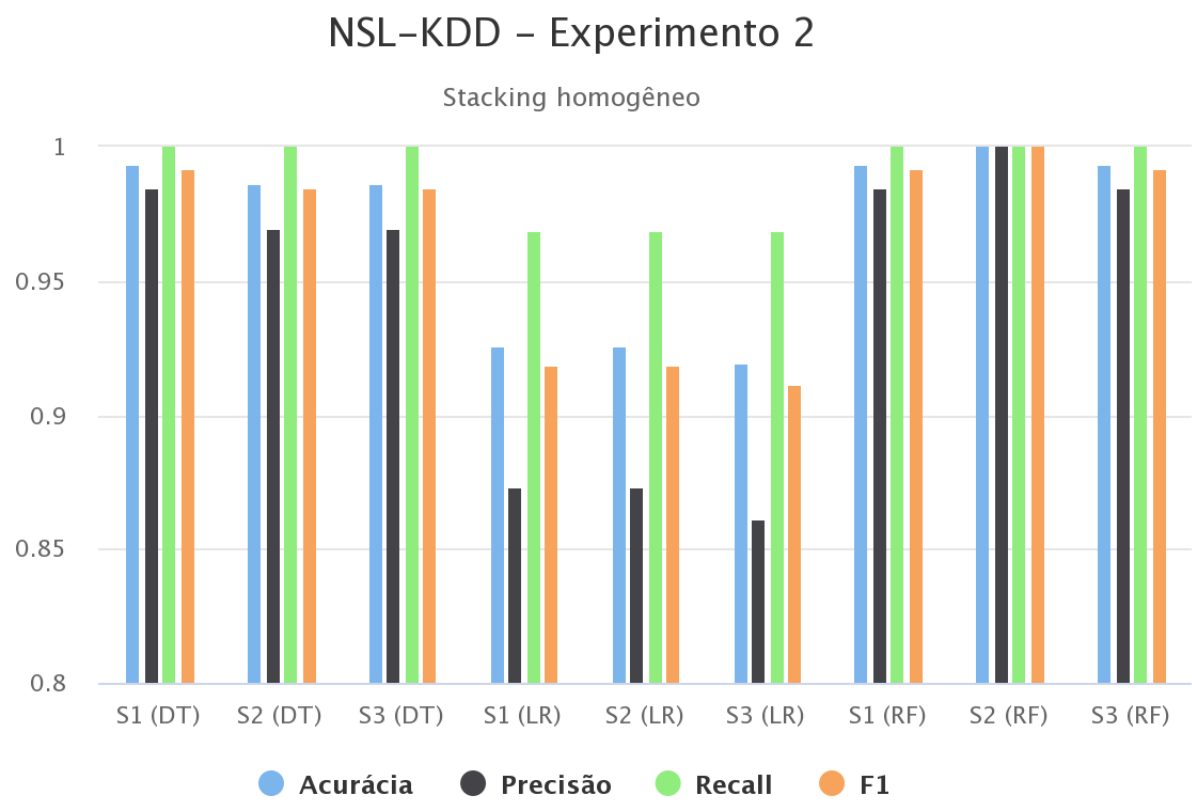


Fonte: Elaborado pelo autor

mesmos provavelmente está associada à pequena divergência entre as atividades normais e anômalas observadas nos conjuntos de dados.

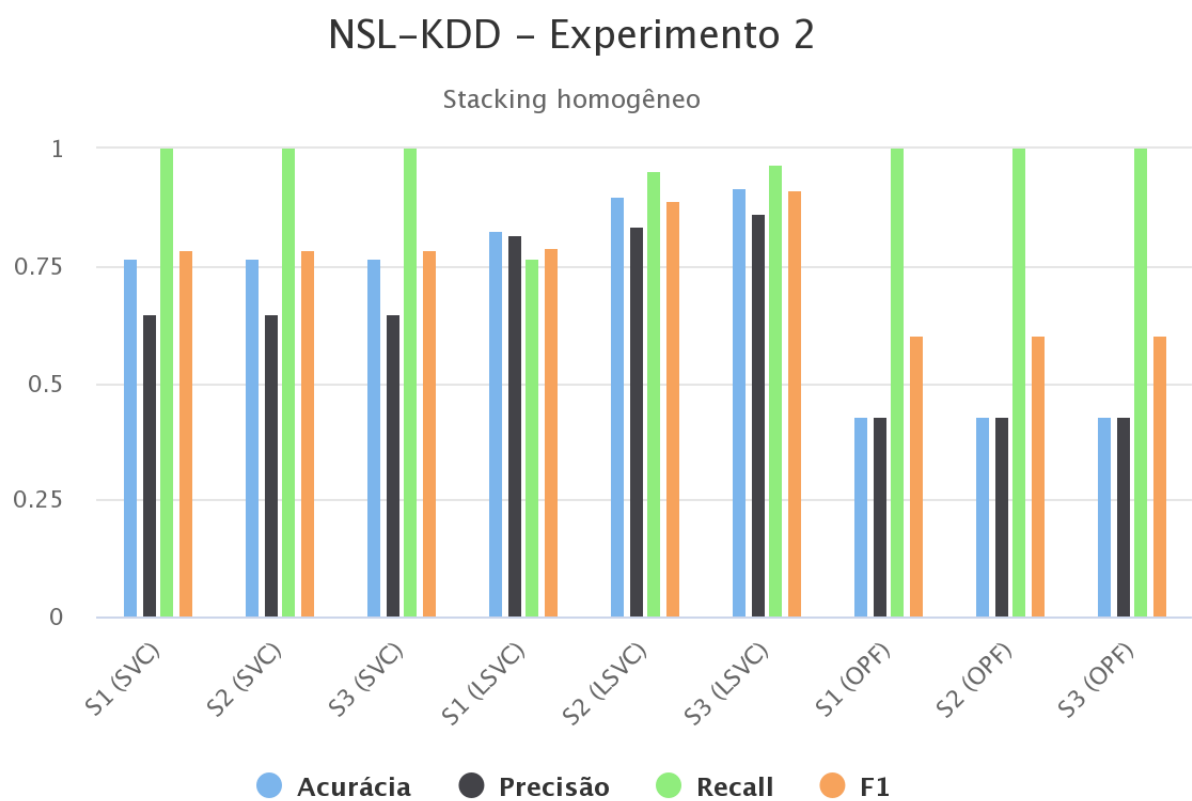
Isso é um problema que pode acontecer em projetos que envolvem Aprendizado de Máquina, mas que só torna-se visível ao realizar o teste estatístico. Nesta solução, se observadas apenas as métricas dos resultados finais, sem aplicação do teste estatístico, a diferença entre os modelos aparenta ser nítida, no entanto, após a implementação do teste, fica evidente que essa é uma percepção equivocada. Assim como apresentado nesta dissertação, inúmeras outras pesquisas apresentam resultados que aparentam ser nitidamente distinguíveis, porém mascarados pela falta de um teste estatístico. Como foi citado antes, na Seção 4.3, o *dataset* é um fator vital para o produto final do trabalho e é uma das maiores dificuldades enfrentadas para obter-se resultados completamente confiáveis.

