UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ

CAMPUS DE FOZ DO IGUAÇU

PROGRAMA DE PÓS-GRADUAÇÃO EM
ENGENHARIA ELÉTRICA E COMPUTAÇÃO

DISSERTAÇÃO DE MESTRADO

COMBINAÇÃO AUTOAJUSTADA DE MODELOS DE APRENDIZAGEM DE MÁQUINA COM OTIMIZAÇÃO DE HIPERPARÂMETROS PARA DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES

EDGARD MOTA DE OLIVEIRA

FOZ DO IGUAÇU
2020

Edgard Mota de Oliveira

Combinação Autoajustada de Modelos de Aprendizagem de Máquina com Otimização de Hiperparâmetros para Detecção de Intrusão em Redes de Computadores

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Computação da Universidade Estadual do Paraná como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica e Computação. Área de concentração: Sistemas Elétricos e Computação.

Orientador: Prof. Dr. Renato Bobsin Machado
Co-orientador: Prof. Dr. Thiago França Naves

Foz do Iguaçu
2020

Ficha de identificação da obra elaborada através do Formulário de Geração Automática do Sistema de Bibliotecas da Unioeste.

Oliveira, Edgard Mota de
Combinação Autoajustada de Modelos de Aprendizagem de Máquina com Otimização de Hiperparâmetros para Detecção de Intrusão em Redes de Computadores / Edgard Mota de Oliveira; orientador(a), Renato Bobsin Machado; coorientador(a), Thiago França Naves, 2020.
162 f.

Dissertação (mestrado), Universidade Estadual do Oeste do Paraná, Centro de Engenharias e Ciências Exatas, Programa de Pós-Graduação em Engenharia Elétrica e Computação, 2020.

1. IDS. 2. Sistemas de Detecção de Intrusões em Redes de Computadores. 3. Aprendizagem de Máquina. 4. Otimização de Hiperparâmetros. I. Machado, Renato Bobsin. II. Naves, Thiago França. III. Título.

Combinação Autoajustada de Modelos de Aprendizagem de Máquina com Otimização de Hiperparâmetros para Detecção de Intrusão em Redes de Computadores

Edgard Mota de Oliveira

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Computação e aprovada pela Banca Examinadora assim constituída:

Prof. Dr. **Prof. Dr. Renato Bobsin Machado** - (Orientador)
Universidade Estadual do Oeste do Paraná - UNIOESTE

Prof. Dr. **Thiago França Naves** - (Co-orientador)
Universidade Tecnológica Federal do Paraná - UTFPR

Prof. Dr. **Davi Marcondes Rocha**
Universidade Tecnológica Federal do Paraná - UTFPR

Prof. Dr. **Romeu Reginatto**
Universidade Estadual do Oeste do Paraná - UNIOESTE

Data da defesa: 9 de dezembro de 2020.

Resumo

O aumento dos incidentes de segurança reportados nos últimos anos acompanha a expansão dos sistemas de informação nos ambientes institucionais/corporativos e o crescimento no uso da Internet. Em resposta a essa situação, iniciativas variadas se ocupam em oferecer proteção aos ambientes digitais, buscando evitar a ocorrência de intrusões em redes de computadores. Dentro do escopo dessas iniciativas, o trabalho proposto nesta dissertação consiste em um método genérico que emprega a combinação autônoma/autoajustada de soluções de aprendizagem de máquina para a detecção de intrusões em redes de computadores. O método genérico proposto faz uso de otimização de hiperparâmetros aplicada à modelagem de aprendizagem de máquina e é validado pela implementação de uma instância sua composta por uma rede *Multi-layer Perceptron* e o método *K-Nearest Neighbors*; a rede *Multi-layer Perceptron*, após ser alvo de um processo de otimização de hiperparâmetros, tem sua classificação realizada em conjunto com o método *K-Nearest Neighbors* segundo critérios autônomos de composição da classificação de ambos os modelos. Por meio de experimentos, concluiu-se que a solução proposta apresenta melhorias em termos de métricas de desempenho em relação a modelos de combinação de classificadores voltados a detecção de intrusão em redes de computadores, com resultados comparáveis a de outras soluções propostas no estado da arte.

Palavras-chave: NIDS, IDS, Sistemas de Detecção de Intrusões em Redes de Computadores, Aprendizagem de Máquina, Otimização de Hiperparâmetros, Combinação de Classificadores.

Abstract

The increase in security incidents reported in recent years accompanies the expansion of information systems in institutional/corporate environments and the growth in the use of the Internet. In response to this situation, various initiatives are concerned with offering protection to digital environments, seeking to avoid the occurrence of intrusions in computer networks. Within the scope of these initiatives, the work proposed in this thesis consists of a generic method that employs the autonomous/self-adjusting combination of machine learning solutions for the detection of intrusions in computer networks. The proposed generic method makes use of hyperparameter optimization applied to machine learning modeling and is validated by the implementation of an instance of it composed of a Multi-layer Perceptron network and the K-Nearest Neighbors method; the Multi-layer Perceptron network, after being the target of a hyperparameter optimization process, has its classification carried out in conjunction with the K-Nearest Neighbors method according to autonomous criteria for composing the classification of both models. Through experiments, it was concluded that the proposed solution presents improvements in terms of performance metrics in relation to models of combination of classifiers aimed at intrusion detection in computer networks, with results comparable to other solutions proposed in recent literature.

Keywords: NIDS, IDS, Network Intrusion Detection System, Machine Learning, Hyperparameter Optimization, Ensemble Learning.

Dedico este trabalho à minha filha, Clara.

Agradecimentos

Inicio esta nota de agradecimento com um extenso prólogo. Acredito que, por meio dele, ficará mais clara a contribuição que cada pessoa citada deu ao longo dessa jornada. Devo isso a mim e a todos que, de alguma forma, contribuíram nos últimos 16 anos para que eu pudesse, enfim, ostentar o título que me é agora outorgado. Esta nota de agradecimento será um pouco diferente da maioria pois irá contemplar um período bem longo, sendo, por esse motivo, igualmente longa. A certeza de que esqueceria de agradecer diversos nomes quase me fez abrir mão de escrevê-la. Seria, no entanto, injusto até mesmo comigo deixar de render algumas palavras em relação a tudo que passou. Foram, afinal, três mestrados cursados - dois dos quais, perdidos - para a obtenção de um único título. Esta nota agradece a quem contribuiu em quaisquer desses três momentos.

Na primeira vez que ingressei em um programa do tipo, recém-saído da graduação, um turbilhão de motivos me levaram a solicitar o desligamento do curso por iniciativa própria, mesmo já estando qualificado. Foi algo bem emblemático pois nunca havia desistido de nada até então. Segui a vida, a carreira; casei, me tornei pai. Com 10 anos a mais de maturidade, decidi voltar a cursar o mesmo programa que havia abandonado anos antes. Iria até o fim dessa vez. Iria.

Nesse meu segundo ingresso, quando, em seu interstício, reprovei pela segunda vez no exame de qualificação, tudo parecia ter sido em vão. Os mais de 700 km semanais de estrada ao quais, durante meses, eu, minha esposa e filha, então bebê, nos submetemos para acompanhamento das aulas e finalização dos créditos. O valor financeiro investido. O desgaste físico e psicológico. Além do sentimento de fracasso, restou desse tempo apenas uma dívida correspondente a um mês do meu salário bruto.

Para minha própria surpresa, o único ímpeto que tive foi o de continuar. Sem lamentar um único dia, procurei por um novo programa em minha área de conhecimento; um que fosse mais próximo de onde, agora, residia. Não queria mais me sujeitar a viagens tão longas e frequentes. Não tinha mais qualquer estrutura para isso. Foi aí que conheci o PGEEC da Unioeste em Foz do Iguaçu.

Finalizado esse longo prólogo, inicio agora os agradecimentos que me comprometi a fazer. Vou iniciar agradecendo à minha família de berço - aquela na qual eu nasci e que me criou. Afinal, o início de tudo foi ali, e nada mais justo do que iniciar por ela. Agradeço à minha mãe, Louise, que mesmo contrariada na vez em que desisti, com um jeito só seu, nunca deixou de acreditar, ajudar e me incentivar para que eu pudesse colher os frutos que hoje colho. Ao meu pai, Toninho, agradeço por jamais ter lançado uma palavra de reprovação para os

meus atos, por me ter sido exemplo de persistência em vários momentos de sua vida, e por ter comemorado junto comigo cada pequena conquista que me levou até o fim desse ciclo. Ao meu irmão, Luciano, agradeço por ter estado comigo sempre que pôde, em todo o processo, em cada qualificação, desde o primeiro mestrado, sempre certo do meu sucesso. Ele assistiu aos meus tropeços mas nunca deixou de esperar para ver meu triunfo.

Agradeço também à minha família estendida - a que me acolheu quando casei com minha virtuosa esposa. Aos seus avós, Seu Lázaro e Dona Deáuzea, agradeço por terem me oferecido sua casa em hospedagem quando precisava me deslocar para cursar as disciplinas na época do segundo mestrado. Agradeço aos meus sogros, Seu Carlos e Dona Sheila, que, nas mesmas ocasiões de hospedagem, sempre me receberam com uma bebida bem gelada e uma refeição bem quentinha, além de terem acreditado em mim durante esse processo de tantos anos. Aos meus cunhados, Guilherme e Michel, agradeço-os pela expressa admiração que tem por mim, algo que me envaidece mas também me custa grande responsabilidade.

Seria injusto, mesmo após tantos anos, deixar de prestar meu agradecimento ao CNPq, pelo fomento em forma de bolsa que me foi concedido na ocasião do primeiro mestrado. O aporte financeiro que recebi naquele momento me foi muito precioso, motivador e bem-vindo, tendo sido o primeiro valor significativo que fiz enquanto jovem adulto que era a época.

Agradeço ainda aos diversos amigos e colegas que fiz por todo o período de estudo compreendido pelos três cursos. Na UEM, onde cursei os dois primeiros mestrados, agradeço especialmente ao meu antigo orientador e amigo, Sérgio, que Deus, sem que tenhamos compreensão do porquê, quis ter ao seu lado antes que pudéssemos estar preparados. Foi ele quem me ensinou os caminhos da pesquisa científica, desde os primeiros anos da graduação. Apesar da severidade que lhe era peculiar, demonstrou compreensão quando optei por não mais continuar o trabalho que fazíamos juntos no primeiro mestrado. Ele se foi antes que tivéssemos a oportunidade de voltar a trabalhar juntos (vínhamos conversando sobre isso), e tenho certeza que, se estivesse aqui, vibraríamos com mais essa conquista, que é nossa, afinal.

Ainda da UEM, agradeço aos colegas de turma que fiz no primeiro e segundo mestrado, parte dos quais hoje levam uma vida de produção acadêmica vigorosa, sendo hoje mestres e doutores nas formais mais digna que carregam esses termos. À Inês, secretária do programa de mestrado da UEM nas duas vezes que o cursei e amiga pessoal, agradeço por ser sempre solícita a quaisquer das necessidades que tive e por torcer de forma incondicional por minhas vitórias. Rendo agradecimentos também ao primeiro grupo de pesquisa do qual participei e, em especial, à Josiane, à Raqueline e ao Roberto, que além de amigos, contribuíram com meu ingresso nos programas de mestrado durante todos esses anos, fornecendo cartas de recomendação ao meu respeito sempre que solicitados.

Prosseguindo com os agradecimentos ao pessoal da UEM, gostaria ainda de agradecer à Valéria, que mesmo conhecedora de minha trajetória acadêmica até então turbulenta, não hesitou em me dar uma chance de ingressar no PGEEC da Unioeste com o fornecimento de uma

carta de recomendação para esse propósito. Por fim, não poderia, de forma alguma, deixar de agradecer à Itana, minha orientadora no segundo mestrado, muito menos à banca de qualificação da época que, na ocasião, entendeu por duas vezes que eu não estava apto a avançar com meu trabalho. As críticas que recebi naquele momento serviram para meu engradecimento pessoal e profissional.

Agradeço ainda ao Tales, da UFLA, meu orientador da especialização, também pelo fornecimento de carta de recomendação concedida para meu ingresso no segundo mestrado: não foi aquele o programa que concluí, mas isso não faz de seu apoio menos importante para a obtenção do título que me foi agora concedido.

Profissionalmente, contei ainda com o auxílio dos meus superiores e colegas de trabalho em ambos os locais em que trabalhei ao longo desses 16 anos, tanto para a realização de procedimentos operacionais, quando na concessão de horários especiais que me permitissem acompanhar as aulas dos cursos, cada qual em seu momento. No BoaCompra, agradeço ao Nelson e ao Carlos (Frodo), meus superiores na época, bem como a toda a equipe que me apoiou e deu suporte naquele momento. Na UTFPR, agradeço à Lígia, então responsável pelos recursos humanos, pelo suporte operacional e coleguismo em tudo que precisei para a redução de carga horária a qual fiz jus quando me foi necessário. Ainda na UTFPR, Agradeço ao professor Carlos Alberto Mucelin, meu diretor, pelas palavras de força e incentivo que precisei ouvir nos piores momentos dessa longa caminhada que agora se encerra.

Agradeço também aos meus amigos, novos e antigos, por me apoiarem na forma de palavras de incentivo e, por vezes, apenas estando presentes. Agradeço ao meu amigo Michel, que sempre expressou suas certezas quanto ao desfecho bem sucedido de cada um dos ingressos que ele acompanhou, mesmo ciente dos reveses que os precediam. Ao pessoal do MUFEM, grupo de amigos que mantenho desde a época de graduação e, em especial ao Muriel, que em razão de sua história pessoal, sempre me foi uma inspiração para a perseverança que consegui manter ao longo dos anos.

Meus agradecimentos se estendem aos professores que tive durante os três mestrados, bem como aos colegas que fiz nesse último ingresso. Com cuidado especial, agradeço ainda ao Cristiano e ao Gustavo, pelas contribuições que puderam dar ao meu trabalho em razão das discussões que tivemos enquanto grupo de pesquisa. Aos demais colegas, seja durante as aulas, seja nos grupos de debate via mensagens instantâneas, muito me foi acrescentado enquanto conhecimento por meio de vocês. Agradeço ainda a Sílvia, que, por inúmeras vezes realizou o trajeto para aula comigo entre as cidades de Santa Helena e Foz do Iguaçu, viabilizando financeiramente a minha continuidade no curso.

Na construção do trabalho final em si, devo agradecimentos ao Thiago, meu coorientador, que nas maiores dificuldades técnicas que tive soube me direcionar para soluções aplicáveis aos problemas que emergiam. Sigo agradecendo à banca de defesa, formada pelos professores Davi e Romeu, que aceitaram o desafio de, na adversidade de uma banca remota, me avaliar e sugerir

melhorias para que o trabalho realizado alcançasse o nível de excelência que justificasse sua aprovação.

No PGEEC da Unioeste, agradeço à Fabiana, assistente do programa, pelo seu pronto auxílio a minhas dúvidas, e ao professor Edgar, coordenador do curso na época do meu ingresso, por ter acreditado que eu finalizaria o mestrado dessa vez mesmo ante contundentes evidências que o fizessem imaginar o contrário.

Em especial, agradeço também ao meu orientador, Renato, por ter confiado em mim desde o início. Sua serenidade como orientador e assertividade em ações foram capazes de me manter sereno, focado e objetivo por todo o tempo em que trabalhamos juntos. Trata-se de uma parceria que espero ter a oportunidade de repetir sempre que possível.

Por fim, quero agradecer à minha família - a que eu e minha esposa edificamos juntos, com muito amor. À minha esposa, Nathalia, pelos momentos em que ela, gentilmente, me permitiu ser omisso em relação a nossa família, trocando o convívio familiar pelas obrigações acadêmicas que me cabiam. Pelas vezes que me cobrou garra e afinco. Não haverá no mundo qualquer palavra capaz de expressar a gratidão que tenho por isso. Esse título, se fosse possível, deveria ser metade dela. Agradeço ainda à minha filha Clara que até o momento, por praticamente toda a sua vida, teve que lidar com um pai quase sem tempo livre, alternando entre trabalho e estudos. Parte do tempo livre que me restará agora será revertido para ela.

Encerro agora esses extensos agradecimentos pedindo sinceras desculpas aos que, por ventura, eu não lembrei. Atribuo esse eventual esquecimento ao longo tempo de vida que essa jornada me consumiu, os quais eu facilmente posso contar. Jamais saberei, no entanto, mensurar o que me foi acrescentado por esse mesmo tempo. Resta-me apenas uma sincera palavra final para servir de desfecho para o longo capítulo de minha vida que agora se encerra: obrigado!

"Work it harder, make it better
Do it faster, makes us stronger
More than ever, hour after, our
Work is never over"

Thomas Bangalter,
Edwin Birdsong,
Guy-Manuel de Homem-Christo
(Daft Punk)

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Pseudocódigos

Lista de Siglas e Abreviaturas

AC	Acurácia
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
EMHOSIR	*Ensemble Method with Hyperparameters Optimization and Self-tuned Intermediate Range*
ES	Especificidade
GCP	*Global Configuration Parameters*
IDS	*Intrusion Detection Systems*
IETF	*Internet Engineering Task Force*
IPS	*Intrusion Prevention Systems*
K-NN	*K-Nearest Neighbors*
MLP	*Multi-layer Perceptron*
PGEEC	Programa de Pós-Graduação em Engenharia Elétrica e Computação, UNIOESTE, Campus de Foz do Iguaçu
OSSSP	*Optimization Search Space Scope Parameters*
SE	Sensibilidade
UNIOESTE	Universidade Estadual do Oeste do Paraná
VANETs	Redes Ad-hoc Veículares

Capítulo 1

Introdução

O grande volume de informações trocadas digitalmente decorrente, em parte, da expansão dos sistemas de informação nos ambientes institucionais/corporativos e do crescimento no uso da Internet, têm seu aumento semelhante àquele de incidentes de segurança reportados nos últimos anos. Na Figura 1.1, apresenta-se um gráfico com informações provenientes do relatório de 2020 divulgado pelo CERT.br[†], combinado com uma projeção a respeito de dispositivos conectados a Internet (Ali, Hamouda & Uysal, 2015) que mostra o quão evidente é essa tendência.

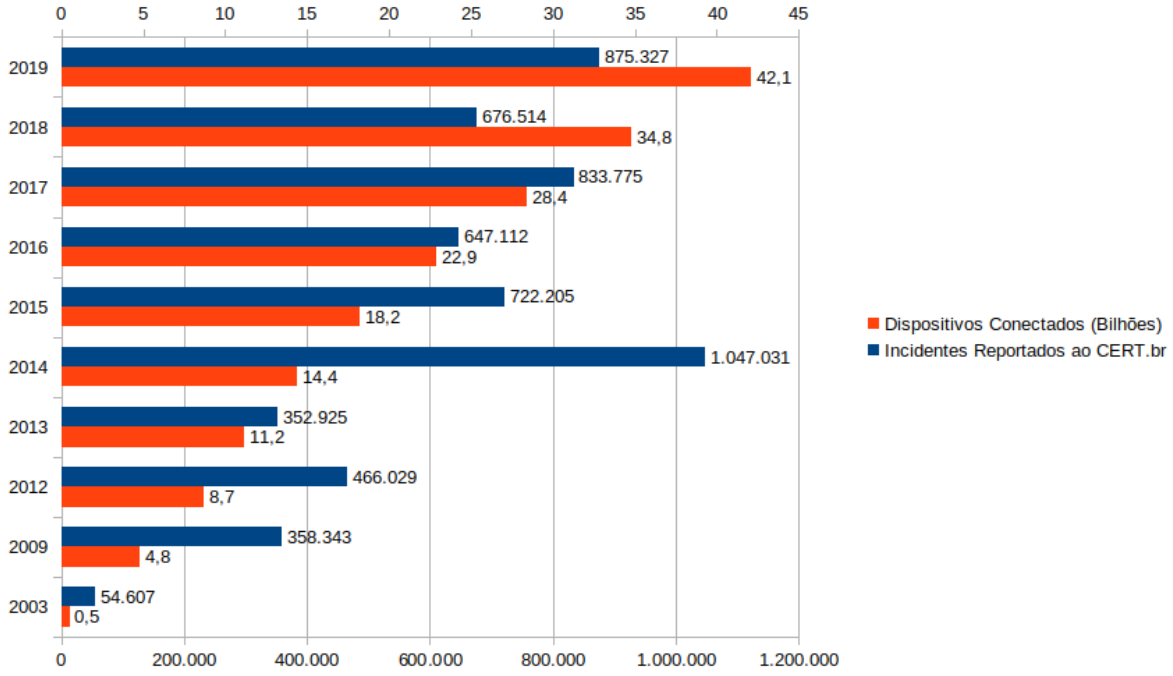

Figura 1.1: Relação Entre Incidentes Reportados ao CERT.br e Projeção de Dispositivos Conectados a Internet

Os referidos incidentes decorrem de ações maliciosas realizadas contra recursos computacionais de toda a sorte (servidores, telefones móveis, equipamentos de IoT, etc.) na forma das chamadas intrusões. A medida que se observa uma transformação digital em negócios convencionais com menor *expertise* na atuação *online*, a quantidade de alvos potenciais disponíveis a

[†]https://www.cert.br/

ação de cibercriminosos na forma de intrusões por meio de redes de computadores aumenta de modo a tornar ainda mais tentadoras suas investidas ilegais.

Em especial, no ano de 2020, em decorrência da pandemia de COVID-19, observa-se uma aceleração sem precedentes desde o início da era digital no que diz respeito ao número de pessoas passando a atuar profissionalmente de forma remota por intermédio de recursos de TIC (Kniffin, Narayanan, Anseel, Antonakis, Ashford, Bakker, Bamberger, Bapuji, Bhave, Choi et al., 2021). Atrelada a essa situação, observa-se que o aumento na ansiedade da população em razão da pandemia tornou mais provável o sucesso de ataques digitais que passam a incorporar a própria situação sanitária global como isca para seus ataques (Lallie, Shepherd, Nurse, Erola, Epiphaniou, Maple & Bellekens, 2020).

Já há alguns anos, a indústria e a academia buscam mecanismos para responder a essa situação de crescente ameaça nos ambientes virtuais. As iniciativas nesse sentido se concretizam de diversas formas, em uma frequente combinação entre políticas, *appliances* e sistemas especificamente projetados para proteger os ambientes digitais das intrusões em redes de computadores. O trabalho proposto nesta dissertação se apresenta como mais uma dessas iniciativas de combate a ações digitais maliciosas que se utilizam das redes de computadores como canal para ocorrerem.

Especificamente, o presente trabalho propõe um método genérico chamado EMHOSIR (*Ensemble Method with Hyperparameters Optimization and Self-tuned Intermediate Range*), que emprega a combinação autônoma/autoajustada de soluções de aprendizagem de máquina, com a aplicação previa de otimização de hiperparâmetros, para a detecção de intrusões em redes de computadores.

A concepção do método e os experimentos realizados permitiram concluir que a solução proposta apresenta melhorias em termos de métricas de desempenho e custo computacional, quanto comparada ao método no qual a mesma se baseou (Souza, 2018). Também foi possível observar que a solução proposta apresenta eficácia compatível com outras soluções concorrentes integrantes do estado da arte no que diz respeito à detecção de intrusões em redes de computadores.

Na seção seguinte, a proposta de dissertação é apresentada de modo mais detalhado, de modo que seja possível compreender os elementos que a constituem.

1.1 Proposta de Dissertação

No presente trabalho foi desenvolvido um método genérico para detecção de intrusão em redes de computadores, caracterizado pela combinação de dois classificadores na forma de modelos de aprendizagem de máquina, incluindo ainda etapas de otimização de hiperparâmetros, chamado EMHOSIR. O método proposto envolve a criação de classificações a partir de com-

binações dinâmicas e autônomas (autoajuste) dos classificadores que o compõe. A proposta implementada se baseou no trabalho desenvolvido por Souza (2018).