Figura 26 – Representação gráfica dos resultados obtidos no Experimento 2 com o *dataset* NSL-KDD



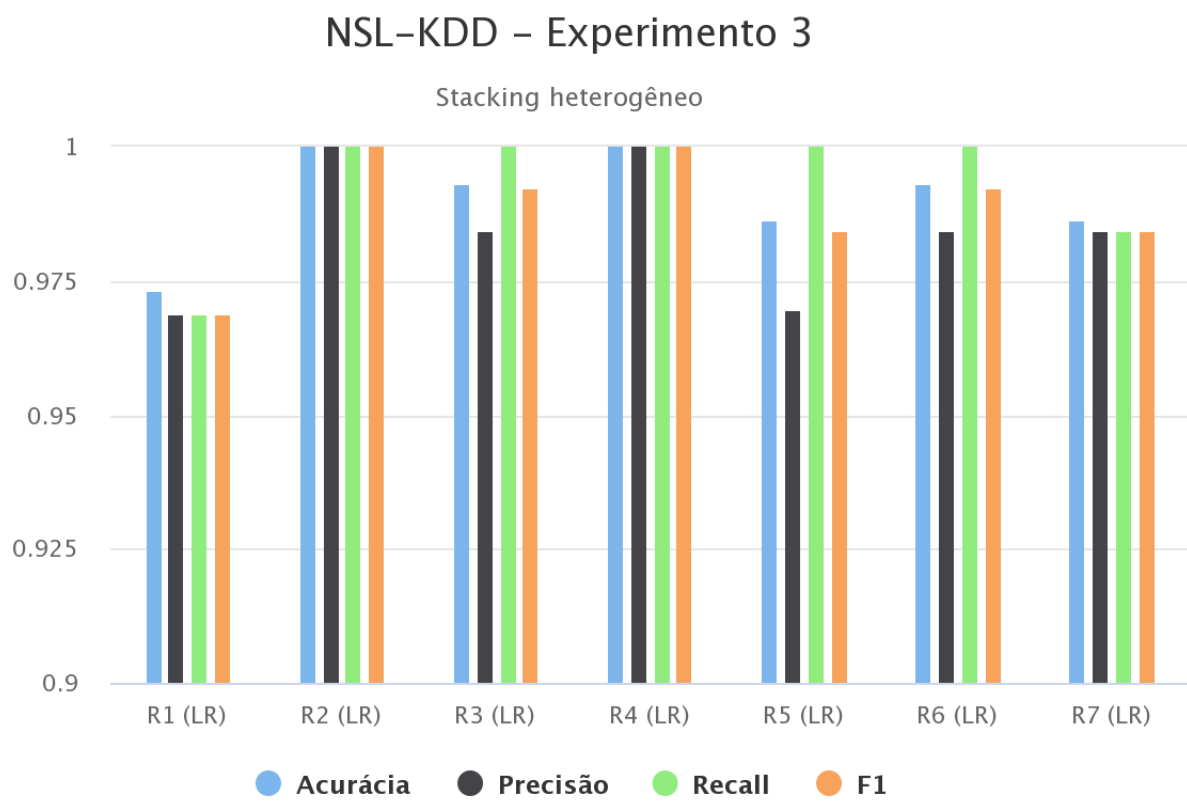
Fonte: Elaborado pelo autor

Figura 27 – Representação gráfica dos resultados obtidos no Experimento 2 com o *dataset* NSL-KDD



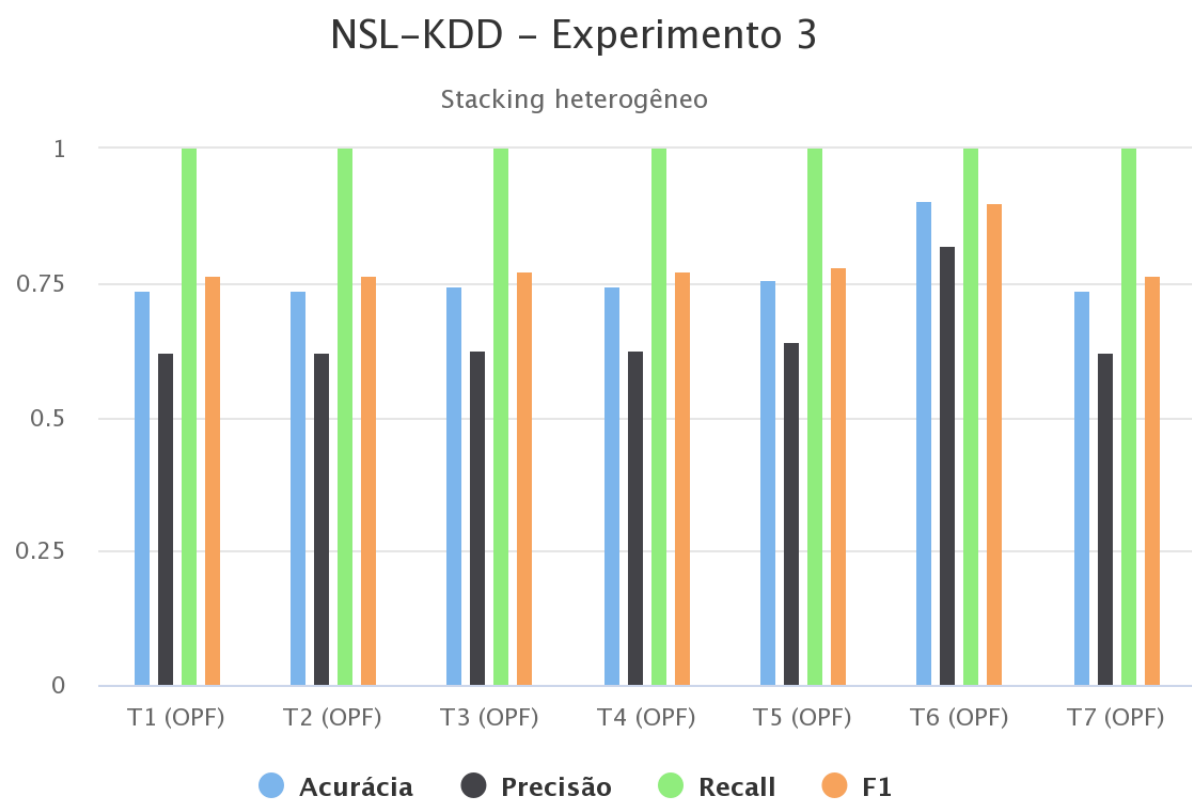
Fonte: Elaborado pelo autor

Figura 28 – Representação gráfica dos resultados obtidos no Experimento 3 com o *dataset* NSL-KDD



Fonte: Elaborado pelo autor

Figura 29 – Representação gráfica dos resultados obtidos no Experimento 3 com o *dataset* NSL-KDD



Fonte: Elaborado pelo autor

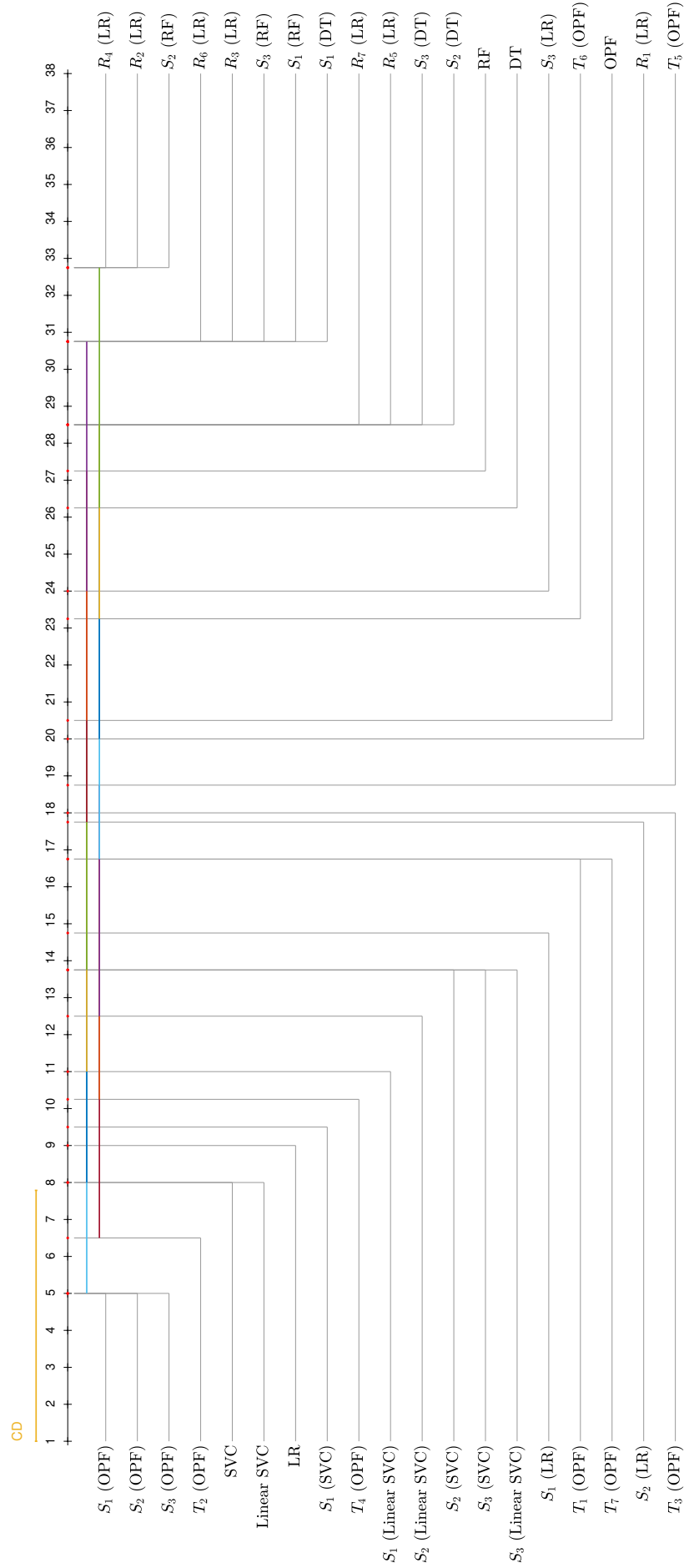


Figura 30 – Análise estatística conduzida pelo teste de Friedman com Nemenyi post-hoc.

6 Conclusão

Com o avanço tecnológico, a quantidade de dispositivos eletrônicos presentes nas atividades das organizações cresceu demasiadamente. É praticamente inimaginável um ambiente de trabalho que funcione sem qualquer auxílio computacional que seja. A computação, por trazer inúmeras vantagens aos modelos de negócios modernos, se enraizou nos processos de negócio de muitas organizações, que necessitam do seu funcionamento constante em alto desempenho para que possam realizar suas operações diárias. Para garantir que as informações, armazenadas eletronicamente, estejam seguras e disponíveis, é preciso entender e aplicar conceitos de Segurança da Informação pertinentes ao contexto considerado.

Este trabalho tem como objetivo contribuir à comunidade de Segurança da Informação, conscientizando sobre a importância desta área tão essencial no panorama atual, onde tudo converge à tecnologia e transmissão de dados. Para realizar esse objetivo, dentro de inúmeros conceitos possivelmente discutíveis nessa área, são considerados os controles lógicos, que procuram garantir a proteção eletrônica de dados sensíveis através da implementação de soluções digitais de segurança. Existem diversas ferramentas implementáveis para desempenhar tal função de proteção, porém este trabalho foca nos Sistemas de Detecção de Intrusão, especialmente àqueles baseados em anomalia, uma vez que o cenário vigente demonstra incessante mutação das ameaças enfrentadas, por parte da defesa, e esta abordagem é a única que apresenta uma solução cabível à detecção de ataques completamente desconhecidos.

A metodologia dos SDIs baseados em anomalia somente tornou-se viável graças às técnicas de Aprendizado de Máquina, que possibilitam que computadores desenvolvam cognição sem o auxílio de programação externa, podendo realizar previsões a partir de um treinamento prévio com dados e, assim, sendo capazes de detectar ameaças desconhecidas. Encontram-se diversas técnicas de ML na literatura, cada uma com sua especialidade, vantagens e desvantagens em relação ao problema em questão. Várias dessas técnicas já foram aplicadas ao problema de detecção de intrusão, no entanto, a maioria delas investigou apenas o uso de classificadores singulares para solucioná-lo. Há uma metodologia em especial, denominada métodos conjuntos, cujo propósito consiste em combinar diversos classificadores singulares para resolver um único problema, que vem obtendo ótimos resultados no problema de detecção de intrusão. Dentre várias técnicas que fazem parte da metodologia dos métodos conjuntos, existe um algoritmo chamado *Stacking*, que tem como objetivo melhorar a capacidade de generalização de um modelo.

Após realizar um estudo extenso sobre quais seriam as melhores técnicas para empregar no problema de detecção de intrusão, decidiu-se, no trabalho desenvolvido, implementar um método conjunto do algoritmo *Stacking* do classificador OPF para o problema de detecção de

intrusão. A literatura afirma que os métodos conjuntos baseiam-se em transformar aprendizes fracos em aprendizes fortes, capazes de realizar previsões melhores que os primeiros. Além disso, a implementação do *Stacking* também visa aprimorar a capacidade de generalização do classificador empregado, que é um fator determinante para a qualidade das previsões. Neste caso, o classificador escolhido como pivô do trabalho foi o OPF, que é eficiente na fase de treinamento, eficaz na fase etapa de classificação e capaz de lidar bem com grandes volumes de dados, elemento inerente ao problema de detecção de intrusão. Todas escolhas tiveram como objetivo alcançar um modelo capaz de identificar anomalias da melhor maneira possível, realizando assim uma grande contribuição para a área de Segurança da Informação.

Sendo assim, este trabalho apresenta um método conjunto baseado no algoritmo *Stacking* utilizando o classificador OPF para o problema de detecção de intrusão. Diversas variantes deste classificador são implementadas com a metodologia *Stacking* em combinações homogêneas e heterogêneas, além de classificadores singulares também, totalizando 38 modelos diferentes gerados. A performance do modelo proposto foi comparada através de outros dois experimentos, que implementam vários classificadores singulares e suas versões homogêneas com o algoritmo *Stacking*, e também mais um que emprega a mesma metodologia utilizada para a implementação do modelo do OPF. Dois conjuntos de dados, uneSPY e NSL-KDD, foram divididos utilizando uma validação cruzada de 10 dobras para evitar viés de treinamento e resultados distorcidos.

A partir da análise dos experimentos, e voltando à hipótese de pesquisa levantada na Introdução, foi possível concluir que implementar o algoritmo *Stacking* realmente melhora a capacidade de generalização de um classificador singular, mas não ao utilizar o classificador OPF. Acredita-se que o procedimento de conquista de nós do OPF, inerente ao funcionamento do algoritmo, não opera corretamente ao ser combinado com a metodologia *Stacking*, e portanto os resultados obtidos são incertos. Além disso, o *dataset* uneSPY, que é fruto de um trabalho interno da UNESP e ainda não havia sido testado até então, aparentou fornecer uma fundação sólida para o treinamento e teste dos modelos, mesmo tendo seus dados desbalanceados devido ao número limitado de potenciais ameaças no conjunto.

6.1 Direcionamentos para Trabalhos Futuros

Tendo em vista que o OPF em sua versão singular obteve resultados tão satisfatórios quanto aqueles que demonstraram o melhor desempenho, fica nítido que a escolha de aplicá-lo para o problema em questão foi adequada. A implementação dos métodos conjuntos produziram resultados convincentes, melhorando as versões singulares dos classificadores e também comprovando sua eficiência na detecção de intrusão. No entanto, como foi possível observar nos resultados dos experimentos, a união do classificador OPF com a metodologia *Stacking* indica ter sido ineficaz, possivelmente devido a fatores inerentes ao algoritmo do

OPF.

Portanto, em trabalhos futuros, para verificar se o problema realmente reside na harmonização das características intrínsecas do OPF com o algoritmo *Stacking*, é interessante implementar a mesma metodologia deste trabalho com outros métodos conjuntos, como o *Boosting* ou *Bagging*, ou até criando um novo método combinado. Diante disso, será possível validar se a adversidade enfrentada estava no algoritmo de *Stacking* ou no classificador OPF, nos *datasets* ou até mesmo na metodologia empregada. Além disso, também pode ser interessante implementar o OPF utilizando outras medidas de distância, e.g. gaussiana ou hamming.

6.2 Publicações Realizadas

O trabalho apresentado foi publicado no periódico *Evolutionary Intelligence*¹⁷ da editora Springer sob o título *Optimum-path forest stacking-based ensemble for intrusion detection*¹⁸.