A ideia da proposição feita foi a de criar um método genérico/abstrato que permitisse a escolha de quaisquer modelos de aprendizagem de máquina[†] para integrar a composição de classificadores do método. Da mesma maneira, as técnicas para determinar de que forma deverá ocorrer a combinação dos classificadores que integram a solução, bem como os mecanismos utilizados para realização de otimização de hiperparâmetros não são intrínsecos ao método proposto, podendo ser variados conforme o interesse daquele que se propõe a fazer a utilização do método.

Especificamente, para validar a proposição do método genérico apresentado neste trabalho, implementou-se ainda uma instância do mesmo com a composição entre uma rede MLP e um modelo gerado pelo método K-NN; a rede MLP, após ser alvo de um processo de otimização de hiperparâmetros, tem sua classificação realizada em conjunto com o método K-NN, com o critério de combinação de classificadores semelhante ao utilizado por Souza (2018), mas realizado de forma dinâmica.

Por meio da implementação do método genérico proposto, criou-se a expectativa de obtenção de melhorias em termos de métricas de desempenho quando em comparação com o trabalho de Souza (2018), no qual o mesmo é baseado. Também se ansiou que os valores das métricas de desempenho obtidas fossem competitivas, inclusive, com valores obtidos por outros trabalhos que compõe o estado da arte da área de detecção de intrusão em redes de computadores por meio de aprendizagem de máquina. A acurácia foi a métrica principal na qual se esperou obter melhorias, tendo sido sobre ela os esforços concentrados no sentido de se obter avanços e diferenciais. Em comparação com o trabalho de Souza (2018), tinha-se ainda a expectativa de que as melhorias obtidas em termos de métricas de desempenho estivessem acompanhadas de uma redução no custo computacional para a classificação das instâncias representando tráfico de rede.

1.1.1 Objetivos

Em termos gerais, o objetivo do presente trabalho consistiu em, por meio da proposição de um método genérico, melhorar a acurácia e, eventualmente, demais métricas de desempenho na detecção de intrusão em redes de computadores, tendo como referência os valores dessas métricas obtidos pelo modelo de detecção de intrusão proposto por Souza (2018). A expectativa de melhorias se justifica pela adoção da otimização de hiperparâmetros e pela forma mais criteriosa de combinar as classificações dos modelos que compõe o novo método, quando em comparação com a forma de combinação de classificadores presente em Souza (2018). O novo método também foi criado com o objetivo de alcançar resultados competitivos em relação a

[†]No Capítulo 4, fica mais claro que algumas características dos modelos os permitem tirar mais vantagem do método genérico proposto

outros métodos de detecção de intrusão em rede de computadores presentes no estado da arte da área que também se utilizam de aprendizagem de máquina.

Objetivos Específicos

Pode-se dizer que os objetivos específicos que se buscou atingir com a proposta de dissertação ora apresentada foram:

- Observar o efeito de diferentes algoritmos de otimização de hiperparâmetros nos resultados da detecção de intrusão, nos termos das métricas de desempenho obtidos pelo método proposto;
- Minimizar o custo computacional do método proposto em comparação a composição realizada em Souza (2018), fazendo com que o escrutíneo do primeiro classificador seja quase sempre suficiente para se obter boas métricas de desempenho;
- Comparar, em termos de significância estatística, as métricas de desempenho obtidas pelo método proposto com as obtidas na proposta original de Souza (2018), superando-as.
- Desenvolver um método que obtenha métricas de desempenho com vantagens em relação às obtidas por técnicas levantadas no estado da arte dentro do domínio do problema.

1.1.2 Hipótese

Modelos de classificadores híbridos, direcionados a detecção de intrusão em redes de computadores, podem ser melhorados em termos de acurácia, custo computacional e possivelmente demais métricas de desempenho, por meio de otimização de hiperparâmetros e combinação criteriosa e inteligentemente ajustada dos participantes da composição, valendo-se das características que a eles são intrínsecas.

1.2 Estrutura do Trabalho

Além do capítulo que ora se finda, no qual uma breve contextualização e apresentação acerca do trabalho desenvolvido é realizada, no Capítulo 2 da presente dissertação, um arcabouço conceitual na forma de referencial teórico é trazido para permitir uma compreensão acerca do presente trabalho.

No Capítulo 3 são apresentados trabalhos no âmbito do estado da arte, os quais se relacionam com a presente dissertação, permitindo ao leitor ter a clara compreensão do modo com a qual este trabalho se relaciona com os demais pré-existentes, bem como compreender suas contribuições à área de estudo em que o mesmo se concentra.

Já no Capítulo 4, de materiais e métodos, é apresentada de forma minuciosa a maneira com a qual se implementou o presente trabalho, com detalhamento em termos de escolha e aplicação de tecnologias e técnicas como as apresentadas no Capítulo 2, bem como explanações a respeito de decisões de projeto realizadas.

O Capítulo 5 avança expondo e discutindo os resultados alcançados com o o método desenvolvido neste trabalho, tratando analiticamente as informações extraídas dos experimentos realizados durante o seu desenvolvimento.

Por fim, no Capítulo 6, as conclusões acerca dos resultados alcançados são contrapostos à proposta apresentada na Seção 1.1 em termos dos objetivos que se tinha.

O trabalho traz ainda alguns apêndices (A e B), nos quais o leitor pode avaliar parte dos dados que permitiram chegar às conclusões apresentadas no Capítulo 6, bem como acompanhar outras discussões gerais realizadas ao longo de todo o texto.

Capítulo 2

Referencial Teórico

2.1 Considerações Iniciais

O presente trabalho se encontra dentro do domínio de conhecimento específico referente à detecção de intrusão em redes de computadores. Sendo assim, no capítulo de referencial teórico que ora se inicia, na Seção 2.2, apresenta-se uma brevemente tema dentro do qual o trabalho em questão se insere. Complementando a discussão referente ao tópico, o capítulo segue com a apresentação da base de dados de *benchmark* NSL-KDD. Tendo em vista o fato da solução proposta no escopo deste trabalho fazer uso de conhecimentos clássicos da literatura científica correlata, faz-se necessário familiarizar-se com diversos conceitos nela presentes para uma plena compreensão da abordagem implementada. Há, portanto, neste capítulo, uma seção voltada a cada um dos conhecimentos que se fazem necessários a tal compreensão.

Na Seção 2.4, a aprendizagem de máquina é caracterizada junto aos problemas de classificação, os quais, entre outros, a mesma se propõe resolver. Em sequência, nas seções 2.5 e 2.6, as abordagens de aprendizagem de máquina conhecidas como Redes Neurais Perceptron de Múltiplas Camadas e método *K-Nearest Neighbours*, respectivamente, são discutidas. Na condição de grande referencial utilizado neste trabalho, a Seção 2.7 apresenta o método de combinação de classificadores de Souza (2018). As seções 2.8 e 2.9 dão seguimento ao capítulo caracterizando, respectivamente, as técnicas de otimização de hiperparâmetros e o método de Evolução Diferencial; os referidos elementos, ao serem incorporados neste trabalho, além de peculiarizá-lo, se tornaram diferenciais do mesmo em relação à abordagem proposta por Souza (2018). Por fim, na Seção 2.10, o presente capítulo é encerrado com uma breve discussão a respeito dos tópicos nele abordados.

2.2 Detecção de Intrusão em Redes de Computadores

Conforme visto no Capítulo 1, o presente trabalho se propõe a apresentar um método para detecção de intrusão em redes de computadores, tendo como referência a abordagem proposta

por Souza (2018). No entanto, para que se possa compreender o método proposto, é necessário definir a tarefa a qual o mesmo se propõe realizar. Tal caracterização se inicia com a apresentação da definição correspondente ao termo intrusão dentro do escopo deste trabalho.

No contexto de redes de computadores, dá-se o nome de intrusão aos eventos, às ações ou ao conjunto de ações com o intuito de comprometer a integridade, a disponibilidade ou a confidencialidade de um recurso computacional (Crosbie & Spafford, 1995; Heady, Luger, Maccabe & Servilla, 1990). A integridade diz respeito a condição de proteção da informação contra degradação ou manipulação não autorizada; a disponibilidade é a qualidade que garante o acesso à informação por parte de usuários autorizados quando necessário e de forma confiável; a confidencialidade está relacionada ao fato da informação ter sua disponibilização limitada às contra-partes a que se destina (Samonas & Coss, 2014).

Há um número expressivo de iniciativas, conduzidas tanto pela academia quanto pela indústria, cujos esforços estão concentrados no combate a ações maliciosas direcionadas a sistemas computacionais na forma de intrusões. Dentre essas iniciativas, são notórios os IDS (*Intrusion Detection Systems*), sistemas baseados em *software* ou *hardware* que automatizam o processo de monitoramento de eventos que ocorrem em sistemas computacionais ou redes, buscando sinais de intrusão (Bace & Mell, 2001). Além de haver alternativas de IDS disponibilizadas ao mercado tanto por meio de soluções de código aberto/livre (Snort*, Suricata†) quanto comerciais, na forma de *appliances* de inúmeors fabricantes, a academia tem contribuído já há alguns anos com proposições dentro do tema, decorrentes dos resultados de pesquisas (Ali Alheeti & McDonald-Maier, 2016; Aljawarneh, Aldwairi & Yassein, 2018; Cepheli, Büyükçorak & Karabulut Kurt, 2016; Elkhadir & Mohammed, 2019; Hamed, Dara & Kremer, 2018; Kanimozhi & Jacob, 2019; Kasongo & Sun, 2020; Liu, Zhang, Tang, Xie, Ma, Zhang, Zhang & Niyoyita, 2020; Lu, Sun, Liu & Li, 2018; Mohan, 2017; Ramkumar & Murugeswari, 2015; Reazul, Rahman & Samad, 2017; Selvakumar & Muneeswaran, 2019; Shenfield, Day & Ayesh, 2018; Vijayanand, Devaraj & Kannapiran, 2018; Zhang & Zhu, 2018; Souza, 2018).

De acordo com Wu & Banzhaf (2010b), um IDS tem seu funcionamento geral ilustrado conforme a Figura 2.1. Segundo apresentado, entre as diversas etapas de funcionamento de um IDS, o detecção de intrusão é a mais crucial, por se tratar da etapa em que os dados auditados são comparados ante a modelos de detecção de intrusão que descrevem tanto padrões de comportamento intrusivo quanto benigno. É justamente nessa etapa que se concentra a proposta de trabalho por este texto apresentada.

Na revisão da literatura realizada por Liao, Richard Lin, Lin & Tung (2012), é apresentada uma taxonomia dos IDS que compila várias outras iniciativas realizadas no mesmo sentido (Debar, Dacier & Wespi, 1999; Axelsson, 2000; Estevez-Tapiador, Garcia-Teodoro & Diaz-Verdejo, 2004; Amer & Hamilton, 2010; Bace & Mell, 2001; Sabahi & Movaghar, 2008; Kumar, Srivastava & Lazarevic, 2006; Xenakis, Panos & Stavrakakis, 2011). Liao et al. (2012).

*https://www.snort.org/

†https://suricata-ids.org/

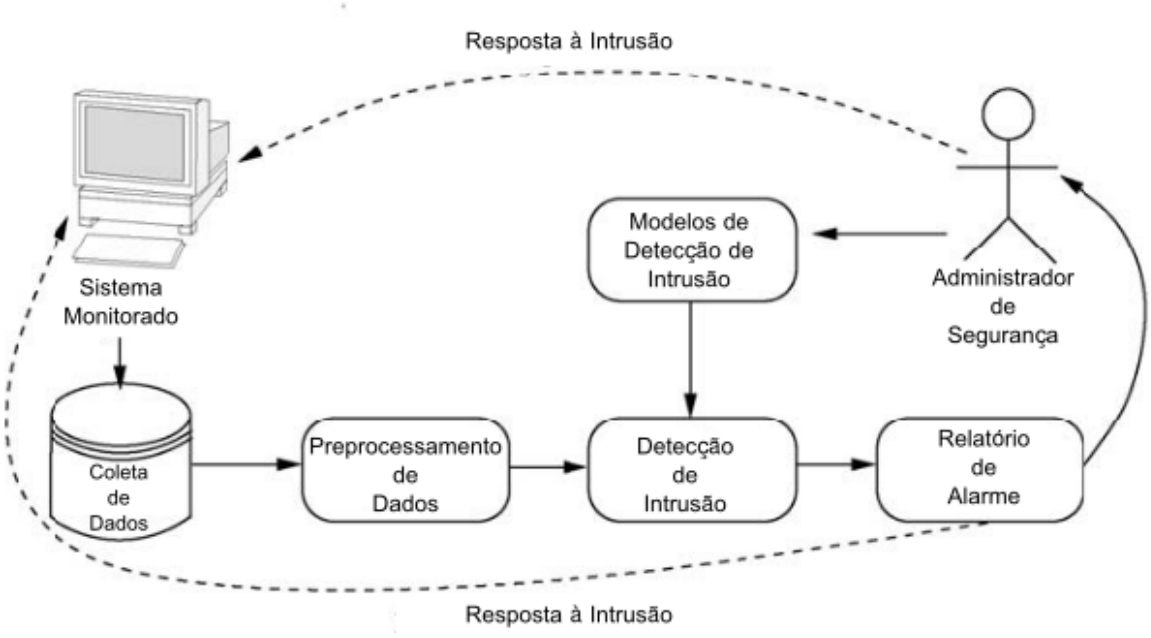

Figura 2.1: Organização Generalizada de um IDS
Fonte: Adaptado de Wu & Banzhaf (2010b)

Tais referências fazem um trabalho que permite uma visão geral de todos os aspectos a serem considerados na caracterização de uma solução de IDS. Uma adaptação da taxonomia por eles apresentada pode ser vista na Figura 2.2.

Conforme se vê na figura, uma solução de IDS pode ser classificada sob diversos aspectos, cada um dos quais com sua própria hierarquia. Em um nível mais amplo, a classificação dos IDS se dá em termos de *entrega do sistema*, *fonte de dados*, *ocasião* e *estratégia de detecção*. O presente texto segue com um detalhamento dessa classificação, considerando até o terceiro nível da taxonomia apresentada.

Em seu trabalho, Liao et al. (2012) prosseguem explanando a classificação apresentada. Com relação à *entrega do sistema*, a mesma diz respeito a maneira como o IDS é colocado em operação para atuar realizando as atividades de sua competência sobre aquilo que será colocado sob sua égide. Em termos de *arquitetura de rede*, a *entrega do sistema* de IDS pode ser *centralizada*, que é quando a coleta e análise de informação para detecção se dá em um único sistema, *distribuída* que é quando os dados da coleta provém de múltiplos sistemas monitorados de modo que ataques com reflexos sobre diversos pontos da rede podem ser detectados ou *híbrida*, que se trata de uma combinação dos dois tipos anteriores mencionados; quanto ao *tipo de rede* que interconecta o IDS ao que se quer monitorar, a conexão pode ser *cabeada*, *sem-fio* e *mista*; quanto ao *tipo de tecnologia* utilizada pelo IDS, pode-se ter as *baseadas em host*, em que há coleta e monitoramento de *hosts* com informações sensíveis e servidores que rodam serviços públicos, as *baseadas em rede cabeada* e *baseadas em rede sem-fio* em que ocorre a captura de tráfico em segmentos específicos da rede por meio de sensores, cada qual para seu tipo de rede, com subsequente análise de aplicações e protocolos, as de *análise de comportamento de rede*

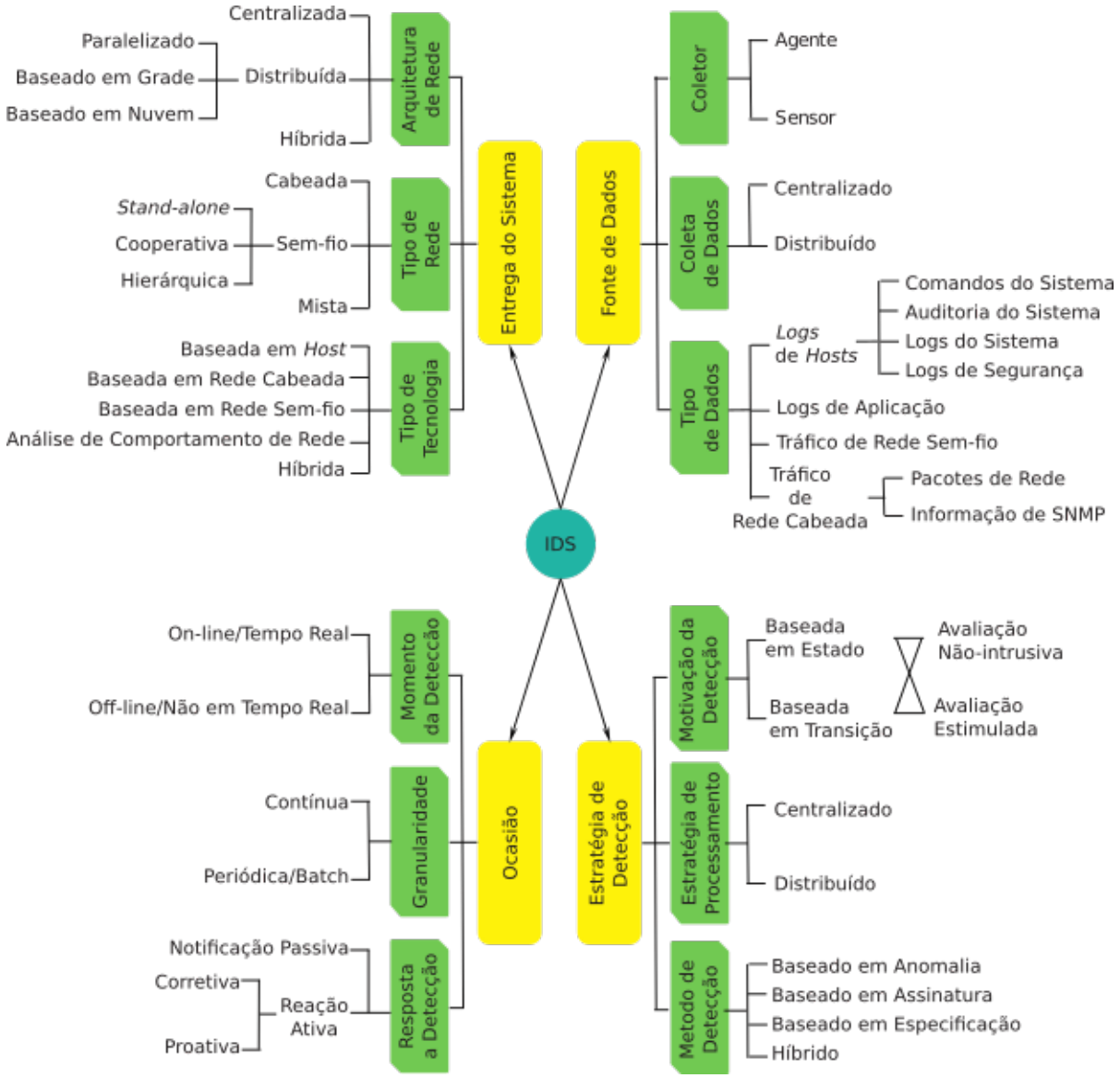

Figura 2.2: Taxonomia de um IDS
Fonte: Adaptado de Liao et al. (2012)

em que a inspeção do tráfego de rede reconhece ataques a partir de tráficos de fluxo inesperado e os tipos *híbrido*, resultantes da composição de quaisquer dos tipos anteriores descritos.

Com relação à *fonte de dados*, Liao et al. (2012) discriminam os IDS em função das diversas características que as mesmas podem ter. Seu *coletor* pode ser tanto do tipo *agente* quanto *sensor*; a *coleta de dados* em si (vide Figura 2.1), pode ser realizada de forma *centralizada* ou mesmo *distribuída*; por fim, com relação aos *tipo de dados* coletados, esses podem ser diversos: na forma de *logs de hosts*, *logs de aplicações*, *tráfico de rede sem-fio* e *tráfico de rede cabeada*.

A *ocasião* de atuação do IDS também é uma dos critérios por meio da qual esses sistemas são caracterizados na taxonomia de Liao et al. (2012). Por essa perspectiva, um IDS, em termos de *momento de detecção*, pode fazê-lo tanto *on-line/tempo real*, conforme os eventos monitorados forem ocorrendo, quanto *off-line/não em tempo real*, posterior a ocorrência que está sendo analisada; a *granularidade* pode ser tanto *contínua*, em que um fluxo de dados constante é

analisado, quanto *periódica/batch*, em que seções de dados são especificamente separadas para análise; em termos de *resposta a detecção*, os IDS podem adotar posturas de *notificação passiva* e *reações ativas*, aproximando-os do comportamento de sistemas chamados de IPS (*Intrusion Prevention Systems*).

Com relação à *estratégia de detecção*, Liao et al. (2012) apresenta as *motivações da detecção*, que podem ser *baseada em estado* (seguro ou inseguro) ou *baseada em transição* (na passagem de seguro para inseguro ou vice-versa). Ainda nesse contexto, a *estratégia de processamento* das detecções também podem ocorrer de forma *centralizada* ou *distribuída* (Zhang & Zhu, 2018). Por último, Liao et al. (2012) apresentam ainda os *métodos de detecção*, que podem ser tanto *baseado em anomalia*, *baseado em assinatura*, *baseado em especificação* ou as abordagens *híbridas* que combinam quaisquer das anteriores.

Os *métodos de detecção*, inclusive, foram deixados como derradeiros não por acaso nesta seção, mas como forma de destaque, por serem de grande relevância para o entendimento da abordagem proposta neste trabalho. Em IDS *baseado em anomalia*, modelos são construídos com base em dados considerados normais e os desvios relacionados a esses modelos são detectados sobre os dados observados (Lazarevic, Ozgur, Ertoz, Srivastava & Kumar, 2003). Já nos IDS cujo *métodos de detecção* são *baseados em assinatura* (também conhecidos como *métodos de detecção baseados em mau-uso*), as intrusões são identificadas observando os dados em relação a descrições previamente conhecidas de comportamentos considerados intrusivos (Wu & Banzhaf, 2010a). Os *métodos de detecção* dos IDS *baseados em especificação* se utilizam dos padrões originais dos protocolos de rede das organizações internacionais (ex. IETF) para identificação de intrusões (Liao et al., 2012). Apesar de, dentro do contexto de IDS, o trabalho ora realizado se concentrar exclusivamente na etapa de detecção de intrusão, conforme será visto mais adiante, tanto a abordagem de combinação de classificadores de Souza (2018), quanto a abordagem desta dissertação são do tipo *híbridas* em termos de *métodos de detecção*, uma vez que são capazes de detectar intrusões tanto baseando-se em anomalias quanto em assinatura/mau-uso.

Esta seção é finalizada a apresentação dos diversos aspectos que caracterizam os IDS. A seção que dá sequência a este texto complementa as noções até o momento adquiridas sobre detecção de intrusão em redes de computadores, apresentando em maiores detalhes a base de dados de *benchmark* NSL-KDD.

2.3 Base de Dados de *Benchmark* NSL-KDD

A relevância da base de dados de *benchmark* NSL-KDD pode ser melhor contextualizada com uma apresentação da história que precede sua concepção. Sendo assim, a presente seção se inicia com essa caracterização, de modo que seja possível compreender o quão oportuno fora seu surgimento e de que forma essa base de dados se relaciona com outras bases de dados

existentes.