¹⁷ Disponível em <<https://www.springer.com/journal/12065>>

¹⁸ Disponível em <<https://link.springer.com/article/10.1007/s12065-021-00609-7>>

Referências

- ABU-NIMEH, S.; NAPPA, D.; WANG, X.; NAIR, S. A comparison of machine learning techniques for phishing detection. In: *Proceedings of the Anti-phishing Working Groups 2Nd Annual eCrime Researchers Summit*. New York, NY, USA: ACM, 2007. (eCrime '07), p. 60–69. ISBN 978-1-59593-939-5. Disponível em: <<http://doi.acm.org/10.1145/1299015.1299021>>.
- AMARUDIN; FERDIANA, R.; WIDYAWAN. A systematic literature review of intrusion detection system for network security: Research trends, datasets and methods. In: *2020 4th International Conference on Informatics and Computational Sciences (ICICoS)*. [S.l.: s.n.], 2020. p. 1–6.
- ANDERSON, J. Computer security technology planning study. In: . [S.l.: s.n.], 1972.
- ARAUJO, M.; NEW, M. Araújo mb, new m.. ensemble forecasting of species distributions. *trends ecol evol* 22: 42-47. *Trends in ecology evolution*, v. 22, p. 42–7, 02 2007.
- ASRAFI, N.; LO, D. C.-T.; PARIZI, R. M.; SHI, Y.; CHEN, Y.-W. Comparing performance of malware classification on automated stacking. In: *Proceedings of the 2020 ACM Southeast Conference*. New York, NY, USA: Association for Computing Machinery, 2020. (ACM SE '20), p. 307–308. ISBN 9781450371056. Disponível em: <<https://doi.org/10.1145/3374135.3385316>>.
- BAUER, E.; KOHAVI, R. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, v. 36, n. 1, p. 105–139, Jul 1999. Disponível em: <<https://doi.org/10.1023/A:1007515423169>>.
- BELL, D.; PADULA, L. Secure computer system: Unified exposition and multics interpretation. p. 133, 03 1976.
- BHUYAN, M. H.; BHATTACHARYYA, D. K.; KALITA, J. K. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys Tutorials*, v. 16, n. 1, p. 303–336, 2014.
- BIBA, K. Integrity considerations for secure computer systems. p. 68, 04 1977.
- BLUM, A.; MITCHELL, T. Combining labeled and unlabeled data with co-training. In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. New York, NY, USA: ACM, 1998. (COLT' 98), p. 92–100. ISBN 1-58113-057-0. Disponível em: <<http://doi.acm.org/10.1145/279943.279962>>.
- BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996.
- BREIMAN, L. Stacked regressions. *Machine Learning*, v. 24, n. 1, p. 49–64, Jul 1996. ISSN 1573-0565. Disponível em: <<https://doi.org/10.1023/A:1018046112532>>.
- BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001.
- BRYNJOLFSSON, E. Information assets, technology, and organization. *Management Science, INFORMS*, v. 40, n. 12, p. 1645–1662, 1994. ISSN 00251909, 15265501. Disponível em: <<http://www.jstor.org/stable/2632943>>.

BRYNJOLFSSON, E.; HITT, L. Computing productivity: Firm-level evidence. *The Review of Economics and Statistics*, v. 85, n. 4, p. 793–808, 2003. Disponível em: <<https://EconPapers.repec.org/RePEc:tp:restat:v:85:y:2003:i:4:p:793-808>>.

BUL'AJOUL, W.; JAMES, A.; PANNU, M. Improving network intrusion detection system performance through quality of service configuration and parallel technology. *J. Comput. Syst. Sci.*, Academic Press, Inc., Orlando, FL, USA, v. 81, n. 6, p. 981–999, set. 2015. ISSN 0022-0000. Disponível em: <<http://dx.doi.org/10.1016/j.jcss.2014.12.012>>.

CATANIA, C.; BROMBERG, F.; GARINO, C. G. An autonomous labeling approach to support vector machines algorithms for network traffic anomaly detection. *Expert Syst. Appl.*, v. 39, p. 1822–1829, 2012.

CHAND, N.; MISHRA, P.; KRISHNA, C. R.; PILLI, E. S.; GOVIL, M. C. A comparative analysis of svm and its stacking with other classification algorithm for intrusion detection. In: *2016 International Conference on Advances in Computing, Communication, Automation (ICACCA) (Spring)*. [S.l.: s.n.], 2016. p. 1–6. ISSN null.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 41, n. 3, jul. 2009. ISSN 0360-0300. Disponível em: <<https://doi.org/10.1145/1541880.1541882>>.

CISCO. *CISCO Annual Cybersecurity Report*. 2018. Disponível em: <https://www.cisco.com/c/dam/m/hu_hu/campaigns/security-hub/pdf/acr-2018.pdf>.

CORONA, I.; GIACINTO, G.; MAZZARIELLO, C.; ROLI, F.; SANSONE, C. Information fusion for computer security: State of the art and open issues. *Information Fusion*, v. 10, n. 4, p. 274 – 284, 2009. ISSN 1566-2535. Special Issue on Information Fusion in Computer Security. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S156625350900030X>>.

DASTANPOUR, A.; IBRAHIM, S.; MASHINCHI, M.; SELAMAT, A. Comparison of genetic algorithm optimization on artificial neural network and support vector machine in intrusion detection system. In: . [S.l.: s.n.], 2014.

DEMSAR, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, JMLR.org, v. 7, p. 1–30, dez. 2006. ISSN 1532-4435.

DENNING, D. E. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13, n. 2, p. 222–232, Feb 1987. ISSN 0098-5589.

DIETTERICH, T. G. Ensemble methods in machine learning. In: *Multiple Classifier Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. p. 1–15. ISBN 978-3-540-45014-6.

DIVYASREE, T.; SHERLY, K. A network intrusion detection system based on ensemble cvm using efficient feature selection approach. *Procedia Computer Science*, v. 143, p. 442 – 449, 2018. ISSN 1877-0509. 8th International Conference on Advances in Computing Communications (ICACC-2018). Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1877050918321136>>.

ENYINDAH, P.; ONWUACHU, U. An intrusion detection system application for an organisation. *Circulation in Computer Science*, v. 2, p. 23–28, 04 2017.

FARID, D. M.; ZHANG, L.; HOSSAIN, M. A.; RAHMAN, C. M.; STRACHAN, R.; SEXTON, G.; DAHAL, K. P. An adaptive ensemble classifier for mining concept drifting data streams. *Expert Syst. Appl.*, v. 40, p. 5895–5906, 2013.

FELLIN, C.; HANEY, M. Preventing the mistraining of anomaly-based idss through ensemble systems. In: *2014 IEEE World Congress on Services*. [S.l.: s.n.], 2014. p. 66–68. ISSN 2378-3818.

FERNANDES, S.; PAPA, J. Improving optimum-path forest learning using bag-of-classifiers and confidence measures. *Pattern Analysis and Applications*, v. 22, 05 2019.

FILHO, P. P.; BARROS, C.; RAMALHO, G.; R., P.; PAPA, J.; ALBUQUERQUE, V.; TAVARES, J. Automated recognition of lung diseases in ct images based on the optimum-path forest classifier. *Neural Computing and Applications*, v. 31, p. 901–914, 02 2019.

FISHER, R. A. *Statistical Methods and Scientific Inference*. New York, Hafner, 1959. Disponível em: <<https://www.worldcat.org/title/statistical-methods-and-scientific-inference/oclc/1516472>>.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, [American Statistical Association, Taylor Francis, Ltd.], v. 32, n. 200, p. 675–701, 1937. ISSN 01621459. Disponível em: <<http://www.jstor.org/stable/2279372>>.

GAUTAM, R. K. S.; DOEGAR, E. A. An ensemble approach for intrusion detection system using machine learning algorithms. *2018 8th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, p. 14–15, 2018.

GIACINTO, G.; PERDISCI, R.; RIO, M. D.; ROLI, F. Intrusion detection in computer networks by a modular ensemble of one-class classifiers. *Inf. Fusion*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 9, n. 1, p. 69–82, jan. 2008. ISSN 1566-2535. Disponível em: <<http://dx.doi.org/10.1016/j.inffus.2006.10.002>>.

GIACINTO, G.; ROLI, F.; DIDACI, L. Fusion of multiple classifiers for intrusion detection in computer networks. *Pattern Recogn. Lett.*, Elsevier Science Inc., New York, NY, USA, v. 24, n. 12, p. 1795–1803, ago. 2003. ISSN 0167-8655. Disponível em: <[http://dx.doi.org/10.1016/S0167-8655\(03\)00004-7](http://dx.doi.org/10.1016/S0167-8655(03)00004-7)>.

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

HANSEN, L. K.; SALAMON, P. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 12, n. 10, p. 993–1001, Oct 1990. ISSN 0162-8828.

HAQ, N. F.; ONIK, A. R.; HRIDOY, M. A. K.; RAFNI, M.; SHAH, F. M.; FARID, D. M. Application of machine learning approaches in intrusion detection system: A survey. *International Journal of Advanced Research in Artificial Intelligence*, The Science and Information Organization, v. 4, n. 3, 2015. Disponível em: <<http://dx.doi.org/10.14569/IJARAI.2015.040302>>.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 2nd. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998. ISBN 0132733501.

HSU, Y.; HE, Z.; TARUTANI, Y.; MATSUOKA, M. Toward an online network intrusion detection system based on ensemble learning. In: *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. [S.l.: s.n.], 2019. p. 174–178. ISSN 2159-6182.

JENSEN, F. V. *Introduction to Bayesian Networks*. 1st. ed. Berlin, Heidelberg: Springer-Verlag, 1996. ISBN 0387915028.

KAJA, N.; SHAOUD, A.; MA, D. An intelligent intrusion detection system. *Applied Intelligence*, v. 49, n. 9, p. 3235–3247, Sep 2019. ISSN 1573-7497. Disponível em: <<https://doi.org/10.1007/s10489-019-01436-1>>.

KASPERSKY. *Kaspersky Security Bulletin: Statistics*. 2020. Disponível em: <https://go.kaspersky.com/rs/802-IJN-240/images/KSB_statistics_2020_en.pdf>.

KAUSHIK, S. S.; DESHMUKH, D. P. P. R. Detection of attacks in an intrusion detection system. In: . [S.l.: s.n.], 2011.

KENDALL, K. K. R. *A database of computer attacks for the evaluation of intrusion detection systems*. Tese (Doutorado) — Massachusetts Institute of Technology, 1999.

KRAEMER, K.; DEDRICK, J.; YAMASHIRO, S. Dell computer: Refining and extending the business model with it. *The Information Society - TIS*, 01 2000.

KRISHNAVENI, S.; SIVAMOHAN, S.; SRIDHAR, S.; PRABAKARAN, S. Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing. In: . Cluster Computing, 2021. Disponível em: <<https://doi.org/10.1007/s10586-020-03222-y>>.

KUNE, R.; KONUGURTHI, P. K.; AGARWAL, A.; CHILLARIGE, R. R.; BUYYA, R. The anatomy of big data computing. *Software: Practice and Experience*, v. 46, n. 1, p. 79–105, 2016. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2374>>.

LANGIN, C. L. *A SOM+ Diagnostic System for Network Intrusion Detection*. 2011. <<https://opensiuc.lib.siu.edu/cgi/viewcontent.cgi?article=1389&context=dissertations>>.

LEE, J. Z.; MOSKOVICS, S.; SILACCI, L. A survey of intrusion detection analysis methods. In: . [S.l.: s.n.], 1999.

LI, S.; FU, Q.; GU, L.; SCHOLKOPF, B.; CHENG, Y. Kernel machine based learning for multi-view face detection and pose estimation. *Proceedings of the IEEE International Conference on Computer Vision*, v. 2, 10 2001.

LIPPMANN, R. P.; FRIED, D. J.; GRAF, I.; HAINES, J. W.; KENDALL, K. R.; MCCLUNG, D.; WEBER, D.; WEBSTER, S. E.; WYSCHOGROD, D.; CUNNINGHAM, R. K.; ZISSMAN, M. A. Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation. In: *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*. [S.l.: s.n.], 2000. v. 2, p. 12–26 vol.2. ISSN null.

LIU, B.; YAN, S.; YOU, H.; DONG, Y.; LI, J.; LI, Y.; LANG, J.; GU, R. An ensembled rbf extreme learning machine to forecast road surface temperature. In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. [S.l.: s.n.], 2017. p. 977–980.

MARTINO, V. D.; WIRTH, L. Telework: A new way of working and living. *Int'l Lab. Rev.*, HeinOnline, v. 129, p. 529, 1990.

MCAFEE, A.; BRYNJOLFSSON, E.; DAVENPORT, T.; PATIL, D.; BARTON, D. Big data: The management revolution. *Harvard Bus Rev*, v. 90, p. 61–67, 01 2012.

- MIRZA, A. H. Computer network intrusion detection using various classifiers and ensemble learning. In: *2018 26th Signal Processing and Communications Applications Conference (SIU)*. [S.l.: s.n.], 2018. p. 1–4.
- MOUSSAILEB, R.; BOUGET, B.; PALISSE, A.; BOUDER, H. L.; CUPPENS, N.; LANET, J. Ransomware's early mitigation mechanisms. In: *Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES 2018, Hamburg, Germany, August 27-30, 2018*. [s.n.], 2018. p. 2:1–2:10. Disponível em: <<https://doi.org/10.1145/3230833.3234691>>.
- MUKHOPADHYAY, I.; CHAKRABORTY, M.; CHAKRABARTI, S. A comparative study of related technologies of intrusion detection prevention systems. *J. Information Security*, v. 2, p. 28–38, 01 2011.
- MUSA, U. S.; CHHABRA, M.; ALI, A.; KAUR, M. Intrusion detection system using machine learning techniques: A review. In: *2020 International Conference on Smart Electronics and Communication (ICOSEC)*. [S.l.: s.n.], 2020. p. 149–155.
- NAQA, I. E.; MURPHY, M. J. What is machine learning? In: *machine learning in radiation oncology*. [S.l.]: Springer, 2015. p. 3–11.
- NEMATI, H. *Information security and ethics: concepts, methodologies, tools, and applications: concepts, methodologies, tools, and applications*. [S.l.]: IGI global, 2007.
- NEMENYI, P. *Distribution-free Multiple Comparisons*. Princeton University, 1963. Disponível em: <<https://books.google.com.br/books?id=nhDMtgAACAAJ>>.
- NENE, M.; SINGH, J. A survey on machine learning techniques for intrusion detection systems. *International Journal of Advanced Research in Computer and Communication Engineering*, v. 11, p. 4349, 12 2013.
- OPITZ, D.; MACLIN, R. Popular ensemble methods: An empirical study. v. 11, 12 1999.
- ÖZGÜR, A.; ERDEM, H. A review of kdd99 dataset usage in intrusion detection and machine learning between 2010 and 2015. *PeerJ PrePrints*, v. 4, p. e1954, 2016.
- PALMIERI, F. Network anomaly detection based on logistic regression of nonlinear chaotic invariants. *Journal of Network and Computer Applications*, v. 148, p. 102460, 2019. ISSN 1084-8045. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1084804519303200>>.
- PANIGRAHI, S.; KUNDU, A.; SURAL, S.; MAJUMDAR, A. K. Credit card fraud detection: A fusion approach using dempster-shafer theory and bayesian learning. *Information Fusion*, v. 10, p. 354–363, 2009.
- PAPA, J. P.; FALCAO, A. X.; SUZUKI, C. T. N. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, v. 19, n. 2, p. 120–131, 2009. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/ima.20188>>.
- POLIKAR, R.; TOPALIS, A.; PARIKH, D.; GREEN, D.; FRYMIARE, J.; KOUNIOS, J.; CLARK, C. M. An ensemble based data fusion approach for early diagnosis of alzheimer's disease. *Inf. Fusion*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 9, n. 1, p. 83–95, jan. 2008. ISSN 1566-2535. Disponível em: <<http://dx.doi.org/10.1016/j.inffus.2006.09.003>>.

PRASETIO, B. H.; TAMURA, H.; TANNO, K. Ensemble support vector machine and neural network method for speech stress recognition. In: *2018 International Workshop on Big Data and Information Security (IWBIS)*. [S.l.: s.n.], 2018. p. 57–62.

RAJADURAI, H.; GANDHI, U. D. A stacked ensemble learning model for intrusion detection in wireless network. In: . *Neural Computing and Applications*, 2020. Disponível em: <<https://doi.org/10.1007/s00521-020-04986-5>>.

RAJAGOPAL, S.; KUNDAPUR, P. P.; HAREESHA, K. S. A stacking ensemble for network intrusion detection using heterogeneous datasets. In: . *Security and Communication Networks*, 2020. Disponível em: <<https://doi.org/10.1155/2020/4586875>>.

RASTEGARI, S.; HINGSTON, P.; LAM, C.-P. Evolving statistical rulesets for network intrusion detection. *Applied Soft Computing*, v. 33, p. 348 – 359, 2015. ISSN 1568-4946. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1568494615002689>>.