Dentro da história de detecção de intrusão em redes de computadores, um relevante ponto de inflexão se deu em razão da 3ª Competição Internacional de Ferramentas para Descoberta e Mineração de Dados, referenciada comumente como *KDD Cup 99* (Stolfo, Fan, Lee et al., 1999). A relevância desse evento se deve ao fato dele ter sido responsável pela popularização de uma base de dados de eventos de intrusão, possível de ser utilizada como conjunto de dados de treinamento para criação de modelos de detecção de intrusão baseados em aprendizagem de máquina (maiores detalhes da técnica na Seção 2.4). Criada a partir de resultados obtidos pelo *MIT Lincoln Labs*, em 1998 no escopo do *DARPA Intrusion Detection Evaluation Program* (Stolfo, Fan, Lee, Prodromidis & Chan, 2000), a base de dados de eventos computacionais recebeu o nome de KDD-99. Essa base apresenta uma variedade de intrusões simuladas no ambiente da rede militar de onde fora extraída, assim como eventos não intrusivos.

Desde sua popularização, a KDD-99 serviu como base de dados de eventos computacionais para comparação (*benchmark*) entre diferentes pesquisas que propõem a detecção de intrusão por meio de aprendizagem de máquina, sendo, em razão disso, a mais conhecida da área (Wu & Banzhaf, 2010a). A despeito de críticas precoces e assertivas a respeito de problemas com a KDD-99 (McHugh, 2000), sua frequente ocorrência nas pesquisas sobre detecção de intrusão se deve ao fato de, por muito tempo, ter sido uma das poucas bases publicamente disponíveis com informações de sua natureza (Tavallaee, 2009).

Para lidar com parte dos problemas apresentados pela KDD-99, especificamente no que diz respeito ao alto índice de duplicação de registros (da ordem de 78% no conjunto de treinamento e 75% no conjunto de testes), uma nova base chamada NSL-KDD consistindo apenas de registros selecionados da KDD-99 fora proposta por Tavallaee (2009). Embora hoje existam outras bases de dados de eventos de segurança públicas (ISCXIDS2012[‡], CICIDS2017[§], CICIDS2018[¶], etc), sobre as quais várias pesquisas vem sendo desenvolvidas, por se tratar de uma versão aprimorada da clássica KDD-99, a NSL-KDD vem gradualmente tomando o lugar de sua predecessora como base para comparação. Face ao exposto, a NSL-KDD tem sido submetida a variados tipos de análise por diversos autores, de modo que, diferentes técnicas e ferramentas já foram utilizadas sobre a mesma (Selvakumar & Muneeswaran, 2019; Elkhadir & Mohammed, 2019; Mohan, 2017; Zhang & Zhu, 2018; Liu et al., 2020; Aljawarneh et al., 2018; Souza, 2018), com o intuito de se desenvolver modelos de detecção de intrusão funcionalmente efetivos (Dhanabal & Shantharajah, n.d.).

A base de dados NSL-KDD é formada por 148.517 exemplos, cada um representando uma conexão de rede. No entanto, a referida base de dados fora disponibilizada à comunidade científica de forma peculiar, em duas partes, nos chamados conjunto de treinamento (125.973 exemplos) e conjunto de testes (22.544 exemplos). A razão para essa forma de distribuição da

[‡]https://www.unb.ca/cic/datasets/ids.html
[§]https://www.unb.ca/cic/datasets/ids-2017.html
[¶]https://www.unb.ca/cic/datasets/ids-2018.html

base de dados está diretamente relacionada com a aplicação de soluções voltadas ao problema de detecção de intrusão em redes de computadores que se utilizam de aprendizagem de máquina, algo que fica mais claro com a leitura da Seção 2.4. A base de dados NSL-KDD é constituída de 42 atributos, sendo 32 proporcionais (discretos e contínuos) e 10 nominais discretos (incluindo a classe). Os valores referentes à classe correspondem aos diferentes tipos de ataque presentes (39 tipos distintos, pertencentes a quatro categorias) ou a conexões normais/benignas. Na Tabela A.1 disponível no Apêndice A, é possível ver o detalhamento referente a constituição da base de dados NSL-KDD em termos de seus atributos.

Tendo sido compreendidas a importância e estrutura da base de dados de *benchmark* NSL-KDD, é possível avançar na apresentação relacionada à aprendizagem de máquina e problemas de classificação, tratados de forma detalhada na seção seguinte.

2.4 Aprendizagem de Máquina e Problemas de Classificação

Uma das definições mais primordiais referentes à aprendizagem de máquina a classifica como sendo a área de estudo que confere aos computadores a habilidade de aprender sem que o mesmo seja explicitamente programado para tal (Samuel, 1959). Com a evolução dos estudos na área, chegou-se a definição de um problema bem-posto (Hadamard, 1902) de aprendizagem de máquina como o postulado a seguir:

> Diz-se que um computador aprende por uma experiência *E* em relação a uma tarefa *T* e alguma medida de desempenho *P* se seu desempenho em *T* medido por *P* melhora com a experiência *E*. (Mitchell et al., 1997)

É possível determinar a natureza do problema de aprendizagem de máquina com o qual se está trabalhando a partir do tipo de *feedback* disponível durante a aprendizagem; por meio da análise dessa propriedade, a área de aprendizagem de máquina é frequentemente dividida em três: aprendizagem supervisionada, aprendizagem não-supervisionada e aprendizagem por reforço (Russell & Norvig, 2002).

Em seu clássico livro, Russell & Norvig (2002) explicam que algoritmos para aprendizagem supervisionada determinística tem como entrada os valores corretos de uma função desconhecida para entradas particulares a partir dos quais tentam recuperar a referida função ou algo próximo a mesma. Conforme o próprio nome sugere, nos problemas de aprendizagem supervisionada, existe o papel ocupado por um supervisor que provê as saídas desejadas de acordo com as entradas para o problema sendo tratado (Ethem, 2016). Desse modo, conforme previamente enunciado, o *feedback* disponível para a aprendizagem é um conjunto de pares no padrão $(x, f(x))$, em que x é a entrada e $f(x)$ a saída da função aplicada a x (Russell & Norvig, 2002). Esse conjunto de pares é frequentemente chamado de conjunto de treinamento, do

mesmo modo que a aprendizagem que se faz com seu uso é chamada de treinamento. Em contrapartida, em razão da ausência de *feedback* em relação a quais são os resultados desejados a partir das entradas, nos problemas de aprendizagem não-supervisionada a tarefa realizada com maior frequência está relacionada à detecção de padrões nos dados, com a realização de agrupamentos (*clustering*) (Ghahramani, 2004) que levam em conta a similaridade e dissimilaridade desses dados entre si. Por fim, em aprendizagem por reforço, o agente computacional sendo treinado interage com o ambiente produzido um conjunto de ações que podem ser recompensadas ou punidas (Ghahramani, 2004).

Para prosseguir, faz-se necessário explicitar que, neste trabalho, o problema de detecção de intrusão em redes de computadores foi abordado como um problema de aprendizagem supervisionada; tal situação foi possibilitada pelo uso da base de dados de *benchmark* NSL-KDD. Conforme antecipado na seção anterior, a base de dados NSL-KDD fora distribuída nos chamados conjunto de treinamento e conjunto de testes; em bases de dados de *benchmark*, tal distribuição é útil para soluções que envolvem aprendizagem de máquina - especificamente as que realizam o tipo de validação conhecida como *holdout* (mais detalhes sobre isso adiante nesta seção). No caso específico da base de dados NSL-KDD, o conjunto de treinamento, constituído de conexões rotuladas como intrusivas ou normais/benignas - a base de dados em si - pode ser utilizado para a aprendizagem, sendo o *feedback* necessário e inerente a problemas de aprendizagem supervisionada; por sua vez, o conjunto de testes pode ser utilizado para se calcular o desempenho correspondente ao aprendizado realizado. A mensuração de desempenho, previstas inclusive na definição de Mitchell et al. (1997) apresentada, também é tratada em maiores detalhes mais adiante nesta mesma seção.

A aplicação de algoritmos de aprendizagem de máquina supervisionada pode ser feita a problemas de regressão ou problemas de classificação. Na aprendizagem supervisionada aplicada a esses dois tipos de problema, o que se busca é a criação de um modelo capaz de reproduzir, em termos de saídas geradas, o padrão existente nos dados de um determinado domínio. Isso é possível pela observação de eventos, instâncias e exemplos do referido domínio. No caso de regressão, as saídas são numéricas enquanto que no caso de classificação, tratam-se de saídas nominais discretas/categóricas, chamadas de classes. Um modelo que realiza classificação também é chamado de classificador.

As classificações realizadas podem ser do tipo binárias, em que existem apenas duas classes, ou de múltiplas classes, a depender do problema sendo solucionado e do algoritmo que se adota para a solução. Existem técnicas chamadas de binarização que permitem que problemas de múltiplas classes possam ser transformados em problemas binários; essas técnicas costumam ser aplicadas nas etapas de pré-processamento sobre as bases de dados de *benchmark* com as quais se está trabalhando. Conforme pode ser visto mais adiante neste texto e especificamente na Seção 4.3, para a solução proposta por este trabalho, ao problema de detecção de intrusão em redes de computadores utilizando a base de dados NSL-KDD, fora aplicada uma técnica de binarização. Em problemas de classificação binária, geralmente se adota uma das classes

como sendo a classe positiva, enquanto que a outra classe é identificada como classe negativa. Essa escolha arbitrária está diretamente relacionada ao problema sendo tratado; neste trabalho, tendo em vista a binarização realizada conforme apresentado na Seção 4.3 que gerou as classes "intrusiva"e "normal"para as conexões, a primeira foi considerada a classe positiva enquanto a última a classe negativa.

No que diz respeito a efetividade de modelos gerados pelos algoritmos de aprendizagem de máquina e a mensuração dessa efetividade, deve-se ter em mente que, na condição de ferramentas capazes de gerar modelos que descrevem de forma simplificada (por meio de uma função hipótese) os dados dos problemas aos quais foram aplicados, diversos modelos podem ser gerados a partir de um mesmo conjunto de dados de treinamento, cada qual relacionado ao algoritmo escolhido para a geração do modelo e à parametrização efetuada sobre o mesmo. Diante de um cenário de modelos concorrentes, se fazem necessários mecanismos que permitam aferir o quão bem cada um destes modelos consegue descrever os dados sobre os quais foram construídos, para que seja possível compará-los entre si ante a uma referência comum. É para isso que as métricas de desempenho são importantes.

Uma vez treinado, um classificador está apto a retornar uma classificação a partir de uma entrada do problema a ele fornecido. Em se tratando de aprendizagem supervisionada, as saídas providas pelos classificadores podem ser comparadas a conjuntos de testes para verificar o quão bem o modelo está realizando a classificação. Para melhor compreensão das métricas de desempenho existentes, é relevante conhecer alguns conceitos básicos, apresentados a seguir em termos de classificação binária. Considerando a aplicação de um modelo classificador sobre uma base de *benchmark*, define-se como:

- Verdadeiros Positivos (VP): Número de instâncias classificadas pelo modelo como positivas que de fato são positivas;
- Verdadeiros Negativos (VN): Número de instâncias classificadas pelo modelo como negativas que de fato são negativas;
- Falsos Positivos (FP): Número de instâncias classificadas pelo modelo como positivas mas que na verdade são negativas;
- Falsos Negativos (FN): Número de instâncias classificadas pelo modelo como negativas mas que na verdade são positivas;

Tendo em vista essas definições, pode-se compreender as métricas de desempenho obtidas a partir das mesmas. Essas métricas são especificadas nas equações (2.1), (2.2), (2.3) e (2.4):

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.1}$$

$$Sensibilidade\ (ou\ Recall) = \frac{VP}{VP + FN} \tag{2.2}$$

$$Especificidade = \frac{VN}{VN + FP} \tag{2.3}$$

$$F1\ Measure = \frac{VP}{VP + \frac{1}{2} \cdot (FP + FN)} \tag{2.4}$$

Conforme antecipado nesta seção, a obtenção dessas métricas em problemas de aprendizagem supervisionada costuma ser feita em uma etapa chamada de validação do modelo. Apesar da existência de outras técnicas (*leave-one-out*, *bootstrap*), são duas as formas mais comuns de validação de modelos: *holdout* e validação cruzada de k *folds*. Nas validações do tipo *holdout*, divide-se a base de dados de *benchmark* em dois conjuntos; pela dinâmica deste texto, neste ponto, a nomenclatura desses conjuntos já deve ser familiar: são os chamados conjunto de treinamento e conjunto de testes. Em uma proporção de geralmente $70\%/30\%$, treina-se o modelo sobre o conjunto de treinamento enquanto se realiza a mensuração das métricas de desempenho utilizando o conjunto de testes. É essa a razão da base de dados de *benchmark* NSL-KDD ser distribuída previamente dividida nesses dois conjuntos.

No caso de validação cruzada de k *folds*, define-se um valor para k (frequentemente 10). Esse valor corresponde a quantidade de conjuntos chamados *folds* nos quais a base de dados de *benchmark* será dividida. Realiza-se, então, o treinamento/aprendizagem sobre os primeiros $k-1$ *folds* e as métricas de desempenho são calculadas tendo como referência o $fold$ não utilizado para treinamento nessa etapa. O processo mencionado deve ser repetido k vezes, de modo que sempre um *fold* diferente seja poupado da fase de treinamento/aprendizagem e utilizado para o cálculo das métricas de desempenho. O valor final de cada métrica de desempenho que se quer calcular é obtido pela média aritmética simples dos valores individuais obtidos nas métricas de desempenho para cada *fold* individual.

As informações da seção que ora se encerra potencializam a compreensão das duas seções seguintes que lidam com métodos específicos voltados a aprendizagem de máquina supervisionada. O primeiro desses métodos, tratados na seção a seguir, é o método conhecido como Rede Neural Perceptron de Múltiplas Camadas.

2.5 Redes Neurais Perceptron de Múltiplas Camadas

Dentro da categoria de soluções computacionais bio-inspiradas, as redes neurais artificiais têm ganhado bastante momento em pesquisas e aplicações práticas nos últimos anos dada a sua versatilidade. Sua inspiração tem como principal elemento o neurônio biológico que vem a ser a unidade fundamental celular de um sistema nervoso (Tommy Wai-shing & David Siu-yeung, 2007). A Figura 2.3 apresenta um típico neurônio biológico; o detalhe de sua interligação com outros neurônios, nas chamadas junções sinápticas, pode ser visto na Figura 2.4.

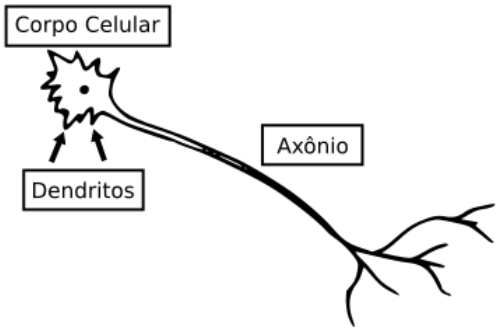

Figura 2.3: Neurônio Biológico

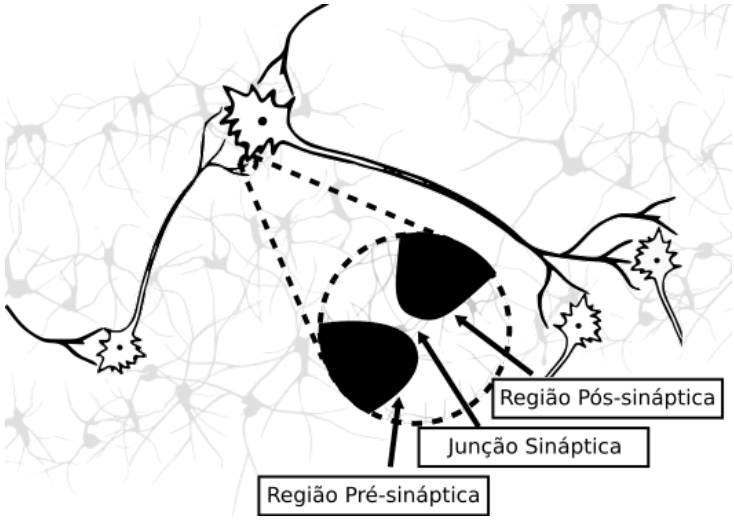

Figura 2.4: Neurônios Biológicos Interligados
Fonte: Adaptado de Tommy Wai-shing & David Siu-yeung (2007)

As junções sinápticas nas redes neurais biológicas ocorrem entre a extremidade dos axônios (região pré-sináptica) e os dendritos (região pós-sináptica); no estreito espaço da junção sináptica, a partir de estímulos de entrada, a atividade neural caminha de uma célula nervosa para outra na forma de gatilhos elétricos gerados por processos eletroquímicos.

As redes neurais, em geral, possuem como principal vantagem a capacidade de aprendizagem e generalização, tolerância a falha, além da rápida capacidade de processamento para gerar classificações uma vez que estejam treinadas (Cain, 2016).

As Redes Neurais Perceptron de Múltiplas Camadas ou redes MLP (*Multi-layer Perceptron*) são variações das Redes Neurais Perceptron (Rosenblatt, 1958) simples que, ao contrário dessas, são capazes de tratar problemas não-lineares. As redes MLP são constituídas de diversas unidades de perceptrons interligadas em forma de camadas. Da mesma forma que nas Redes Neurais Perceptron simples, a inspiração biológica nas MLP se dá em razão dos chamados perceptrons que são as unidades computacionais primordiais que formam ambos os tipos de redes. Essas unidades possuem relação direta com contrapartes biológicas nas quais se inspiram: o perceptron está para os neurônios, enquanto que as MLP estão para um sistema nervoso; da mesma forma, as interconexões entre as camadas das rede MLP, formadas por perceptrons

que se ligam diretamente, emprestam a nomenclatura biológica, sendo também chamadas de sinapses. Na Figura 2.5 é possível ver a representação de uma rede MLP. Por meio da mesma, pode-se observar a similaridade decorrente da analogia com um sistema nervoso biológico e seus elementos.

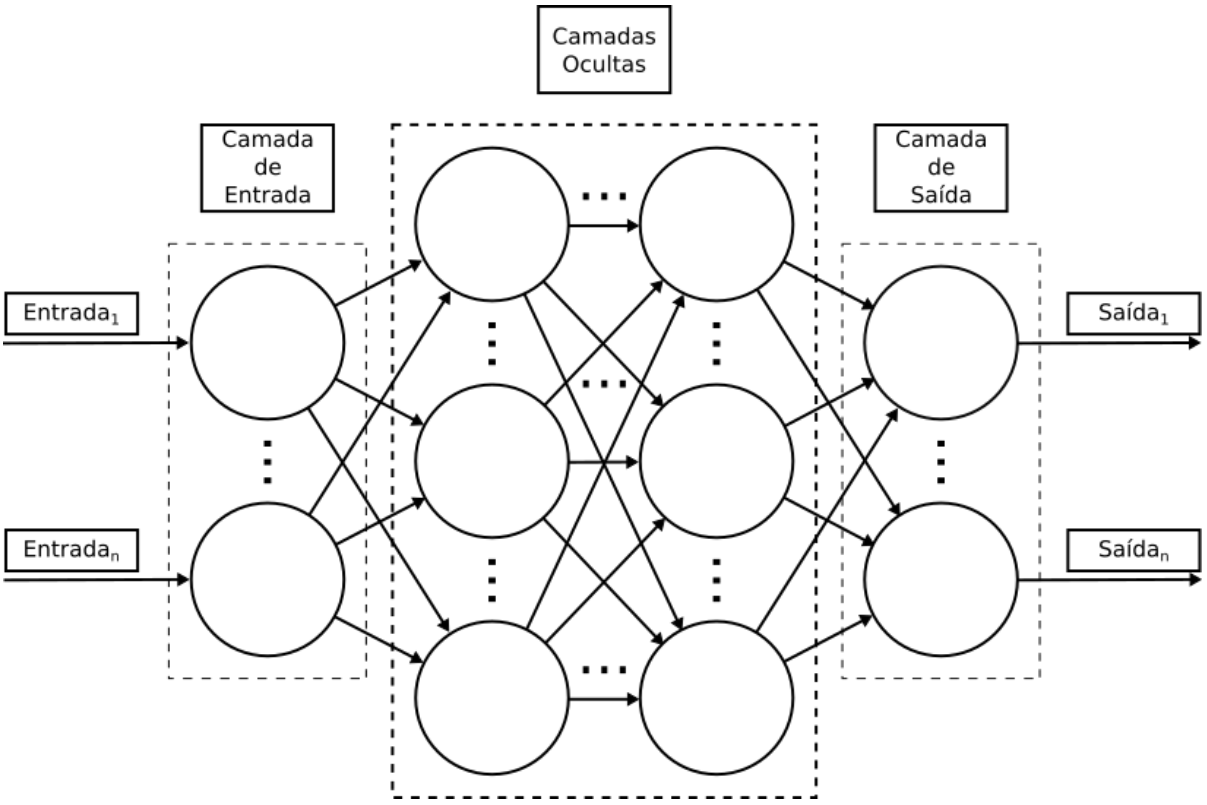

Figura 2.5: Representação de uma Rede MLP

Na figura em questão, os elementos circulares representam os perceptrons ligados por meio de camadas uns aos outros. Dada sua relação direta com os neurônicos biológicos, neste trabalho e em boa parte da literatura relacionada, usa-se de forma intercambiada o termo neurônio e perceptron para se referir ao mesmo elemento em redes do tipo MLP. A rede MLP, estimulada por sinais de entradas fornecidas via camada de entrada, liga-se camada a camada por meio de junções sinápticas artificiais representadas pelas setas; um conjunto de entradas fornecido a rede MLP gera saídas na chamada camada de saída. O procedimento interno da rede MLP, cujo funcionamento encontra-se melhor explicado adiante, envolve a somatória ponderada de pesos distintos (um peso por junção sináptica) cujo valor é utilizado como entrada de uma função matemática conhecida como função de ativação. O valor produzido pela função de ativação é utilizado como entrada do perceptron seguinte, conferindo a característica de rede que lhe é peculiar.

Conforme explanado em Haykin (2001), a propagação do sinal nas redes MLP ocorre camada a camada, no sentido da camada de entrada, passando pelas camadas ocultas, até a camada de saída, sendo, por esse motivo, classificada entre as redes neurais alimentadas para frente (*feedforward*); seu treinamento, cujo objetivo é justamente determinar os valores dos

pesos sinápticos, se dá pelo algoritmo de retropropagação de erros (*backpropagation*), que consiste de dois passos principais (Haykin, 2001):

1. O padrão apresentado à camada de entrada se propaga pela rede neural até chegar a camada de saída, sem realizar qualquer alteração dos pesos sinápticos;

2. De acordo com a regra de correção definida no algoritmo de retropropagação de erros, ocorre o ajuste de pesos sinápticos, considerando a diferença entre saída obtida e saída desejada (erro). O referido erro se propaga pelas camadas da rede em ordem oposta a da sua alimentação, partindo da camada de saída, passando pelas camadas ocultas até chegar a camada de entrada, realizando correções nos pesos sinápticos pelo caminho de acordo com a contribuição de cada peso ao erro final obtido; as correções nos pesos sinápticos ocorrem de modo a fazer com que novas entradas apresentadas a rede MLP produzam respostas com maior aproximação estatística em relação aos valores das respostas que se deseja obter.

A convergência do algoritmo de retropropagação se dá na ocorrência de um erro suficientemente pequeno dentro do critério sendo considerado, fato que caracteriza a finalização do treinamento da rede MLP e possibilidade de seu uso sem que seja necessário realizar a retropropagação (Lima, 2005).

Diversos são os parâmetros que caracterizam uma rede MLP, bem como o treinamento sobre ela realizado. Apenas para citar alguns, em termos de rede MLP, além dos elementos que determinam sua dimensão, tal como o número de neurônios em cada camada e o número de camadas ocultas, há ainda a já mencionada função de ativação, a taxa de aprendizagem o *dropout* e o decaimento dos pesos. A taxa de aprendizagem é um parâmetro que especifica o quão brusca ou sútil deve ser a atualização dos pesos sinápticos durante a aprendizagem, tornado-a estável e permitindo que ótimos locais sejam evitados; tanto o *dropout* quanto o decaimento de pesos são aplicados como técnicas chamadas de regularização, cujo objetivo é minimizar o hiperajuste de modelos (*overfitting*), situação em que após treinado, o modelo só obtém bons resultados nos dados sobre os quais treinou, não generalizando seu desempenho para novos conjuntos de dados (Dietterich, 1995). O *dropout* o faz especificando um percentual de neurônios a terem seus valores descartados aleatoriamente. O decaimento de pesos penaliza pesos de maior valor escalar, minimizando sua influência sobre o modelo final treinado.