RAZA, R. A.; WANG, X.-Z.; HUANG, J. Z.; ABBAS, H.; HE, Y.-I. Fuzziness based semi-supervised learning approach for intrusion detection system. *Information Sciences*, v. 378, 04 2016.

REVATHI, S.; MALATHI, A. A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection. In: . [S.l.: s.n.], 2013.

ROSA, G. H. de; PAPA, J. P.; FALCAO, A. X. *OPFython: A Python-Inspired Optimum-Path Forest Classifier*. 2020.

RUBIO, B. de C. Classificação e otimização de características para detecção de anomalias em redes de computadores. 2016.

RUBIO, J. E.; ALCARAZ, C.; ROMAN, R.; LOPEZ, J. Current cyber-defense trends in industrial control systems. *Computers Security*, v. 87, p. 101561, 2019. ISSN 0167-4048. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167404819301245>>.

SALTZER, J. H.; SCHROEDER, M. The protection of information in computer systems. *Proceedings of the IEEE*, v. 63, p. 1278–1308, 1975.

SALZBERG, S. L. C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, v. 16, n. 3, p. 235–240, Sep 1994. ISSN 1573-0565. Disponível em: <<https://doi.org/10.1007/BF00993309>>.

SCHAPIRE, R. E. The strength of weak learnability. *Machine Learning*, v. 5, n. 2, p. 197–227, Jun 1990. ISSN 1573-0565. Disponível em: <<https://doi.org/10.1007/BF00116037>>.

SCHOU, C. D.; TRIMMER, K. J. Information assurance and security. *Journal of Organizational and End User Computing*, IGI Global, v. 16, n. 3, p. 123–145, 2004.

SCHUBERT, P.; LEIMSTOLL, U. Importance and use of information technology in small and medium-sized companies. *Electronic Markets*, v. 17, p. 38–55, 01 2007.

SCHULTZ, M. G.; ESKIN, E.; ZADOK, E.; STOLFO, S. J. Data mining methods for detection of new malicious executables. In: *IEEE Symposium on Security and Privacy*. [S.l.: s.n.], 2001.

SEEMAKURTHI, P.; ZHANG, S.; QI, Y. Detection of fraudulent financial reports with machine learning techniques. In: *2015 Systems and Information Engineering Design Symposium*. [S.l.: s.n.], 2015. p. 358–361.

SOLMS, B. Corporate governance and information security. *Computers Security*, v. 20, p. 215–218, 05 2001.

STALLINGS, W. *Network Security Essentials: Applications and Standards*. 4th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010. ISBN 0136108059, 9780136108054.

TAMA, B. A.; PATIL, A. S.; RHEE, K. An improved model of anomaly detection using two-level classifier ensemble. In: *2017 12th Asia Joint Conference on Information Security (AsiaJCIS)*. [S.l.: s.n.], 2017. p. 1–4.

TANENBAUM, A. S.; FEAMSTER, N.; WETHERALL, D. J. *Computer Networks*. 6th. ed. [S.l.]: Pearson, 2020. ISBN 9780135408001.

TAVALLAEE, M.; BAGHERI, E.; LU, W.; GHORBANI, A. A. A detailed analysis of the kdd cup 99 data set. In: *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. [S.l.: s.n.], 2009. p. 1–6. ISSN 2329-6275.

TAX, D. M. J.; DUIN, R. P. W. Data domain description using support vectors. In: *Proceedings of the European Symposium on Artificial Neural Networks*. [S.l.: s.n.], 1999. p. 251–256.

TIMČENKO, V.; GAJIN, S. Ensemble classifiers for supervised anomaly based network intrusion detection. In: *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*. [S.l.: s.n.], 2017. p. 13–19.

VALDES, A.; SKINNER, K. Probabilistic alert correlation. In: *Recent Advances in Intrusion Detection (RAID 2001)*. [S.l.]: Springer-Verlag, 2001. (Lecture Notes in Computer Science, 2212).

VANERIO, J.; CASAS, P. Ensemble-learning approaches for network security and anomaly detection. In: *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*. New York, NY, USA: ACM, 2017. (Big-DAMA '17), p. 1–6. ISBN 978-1-4503-5054-9. Disponível em: <<http://doi.acm.org/10.1145/3098593.3098594>>.

VERIZON. *Data Breach Investigations Report*. 2020. Disponível em: <<https://enterprise.verizon.com/resources/reports/2020-data-breach-investigations-report.pdf>>.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. [S.l.: s.n.], 2001. v. 1, p. I–I. ISSN 1063-6919.

VIOLA, P.; JONES, M. J. Robust real-time face detection. *International Journal of Computer Vision*, v. 57, n. 2, p. 137–154, May 2004. ISSN 1573-1405. Disponível em: <<https://doi.org/10.1023/B:VISI.0000013087.49260.fb>>.

WANG, B.; LIU, Y.; QIAN, J.; PARKER, S. K. Achieving effective remote working during the covid-19 pandemic: A work design perspective. *Applied psychology*, Wiley Online Library, v. 70, n. 1, p. 16–59, 2021.

WEST, D. Neural network ensemble strategies for financial decision applications. *Computers Operations Research*, v. 32, 05 2004.

WHITMAN, M. E.; MATTORD, H. J. *Principles of information security*. [S.l.]: Cengage Learning, 2011.

WILCOXON, F. Individual comparisons by ranking methods. *Biometrics Bulletin*, [International Biometric Society, Wiley], v. 1, n. 6, p. 80–83, 1945. ISSN 00994987. Disponível em: <<http://www.jstor.org/stable/3001968>>.

WOLPERT, D. H. Stacked generalization. *Neural Networks*, v. 5, n. 2, p. 241 – 259, 1992. ISSN 0893-6080. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0893608005800231>>.

XU, L.; KRZYZAK, A.; SUEN, C. Y. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 22, n. 3, p. 418–435, May 1992. ISSN 0018-9472.

YAN, W.; XUE, F. Jet engine gas path fault diagnosis using dynamic fusion of multiple classifiers. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. [S.l.: s.n.], 2008. p. 1585–1591. ISSN 2161-4393.

YANG, P.; WAN, X.; SHI, G.; QU, H.; LI, J.; YANG, L. Naruto: Dns covert channels detection based on stacking model. In: _____. *Proceedings of the 2020 The 2nd World Symposium on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2020. p. 109–115. ISBN 9781450387873. Disponível em: <<https://doi.org/10.1145/3425329.3425336>>.

ZHOU, Y.; CHENG, G. An efficient network intrusion detection system based on feature selection and ensemble classifier. *CoRR*, abs/1904.01352, 2019. Disponível em: <<http://arxiv.org/abs/1904.01352>>.

ZHOU, Z.-H. *Ensemble methods : foundations and algorithms*. 1. ed. CRC Press, 2012. Hardcover. ISBN 1439830037. Disponível em: <<http://www.worldcat.org/isbn/1439830037>>.

ZHOU, Z.-H.; JIANG, Y.; YANG, Y.-B.; CHEN, S.-F. Lung cancer cell identification based on artificial neural network ensembles. *Artificial Intelligence in Medicine*, v. 24, n. 1, p. 25 – 36, 2002. ISSN 0933-3657. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S093336570100094X>>.

Apêndices

APÊNDICE A – Propriedades do *dataset* uneSPY

Este apêndice descreve detalhes do *dataset* uneSPY que podem ser úteis para aqueles que desejarem utilizá-lo no futuro. Para atender as necessidades do classificador OPF, foi necessário transformar o *dataset* em dados numéricos. Uma vez que manter os controles e padrões dos dados era essencial, tabelas de codificação foram criadas para cada dado alfanumérico transformado. Assim, o *dataset* preserva sua informação e pode ser utilizado sem comprometer sua essência.

Como foi mencionado na Seção 4.3.1, o conjunto de dados uneSPY possui 23 características, que são apresentadas na Tabela 4. O *dataset* original têm três características que foram ignoradas, que são "*Source*", "*Destination*" e "*Info*". A remoção foi necessária devido à incompatibilidade das mesmas com o classificador OPF. Além disso, Tabela 5 apresenta os dados relacionados aos "Tipos IGMP"; Tabela 6 descreve a codificação das mensagens DHCPv6; Tabela 7 introduz a classificação dos tipos de pacotes; e Tabela 8 exibe os protocolos codificados do conjunto de dados.