Em relação ao treinamento, há também parâmetros que o caracterizam, como é o caso do número de épocas que determina quantas vezes o conjunto de treinamento deve ser completamente apresentado ao modelo para que o mesmo seja treinado.

A seção seguinte trata do método de *K-Nearest Neighbours*, cujo entendimento é primordial para compreensão acerca do trabalho por este texto apresentado.

2.6 Método *K-Nearest Neighbours*

Diferentemente do que ocorre nas redes MLP, em que o modelo corretamente escolhido e adequadamente parametrizado melhora incrementalmente sua capacidade de classificação em decorrência de um processo de treinamento, no método *K-Nearest Neighbours* (K-NN) as classificações são realizadas por similaridade entre a nova instância e as instâncias conhecidas do domínio do problema. Por esse motivo, o K-NN é classificado como sendo um método de aprendizagem de máquina baseado em instâncias (Russell & Norvig, 2010). Sua característica o diferencia dos algoritmos incrementais que são sensíveis a ordem de apresentação das instâncias e podem tomar decisões precocemente com base já nas primeiras informações obtidas do problema (Salzberg, 1991).

De forma simplificada, o algoritmo K-NN considera cada instância como sendo um ponto em um plano n-dimensional R^n em que n é o número de atributos da instância. Utilizando-se de uma função para cálculo de distância/similaridade, o K-NN determina o quão próximo uma instância sendo classificada está das demais instâncias cuja classificação já é conhecida (Cover & Hart, 1967). O k, que faz parte da nomenclatura do algoritmo, é o parâmetro que especifica quantas instâncias estão sendo consideradas para se determinar a classificação de novas instâncias, sendo preferencialmente um número ímpar[‡]: a classificação da nova instância será a mesma que a classificação majoritária entre as k instâncias conhecidas mais próximas/similares de acordo com a métrica utilizada no cálculo de similaridade. A distância Euclidiana (Mitchell et al., 1997) é uma das medidas de similaridade mais utilizadas com o K-NN.

A despeito de sua simplicidade, o K-NN com $k = 1$ [†] costuma obter bons resultados em termos de classificação quando o volume de instâncias conhecidas é grande. Isso ocorre porque para um conjunto suficientemente grande de instâncias conhecidas, sua taxa de erro é menor que o dobro da taxa base de erro de Bayes, que é a menor taxa de erro que se pode atingir utilizando o melhor classificador possível (Elkan, 2011).

Contudo, do ponto de vista prático, o método K-NN se mostra extremamente sensível a dados de alta dimensionalidade em termos de complexidade de tempo, além de ter seu comportamento fortemente comprometido por incorretudes e imperfeições nos dados rotulados (Dasarathy, 1991; Devijver & Kittler, 1982).

Indo adiante com o embasamento necessário a compreensão da solução implementada e apresentada por este trabalho, a seção seguinte trata do método de combinação de classificadores de Souza (2018).

[‡]Números pares podem ensejar empates na hora de se identificar a classificação majoritária

[†]Também chamado de 1-NN ou *NN Nearest Neighbor* (NN)

2.7 O Método de Combinação de Classificadores de Souza (2018)

Para detecção de intrusão em redes de computadores, o método de combinação de classificadores proposto em Souza (2018) classifica eventos por meio de uma rede MLP, gerando saídas contínuas com valores variando entre 1 e -1 resultantes de uma função de ativação tangente hiperbólica. As saídas cujos valores estão próximos a 1 indicam a detecção de um padrão de comportamento normal; por conseguinte, a obtenção de valores próximos de -1 indicam a detecção de um padrão de comportamento intrusivo. Os valores intermediários são submetidos ao classificador K-NN. Neste trabalho, deu-se o nome de *pipeline* de fluxo condicional a essa forma de interação entre os dois classificadores.

Em seu trabalho, o critério para a definição de quais valores são considerados intermediários, compondo a chamada faixa intermediária, foram obtidos de forma experimental, com apresentação detalhada em seu trabalho (Souza, 2018). Tal faixa envolve valores de saída da rede MLP que englobam a maioria dos falsos positivos e falsos negativos gerados.

A arquitetura do método originalmente proposto por Souza (2018) pode ser observada na Figura 2.6. Observa-se que a base de dados NSL-KDD pré-processada, ao ser submetida à abordagem híbrida composta por rede MLP e pelo método K-NN, gera um resultado final correspondente à classificação dos exemplos consumidos. O pré-processamento ao qual à NSL-KDD fora submetida pode ser melhor compreendido na Seção 4.3.

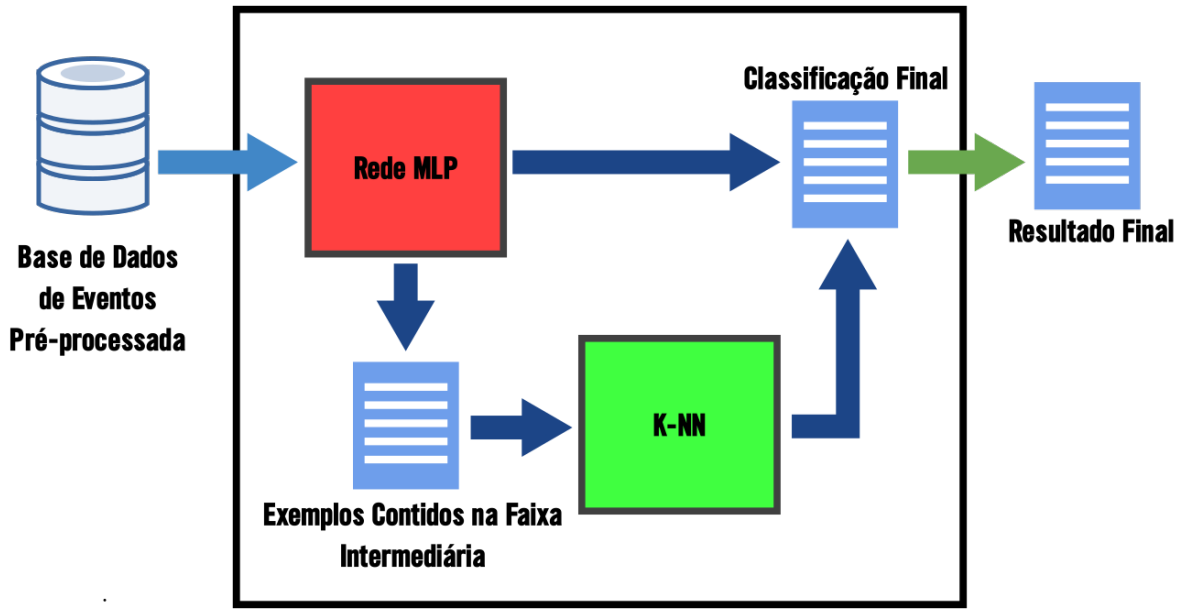

Figura 2.6: Arquitetura do Método Híbrido de Souza (2018)
Fonte: Adaptado de Souza (2018)

Para cada exemplo processado pela rede MLP, uma saída contínua entre -1 e 1 é gerada, correspondente ao valor retornado pela função de ativação tangente hiperbólica. Desse modo,

existem duas distribuições de frequência referentes aos valores de saída retornados pela rede MLP: uma relacionada aos valores negativos entre -1 e 0, e a outra aos valores positivos entre 0 e 1. Para cada uma delas, Souza (2018) identificou o percentil com a maior incidência de classificações errôneas. As composições de percentis avaliadas são demonstradas na Figura 2.7. Conforme pode ser visto, foram avaliadas oito configurações distintas de percentis, cada composição representada por uma das linhas verticais transpassadas por uma reta horizontal que indica o percentil 0. Da esquerda para a direita, é possível ver que os primeiros percentis testados foram o que corresponde ao 1^{o} valor na parte positiva da faixa intermediária, combinado com o 75^{o} percentil da parte negativa. Testes subsequentes se deram utilizando outras variações. Mantendo o sentido da esquerda para a direita, é o caso representado pela linha vertical subsequente, com o 1^{o} valor na parte positiva da faixa intermediária combinado com o 75^{o} percentil da parte negativa, bem como o das demais combinações de percentis apresentadas na figura. É válido observar que foram avaliadas mais variações de percentis para os valores negativos do que para os positivos, assim como maiores proporções dessa distribuição de frequência. A grande elusividade das conexões intrusivas, bem como a menor frequência de ocorrência na base de dados quando comparada a conexões normais, justifica tal escolha, dado que a rede MLP se mostrou mais propensa a erros quando da classificação de conexões do tipo intrusiva. A determinação dos percentis serviu para se estabelecer os limites da faixa intermediária. A faixa intermediária diz respeito aos valores de saída da rede MLP em termos de percentil que, tanto para a parte positiva, quanto para a parte negativa, determinam quais instâncias são enviadas para análise do K-NN.

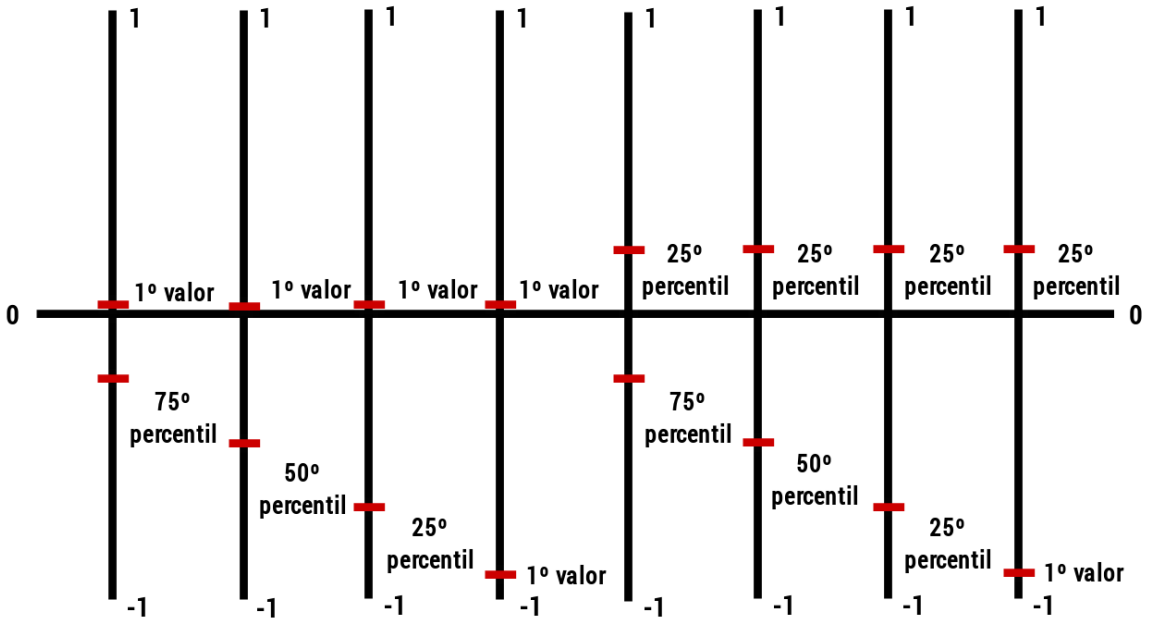

Figura 2.7: Percentis Testados no Método de Combinação de Classificadores de Souza (2018)
Fonte: Adaptado de Souza (2018)

Concluída a apresentação do método de combinação de classificadores de Souza (2018), a seção subsequente trata de otimização de hiperparâmetros, que figura como uma das principais evoluções aplicadas ao método de Souza (2018) na proposta apresentada neste trabalho.

2.8 Otimização de Hiperparâmetros

Resultados empíricos sugerem que a parametrização dos algoritmos de aprendizagem de máquina exerce impacto significativo nas métricas de desempenho obtidas pelos modelos correspondentes gerados. Tal percepção justifica o esforço de se utilizar técnicas capazes de identificar os parâmetros que otimizam as referidas métricas dos modelos que tais algoritmos geram. Ocorre, no entanto, que cada algoritmo possui um número variado (e possivelmente grande) de parâmetros que, em contraste com os atributos aprendidos durante a fase de treinamento do modelo, devem ser definidos a priori, ou seja, antes mesmo de qualquer dado lhes ser apresentado, sendo, por esse motivo, chamados de hiperparâmetros. Em redes MLP, por exemplo, o número de camadas ocultas, a quantidade de neurônios em cada camada e a função de ativação são exemplos de hiperparâmetros. Dá-se o nome de otimização de hiperparâmetros ao conjunto de técnicas que, sistematicamente, efetuam escolhas para os hiperparâmetros de um dado algoritmo de aprendizagem de máquina. Essas técnicas buscam obter modelos que alcancem valores ótimos para as métricas de desempenho consideradas, tendo em vista o problema em questão.

Conforme apresentado em Bergstra & Bengio (2012), o objetivo final de um típico algoritmo de aprendizagem de máquina $\mathcal{A}$ é encontrar a função f (função hipótese) que minimiza uma perda esperada $\mathcal{L}(x; f)$ em um conjunto de exemplos x pertencentes à distribuição natural G_x. Um algoritmo de aprendizagem de máquina $\mathcal{A}$ é uma função que mapeia uma base de dados $X^{(treinamento)}$, composta de um conjunto finito de exemplos de G_x, em uma função f. Segundo os autores, é comum que f seja obtida a partir da otimização de critérios considerados para o treinamento, resultando em um conjunto de parâmetros θ. Contudo, os autores prosseguem explanando que o próprio algoritmo de aprendizagem de máquina deve ser previamente configurado por meio de hiperparâmetros $\lambda \in \Lambda^{\dagger}$, de modo que o verdadeiro algoritmo de aprendizagem de máquina é aquele obtido após a escolha de λ sendo, por isso, denotado $\mathcal{A}_\lambda$ e $f = \mathcal{A}_\lambda(X^{(treinamento)})$ para o conjunto de treinamento $X^{(treinamento)}$.

Bergstra & Bengio (2012) dizem ainda que no contexto em questão, a otimização de hiperparâmetros diz respeito a escolha de λ que minimiza os erros de generalização representados pela expressão matemática $\mathbb{E}_{x \sim G_x}\left[\mathcal{L}\left(x; \mathcal{A}_\lambda(X^{(treinamento)})\right)\right]$. Para os autores, é importante perceber que as operações realizadas pelo algoritmos $\mathcal{A}$ envolvem uma otimização interna, geralmente feita de forma iterativa e aproximada (treinamento). Segundo apresentado pelos autores, o problema de identificar bons valores para os hiperparâmetros λ é chamado de otimização de hiperparâmetros e é formalmente definido conforme se vê a seguir:

[†] Λ é o conjunto de todas as possibilidades de valores de hiperparâmetros possíveis

$$\lambda^{(*)} = \underset{\lambda \in \Lambda}{\operatorname{argmin}} \, \mathbb{E}_{x \sim G_x} \left[\mathcal{L}\Big(x; \mathcal{A}_\lambda(X^{(treinamento)})\Big) \right] \tag{2.5}$$

As subseções seguintes tratam das técnicas de otimização de hiperparâmetros que foram utilizadas no presente trabalho, iniciando-se pela Busca em Grade.

2.8.1 Busca em Grade

No método de Busca em Grade (*Grid Search*), adota-se a abordagem mais ingênua que se pode ter para um caso de otimização: a partir de um conjunto de hiperparâmetros e seus valores potenciais, percorre-se todo o espaço de atribuições possíveis desses hiperparâmetros de forma exaustiva, escolhendo aqueles que obtém melhor desempenho (Hazan, Klivans & Yuan, 2017). Em termos matemáticos, considerando o conjunto de tentativas $T = \{\lambda^{(1)}...\lambda^{(S)}\}$ para se obter $\lambda^{(*)}$ conforme a equação (2.5), tendo S como o número de instâncias de hiperparâmetros tentados, temos que, na Busca em Grade, $T = \Lambda \therefore S = |\Lambda|$.

Conforme pode ser facilmente inferido, a medida que o número de hiperparâmetros e a quantidade de valores considerados para os mesmos cresce, o espaço de busca aumenta de forma exponencial. Sendo assim, a utilização da Busca em Grade se torna rapidamente impraticável, situação que, neste trabalho, motivou a redução de dimensionalidade do espaço de busca dos hiperparâmetros apresentada na Seção 4.4. Levando em conta tal limitação da Busca em Grade, recomenda-se utilizá-la em situações de menor dimensionalidade dos hiperparâmetros e/ou junto à paralelização computacional.

Ademais, sem que se modifique o funcionamento padrão da Busca em Grade, a solução não é capaz de lidar com hiperparâmetros contínuos uma vez que haveria um número infinito de combinações de atribuições a serem geradas para se detectar aquelas que geram melhores modelos.

Felizmente existem alternativas melhores em termos de desempenho do que a Busca em Grade com um bom potencial de encontrar hiperparâmetros que garantam aos modelos bons resultados em termos de métricas de desempenho. A Busca Aleatória, tratada na seção seguinte é uma dessas alternativas.

2.8.2 Busca Aleatória

A chamada Busca Aleatória (*Random Search*) possui as mesmas vantagens práticas da Busca em Grade sendo, da mesma forma, simples em termos conceituais, de fácil implementação e de paralelização trivial, com a vantagem de ser mais ágil que a última em situações nas quais se está diante de grandes dimensionalidades no espaço de busca de hiperparâmetros (Bergstra & Bengio, 2012).

De acordo com o que se apresenta na Seção 4.4, nem todo o espaço de busca de hiperparâmetros é igualmente promissor e a Busca Aleatória tira proveito dessa característica. Conforme sugere seu nome, na Busca Aleatória, os valores de hiperparâmetros são explorados de modo que o conjunto de tentativas $T \subseteq \Lambda$, conforme a equação (2.5), possua elementos aleatoriamente selecionados de Λ, buscando satisfazer a propriedade $S = |T| \ll |\Lambda|$ na obtenção de $\lambda^{(*)}$.

Os propositores da Busca Aleatória tiram vantagem e extrapolam um princípio chamado de dimensão de baixa efetividade (*low effective dimension*) que postula que se uma função f de duas variáveis pode ser aproximada por uma outra função g de uma única variável ($f(x_1, x_2) \approx g(x_1)$), pode-se dizer que f possui uma dimensão de baixa efetividade (Bergstra & Bengio, 2012). O princípio das dimensões de baixa efetividade é exatamente o que explica o fato de nem todo o espaço de busca de hiperparâmetros ser promissor em termos de identificação de $\lambda^{(*)}$.

Tendo em vista o princípio das dimensões de baixa efetividade, em seu trabalho, Bergstra & Bengio (2012), inclusive, defendem o potencial da Busca Aleatória encontrar melhores resultados que a Busca em Grade conforme pode ser visto na Figura 2.8. Na referida figura, são apresentadas, para Busca em Grade e Busca Aleatória, nove tentativas de otimização de uma função $f(x, y) = g(x) + h(y) \approx g(x)$ com dimensão de baixa efetividade. Sobre os quadrados, em verde, a função $g(x)$ pode ser vista, enquanto que a esquerda dos mesmos, em amarelo, a função $h(y)$. Segundo observam Bergstra & Bengio (2012), as nove tentativas de otimização da Busca em Grade exploram apenas três pontos distintos de g enquanto que na Busca Aleatória, todas as nove tentativas exploram pontos distintos de g. Os autores finalizam essa análise afirmando que essa situação é a regra e não a exceção em se tratando de Busca em Grade, principalmente em situações de alta dimensionalidade do espaço de busca de hiperparâmetros.

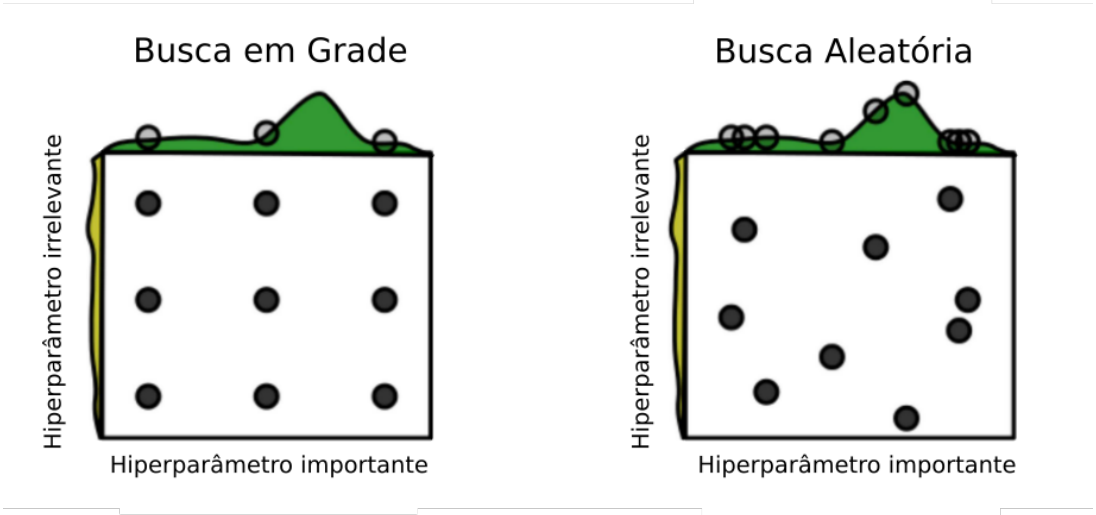

Figura 2.8: Exploração do Espaço de Busca: Busca em Grade *vs* Busca Aleatória
Fonte: Adaptado de Bergstra & Bengio (2012)

Compreendida a Busca Aleatória e seu potencial quando em comparação com a Busca

em Grade, na subseção seguinte a técnica de Otimização Bayesiana é apresentada.

2.8.3 Otimização Bayesiana

Fechando a apresentação das técnicas de otimização de hiperparâmetros adotadas neste trabalho, a presente subseção trata da Otimização Bayesiana em sua vertente que faz uso de Processos Gaussianos.

A Otimização Bayesiana é extremamente útil para problemas em que se quer realizar otimização de funções matemáticas f (encontrar o $\mathrm{argmax}\, f$) do tipo caixa-preta com alto custo computacional para cálculo e que retornam saídas ruidosas[†]. Pelo fato de serem funções caixa-preta, as soluções de otimização aplicadas as mesmas não podem contar com gradientes; por terem alto custo computacional para cálculo, deve-se minimizar ao máximo as avaliações de valor que se faz sobre elas. Apesar de não envolver funções caixa-preta em seu sentido mais estrito, as características marcantes de problemas de otimização de hiperparâmetros são as mesmas que aquelas para as quais a Otimização Bayesiana é adequada.

O leitor mais atento pode ter notado que, ao contrário do que se observa na equação (2.5), a Otimização Bayesiana diz respeito a problemas de maximização e não de minimização. Nesse sentido, para o uso de Otimização Bayesiana na otimização de hiperparâmetros, faz-se necessário considerar o problema de cálculo de argmax sobre o cálculo de erros de generalização com sinal invertido, conforme a equação (2.6).

$$\lambda^{(*)} = \underset{\lambda \in \Lambda}{\mathrm{argmax}} -\mathbb{E}_{x \sim G_x}\Big[\mathcal{L}\Big(x; \mathcal{A}_\lambda(X^{(treinamento)})\Big)\Big] \tag{2.6}$$

A Otimização Bayesiana recebe esse nome em razão do Teorema de Bayes, do qual se permite derivar que a probabilidade *a posteri* de uma hipótese H dada uma observação O é proporcional à chance da ocorrência O tendo em vista H, multiplicada pela probabilidade de H, conforme pode-se observar a seguir (Brochu, Cora & De Freitas, 2010):

$$P(H|O) \propto P(O|H) \cdot P(H)$$

O Teorema de Bayes é incorporado na Otimização Bayesiana pelo fato de se assumir algumas crenças *a priori* sobre propriedades da função que se quer otimizar (ex: continuidade), que vão sendo atualizadas conforme novas observações de relações entre entradas e saídas vão sendo realizadas.

O que ocorre, conforme se vê em Brochu et al. (2010), é que dado um x_i correspondendo à i-ésima entrada fornecida à função desconhecida f que se quer otimizar e $f(x_i) = y_i$

[†]Ruidosas no sentido de poderem apresentar variações mínimas de valor entre avaliações subsequentes

como a observação da saída correspondente retornada por f, as t observações acumuladas $y_{1:t} = \{y_1, ..., y_t\}$[‖] servem de subsídio para atualização das crenças sobre a função desconhecida f. Com as relações $\mathcal{R}_{1:t} = \{x_{1:t}, y_{1:t}\}$, a distribuição *a priori* é combinada com a função de probabilidade $P(\mathcal{R}_{1:t}|f)$, permitindo que se avalie o quão provável as observações obtidas são, dado o conhecimento que se achava ter inicialmente. Conforme seguem Brochu et al. (2010), o fato de se ter assumido certas propriedades sobre f *a priori* faz com que algumas saídas sejam consideradas menos prováveis do que outras. A situação permite uma atualização *a posteriori* da distribuição de f, dada por:

$$P(\mathcal{R}_{1:t}|f) \propto P(f|\mathcal{R}_{1:t}) \cdot P(f)$$

Brochu et al. (2010) finalizam suas considerações sobre esse processo bayesiano tecendo comentários a respeito do fato de tal atualização corresponder ao uso de uma função substituta que se ajusta dinamicamente a cada nova observação, sendo ela a função que de fato está sendo otimizada. A Figura 2.9, adaptada de Rasmussen & Williams (2005), apresenta a evolução da crença a respeito da função sobrejacente a medida que se realizam novas observações. No quadro esquerdo, são apresentadas quatro possíveis configurações da função f baseando-se na crença que se tinha *a priori* a respeito da mesma, antes de se realizar observações. No quadro da direita, após duas observações, o que se vê nas linhas pontilhadas é a atualização das quatro possíveis configurações que se tinha *a priori* para f, com a linha sólida representando a previsão média da curva de f baseada nessas quatro configurações. As regiões sombreadas são projeções referentes a duas vezes o desvio padrão de cada entrada x.

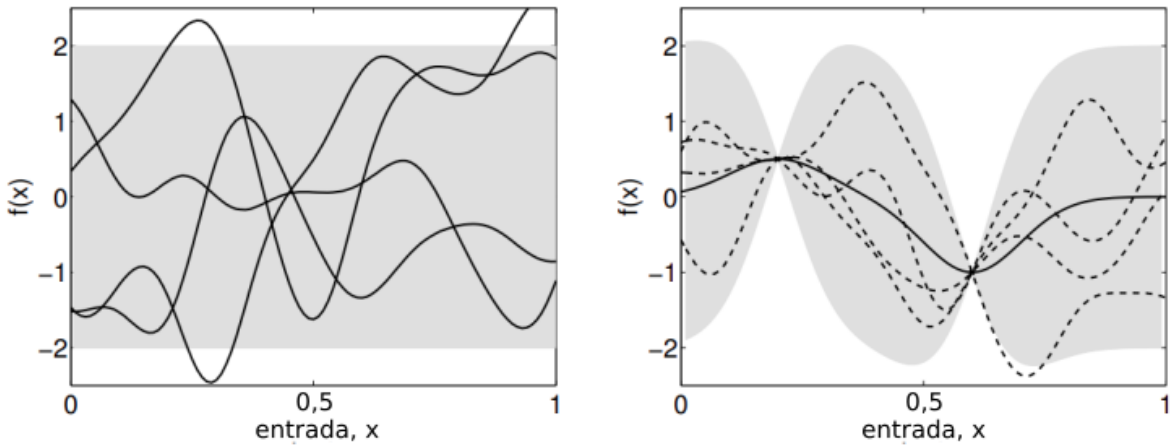

Figura 2.9: Evolução de Crenças sobre Função Desconhecida f
Fonte: Adaptado de Rasmussen & Williams (2005)

Os Processos Gaussianos que caracterizam a técnica estão justamente relacionados à função substituta utilizada por esta vertente da Otimização Bayesiana. O que ocorre é que nessa

[‖] A contraparte correspondendo as entradas das observações é dada por $x_{1:t} = \{x_1, ..., x_t\}$

vertente da Otimização Bayesiana, assume-se que a função f desconhecida é uma amostra de um Processo Gaussiano, de modo que a atualização da distribuição feita segundo o Teorema de Bayes diz respeito a distribuição da referida função, a partir das observações feitas. Mais informações sobre Processos Gaussianos podem ser obtidos em Rasmussen (2003).

Por se tratar de uma técnica de otimização global, e não local, a Otimização Bayesiana lida constantemente com um *tradeoff* entre exploração e aprofundamento: enquanto estima as características da função f desconhecida, deve decidir entre explorar regiões já tidas como promissoras para a maximização ao passo que não negligencia regiões de menor potencial da referida função.

O presente parágrafo encerra a subseção a respeito da técnica de Otimização Bayesiana, bem como a seção relacionada à otimização de hiperparâmetros. A próxima seção trata do algoritmo de Evolução Diferencial, utilizado no trabalho como parte da solução proposta.

2.9 Evolução Diferencial

O algoritmo de Evolução Diferencial é um algoritmo evolutivo da mesma classe dos Algoritmos Genéticos. Trata-se de um método para resolução de problemas com a característica de meta-heurística, no sentido de fazer pouca ou nenhuma suposição a respeito do problema de otimização sendo resolvido. Assim como outras meta-heurísticas, o algoritmo de Evolução Diferencial é capaz de buscar por soluções candidatas em grandes espaços de busca sendo, até por isso, recomendado para a realização de otimizações globais. Assim como outros algoritmos da mesma natureza, o algoritmo de Evolução Diferencial não tem garantia de encontrar soluções ótimas.

Ao lidar com algoritmos evolutivos, é importante conhecer o vocabulário que lhes é peculiar, bem como algumas características inerentes ao seu escopo. No escopo de algoritmos evolutivos, cuja inspiração provém das teorias de evolução biológica, o termo população diz respeito a um conjunto de entradas a ser testado como eventual solução para o problema sendo solucionado (problema de minimização de uma função f). Cada entrada pode, eventualmente, ser chamada de indivíduo da população. Os indivíduos são formados por vetores que contém suas características, sendo cada posição desse vetor conhecida como gene. No caso de otimização de hiperparâmetros, cada gene pode ser o valor de um hiperparâmetro. Geração é o termo dado a uma população em um dado tempo/momento da execução do algoritmo. O termo filho(s) está associado aos indivíduos gerados a partir de operações entre outros indivíduos de uma geração precedente.

Proposto originalmente em Storn & Price (1997), o algoritmo conhecido como Evolução Diferencial é explicado em Tirronen & Neri (2009) de forma a ser facilmente compreendido. Segundo Tirronen & Neri (2009), o algoritmo de Evolução Diferencial consiste da geração de

uma amostra inicial de população S_{pop} de forma pseudoaleatória, com o uso de uma distribuição uniforme dentro do espaço de busca em que a otimização está sendo realizada sobre uma função f. A cada geração, para cada indivíduo $x(i)$ de S_{pop}, são extraídos, também de forma pseudoaleatória, três indivíduos $x(i)_r$, $x(i)_s$ e $x(i)_t$. São os indivíduos $x(i)_r$, $x(i)_s$ e $x(i)_t$ que dão origem a um filho provisório tal como na equação (2.7).

$$x(i)'_{filho} = x(i)_t + F\big(x(i)_r - x(i)_s\big) \tag{2.7}$$

A equação (2.7) diz respeito ao chamado operador de mutação da Evolução Diferencial[†]. Conforme apresentado em Tirronen & Neri (2009), $F \in [0, 1[$ é um fator de escala que controla o tamanho do chamado vetor de exploração $\big(x(i)_r - x(i)_s\big)$, determinando a distância do filho sendo gerado em relação ao indivíduo $x(i)$.

Tirronen & Neri (2009) prosseguem explanando que, para aumentar a exploração do espaço de busca, cada gene do novo indivíduo $x(i)'_{filho}$ tem uma chance aleatória de ser trocado com o gene correspondente de $x(i)$, conforme pode ser visto na equação (2.8).

$$x(i)_{filho,j} = \begin{cases} x(i)_j & \text{se } rand(0,1) < CR \\ x(i)'_{filho,j} & \text{caso contrário} \end{cases} \tag{2.8}$$

A equação (2.8) determina a geração do filho $x(i)_{filho}$, com CR sendo uma constante que determina o quão provável será a troca de genes de $x(i)'_{filho}$ pelos de $x(i)$ para a geração de $x(i)_{filho}$. Trata-se do operador de recombinação (*crossover*) do algoritmo de Evolução Diferencial. Por fim, Tirronen & Neri (2009) explicam que, para a nova geração da população, os indivíduos $x(i)_{filho}$ substituem os indivíduos $x(i)$ apenas se $f\big(x(i)_{filho}\big) < f\big(x(i)\big)$ (trata-se de minimização).

Para facilitar ainda mais a compreensão do algoritmo de Evolução Diferencial, pode-se observar seu funcionamento em termos de pseudocódigo no Pseudocódigo 2.1. Conforme pode ser observado no Pseudocódigo 2.1, para a finalização do algoritmo de Evolução Diferencial, deve-se ainda definir um critério de parada. O critério de parada pode ser um valor mínimo atingido em f, uma variação desse valor entre as diversas gerações ou até um número de gerações.

A presente seção se encerra, finalizando as considerações entendidas como pertinentes à compreensão do algoritmo de Evolução Diferencial. A seção seguinte encerra este capítulo, tecendo algumas considerações finais a respeito do mesmo.

[†]Existem outros operadores de mutação que podem ser usados no algoritmo de Evolução Diferencial conforme pode ser visto em Tirronen & Neri (2009)

Pseudocódigo 2.1 Evolução Diferencial

```
1: Gera aleatoriamente uma população inicial S_pop
2: while not criterio_parada() do
3:     for x(i) in S_pop do
4:         Calcula f(x(i))
5:     end for
6:     for x(i) in S_pop do
7:         Seleciona x(i)_r, x(i)_s e x(i)_s aleatoriamente em S_pop
8:         x(i)'_filho ← x(i)_t + F(x(i)_r − x(i)_s)        ▷ Operador de Mutação
9:         x(i)_filho ← x(i)'_filho
10:        for j = 1 to n do                                ▷ n é o número de genes
11:            if rand(0,1) > CR then                       ▷ Operador de Recombinação
12:                x(i)_filho,j ← x(i)_j
13:            end if
14:        end for
15:        if f(x(i)_filho) < f(x(i)) then
16:            x(i) ← x(i)_filho
17:        end if
18:    end for
19: end while
```

2.10 Considerações finais

A presente seção finaliza a apresentação do referencial teórico necessário a compreesão da proposta feita e implementada neste trabalho de dissertação. O Capítulo 3 que se inicia a seguir trata dos trabalhos relacionados, apresentando o estado da arte dentro do qual este trabalho se insere.

Capítulo 3

Trabalhos Relacionados

3.1 Considerações Iniciais

Toda ciência é construída a partir do enredamento do conhecimento que provém do esforço coletivo da comunidade científica que a compõe enquanto corpo e sua produção. Desse modo, não há conhecimento científico isolado, uma vez que cada trabalho desenvolvido em um determinado tópico/domínio se conecta aos demais de alguma maneira. O presente capítulo apresenta os trabalhos que, em razão do seu objeto de estudo ou método, relacionam-se com este trabalho. Por meio de sua leitura, será possível ter um vislumbre do estado da arte em termos de detecção de intrusão em redes de computadores.

3.2 O Estado da Arte

Algo comum ao se realizar pesquisa científica é lidar com pesquisas que, apesar de abordarem um mesmo tópico, o fazem de forma diferente e com objetivos divergentes. O resultado disso é que nem sempre é possível realizar a comparação direta de seus resultados. Ainda assim, o contato com as mesmas enriquece a visão a respeito do problema sendo abordado, compondo uma fotografia correspondente a tudo aquilo que já se pesquisou sobre o assunto e permitindo a identificação de oportunidades para se contribuir com o chamado estado da arte.

Conforme visto na Seção 2.2, a detecção de intrusão em redes de computadores possui uma taxonomia própria que permite classificá-la sobre diversos aspectos, fato que evidencia a abrangência e variabilidade das técnicas disponíveis para a execução da tarefa que a caracteriza. Todavia, a escassez de estudos secundários recentes na forma de mapeamentos e revisões sistemáticas da literatura (Kitchenham, Budgen & Pearl Brereton, 2011; Petersen, Vakkalanka & Kuzniarz, 2015; Dyba, Kitchenham & Jorgensen, 2005) dentro do tópico em seu nível mais abrangente dificultam a exposição exaustiva do que há de mais atualizado para tratamento do problema. Assim, é mais comum que sejam encontrados estudos recentes dessa natureza focando em algum tipo de rede ou subcategoria do problema (Goncalves, Ribeiro, Gama, Santos,

Costa, Dias, MacEdo & Nicolau, 2019). Tendo em vista que a condução de estudos secundários dessa natureza poderia em si constituir em uma derivação de trabalho, optou-se por, para a apresentação desta seção, usar como referência tanto a composição dos estudo sistemáticos existentes, quanto pesquisas não sistemáticas realizadas no contexto deste trabalho, além de estudos identificados na forma de *snowballing* a partir de tais pesquisas. Utilizou-se ainda, como referência para apresentação desta seção, estudos secundários não sistemáticos que tangenciam o tópico (Liao, Richard Lin, Lin & Tung, 2013; Shashank & Balachandra, 2018; Fernandes, Rodrigues, Carvalho, Al-Muhtadi & Proença, 2019; Khraisat, Gondal, Vamplew & Kamruzzaman, 2019; Thomas & Pavithran, 2019; Bhuyan, Bhattacharyya & Kalita, 2014; Ahmad, Zainudin, Kama, Idris & Saudi, 2018). Foi mantido o foco em trabalhos recentes, desenvolvidos e publicados nos últimos quatro anos, com a preferência, sempre que possível, por aqueles que tenham sido divulgados em revistas e eventos com Qualis segundo o CNPq.

No contexto apresentado, um dos problemas recorrentes identificados na detecção de intrusão em redes de computadores diz respeito à obtenção de altas taxas de acerto[†] ao menor tempo possível. Soluções que apresentam essas características tendem a ser preferíveis às que carecem de quaisquer uma delas por viabilizarem à construção de bons sistemas de detecção de intrusão em redes de computadores que atuem em tempo real. Entendendo a inevitabilidade das intrusões e suas tentativas, a razão para a preferência por sistemas com as características a pouco citadas está na minimização do potencial destrutivo das ameaças virtuais direcionadas a sistemas sob sua proteção. Esses, com maiores taxas de acerto, estão relacionadas a menores distrações com falsos alarmes, por parte do gestor do sistema de detecção e/ou a um menor número de ataques passando despercebidos, junto a identificação precoce da situação a tempo de se atuar para mitigar seus danos.

Além disso, nota-se que, em se tratando de soluções que empregam um único classificador, antagônicas às que, como a proposta por este trabalho, realizam a combinação de classificadores, há uma pulverização de técnicas utilizadas na detecção de intrusão em redes de computadores. Uma outra observação frequente na literatura é o uso de pré-processamentos relacionados tanto a seleção de atributos (Mohan, 2017; Selvakumar & Muneeswaran, 2019; Liu et al., 2020; Vijayanand et al., 2018; Lu et al., 2018; Aljawarneh et al., 2018; Zhang & Zhu, 2018; Reazul et al., 2017; Hamed et al., 2018) quanto à extração de atributos (Kasongo & Sun, 2020; Elkhadir & Mohammed, 2019; Aljawarneh et al., 2018; Cepheli et al., 2016), ao ponto de parte dos trabalhos identificados caracterizarem suas soluções em seus títulos ou objetivos mais por esses aspectos do que pelos classificadores subjacentes. Enquanto a seleção de atributos permite reduzir a base de dados sobre a qual se está trabalhando por meio da manutenção de um subconjunto mínimo de atributos informativos, a extração de atributos substitui atributos pré-existentes por novos atributos mais densos do ponto de vista informacional, combinando-os ou identificando relações entre os mesmos. A redução de dimensionalidade proporcionada por ambas as técnicas agiliza a classificação ao mesmo tempo que contribui com a capacidade dos

[†]Utilizou-se aqui o termo "taxas de acerto"como uma substituição genérica às métricas de desempenho formais como as apresentadas na Seção 2.4 que de fato aferem as qualidades dos modelos

modelos gerados realizarem predições corretas, tendo em vista que preza pela manutenção de atributos mais informativos em detrimento de atributos menos informativos.

Um dos trabalhos identificados cujo esforço se concentrou na seleção de atributos foi o de Hamed et al. (2018). Sua técnica de Adição Recursiva de Atributo (*Recursive Feature Addition*) com uso de bigramas se utiliza de um algoritmo SVM para a seleção incremental dos melhores atributos para se realizar a classificação de um fluxo de dados de rede em termos de dados normais ou intrusivos. A solução proposta, com menor enfoque no classificador utilizado, fez uso da base de dados de *benchmark* ISCX 2012.

Foi observada a recorrência de soluções relacionadas à algoritmos evolutivos (Mohan, 2017; Vijayanand et al., 2018; Lu et al., 2018; Reazul et al., 2017) em diversas etapas das soluções propostas voltadas à detecção de intrusão em redes de computadores apresentadas nesta seção. Nos trabalhos de Reazul et al. (2017) e Vijayanand et al. (2018), os algoritmos evolutivos estão especificamente ligados à etapa de pré-processamento no que diz respeito à seleção de atributos. No trabalho de Lu et al. (2018), mais concentrado na detecção de intrusão em redes sem-fio, faz-se uso de algoritmos evolutivos na vertente de Programação de Redes Genéticas (*Genetic Network Programming*) para a seleção de regras de detecção de intrusão. Em contrapartida, no trabalho de Mohan (2017), é feito o uso de um algoritmo genético para a geração de uma nova população que auxilia no treinamento do classificador proposto: uma rede MLP como a que compõe a solução proposta neste trabalho.

As redes MLP, inclusive, se mostraram como soluções frequentes ao problema de detecção de intrusão em redes de computadores. No trabalho que se citou previamente Mohan (2017), a abordagem proposta foi capaz de atingir um nível de acurácia de $99,67\%$ sobre a base de dados NSL-KDD. Tendo em vista a recorrência do problema de se identificar intrusões de forma correta e rápida bem como a desejabilidade de se incorporar as virtudes correspondentes nas soluções propostas, o estudo de Shenfield et al. (2018) também propõe a utilização de uma rede MLP que, conforme apresentado na Seção 2.5, possui a marcante característica de realizar classificações rápidas uma vez que tenham sido adequadamente treinadas. Utilizando validação cruzada de 10 *folds*, sua solução foi capaz de atingir desempenho médio em termos de acurácia da ordem de 98% e sensibilidade de 95% (a especifidade não foi informada). Infelizmente, por fazer uso de uma base de dados própria não publicada, a reprodutibilidade dos seus resultados fica comprometida e não verificável.

Também por meio do uso de redes MLP, o trabalho de Ali Alheeti & McDonald-Maier (2016), mais focado em redes *ad-hoc* veiculares (VANETs) para carros autônomos, foi capaz de obter bons resultados em termos de métricas de acurácia. Para *benchmark*, sua solução fez uso da base de dados de eventos Kyoto (Song, Takakura, Okabe, Eto, Inoue & Nakao, 2011), extraindo 60.000 registros da mesma que antes de servirem de entrada à rede neural, passam por um processo de pré-processamento referente à seleção de atributos e posterior fuzzificação (Ramkumar & Murugeswari, 2015). Com a implementação realizada, foi possível obter acurácias entre $99,05\%$ e $99,18\%$, a depender da tarefa sendo avaliada, uma vez que trata-se de uma

solução híbrida em termos de método de detecção. O trabalho distingue em seus resultados a detecção de mau-uso da detecção de anomalias na rede. Na Figura 3.1, extraída do trabalho original de Ali Alheeti & McDonald-Maier (2016), é possível ver com maior clareza o método proposto. Na fase de pré-processamento (*preprocessing phase*), atividades diversas tais como normalização e codificação são realizadas, precedendo a seleção de atributos realizada pela técnica de POS (*Proportional Overlapping Score*) (Mahmoud, Harrison, Perperoglou, Gul, Khan, Metodiev & Lausen, 2014). Após a fuzzificação, na fase de treinamento, são reduzidos atributos de modo a se manter um determinado limiar de acurácia previamente definido. Por fim, a rede MLP pode ser testada, retornando classificações de um dos três tipos: desconhecido, anormal ou normal.

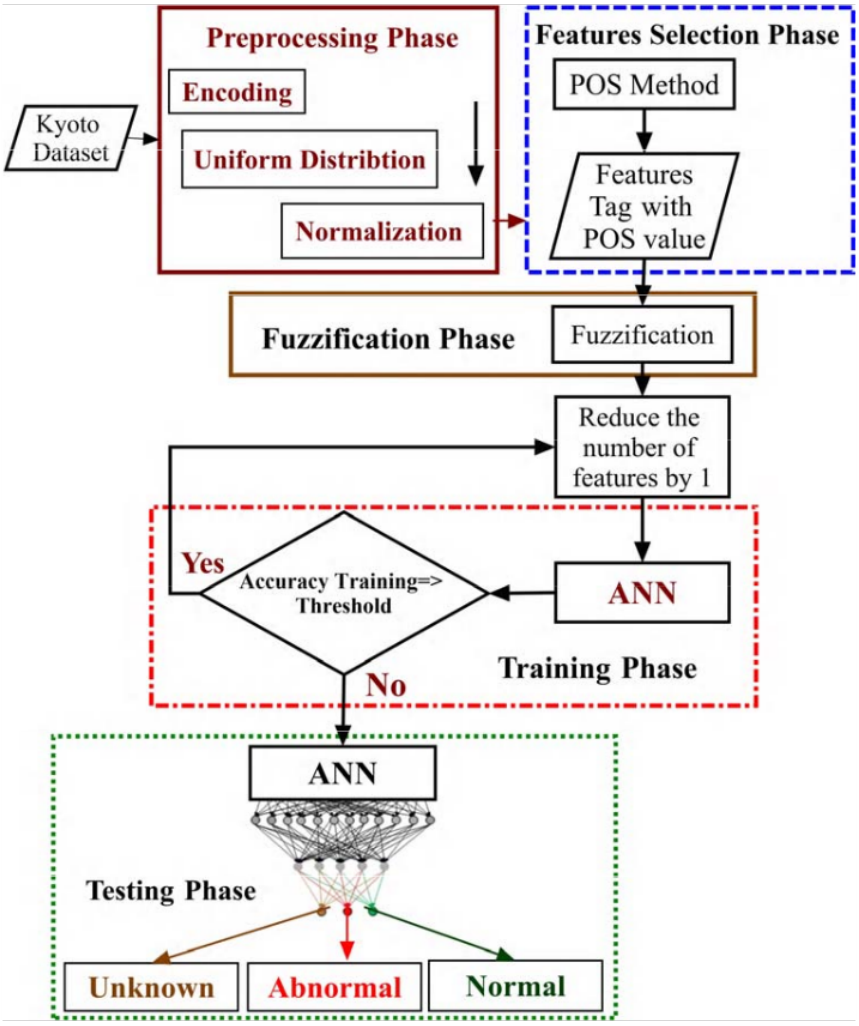

Figura 3.1: Arquitetura Geral da Proposta de Ali Alheeti & McDonald-Maier (2016)
Fonte: Ali Alheeti & McDonald-Maier (2016)

Outros tipos de redes neurais, além das MLP, vêm sendo utilizadas para a detecção de intrusão em redes de computadores. No caso do trabalho de Kasongo & Sun (2020), foi feito o uso de Redes Neurais Profundas Alimentadas Adiante (*Feed-Forward Deep Neural Networks*).