¹⁹ IANA - Internet Assigned Numbers Authority. Disponível em <<http://www.iana.org/>>

Tabela 4 – Descrição das características do conjunto de dados uneSPY.

Código	Descrição
1	<i>Attack Code</i>
2	<i>Time</i>
3	<i>Same IP</i>
4	<i>Broadcast Destination</i>
5	<i>Protocol</i>
6	<i>Packet Length</i>
7	<i>UDP Source Port</i>
8	<i>UDP Destination Port</i>
9	<i>IPv4 Header Length</i>
10	<i>DHCPv6 Message type</i>
11	<i>Malformed Packet</i>
12	<i>Host Identity Protocol</i>
13	<i>ICMP Type</i>
14	<i>ICMP Code</i>
15	<i>IGMP Type</i>
16	<i>IPv4 Time to live</i>
17	<i>TCP Source Port</i>
18	<i>TCP Destination Port</i>
19	<i>TCP Header Length</i>
20	<i>IPv4 Total Length</i>
21	<i>Type</i>
22	<i>UDP Length</i>
23	<i>IPv6 Hop limit (time to live)</i>

Fonte: Bruna C. Rubio (RUBIO, 2016).

Tabela 5 – Descrição dos tipos IGMP empregados no conjunto de dados uneSPY.

Protocolo	Tipo IGMP	IANA ¹⁹	Código
-	Vazio	-	-1
IGMPv1	<i>Membership Report</i>	0x12	12
IGMPv2	<i>Membership Query</i>	0x11	11
IGMPv2	<i>Membership Report</i>	0x16	16
IGMPv2	<i>Leave Group</i>	0x17	17
IGMPv3	<i>Membership Report</i>	0x22	22

Fonte: Bruna C. Rubio (RUBIO, 2016).

Tabela 6 – Descrição da codificação das mensagens DHCPv6 empregadas no conjunto de dados uneSPY.

Tipo de DHCPv6 mensagem	Código (IANA)
Vazio	$-(-1)^2$
<i>Solicit</i>	1
<i>Advertise</i>	2
<i>Request</i>	3
<i>Confirm</i>	4
<i>Renew</i>	5
<i>Rebind</i>	6
<i>Reply</i>	7
<i>Release</i>	8
<i>Information-request</i>	11

Fonte: Bruna C. Rubio (RUBIO, 2016).

Tabela 7 – Descrição da codificação dos tipos de pacotes empregados no conjunto de dados uneSPY.

Tipo	Código
Vazio	0
IPv4	1
IPv6	2
ARP	3
802.1X Authentication	4

Fonte: Bruna C. Rubio (RUBIO, 2016).

Tabela 8 – Descrição da codificação dos protocolos empregados no conjunto de dados uneSPY.

Protocolo	Código
ALLJOYN-NS	1
ARP	2
BJNP	3
BROWSER	4
CUPS	5
DB-LSP-DISC	6
DHCP	7
DHCPv6	8
DNS	9
EAP	10
EAPOL	11
Elasticsearch	12
ESP	13
FTP	14
HIP	15
HPEXT	16
HTTP	17
ICMP	18
IGMPv1	19
IGMPv2	20
IGMPv3	21
IMAP	22
IPv4	23
IPv6	24
LLC	25
LLMNR	26
MDNS	27
NBNS	28
NetBIOS	29
OSPF	30
POP3	31
QUIC	32
SMB Mailslot	33
SNMP	34
SRVLOC	35
SSDP	36
TCP	37
TELNET	38
TLSv1	39
TLSv1.2	40
UDP	41
WOL	42
XID	43

Fonte: Bruna C. Rubio (RUBIO, 2016).

APÊNDICE B – Tabelas dos Resultados Experimentais

Este apêndice exibe todos os resultados provenientes dos experimentos apresentados no Capítulo ???. Como havia sido mencionado previamente, em relação ao *dataset* *uneSPY*, a Tabela 9 refere-se ao experimento dos classificadores singulares, a Tabela 10 demonstra os resultados dos conjuntos *stack* homogêneos e a Tabela 11 apresenta os modelos conjuntos *stack* heterogêneos. Além disso, sobre o *dataset* *NSL-KDD*, a Tabela 12 demonstra os resultados dos classificadores singulares, a Tabela 13 representa o experimento dos conjuntos *stack* homogêneos e a Tabela 14 exibe os resultados obtidos pelos conjuntos *stack* heterogêneos.

Tabela 9 – Média e desvio padrão dos classificadores singulares treinados no *dataset* *uneSPY*

Classificador	Acurácia	Precisão	<i>Recall</i>	F1	Tempo (s)
DT	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	0.068 ± 0.004
LR	0.78339 ± 0.08644	0.95891 ± 0.01433	0.79233 ± 0.09114	0.86534 ± 0.05903	5.126 ± 0.166
RF	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	0.431 ± 0.006
SVC	0.90002 ± 0.01167	0.90002 ± 0.01167	1.00000 ± 0.00000	0.94734 ± 0.00646	150.689 ± 3.929
Linear SVC	0.70283 ± 0.22971	0.92951 ± 0.02439	0.72855 ± 0.289279	0.77821 ± 0.20811	12.223 ± 0.003
OPF	0.99813 ± 0.00149	0.99858 ± 0.00155	0.99935 ± 0.00110	0.99896 ± 0.00082	17684.7 ± 114.200

Fonte: Elaborado pelo autor.

Tabela 10 – Média e desvio padrão dos *Stackings* homogêneos treinados no *dataset* uneSPY

Classificador	Acurácia	Precisão	Recall	F1	Tempo (s)
S_1 (DT)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	0.198 ± 0.000
S_2 (DT)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	0.322 ± 0.000
S_3 (DT)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	0.447 ± 0.000
S_1 (LR)	0.92224 ± 0.00000	0.92224 ± 0.00000	1.00000 ± 0.00000	0.95955 ± 0.00000	11.684 ± 0.000
S_2 (LR)	0.98327 ± 0.00000	0.98729 ± 0.00000	0.99466 ± 0.00000	0.99096 ± 0.00000	96.012 ± 0.000
S_3 (LR)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	107.793 ± 0.000
S_1 (RF)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.291 ± 0.000
S_2 (RF)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	2.097 ± 0.000
S_3 (RF)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	2.796 ± 0.000
S_1 (SVC)	0.92224 ± 0.00000	0.92224 ± 0.00000	1.00000 ± 0.00000	0.95955 ± 0.00000	246.034 ± 0.000
S_2 (SVC)	0.99803 ± 0.00000	0.99787 ± 0.00000	1.00000 ± 0.00000	0.99893 ± 0.00000	254.044 ± 0.000
S_3 (SVC)	0.99803 ± 0.00000	0.99787 ± 0.00000	1.00000 ± 0.00000	0.99893 ± 0.00000	194.757 ± 0.000
S_1 (Linear SVC)	0.92224 ± 0.00000	0.92224 ± 0.00000	1.00000 ± 0.00000	0.95955 ± 0.00000	33.007 ± 0.000
S_2 (Linear SVC)	0.92224 ± 0.00000	0.92224 ± 0.00000	1.00000 ± 0.00000	0.95955 ± 0.00000	344.403 ± 0.000
S_3 (Linear SVC)	0.92224 ± 0.00000	0.92224 ± 0.00000	1.00000 ± 0.00000	0.95955 ± 0.00000	618.311 ± 0.000
S_1 (OPF)	0.92224 ± 0.00000	0.92224 ± 0.00000	1.00000 ± 0.00000	0.95955 ± 0.00000	39708.4 ± 0.000
S_2 (OPF)	0.92224 ± 0.00000	0.92224 ± 0.00000	1.00000 ± 0.00000	0.95955 ± 0.00000	65582.5 ± 0.000
S_3 (OPF)	0.92224 ± 0.00000	0.92224 ± 0.00000	1.00000 ± 0.00000	0.95955 ± 0.00000	93661.7 ± 0.000

Fonte: Elaborado pelo autor.

Tabela 11 – Média e desvio padrão dos *Stackings* heterogêneos treinados no *dataset* uneSPY

Classificador	Acurácia	Precisão	Recall	F1	Tempo (s)
R_1 (LR)	0.99606 ± 0.00000	0.99893 ± 0.00000	0.99679 ± 0.00000	0.99786 ± 0.00000	13394.1 ± 0.000
R_2 (LR)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	12636.9 ± 0.000
R_3 (LR)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	12967.3 ± 0.000
R_4 (LR)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	12651.6 ± 0.000
R_5 (LR)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	14978.9 ± 0.000
R_6 (LR)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	13160.5 ± 0.000
R_7 (LR)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	13031.1 ± 0.000
T_1 (OPF)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	2195.3 ± 0.000
T_2 (OPF)	0.92224 ± 0.00000	0.92224 ± 0.00000	1.00000 ± 0.00000	0.95955 ± 0.00000	2423.3 ± 0.000
T_3 (OPF)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	2263.8 ± 0.000
T_4 (OPF)	0.97933 ± 0.00000	0.97808 ± 0.00000	1.00000 ± 0.00000	0.98892 ± 0.00000	2076.7 ± 0.000
T_5 (OPF)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	2396.1 ± 0.000
T_6 (OPF)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1746.9 ± 0.000
T_7 (OPF)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1674.7 ± 0.000

Fonte: Elaborado pelo autor.

Tabela 12 – Média e desvio padrão dos classificadores singulares treinados no *dataset* NSL-KDD

Classificador	Acurácia	Precisão	Recall	F1	Tempo (s)
DT	0.97237 ± 0.01256	0.97531 ± 0.01036	0.97163 ± 0.01895	0.97337 ± 0.01177	0.091 ± 0.001
LR	0.88816 ± 0.01668	0.88403 ± 0.03666	0.90288 ± 0.036751	0.89219 ± 0.01923	0.386 ± 0.006
RF	0.97979 ± 0.01127	0.97336 ± 0.02454	0.98803 ± 0.01223	0.98036 ± 0.01115	0.214 ± 0.001
SVC	0.80794 ± 0.02522	0.72824 ± 0.04414	1.00000 ± 0.00000	0.84199 ± 0.029916	2.586 ± 0.075
Linear SVC	0.83831 ± 0.04035	0.86769 ± 0.05514	0.82151 ± 0.10301	0.83809 ± 0.04491	1.281 ± 0.004
OPF	0.96632 ± 0.01505	0.96917 ± 0.02269	0.96484 ± 0.01752	0.96683 ± 0.01599	333.5 ± 0.111

Fonte: Elaborado pelo autor.

Tabela 13 – Média e desvio padrão dos *Stackings* homogêneos treinados no *dataset* NSL-KDD

Classificador	Acurácia	Precisão	Recall	F1	Tempo (s)
S_1 (DT)	0.99329 ± 0.00000	0.98462 ± 0.00000	1.00000 ± 0.00000	0.99225 ± 0.00000	0.235 ± 0.000
S_2 (DT)	0.98658 ± 0.00000	0.96969 ± 0.00000	1.00000 ± 0.00000	0.98462 ± 0.00000	0.318 ± 0.000
S_3 (DT)	0.98658 ± 0.00000	0.96969 ± 0.00000	1.00000 ± 0.00000	0.98462 ± 0.00000	0.337 ± 0.000
S_1 (LR)	0.92617 ± 0.00000	0.87324 ± 0.00000	0.96875 ± 0.00000	0.91852 ± 0.00000	5.161 ± 0.000
S_2 (LR)	0.92617 ± 0.00000	0.87324 ± 0.00000	0.96875 ± 0.00000	0.91852 ± 0.00000	19.243 ± 0.000
S_3 (LR)	0.91946 ± 0.00000	0.86111 ± 0.00000	0.96875 ± 0.00000	0.91176 ± 0.00000	21.963 ± 0.000
S_1 (RF)	0.99329 ± 0.00000	0.98462 ± 0.00000	1.00000 ± 0.00000	0.99225 ± 0.00000	0.672 ± 0.000
S_2 (RF)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	0.996 ± 0.000
S_3 (RF)	0.99329 ± 0.00000	0.98462 ± 0.00000	1.00000 ± 0.00000	0.99225 ± 0.00000	1.315 ± 0.000
S_1 (SVC)	0.76511 ± 0.00000	0.64646 ± 0.00000	1.00000 ± 0.00000	0.78527 ± 0.00000	6.456 ± 0.000
S_2 (SVC)	0.76511 ± 0.00000	0.64646 ± 0.00000	1.00000 ± 0.00000	0.78527 ± 0.00000	9.037 ± 0.000
S_3 (SVC)	0.76511 ± 0.00000	0.64646 ± 0.00000	1.00000 ± 0.00000	0.78527 ± 0.00000	12.928 ± 0.000
S_1 (Linear SVC)	0.82551 ± 0.00000	0.81667 ± 0.00000	0.76563 ± 0.00000	0.79032 ± 0.00000	54.03 ± 0.000
S_2 (Linear SVC)	0.89933 ± 0.00000	0.83562 ± 0.00000	0.95313 ± 0.00000	0.89051 ± 0.00000	109.3 ± 0.000
S_3 (Linear SVC)	0.91946 ± 0.00000	0.86111 ± 0.00000	0.96875 ± 0.00000	0.91176 ± 0.00000	107.6 ± 0.000
S_1 (OPF)	0.42953 ± 0.00000	0.42953 ± 0.00000	1.00000 ± 0.00000	0.60094 ± 0.00000	993.9 ± 0.000
S_2 (OPF)	0.42953 ± 0.00000	0.42953 ± 0.00000	1.00000 ± 0.00000	0.60094 ± 0.00000	1574.2 ± 0.000
S_3 (OPF)	0.42953 ± 0.00000	0.42953 ± 0.00000	1.00000 ± 0.00000	0.60094 ± 0.00000	2069.8 ± 0.000

Fonte: Elaborado pelo autor.

Tabela 14 – Média e desvio padrão dos *Stackings* heterogêneos treinados no *dataset* uneSPY

Classificador	Acurácia	Precisão	Recall	F1	Tempo (s)
R_1 (LR)	0.97315 ± 0.00000	0.96875 ± 0.00000	0.96875 ± 0.00000	0.96875 ± 0.00000	290.7 ± 0.000
R_2 (LR)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	307.9 ± 0.000
R_3 (LR)	0.99329 ± 0.00000	0.98462 ± 0.00000	1.00000 ± 0.00000	0.99225 ± 0.00000	294.2 ± 0.000
R_4 (LR)	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	1.00000 ± 0.00000	297.5 ± 0.000
R_5 (LR)	0.98658 ± 0.00000	0.96969 ± 0.00000	1.00000 ± 0.00000	0.98462 ± 0.00000	291.5 ± 0.000
R_6 (LR)	0.99329 ± 0.00000	0.98462 ± 0.00000	1.00000 ± 0.00000	0.99225 ± 0.00000	289.9 ± 0.000
R_7 (LR)	0.98658 ± 0.00000	0.98438 ± 0.00000	0.98438 ± 0.00000	0.98438 ± 0.00000	299.8 ± 0.000
T_1 (OPF)	0.73826 ± 0.00000	0.62136 ± 0.00000	1.00000 ± 0.00000	0.76647 ± 0.00000	33.127 ± 0.000
T_2 (OPF)	0.73826 ± 0.00000	0.62136 ± 0.00000	1.00000 ± 0.00000	0.76647 ± 0.00000	33.783 ± 0.000
T_3 (OPF)	0.74497 ± 0.00000	0.62745 ± 0.00000	1.00000 ± 0.00000	0.77108 ± 0.00000	32.595 ± 0.000
T_4 (OPF)	0.74497 ± 0.00000	0.62745 ± 0.00000	1.00000 ± 0.00000	0.77108 ± 0.00000	28.169 ± 0.000
T_5 (OPF)	0.75839 ± 0.00000	0.64 ± 0.00000	1.00000 ± 0.00000	0.78049 ± 0.00000	26.394 ± 0.000
T_6 (OPF)	0.90604 ± 0.00000	0.82051 ± 0.00000	1.00000 ± 0.00000	0.90141 ± 0.00000	19.307 ± 0.000
T_7 (OPF)	0.73826 ± 0.00000	0.62136 ± 0.00000	1.00000 ± 0.00000	0.76647 ± 0.00000	34.295 ± 0.000

Fonte: Elaborado pelo autor.