Com foco mais específico em redes sem-fio, Kasongo & Sun (2020) realizaram o *benchmark* do modelo criado utilizando-se da base de dados AWID (Kolias, Kambourakis, Stavrou & Gritzalis, 2015). Precedido por pré-processamentos referentes à normalização e extração de atributos, com uma validação do tipo *holdout*, seu modelo foi capaz de obter acurácia de $99,66\%$ para a classificação binária.

As Redes Bayesianas também obtiveram destaque devido sua recorrência entre as soluções apresentadas nesta seção. O *framework* proposto por Reazul et al. (2017) as utiliza como classificador base em sua solução, obtendo taxa de acurária da ordem de $98,26\%$ com validação cruzada de 10 *folds* sobre a base de *benchmark* NSL-KDD. No trabalho de Selvakumar & Muneeswaran (2019), após o pré-processamento referente à seleção de atributos, foi realizado, separadamente, o *benchmark* por meio da base de dados KDD-99, tanto de uma Rede Bayesiana como classificador, como de uma árvore de decisão obtida a partir do algoritmo C4.5.

Adotados como parte da solução proposta por este trabalho, os modelos gerados a partir do K-NN se fazem presentes no estado da arte apresentado nesta seção. Na solução proposta por Elkhadir & Mohammed (2019), utiliza-se um desses modelos como classificador base após o preprocessamento de seleção de atributos efetuado por meio de uma nova abordagem variante da Análise Linear Discriminante (*Linear Discriminant Analysis*). A validação da solução é realizada tanto sobre a base de dados KDD-99 quanto sobre a base de dados NSL-KDD, com as métricas de desempenho obtidas em termos dos tipos de ataques presentes nessas bases.

Com uma proposta um pouco diferente da dos trabalhos que se apresentou até o momento, no trabalho de Zhang & Zhu (2018), além do foco em VANETs, há uma preocupação com questões relacionadas à privacidade, uma vez que sua proposta se concentra na detecção de intrusão por meio de aprendizagem de máquina distribuída entre os veículos da rede. A preocupação com a privacidade em seu trabalho diz respeito às informações compartilhadas entre os veículos com o intuito de aprimorar a detecção de intrusões na rede. Analisando os *trade-offs* entre privacidade e desempenho em termos de métricas, foi feito o uso da base de dados NSL-KDD como base de *benchmark* para validação do modelo proposto.

Da mesma forma que no trabalho de Zhang & Zhu (2018), em uma linha bastante vanguardista, Liu et al. (2020) propõem uma solução de detecção de intrusão em redes de computadores adaptativa. Por meio do uso de Modelo de Misturas Gaussianas (*Gaussian Mixture Model*), gera-se por agrupamento, bibliotecas específicas de padrões de ataque e de conexões normais a partir de atributos selecionados por uma etapa de pré-processamento. Os referidos padrões sofrem atualizações automáticas que conferem a característica adaptativa da solução proposta. Os resultados obtidos com a técnica foram avaliados por meio da base de dados NSL-KDD e reportados em termos de diversas métricas de desempenho.

Em termos de soluções que adotam a combinação de classificadores, técnicas já apresentadas nesta seção também se fazem presentes. É o caso das soluções baseadas em regras (Lu et al., 2018), do Modelo de Misturas Gaussianas (Liu et al., 2020), das árvores de decisão

(Selvakumar & Muneeswaran, 2019) e do classificador SVM (Hamed et al., 2018).

Para a construção de um modelo híbrido de classificação em termos de método de detecção, capaz de detectar intrusões tanto por assinaturas/mau uso quanto por anomalia, Cepheli et al. (2016) realizaram a combinação de regras da ferramenta SNORT, especializadas no primeiro tipo de intrusão, com Modelos de Misturas Gaussianas para a detecção do segundo tipo. Com a base de dados DARPA 2000 utilizada para validação e aplicando pré-processamento para extração de atributos, a referida solução foi capaz de alcançar uma taxa de detecção de verdadeiros positivos da ordem de $98,7\%$, ao passo que a taxa de detecção de falso positivos ficou próxima a $0,73\%$.

Voltada para redes sem-fio do tipo *mesh*, a solução envolvendo composição de classificadores proposta em Vijayanand et al. (2018) faz uso de múltiplos classificadores SVM, cada um especializado em um tipo de ataque. Em sua proposta, múltiplas seleções locais de atributos são realizadas por algoritmos genéticos, de modo que o conjunto de atributos selecionado esteja diretamente associado ao tipo de ataque que se está tentando identificar. A solução proposta em seu trabalho foi validada tanto com o uso das bases de dados de *benchmark* DFA-LD(Creech & Hu, 2013) e CICIDS 2017(Sharafaldin, Lashkari & Ghorbani, 2018), quanto com o uso de uma base de dados construída especificamente para esse propósito em um simulador de redes sem-fio do tipo *mesh*, obtendo acurácia de $96,95\%$, $99,85\%$ e $95,7\%$, respectivamente, em cada uma dessas bases.

A solução envolvendo combinação de classificadores proposta por Aljawarneh et al. (2018) tem o foco na obtenção de bons valores de acurácia sobre a base de dados NSL-KDD. Com pré-processamentos referentes tanto à normalização de atributos, quanto a seleção e extração de atributos, utiliza-se de uma árvore de decisão obtida pelo algoritmo J48 e de outros seis classificadores combinados. De simples compreensão, a abordagem proposta consiste em escolher o classificador com melhor acurácia entre os sete para compor o modelo final. Na validação, realizada com *holdout* sobre a distribuição pública da base de dados NSL-KDD já separada em conjunto de treinamento e de testes para uma amostra de 20%, sua solução foi capaz de obter $99,75\%$ de acurácia*.

Para finalizar a caracterização do estado da arte, apresenta-se também uma solução para o problema de detecção de intrusão em redes de computadores envolvendo otimização de hiperparâmetros. Tendo como objeto de otimização os hiperparâmetros de uma MLP, a solução apresentada por (Kanimozhi & Jacob, 2019), considerando a versão binarizada do problema, foi capaz de obter uma acurácia sobre o conjunto de validação da ordem de $99,97\%$ sobre a base de dados de *benchmark* CICIDS 2018† (Sharafaldin et al., 2018). Os resultados obtidos foram provenientes de uma validação cruzada de 10 *folds*. Sua solução partiu de uma MLP com duas camadas ocultas/internas e fez uso da Busca em Grade, apresentada na Subseção 2.8.1; os

*O artigo reporta de forma incorreta ter obtido 99.81% de acurácia, sendo que sua taxa de erro (valor complementar) é reportada como $0,25\%$

†http://www.unb.ca/cic/datasets/ids-2018.html

hiperparâmetros otimizados foram o número de neurônios em cada uma das camadas ocultas e o valor do hiperparâmetro *alfa* inerente a uma regularização do tipo L2. O trabalho de Shenfield et al. (2018), já citado anteriormente, também fez uso de otimização de hiperparâmetros para definir a estrutura de sua rede MLP. A Busca em Grade também foi a escolha em sua proposta.

O trabalho proposto nesta dissertação se diferencia dos demais trabalhos elencados no estado da arte por utilizar otimização de hiperparâmetros aplicada junto à composição de classificadores de aprendizagem de máquina, algo que, pelo que se levantou enquanto da construção da revisão da literatura na qual a presente seção se fundamenta, não fora utilizado no escopo de detecção de intrusão em redes de computadores. Conforme relatado no Capítulo 5, o método, enquanto proposta, não se restringe à busca pela vanguarda em termos de aplicação de tecnologias ao domínio do problema, mas também alcança resultados competitivos em relação aos demais métodos que nesta seção se apresentam, superando, com a geração de um modelo final simples, boa parte dos mesmos no que diz respeito a métricas de desempenho.

3.3 Considerações Finais

O Capítulo que se encerra apresentou diversos trabalhos relacionados à detecção de intrusão em redes de computadores, de modo a contextualizar as contribuições do presente trabalho ao estado da arte. Por meio de sua leitura, foi possível observar as convergências e heterogeneidades existentes entre as soluções propostas para o tópico. O Capítulo subsequente trata de apresentar os materiais e métodos que se utilizou para o desenvolvimento da pesquisa proposta e executada, a qual este texto relata.

Capítulo 4

Materiais e Métodos

4.1 Considerações Iniciais

Este capítulo apresenta os materiais utilizados para o desenvolvimento deste trabalho, compreendendo também a aplicação dos métodos e do referencial teórico apresentados no Capítulo 2, cuja escolha proveio da análise dos trabalhos relacionados e apresentados no Capítulo 3. Por meio da leitura deste capítulo, é possível ter uma visão holística do que foi feito durante o desenvolvimento do trabalho que aqui se apresenta.

A Seção 4.2 se inicia com a apresentação do EMHOSIR, um método de combinação de classificadores genérico proposto com base no método de combinação de classificadores de Souza (2018), que prevê a existência de fases específicas de otimização de hiperparâmetros e de autoajuste de faixa intermediária. As Subseções (4.2.1, 4.2.2, 4.2.3) apresentam ainda as três fases internas que compõe o EMHOSIR, aprofundando-se em questões relacionadas ao seu funcionamento e à implementação realizada. A Seção 4.3 apresenta as atividades de pré-processamento executadas sobre a base de dados de eventos NSL-KDD. A Seção 4.4 apresenta mecanismos aplicados para a redução de dimensionalidade do espaço de busca da otimização de hiperparâmetros. A caracterização dos experimentos feitos a fim de validar e extrair métricas referentes ao EMHOSIR é apresentada na Seção 4.5. Em seguida, na Seção 4.6, é apresentado o ferramental de *software* e de *hardware* empregados durante as etapas de implementação e experimentação. Por fim, na Seção 4.7, são apresentadas as considerações finais a respeito deste capítulo.

4.2 EMHOSIR: Um Método de Combinação de Classificadores com Otimização de Hiperparâmetros e Autoajuste de Faixa Intermediária

Proposto no escopo deste trabalho, o EMHOSIR é um método de combinação de classificadores que se baseia no método de combinação de classificadores de Souza (2018) e prevê a implementação de fases específicas voltadas à otimização de hiperparâmetros e ao autoajuste de faixa intermediária. Com a aplicação direcionada à detecção de intrusão em redes de computadores, o EMHOSIR precisa ser entendido nas duas apresentações por meio das quais se faz presente ao longo deste texto: uma como método genérico de aprendizagem de máquina e outra como instância/implementação do referido método.

O EMHOSIR enquanto método genérico é uma abstração intimamente ligada às fases e mecanismos internos que dele fazem parte, discutidos em maiores detalhes no decorrer desta seção e suas subseções. Apesar de concebido para ser aplicado ao problema de detecção de intrusão em redes de computadores, o EMHOSIR, na condição de método, pode ser implementado e/ou instanciado de diversas maneiras, sobre outros escopos e domínios, de formas tão distintas e alheias à da implementação deste trabalho quanto se fizer necessário, desde que respeitando as fases e mecanismos internos que o caracterizam.

Por outro lado, o EMHOSIR enquanto implementação realizada do método abstrato homônimo contém especificidades que não se estendem ao método sobrejacente. As técnicas utilizadas para otimização de hiperparâmetros e autoajuste de faixa intermediária são só dois exemplos dessas especificidades cujas características são exclusivas da implementação realizada, não necessariamente inerentes ao EMHOSIR enquanto método que apenas prevê sua existência sem especificar detalhes de como fazê-lo.

No EMHOSIR enquanto método, apesar de haver a previsão tanto dos classificadores quanto dos mecanismos referentes a otimização de hiperparâmetros e ao autoajuste da faixa intermediária, esses não estão previamente definidos no método: são as instâncias/implementações, como a realizada neste trabalho, que os determinam. A Figura 4.1 ilustra a relação entre o método EMHOSIR e uma de suas instâncias.

É válido dizer que há momentos no texto em que as referências ao EMHOSIR fazem alusão a ele como método abstrato; em outros instantes, essas referências estão direcionadas à implementação que se fez da referida abstração. Sempre que não há comprometimento da compreensão a respeito do que se quer apresentar, para simplificação da leitura, optou-se pela não diferenciação entre os termos. Especificidades, quando presentes, no entanto, estão fortemente marcadas por expressões que denotam seu caráter, tais como *"na implementação do EMHOSIR realizada"* ou então *"na instância do EMHOSIR implementada"*.

Apesar do víes teórico inerente à proposição de um método genérico, o presente traba-

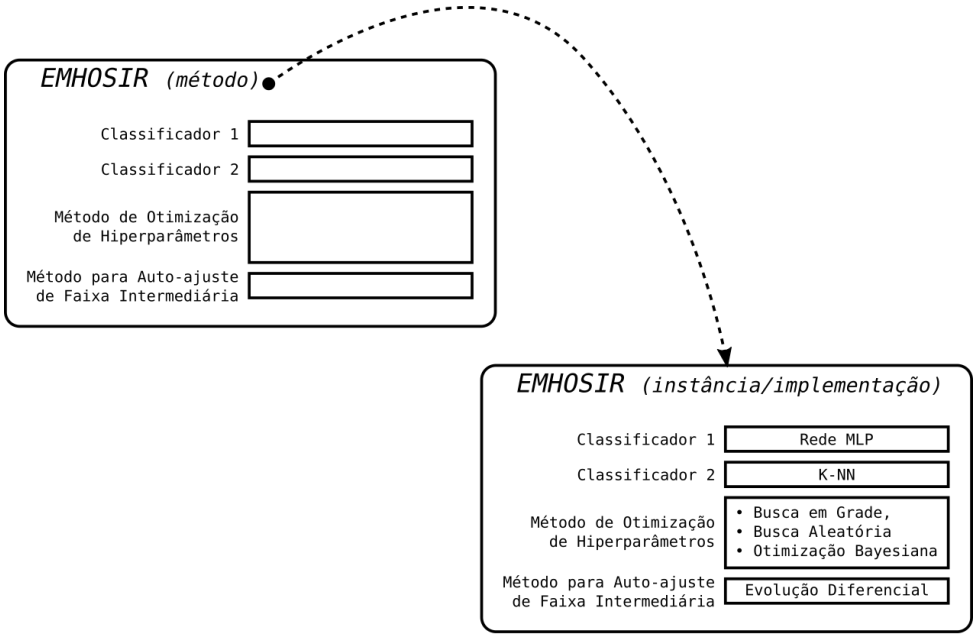

Figura 4.1: EMHOSIR vs implementação do EMHOSIR

lho não se limitou à idealização do EMHOSIR. A implementação realizada de uma instância funcional do mesmo, cujos detalhes se apresentam no decorrer deste capítulo, possibilita tanto a validação da viabilidade de se criar instâncias variadas do método, quanto a averiguação do comportamento da instância específica criada diante de possíveis cenários de utilização.

Um outro relevante ponto a ser discutido diz respeito ao fato do EMHOSIR ser baseado no método de combinação de classificadores de Souza (2018). Tal situação faz com que ambos os métodos sejam semelhantes em diversos aspectos. Face ao exposto, esta seção se dedica ainda a destacar as diferenças e semelhanças existentes entre os dois métodos. Também foi dada especial atenção ao detalhamento do novo método desenvolvido, de modo a permitir uma melhor compreensão acerca das contribuições realizadas.

Em termos de semelhança, é importante observar que, por se tratarem de soluções de combinação de classificadores (*ensemble*), ambos os métodos adotam a composição dos resultados obtidos por múltiplos classificadores para chegarem a uma classificação final. Na condição de método abstrato, o EMHOSIR prevê, pela mesma técnica de *pipeline* de fluxo condicional utilizada no trabalho de Souza (2018), a combinação de dois classificadores, com a premissa de que o primeiro deles seja mais rápido que o segundo em termos de tempo. A ideia é maximizar a quantidade de exemplos classificados pelo primeiro classificador, mais rápido, relegando ao segundo classificador apenas os exemplos para os quais o primeiro não foi capaz de realizar uma classificação assertiva. Da mesma forma que no trabalho de Souza (2018), o critério para se determinar a assertividade da classificação feita pelo primeiro classificador se utiliza da relação entre a métrica de desempenho que se está considerando e as faixas de intervalos de saída dos valores retornados por esse classificador. No caso específico da instância do EMHOSIR implementada e do método de combinação de classificadores de Souza (2018), a composição

em *pipeline* de fluxo condicional (detalhada na Seção 2.7) é realizada com uma rede MLP seguida de um modelo obtido a partir do método K-NN. O critério para determinar se um exemplo precisa ser analisado pelo segundo classificador, quando comparado ao método de combinação de classificadores de Souza (2018), permanece inalterado no EMHOSIR, de modo que apenas os exemplos para os quais a rede MLP gera saídas com valores dentro do intervalo da faixa intermediária (discutida na Seção 2.7) seguem por esse fluxo no *pipeline* de fluxo condicional. Também é inerente ao EMHOSIR a utilização de mecanismos que possibilitem tanto a otimização de hiperparâmetros quanto o autoajuste da faixa intermediária, a qual é justamente responsável por definir quais das instâncias deverão ser classificadas pelo segundo classificador. Essa é a diferença mais marcante entre os dois métodos.

Tanto o EMHOSIR quanto o método de combinação de classificadores de Souza (2018) geram modelos capazes de realizar a classificação de exemplos de uma base de dados de eventos ao fim de seu ciclo de execução. Sendo assim, para que se possa prosseguir com as diferenças e semelhanças entre os métodos, faz-se necessário introduzir a Notação 4.1, capaz de representar tanto um modelo gerado a partir do método de combinação de classificadores de Souza (2018) quanto a partir do EMHOSIR:

$$E_{P_l}^{P_h}(C1,\ C2) \tag{4.1}$$

Considerando o método de combinação de classificadores de Souza (2018) como exemplo, na Notação 4.1, os valores de fronteira da parte positiva e negativa da faixa intermediária são representados por h e l, respectivamente. De forma semelhante, os classificadores que compõe o *pipeline* de fluxo condicional do método são representados por $C1$ e $C2$, na ordem respectiva de sua composição. Desse modo, segundo a notação ora apresentada, um modelo E obtido pelo método de combinação de classificadores de Souza (2018), com o percentil 1 para a parte positiva da faixa intermediária e 25 para a parte negativa, possui a representação $E_{P_{25}}^{P_1}$*(MLP, K-NN)* na Notação 4.1.

Conforme fora adiantado, a diferença crucial entre o método proposto e o método de combinação de classificadores de Souza (2018) está na forma com a qual cada um deles realiza a configuração de dois elementos: os hiperparâmetros e os valores de fronteira da faixa intermediária. Enquanto no método de combinação de classificadores de Souza (2018), tanto os valores de fronteira da faixa intermediária (valores de h e l, segundo a Notação 4.1) quanto os valores dos hiperparâmetros são pré-definidos/fixados com base em dados empíricos que devem ser previamente obtidos, no EMHOSIR esses dois elementos têm seus valores determinados automaticamente, em função da tarefa de classificação corrente. Na forma de um comparativo entre suas principais características, a Tabela 4.1 sumariza as diferenças e semelhanças entre o método de combinação de classificadores de Souza (2018) e a instância do EMHOSIR implementada.

Outra questão relevante diz respeito a diferenças de nomenclatura existentes entre o EMHOSIR e o método de combinação de classificadores de Souza (2018) no que diz respeito às saídas

Tabela 4.1: Comparativo entre o Método de Combinação de Classificadores de Souza (2018) e a Instância do EMHOSIR Implementada

Característica	*Método de Combinação de Classificadores de Souza (2018)*	*Instância do EMHOSIR Implementada*
Primeiro Classificador do *Pipeline* de Fluxo Condicional	Rede MLP	
Segundo Classificador do *Pipeline* de Fluxo Condicional	K-NN	
Critério para Direcionar Fluxo do *Pipeline* ao Segundo Classificador	Valor de Saída da rede MLP dentro da Faixa Intermediária	
Valores de Fronteira da Faixa Intermediária	Pré-definidos/Fixos	Autoajustados
Valores dos Hiperparâmetros	Pré-definidos/Fixos	Otimizados

do primeiro classificador $C1$ e a faixa intermediária relacionada à essas saídas que determina quais instâncias serão submetidas à classificação do segundo classificador $C2$. Conforme apresentado na Seção 2.7, a faixa intermediária no método de combinação de classificadores de Souza (2018) é, na verdade, formada por duas partes: uma responde pelo nome de parte positiva e a outra por parte negativa, cada uma com seu próprio valor de fronteira em termos de percentil que determina, para a parte considerada, as instâncias a serem classificadas pelo segundo classificador $C2$. O nome das partes da faixa intermediária advém do limiar que separa as saídas do classificador $C1$ dividindo-as em duas, a chamada parte positiva, cujos valores são superiores ao limiar $0,5$ enquanto que a parte negativa é menor ou igual a esse valor. Logicamente, a parte positiva da faixa intermediária corresponde aos valores da faixa intermediária compreendidos entre $0,5$ e o limite positivo da mesma, igualmente superior à $0,5$; por sua vez, a parte negativa da faixa intermediária corresponde aos valores menores ou iguais à $0,5$ até os valores limítrofes correspondentes à essa parte.

No mecanismo de *pipeline* de fluxo condicional utilizado por Souza (2018), dividir e nomear as saídas de $C1$ e as partes correspondentes da faixa intermediária é bastante coerente uma vez que a nomenclatura dada tira proveito de duas situações inerentes a solução adotada em seu trabalho. A primeira delas está relacionada ao fato de se tratar de uma classificação binária dentro do escopo de detecção de intrusão em redes de computadores, em que as instâncias sendo classificadas podem ou não ser identificadas como intrusivas. Nesse cenário, uma instância, ao ser classificada como intrusiva, recebe, na verdade, uma classificação positiva; de forma antagônica, uma instância, ao ser classificada como não-intrusiva, recebe uma classificação negativa. A segunda situação diz respeito ao fato de, em termos matemáticos, a função

tangente hiperbólica, utilizada como função de ativação da camada de saída da rede MLP no método de combinação de classificadores de Souza (2018), retornar valores de saída pertencentes ao intervalo $(-1, +1)$, com partes positivas e negativas, reforçando a nomenclatura adotada para as saídas providas por $C1$ e para as partes da faixa intermediária correspondentes. No método em questão, um modelo $E^{P_h}_{P_l}$*(C1, C2)* gerado para o qual as saídas providas por $C1$ sejam valores acima de $0,5$ (parte positiva da faixa intermediária) serão classificadas como intrusivas ou condicionalmente enviadas à $C2$ de acordo com os valores de fronteira h da parte positiva da faixa intermediária; as demais instâncias, cujos valores de saída de $C1$ sejam inferiores ou iguais a $0,5$ (pertencentes a parte negativa da faixa intermediária), serão classificadas como não intrusivas ou condicionalmente enviadas à $C2$ de acordo com o valor de fronteira l definido para a parte negativa da faixa intermediária.

Por meio do esboço da função de ativação tangente hiperbólica (curva em vermelho), a Figura 4.2 apresenta os possíveis valores de saída de $C1$ no método de combinação de classificadores de Souza (2018), a relação entre as saídas de $C1$ e as partes positiva e negativa da faixa intermediária também fica explícita. É possível observar que o limiar $0,5$ que separa a parte positiva da parte negativa das saídas de $C1$ não o faz em iguais proporções, mas na relação $1/4$ para $3/4$, respectivamente. Essa desproporção se reflete na extensão das partes positiva e negativa das saídas de $C1$ dentro do intervalo de valores retornados pela função de ativação da camada de saída (no caso, a tangente hiperbólica). Note-se contudo que, apesar da parte positiva das saídas de $C1$ possuir, em termos visuais, o tamanho de três vezes a parte negativa em relação ao conjunto imagem da função tangente hiperbólica, é a distribuição de frequência dos efetivos valores de saída de $C1$ que determina a verdadeira extensão de cada uma das partes em termos da quantidade de valores a cada uma delas pertencente.

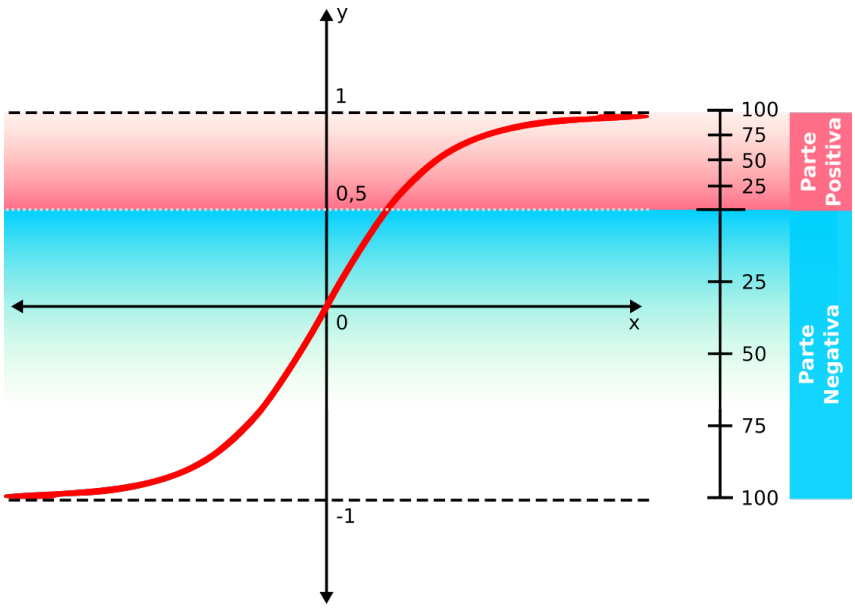

Figura 4.2: Relação entre Saída da Função Tangente Hiperbólica e Partes da Faixa Intermediária de Souza (2018)

As variadas possibilidades de função de ativação da camada de saída de $C1$ na implementação do EMHOSIR implicam em valores de saída de $C1$ potencialmente fora do intervalo fixo $(-1, +1)$ intrínseco ao método de combinação de classificadores de Souza (2018). Além disso, para um modelo $E^{P_h}_{P_l}$*(C1, C2)* gerado, o fato do valor de saída provido por $C1$ estar em quaisquer das duas partes da faixa intermediária não implica necessariamente na classificação positiva ou negativa correspondente, uma vez que, dependendo dos valores de h e l, o acionamento do classificador $C2$ pode vir a gerar uma classificação final divergente da sugerida por $C1$. Isso significa dizer que o pertencimento a uma das partes da saída de $C1$ não necessariamente implica na classificação que aquela parte sugere. Por conta disso, no EMHOSIR optou-se por substituir os termos "parte positiva"e "parte negativa"das saídas de $C1$ por "parte superior"e "parte inferior"respectivamente. A mesma terminologia foi adotava para as partes da faixa intermediária correspondente, mantido, em ambos os casos, o limiar de $0,5$ delimitando partes superiores de inferiores. Sendo assim, em termos de faixa intermediária, ao invés de valor de fronteira da parte positiva, tem-se o valor de fronteira da parte superior, cujo percentil h determina quais instâncias devem ser submetidas ao escrutínio do classificador $C2$, de maneira análoga, ao invés de valor de fronteira da parte negativa, no EMHOSIR o que existe é o valor de fronteira da parte inferior, correspondente ao percentil l que determina, para essa parte, as instâncias a serem submetidas ao classificador $C2$. A nomenclatura se baseia na posição dos valores que constituem a parte em questão em relação ao limiar que separa tanto os valores de saída de $C1$ quanto cada uma delas em termos do plano cartesiano (acima ou abaixo do referido limiar).

Outras questões emergem quando se realiza uma análise mais criteriosa do EMHOSIR enquanto proposta teórica. Considerando-se, por exemplo, o sentido mais amplo da definição de hiperparâmetros apresentada na Seção 2.8. Os valores de fronteira da faixa intermediária são, por definição, um subconjunto dos hiperparâmetros do método de combinação de classificadores de Souza (2018), em razão disso, precisam ser definidos *a priori*, assim como os valores dos demais hiperparâmetros do método. Desse modo, seria suficiente caracterizar o EMHOSIR como sendo uma variação do método de combinação de classificadores de Souza (2018) com otimização de hiperparâmetros, uma vez que tal característica abrangeria também a auto-definição dos valores de fronteira da faixa intermediária. Contudo, dois motivos fizeram com que se optasse por tratar essa porção dos hiperparâmetros de maneira especial, situação que também contribuiu para compor a nomenclatura e caracterização do novo método. O primeiro motivo está relacionado ao fato de, no EMHOSIR, o mecanismo de obtenção dos valores de fronteira da faixa intermediária não ser necessariamente o mesmo utilizado para obtenção dos valores dos demais hiperparâmetros.

Na implementação realizada do EMHOSIR, por exemplo, enquanto a determinação dos demais hiperparâmetros do método se dá por meio das técnicas de otimização descritas na Seção 2.8, a fase de obtenção dos valores de fronteira da faixa intermediária faz uso da técnica de Evolução Diferencial (Seção 2.9). Assim, diferenciando-se da otimização de hiperparâmetros, essa fase no EMHOSIR, inclusive, recebe o nome de autoajuste de faixa intermediária. Os

detalhes do seu funcionamento e da implementação realizada são descritos mais adiante, na Subseção 4.2.3. O segundo motivo está no fato de, no novo método, os valores de fronteira da faixa intermediária serem autoajustados após a fase de treinamento, com um momento distinto e posterior ao da fase de otimização que atribui valores aos demais hiperparâmetros do método. Essa cronologia torna o autoajuste da faixa intermediária dependente da otimização, tendo seu resultado diretamente afetado pelos valores encontrados para os hiperparâmetros otimizados.

Compreendidas as diferenças e semelhanças existentes entre o método de combinação de classificadores de Souza (2018) e o EMHOSIR, por meio da Figura 4.3, é possível refinar a entendimento acerca de detalhes de funcionamento do novo método, incluindo as fases de otimização de hiperparâmetros e de autoajuste de faixa intermediária, bem como a cronologia existente entre elas e as fases precedentes e subsequentes. Trata-se de uma visão em alto-nível das fases do método proposto, a serem detalhadas em Subseções próprias.

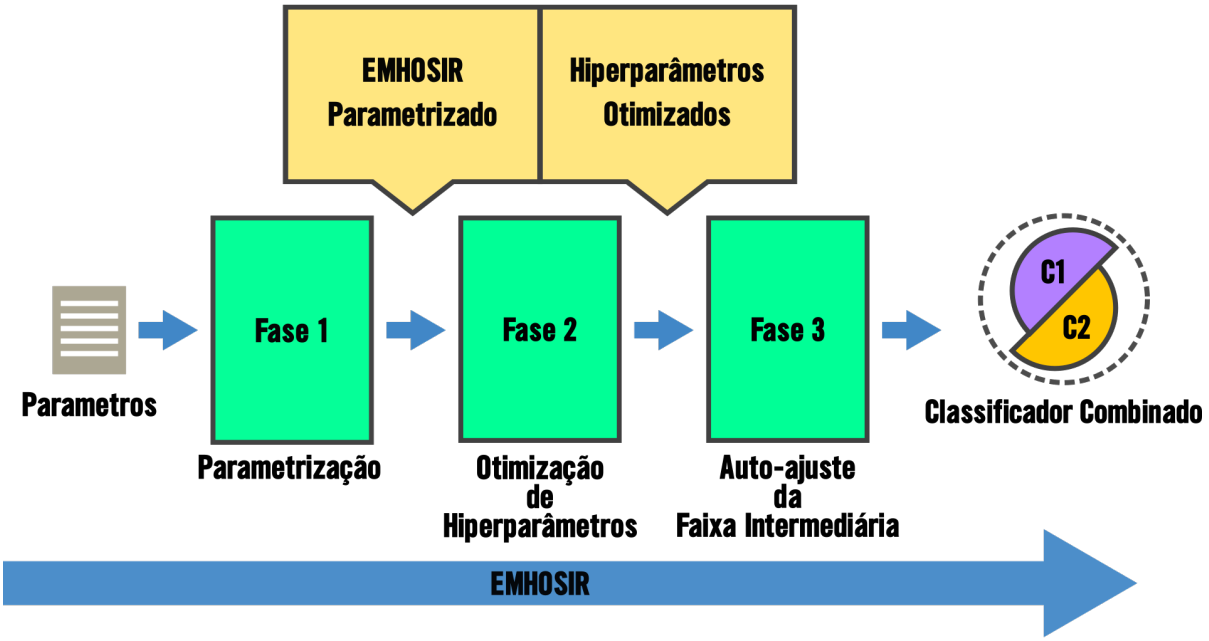

Figura 4.3: Fases do EMHOSIR: Uma Visão em Alto-Nível

A Fase 1 do EMHOSIR diz respeito a sua parametrização. Na implementação que se fez, essa fase se inicia pelo consumo de um arquivo de configuração que determina o comportamento global do método ante as possíveis variações implementadas. Nessa instância do método, o EMHOSIR pode, por exemplo, trabalhar com diferentes algoritmos de otimização de hiperparâmetros, de acordo com a parametrização sobre ele realizada. Mais detalhes sobre as capacidades de parametrização implementadas estão disponíveis na Subseção 4.2.1.

Uma vez parametrizado, na Fase 2 ocorre a otimização de hiperparâmetros. Esta fase se utiliza da mesma base de dados de eventos[†] utilizada para treinamento na subsequente Fase

[†]Na verdade apenas 90% do primeiro *fold* de treinamento da base de dados de eventos, mas isso é melhor detalhado na Subseção 4.2.2

3. Por meio dela é que são determinados os valores dos hiperparâmetros do classificador propriamente dito, constituído pela composição em *pipeline* de fluxo condicional de dois outros classificadores que, conforme já apresentado anteriormente, na implementação deste trabalho, são uma rede MLP e um modelo gerado pelo K-NN. Na instância que se implementou do EMHOSIR, apenas os hiperparâmetros da rede MLP são otimizados. Os detalhes da Fase 2, conforme implementação realizada, estão melhor descritos na Subseção 4.2.2.

A Fase 3 instancia os modelos que compõe a abordagem de combinação de classificadores e em seguida realiza o treinamento do modelo com hiperparâmetros otimizados e o autoajuste dos valores de fronteira da faixa intermediária. Ao fim de sua execução, gera como artefato um modelo final de classificador combinado por meio de um *pipeline* de fluxo condicional entre o primeiro classificador $C1$ e o segundo classificador $C2$ implementados ($E^{P_h}_{P_l}$*(C1, C2)*). Também nessa fase, são extraídas métricas de desempenho do modelo final. Na implementação realizada, as métricas foram obtidas por meio de validação cruzada de k *folds*. O modelo gerado é semelhante ao proposto por Souza (2018) em seu trabalho, mas com uma hiperparametrização potencialmente diferente (incluídos os limites da faixa intermediária). A Subseção 4.2.3 apresenta melhor os detalhes de como essa fase fora implementada.

As Subseções subsequentes tratam de cada uma das fases a pouco apresentadas em um maior nível de aprofundamento.

4.2.1 Fase 1 do EMHOSIR

Apesar da capacidade de parametrização estar prevista no EMHOSIR enquanto método na chamada Fase 1, não há qualquer prescrição a respeito de como fazê-lo ou de quais parâmetros devem ser considerados para a implementação. Neste trabalho, a instância dessa fase foi criada de modo a conferir ao método a capacidade de ser submetido a diversos experimentos distintos pelo simples ajuste de valores em um arquivo de configuração que contém a parametrização que se quer utilizar. Tendo em vista a quantidade de experimentos que se realizou ao longo do desenvolvimento do trabalho, o esforço no sentido de prover tal capacidade à instância do método implementada se mostrou extremamente válido.

A Fase 1 do EMHOSIR expõe um aspecto interessante e recorrente de métodos de otimização de hiperparâmetros que não pode deixar de ser mencionado. Apesar de, em sua essência, o EMHOSIR e outros métodos que otimizam parâmetros tomarem para si a responsabilidade de encontrar os melhores valores de hiperparâmetros para os modelos que geram, esses métodos também precisam ter seu comportamento previamente configurado por meio de uma (hiper)parametrização. Sendo assim, é comum que a utilização de tais métodos não signifique a completa extinção da tarefa de hiperparametrização, o que ocorre é a substituição da hiperparametrização dos modelos, frequentemente complexa, por uma outra "mais leve", referente ao método de otimização de hiperparâmetros utilizado, que simplifica o trabalho de seu utilizador quando em comparação com a tarefa original a qual ele substitui. Desse modo, por conta de

suas características, seria perfeitamente aceitável dizer que a Fase 1 do EMHOSIR consiste em uma hiperparametrização, a opção por chamá-la apenas de parametrização é um esforço para minimizar a sobrecarga do termo e melhor discriminar essa fase com relação ao que ocorre na fase subsequente.

A Fase 1 do EMHOSIR, conforme fora implementada, tem a capacidade de tratar diversos parâmetros. Os parâmetros tratados se dividem em Parâmetros Globais de Configuração (GCP de *Global Configuration Parameters*) e Parâmetros de Escopo do Espaço de Busca da Otimização (OSSSP de *Optimization Search Space Scope Parameters*), e compartilham um mesmo arquivo de configuração. Os GCP determinam o comportamento da implementação do EMHOSIR, do autoajuste de faixa intermediária e dos classificadores. Por sua vez, os OSSSP são os parâmetros que determinam quais hiperparâmetros da rede MLP serão otimizados e quais os possíveis valores que esses hiperparâmetros podem assumir. Os GCP implementados, bem como a descrição dos mesmos, são:

- ***debug***: Parâmetro booleano que permite gerar relatórios adicionais e aumenta a verbosidade do método quando em execução. Sua utilização costuma tornar a execução do método mais morosa em razão de cálculos adicionais que são realizados;
- ***experiment_suffix***: Os resultados dos testes efetuados são armazenados na forma de arquivos em diretórios específicos. Esse parâmetro é uma *String* que determina um sufixo para o diretório dos resultados;
- ***folds***: Na Fase 3 do EMHOSIR, uma etapa de validação cruzada de k *folds* (*k-fold cross-validation*), responsável pelo obtenção das métricas de desempenho que se tem interesse é realizada. Esse parâmetro determina o número de *folds* a serem utilizados na validação;
- ***k***: Número de vizinhos a se considerar na classificação do K-NN;
- ***ho_method***: Método de otimização de hiperparâmetros a ser utilizado na Fase 2 do EMHOSIR. Foram implementados três métodos de otimização de hiperparâmetros: Busca em Grade, Busca Aleatória e Otimização Bayesiana. Esse parâmetro permite escolher qual dessas técnicas utilizar;
- ***random_search_percent***: Percentual do espaço de hiperparâmetros a ser buscado no caso de se estar utilizando a técnica de otimização Busca Aleatória. Esse parâmetro não tem efeito para o caso das demais técnicas;
- ***metrics***: Lista de métricas de desempenho a serem utilizadas pelo EMHOSIR, tanto para a geração de relatórios de desempenho, quanto para otimização de hiperparâmetros e treinamento do modelo. Da forma como fora implementado, tanto métricas *built-in* dos módulos e bibliotecas utilizados podem ser usadas quanto implementações de métricas próprias/personalizadas. A otimização de hiperparâmetros é realizada tendo como referência a primeira métrica da lista de métricas;
- ***verbose***: Verbosidade do método EMHOSIR durante sua execução, com valores que podem ser definidos entre 0 e 2;

- ***restore_best_weights***: Valor booleano que determina se, ao fim do treinamento da rede MLP, os valores de seus pesos devem ser restaurados aos da época do treinamento em que obtiveram melhor desempenho em termos da métrica utilizada para otimização de hiperparâmetros;
- ***monitor***: A implementação do EMHOSIR realizada se utiliza da técnica de *Early Stopping*, que permite o treinamento da rede MLP ser encerrado sempre que não se observar melhorias significativas em épocas adicionais de treinamento. O parâmetro *monitor* especifica qual métrica é considerada para se determinar se há ou não melhoria no treinamento conforme as épocas avançam;
- ***patience***: Combinado com o parâmetro *monitor*, permite especificar quantas épocas de tolerância serão dadas à tentativa de melhoria das métricas antes do treinamento da rede MLP ser encerrado: caso o número de épocas determinado pelo valor de *patience* transcorra sem uma melhoria na métrica determinada pelo GCP *monitor*, o treinamento é finalizado;
- ***higher_percentile***: Apesar do EMHOSIR enquanto método prever a etapa de autoajuste de faixa intermediária, para a realização de testes, o parâmetro *higher_percentile* foi implementado para permitir que o valor de fronteira superior da faixa intermediária fosse fixado (h, na Notação 4.1). Quando o valor desse parâmetro não é fornecido, o autoajuste de faixa intermediária é realizado;
- ***lower_percentile***: Da mesma forma que o parâmetro *higher_percentile*, esse parâmetro permite que o valor de fronteira inferior da faixa intermediária seja fixado (l na Notação 4.1) para a realização de testes. Quando o valor desse parâmetro não é fornecido, o autoajuste de faixa intermediária é realizado [†];
- ***model2_full_set_training***: Na implementação realizada, em que se tem o segundo classificador $C2$ fixo como sendo um modelo gerado pelo K-NN, a base de dados de eventos utilizada para o treino desse pode tanto ser a base completa de treinamento, quanto apenas os exemplos que, durante o treino, tiveram os valores de saída da classificação realizada pela rede MLP dentro do intervalo da faixa intermediária. Quando o valor desse parâmetro booleano é configurado como verdadeiro, utiliza-se a base de dados de eventos de treinamento completa para a criação do modelo do K-NN. É um parâmetro que tem efeito apenas quando os valores de *higher_percentile* e *lower_percentile* estão fixos, situação em que o autoajuste de faixa intermediária não ocorre; a simples necessidade de realização do autoajuste de faixa intermediária obriga a utilização do K-NN com a base de treinamento completa (valor do parâmetro *model2_full_set_training* é forçado a ser verdadeiro), uma vez que, por ainda serem indeterminados durante o treino, não é possível, *a priori*, saber se o valor de saída da classificação de um exemplo/instância realizada pela rede MLP está dentro das fronteiras da faixa intermediária;

[†]A implementação realizada neste trabalho exige que os GCP *higher_percentile* e *lower_percentile* estejam ambos definidos ou indefinidos.

- ***dataset_path***: Caminho na estrutura de diretórios em que se encontra a base de dados de eventos sobre a qual o método será executado;
- ***sampling***: Permite que, ao invés da base de dados de eventos completa, apenas uma amostra estratificada dessa base seja utilizada no método para a geração do modelo de combinação de classificadores final. O valor do parâmetro *sampling* pode tanto ser um número inteiro representando o número absoluto de exemplos a serem utilizados na amostra, quanto um número decimal inferior a 1, representando um percentual da base;
- ***epochs***: Número de épocas de treinamento da rede MLP;
- ***decay***: Especifica o valor de decaimento de pesos (*weight decay*) a ser utilizado na rede MLP.

Apresentados os GCP e sua funcionalidade na implementação realizada, faz-se ainda necessária a apresentação dos OSSSP. Os OSSSP influenciam diretamente na Fase 2 do EMHOSIR e sua natureza é totalmente diversa da natureza dos GCP previamente apresentados. Como a otimização de hiperparâmetros ocorre dentro de um espaço de busca extenso porém finito, é importante, antes de tudo, delimitar esse espaço (mais detalhes em 4.4), de modo a tornar possível sua execução em um tempo aceitável. Delimitar o espaço de busca de otimização significa especificar quais hiperparâmetros são passíveis de otimização e especificar quais valores são possíveis para cada um destes (determinar o escopo de cada hiperparâmetro). Na implementação do EMHOSIR realizada, são os OSSSP que se incumbem de tais tarefas. Nessa implementação, cada OSSSP está diretamente associado à um hiperparâmetro da rede MLP (relação um-para-um), de modo que apenas os hiperparâmetros para os quais há um OSSSP correspondente têm sua otimização possível de ser realizada. A atribuição de valores a um OSSSP é, na verdade, a especificação de limites para o hiperparâmetro relacionado.

A especificação de limites pode ser realizada tanto por arrolamento de valores quanto pela especificação de um intervalo de valores. No caso de uso de arrolamento, especifica-se um rol de valores, representando aqueles possíveis de serem assumidos pelo hiperparâmetro correspondente; no caso de intervalo, os limites inferior e superior dos valores que podem ser atribuídos ao hiperparâmetro relacionado, assim como o número de valores a serem considerados dentro do intervalo de tais limites, são especificados. A utilização de arrolamento de valores ou de intervalo depende exclusivamente do tipo do hiperparâmetro sobrejacente (se *String*, inteiro, etc.) uma vez que, em ambos os casos, os valores atribuídos restringem o espaço de busca da otimização para o hiperparâmetro em questão. O resultado que se tem é que apenas os valores (arrolados ou do intervalo) são considerados durante a otimização da Fase 2 do EMHOSIR. Também é possível especificar um valor fixo para quaisquer dos parâmetros da categoria OSSSP, que ao fazê-lo, utiliza no hiperparâmetro associado o valor especificado, sendo um fato que também reduz o espaço de busca da otimização. São parâmetros do tipo OSSSP da implementação realizada:

OSSSP por Arrolamento de Valores

- ***optimizer***: Função de otimização da rede MLP;
- ***loss_function***: Função de perda/custo da rede MLP;
- ***input_activation***: Função de ativação da camada de entrada da rede MLP;
- ***internal_activation***: Função de ativação das camadas internas/ocultas da rede MLP. Na implementação realizada, todas as camadas ocultas, independente do seu quantitativo, possuem a mesma função de ativação;
- ***output_activation***: Função de ativação da camada de saída da rede MLP;
- ***internal_units***: Quantidade de neurônios em cada uma das camadas internas/ocultas da rede MLP (todas as camadas ocultas contam com o mesmo número de neurônios definidos por meio desse OSSSP);
- ***n_hidden_layers***: Quantidade de camadas internas/ocultas da rede MLP.

OSSSP por Intervalo ou Arrolamento de Valores (ambos aceitos)

- ***dropout***: Valor de *dropout* da rede MLP. O mesmo *dropout* é aplicados a todas as camadas ocultas;
- ***learning_rate***: Valor de taxa de aprendizagem do otimizador da rede MLP.

A Seção seguinte apresenta a Fase 2 do EMHOSIR, em que ocorre a otimização de hiperparâmetros.

4.2.2 Fase 2 do EMHOSIR

A Fase 2 do EMHOSIR ocorre de forma subsequente à parametrização do método realizada na Fase 1. Nessa fase, o EMHOSIR realiza a otimização de hiperparâmetros conforme fora parametrizado pelos GCP e os OSSSP. Conforme apresentado na subseção anterior, as técnicas de otimização de hiperparâmetros implementadas são as seguintes:

- Busca em Grade (Hsu, Chang, Lin et al., 2003), abordada na Subseção 2.8.1;
- Busca Aleatória (Bergstra & Bengio, 2012), apresentada na Subseção 2.8.2;
- Otimização Bayesiana com Processos Gaussianos (Snoek, Larochelle & Adams, 2012), tratada em detalhes na Subseção 2.8.3.

É importante observar que, em termos de tempo de execução, a Fase 2 é prevista como a fase mais longa entre as fases do EMHOSIR enquanto método. Conforme visto na Seção

2.8, tal situação se deve ao fato de, durante essa fase, diversas configurações distintas de hiperparâmetros serem experimentadas, o que implica na realização de repetidos treinamentos dos classificadores que compõe a solução e cujos hiperparâmetros estão sendo otimizados. Técnicas de otimização de hiperparâmetros mais eficientes em termos de tempo, como a Otimização Bayesiana, podem, no entanto, minimizar o espaço de busca percorrido durante a otimização. Isso se traduz em uma menor quantidade de configurações de hiperparâmetros distintas testadas, o que implica em uma diminuição no tempo dispendido pela execução dessa fase do método.

O EMHOSIR enquanto método não prescreve quais hiperparâmetros devem ser otimizados nem quais dos classificadores que o constituem devem passar pela etapa de otimização. Na implementação do método realizada, apenas os hiperparâmetros do primeiro classificador (rede MLP), descritos na Subseção 4.2.1 por meio dos OSSSP, são passíveis de otimização. Uma vez que a técnica de otimização utilizada tenha sido capaz de obter um conjunto de hiperparâmetros otimizados, a informação referente aos valores destes é disponibilizada para a Fase 3, que, na implementação realizada, se utiliza das mesmas para instanciar a rede MLP conforme a otimização feita.

Conforme pode ser visto na Figura 4.4, além da validação final do modelo combinado gerado, realizada ao fim da Fase 3 (detalhamento na Subseção 4.2.3), há ainda, na implementação do EMHOSIR deste trabalho, uma validação interna para a otimização de hiperparâmetros dessa fase. Essa utiliza-se da técnica de validação cruzada de 10 *folds*, que visa determinar qual entre as 10 melhores configurações de hiperparâmetros encontradas após a otimização[†] possui o melhor desempenho em termos gerais. Algumas importantes decisões de projeto foram tomadas para se realizar tal validação. A primeira delas foi a realização da otimização de hiperparâmetros utilizando uma amostra estratificada de 81% da base de dados de eventos. O tamanho peculiar da amostra deve-se ao fato da mesma se tratar de 90% do conjunto de treinamento utilizado para o primeiro *fold* da validação que ocorre na Fase 3, os 10% restantes foram destinados à validação da otimização de hiperparâmetros, em sua própria validação cruzada.

A opção por fazer a otimização de hiperparâmetros da maneira descrita anteriormente deve-se ao fato do processo de otimização de hiperparâmetros ser, como já informado outras vezes neste texto, extremamente moroso. A segunda decisão de projeto importante nessa fase foi o uso da métrica *F1 Measure* como critério para se determinar o melhor entre os conjuntos de hiperparâmetros. A utilização da *F1 Measure* nesse momento independe da métrica utilizada para a otimização de hiperparâmetros propriamente dita (parametrizada via GCP *metrics*), sendo o critério final para se selecionar o melhor conjunto de hiperparâmetros entre os 10 de melhor desempenho. A escolha dessa métrica não-parametrizável em detrimento de qualquer outra se deve ao fato dela ponderar o desempenho de outras métricas importantes para a implementação realizada (especificidade e sensibilidade), minimizando a chance de se escolher um conjunto de hiperparâmetros que tenha bom desempenho apenas sobre um conjunto particular de dados ou sobre uma única métrica. Tal situação prioriza a seleção de configurações que

[†]Conforme definido via GCP *metrics* de acordo com o apresentado na Subseção 4.2.1

melhor generalizem para conjuntos de dados diversos.

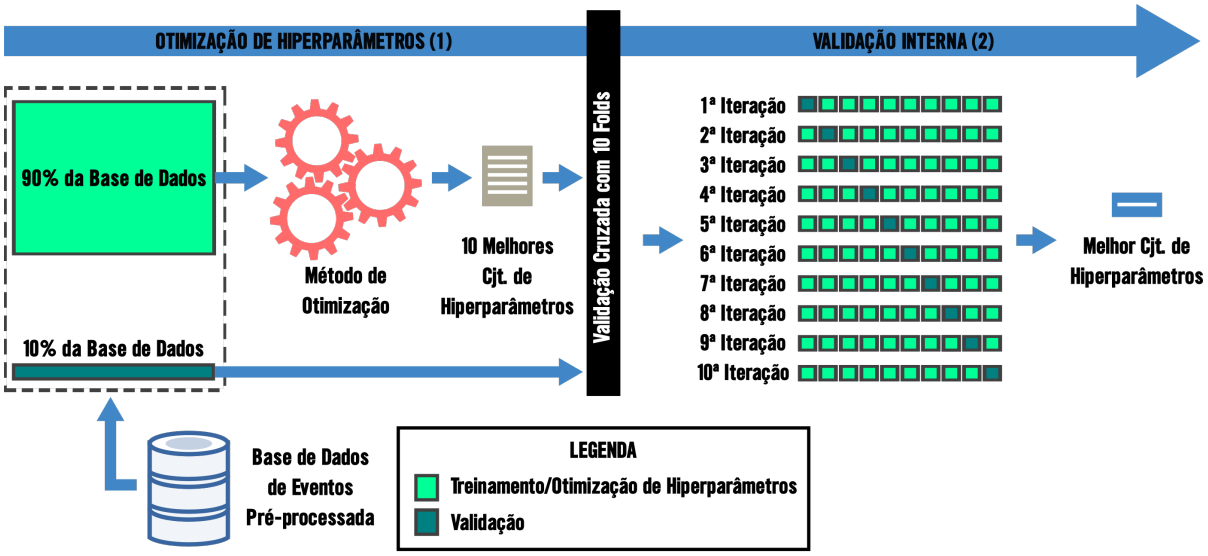

Figura 4.4: Detalhamento da Fase 2 da Implementação do EMHOSIR Realizada

Várias linguagens de programação contam com módulos e bibliotecas que implementam as técnicas de otimização incorporadas na instância do EMHOSIR deste trabalho. Na Seção 4.6, são apresentados os detalhes a respeito de quais dessas implementações foram utilizadas, bem como peculiaridades específicas da implementação do EMHOSIR feita, herdadas das referidas bibliotecas.

4.2.3 Fase 3 do EMHOSIR

Na Fase 3 do EMHOSIR ocorre o treinamento do classificador combinado, instanciado conforme a otimização de hiperparâmetros realizada na fase anterior. Nessa mesma fase, é realizada a etapa de autoajuste de faixa intermediária. Também é a ela inerente a extração das métricas de desempenho as quais se tem interesse. O entendimento de tais mecanismos se fazem indispensáveis à compreensão das contribuições do presente trabalho quando em comparação com a proposta de Souza (2018). Entre as três fases do método, é a fase em que se concentrou a maior parte das decisões de projeto que caracterizaram a implementação do EMHOSIR apresentada, sendo, portanto, a mais densa em termos de explicação a respeito do que fora feito.

Ao se pensar em treinamento de um modelo combinado de classificador $E_{P_l}^{P_h}(C1,\ C2)$, deve-se ter em mente que tal modelo, enquanto classificador, é uma composição entre um primeiro classificador $C1$ e um segundo classificador $C2$. Sendo assim, para que o classificador combinado seja considerado treinado, cada um dos classificadores que o compõe deve ser treinado individualmente, ainda que seu treinamento ocorra de forma paralela ou conjunta. Desse modo, considera-se "tempo de treinamento"os momentos posteriores ao início do treinamento de quaisquer dos classificadores e anteriores à finalização do treinamento de ambos. O treina-

mento em si, realizado nessa fase do EMHOSIR, não possui qualquer especificidade alheia ao mecanismo usual de treinamento dos classificadores individuais da composição.

Enquanto método abstrato, o EMHOSIR não prevê nessa fase uma técnica especifica para a realização do autoajuste de faixa intermediária. No entanto, para possibilitar o cálculo dos valores de fronteira da referida faixa, deve haver uma forma de, com base em uma hipótese de faixa intermediária especificada pelos valores de fronteiras de suas partes, estimar o desempenho do modelo combinado de classificador final a ser gerado em termos da métrica de interesse para a qual a otimização de hiperparâmetros fora realizada na Fase 2. Na implementação que se fez, a estimativa de tal desempenho se dá individualmente em termos das partes superior e inferior das saídas de $C1$, de acordo com o valor de fronteira considerado para cada uma das partes da faixa intermediária correspondente, por meio de uma função matemática e. Para cada parte das saídas de $C1$, a referida função é capaz de estimar de forma independente o valor da métrica de desempenho de otimização m[†] para um dado valor de fronteira em termos de percentil i (notação P_i representando o i-ésimo percentil), podendo i corresponder tanto a l quanto a h na Notação 4.1, de acordo com a parte das saídas de $C1$ para a qual a estimativa está sendo realizada. A estimativa realizada pela função e é, na verdade, o cálculo real do desempenho m do modelo combinado de classificador $E^{P_h}_{P_l}$*(C1, C2)* sobre o conjunto de treinamento para a parte das saídas de $C1$ e a faixa intermediária correspondente em questão, trata-se, no entanto, de uma estimativa em relação a esse mesmo desempenho no conjunto de validação.

Para a estimativa da métrica de desempenho considerada para otimização, a notação $\overline{m}_{P_i}$ é usada para se referir a parte superior das saídas de $C1$ enquanto que $\underline{m}_{P_i}$ diz respeito à mesma estimativa para a parte inferior. Além de depender dos valores de i, a função e implementada realiza a estimativa de m para as partes das saídas de $C1$ utilizando-se das classificações verdadeiras Y referentes ao conjunto de dados de treinamento na parte em questão e das saídas O^{C1} e O^{C2}, correspondentes à classificação de $C1$ e de $C2$ para esse conjunto, respectivamente. Nos casos em que a saída O^{C1} se refere especificamente à saída do classificador $C1$ para a parte superior das saídas de $C1$, a notação $\overline{O}^{C1}$ é adotada; de forma complementar, a notação $\underline{O}^{C1}$ cabe às saídas do classificador $C1$ para as partes inferiores das saídas de $C1$. Para as saídas O^{C2} do classificador $C2$ e para as classificações verdadeiras Y, sempre que necessário discriminar a qual das partes das saídas de $C1$ cada uma se refere, uma notação semelhante é utilizada: $\overline{O}^{C2}$ e $\underline{O}^{C2}$ sendo respectivamente as saídas do classificador $C2$ para as partes superior e inferior das saídas de $C1$, enquanto que $\overline{Y}$ e $\underline{Y}$ denotam as classificações verdadeiras na mesma ordem respectiva. A equação (4.1) e as demais que a compõe (de 4.2 à 4.5) representam matematicamente a referida função.

$$e(i, Y, O^{C1}, O^{C2}) = (1-p) \cdot \mathcal{M}(O^{C1}_{j \in J^{outer}}, Y_{j \in J^{outer}}) + p \cdot \mathcal{M}(O^{C2}_{j \in J^{inner}}, Y_{j \in J^{inner}}) \qquad (4.1)$$

[†]Na implementação realizada, a métrica m é a métrica considerada para a otimização de hiperparâmetros, parametrizada por meio do GCP *metrics* conforme apresentado na Subseção 4.2.1

$$p = 1 - \left(\frac{|J^{outer}|}{|J|}\right) \text{ ou, alternativamente, } p = \frac{|J^{inner}|}{|J|} \tag{4.2}$$

$$J = \{j \in \mathbb{N}^* / j \leq |Y|\} \tag{4.3}$$

$$J^{outer} = J - J^{inner} \tag{4.4}$$

$$n = \frac{|O^{C1}| \cdot i}{100} = k + f, \quad k \in \mathbb{N}^*, \quad f \in \mathbb{R} / 0 \leq f < 1 \tag{4.5}$$

Nas equações (4.6a) e (4.6b), tem-se $i = h \therefore O^{C1} = \overline{O}^{C1}$ (parte superior da faixa intermediária):

$$P_i = P_h = \begin{cases} \overline{O}_k^{C1} & \text{se } n \in \mathbb{N}^* \therefore k = n, f = 0 \\ (1-f) \cdot \overline{O}_k^{C1} + f \cdot \overline{O}_{k+1}^{C1} & \text{se } n \notin \mathbb{N}^* \therefore 0 < f < 1 \end{cases} \tag{4.6a}$$

$$J^{inner} = \{j \in J / \overline{O}_j^{C1} \leq P_h\} \tag{4.6b}$$

Nas equações (4.7a) e (4.7b), tem-se $i = l \therefore O^{C1} = \underline{O}^{C1}$ (parte inferior da faixa intermediária); de forma análoga, para essas equações, tem-se $Y = \underline{Y}$, de modo que $|\underline{O}^{C2}| = |\underline{Y}|$ e $|\underline{O}^{C1}| = |\underline{Y}|$:

$$P_i = P_l = \begin{cases} \underline{O}_{|Y|-k}^{C1} & \text{se } n \in \mathbb{N}^* \therefore k = n, f = 0 \\ f \cdot \underline{O}_{(|Y|-1)-k}^{C1} + (1-f) \cdot \underline{O}_{|Y|-k}^{C1} & \text{se } n \notin \mathbb{N}^* \therefore 0 < f < 1 \end{cases} \tag{4.7a}$$

$$J^{inner} = \{j \in J / \underline{O}_j^{C1} \geq P_l\} \tag{4.7b}$$

O fato da função e se valer tanto de O^{C1} quanto de O^{C2} para estimar $\overline{m}_{P_i}$ e $\underline{m}_{P_i}$ tipifica a necessidade dos dois modelos $C1$ e $C2$ estarem previamente treinados. A despeito de sua aparente complexidade, a função e nada mais é do que uma média ponderada do desempenho em termos da métrica m dos classificadores $C1$ e $C2$ no conjunto de treinamento, a ponderação se dá pela proporção de exemplos/instâncias destinadas a cada classificador em relação ao desempenho individual obtido por cada um deles. O que a função e faz, na verdade, é calcular a contribuição dos classificadores $C1$ e $C2$ ao valor da métrica m final do classificador combinado do EMHOSIR, no conjunto de treinamento para a parte da faixa intermediária cujo valor de fronteira é o valor P_i. O valor de p, calculado conforme equação (4.2), é o percentual de instâncias a serem enviadas à $C2$ por se encontrarem dentro da faixa intermediária na parte considerada tendo P_i como valor de fronteira da referida parte. Por fim, $\mathcal{M}$ é a função capaz de calcular o valor da métrica m, dado um conjunto de saídas e as classificações verdadeiras correspondentes. No caso da equação (4.1), no primeiro termo, a função $\mathcal{M}$ é parametrizada pela porção das saídas O^{C1} fora da parte considerada da faixa intermediária ($O_{j \in J^{outer}}^{C1}$) bem como pelas classificações verdadeiras correspondentes a essa porção ($Y_{j \in J^{outer}}$). No segundo termo, a função $\mathcal{M}$ é parametrizada pela porção das saídas O^{C2} dentro da faixa intermediária ($O_{j \in J^{outer}}^{C2}$) bem como pelas classificações verdadeiras correspondentes a essa porção ($Y_{j \in J^{outer}}$).

Para o cálculo de $\overline{m}_{P_i}$, as equações (4.6a) e (4.6b) são utilizadas com o intuito de se de-

terminar J^{inner} que, em razão dos valores de h ($i = h$), são os índices das instâncias dentro da parte superior da faixa intermediária; com relação ao cálculo de $\underline{m}_{P_i}$, referente à parte inferior das saídas de $C1$, as equações (4.7a) e (4.7b) é que são utilizadas no lugar das equações (4.6a) e (4.6b) respectivamente. Por meio de J^{inner}, é possível obter J^{outer} (equação (4.4)), correspondente aos índices de instâncias fora da faixa intermediária na parte considerada, sua obtenção se dá pelo complemento de J^{inner} em relação ao conjunto total de índices J das instâncias (equação (4.3)). O cálculo do valor de fronteira das partes da faixa intermediária se utiliza do percentil i. Por meio de i, é possível identificar o valor limítrofe P_i (equações (4.6a) ou (4.7a), conforme a parte da faixa intermediária sendo calculada), responsável por determinar quais instâncias estão dentro da faixa intermediária da parte em questão. Sempre que n, obtido pela equação (4.5) e necessário para o cálculo de P_i, puder ser decomposto em um valor k inteiro não-nulo e um valor f decimal, P_i será obtido por interpolação, conforme se vê nas equações (4.6a) e (4.7a). Nos demais casos, conforme as mesmas equações, P_i será meramente o k-ésimo elemento de $\overline{O}^{C1}$ para a parte superior das saídas de $C1$ e o elemento da posição $|Y| - k$ de $\underline{O}^{C1}$ para a parte inferior das saídas de $C1$.

Dado um valor de hipótese i (referente ao percentil P_i) para a fronteira da parte da faixa intermediária considerada, a função e (equação (4.1)) permite estimar o valor da métrica m para aquela partes das saídas de $C1$. O EMHOSIR enquanto método prevê a existência de um mecanismo que permita realizar o autoajuste da faixa intermediária de modo a maximizar o desempenho do modelo final a ser gerado. Na implementação do EMHOSIR realizada, a solução dada para o problema em questão foi a definição de uma função objetivo z a partir de e que, ao ser minimizada, maximiza o desempenho previsto para m do modelo combinado de classificador $E^{P_h}_{P_l}$***(C1, C2)*** sendo gerado em termos dos limites h e l da faixa intermediária. A forma geral dessa função objetivo z pode ser vista conforme apresentado na equação (4.8):

$$z(h, l, \overline{\underline{m}}_{C1}, \overline{\underline{Y}}, \overline{\underline{O}}^{C1}, \overline{\underline{O}}^{C2}) = \sum_{q=1}^{s} w_q \cdot t_q \tag{4.8}$$

Na equação 4.8, ao invés de cálculos por partes das saídas de $C1$, o cálculo se dá em termos da saída integral de $C1$, composta pela parte superior e inferior da mesma. Desse modo, além dos percentis de fronteira h e l da faixa intermediária, também parametrizam a função objetivo z o valor $\overline{\underline{m}}_{C1}$ referente ao desempenho obtido pelo classificador $C1$, em termos da métrica m no conjunto de treinamento em ambas as partes das saídas de $C1$. O conjunto de classificações verdadeiras $\overline{\underline{Y}}$ também referente a ambas as partes ($\overline{\underline{Y}} = \overline{Y} + \underline{Y}$) e as saídas dos classificadores $C1$ e $C2$ para as mesmas, correspondentes a notação $\overline{\underline{O}}^{C1}$ e $\overline{\underline{O}}^{C2}$, respectivamente, em que $\overline{\underline{O}}^{C1} = \overline{O}^{C1} + \underline{O}^{C1}$ e $\overline{\underline{O}}^{C2} = \overline{O}^{C2} + \underline{O}^{C2}$. Tendo sido definida a função objetivo z em termos gerais, foram posteriormente definidos os s pesos w e critérios t, esses últimos em razão tanto da função e quanto dos parâmetros da função objetivo z. A definição que se fez pode ser vista nas equações (4.9), (4.11), (4.10) e (4.12).

$$W = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, T = \begin{bmatrix} t_1 & t_2 \end{bmatrix} \tag{4.9}$$

$$t_1 = \frac{h + l - \beta_1}{\beta_2 - \beta_1} \tag{4.10}$$

$$t_2 = 1 - \left[\frac{\left(\frac{|\overline{Y}|}{|\underline{Y}|}\right) \cdot e(h, \overline{Y}, \overline{O}^{C1}, \overline{O}^{C2}) + \left(\frac{|\underline{Y}|}{|\overline{Y}|}\right) \cdot e(l, \underline{Y}, \underline{O}^{C1}, \underline{O}^{C2}) - \overline{\underline{m}}_{C1}}{1 - \overline{\underline{m}}_{C1}} \right] \tag{4.11}$$

$$z(h, l, \overline{\underline{m}}_{C1}, \overline{\underline{Y}}, \overline{\underline{O}}^{C1}, \overline{\underline{O}}^{C2}) = \sum_{q=1}^{2} w_q \cdot t_q = W \cdot T \tag{4.12}$$

A matriz de critérios T formada pelos critérios t_1 e t_2 (equação (4.9)) foi definida de tal modo que o valor da função objetivo z sofresse influência tanto do tamanho da faixa intermediária delimitada por h e l quanto da melhoria que essa faixa intermediária proporciona em termos da métrica m na classificação final quando em comparação com o desempenho $\overline{\underline{m}}_{C1}$ referente ao classificador $C1$ também em termos da métrica m para a saída integral de $C1$ (parte superior e inferior). Cada um dos critérios está diretamente associado a uma das influências que se quis imprimir sobre a função objetivo z.

Por sua vez, a matriz de pesos W, com os pesos w_1 e w_2 foi definida de tal modo que, a cada um dos critérios de T, fosse atribuído um peso respectivo. O critério t_1 corresponde ao valor normalizado do tamanho da faixa intermediária definido por h e l em relação aos limites mínimo e máximo definidos respectivamente por β_1 e β_2. O critério t_2 é formado por dois termos em que o primeiro é subtraído do segundo. Posterga-se, por agora, a discussão a respeito do primeiro termo, cuja a análise ocorrerá em sequência, de modo mais oportuno. O segundo termo, mais interessante e entre colchetes, diz respeito a melhoria da métrica m na classificação final normalizada em relação à $\overline{\underline{m}}_{C1}$, obtida com o uso da faixa intermediária com limites h e l. A subtração entre os dois termos que compõe t_2 tem por objetivo fazer com que o valor de t_2 seja menor conforme a faixa intermediária se mostre mais apta a melhorar a métrica m do classificador final a ser gerado em relação à métrica $\overline{\underline{m}}_{C1}$. Oportunamente, vale dizer que o primeiro termo, correspondente ao valor escalar 1, não seria estritamente necessário, mas auxilia na percepção da melhoria da métrica m que faz com que t_2 valha 0 no caso da métrica m ter atingido seu máximo valor. É importante observar que o critério t_2 faz uso da função e definida pela equação (4.1) aplicada a cada uma das partes das saídas de $C1$, em uma soma que pondera a quantidade de instâncias pertencentes a cada uma dessas partes.

Os valores dos pesos w_1 e w_2 foram definidos como sendo 1 e 3, respectivamente. A definição arbitraria desses valores foi feita de modo que, sobre o valor da função objetivo z, o impacto da melhoria da métrica m no modelo $E^{P_h}_{P_l}$*(C1, C2)* sendo gerado pela faixa intermediária definida por h e l (critério t_2) fosse maior que impacto do tamanho da faixa intermediária (critério t_1). Tendo em vista que valores menores para a função objetivo z são preferíveis, a

definição da matriz W conforme feito faz com que a função objetivo z expresse predileção por faixas intermediárias menores desde que as mesmas gerem modelos $E^{P_h}_{P_l}$*(C1, C2)* de maior desempenho em relação a métrica m.

Com relação aos limites mínimo β_1 e máximo β_2 definidos para o tamanho da faixa intermediária, também se utilizou, de forma arbitrária, os valores 2 e 124 respectivamente. O valor para β_1 corresponde a uma faixa intermediária mínima em que cada uma das partes (superior e inferior) utiliza o percentil 1 ($\frac{\beta_1}{2}$ no domínio de h e l). O valor para β_2 foi definido a partir do tamanho de faixa intermediária que obteve melhor desempenho da métrica acurácia nos experimentos realizados por Souza (2018) em seu trabalho.

Considerando $z'(h,l) = z(h,l,\overline{m}_{C1},\overline{Y},\overline{O}^{C1},\overline{O}^{C2})$, o problema do autoajuste da faixa intermediária, previsto para a Fase 3 do EMHOSIR, foi transformado em um problema de minimização da função objetivo $z'(h,l)$, conforme definido a seguir:

$$\begin{aligned} \text{Minimizar:} \quad & z'(h,l) \\ \text{Sujeito a:} \quad & h + l \leq \beta_2 \\ & h, l \in [\frac{\beta_1}{2}, 99] \end{aligned}$$

Ao problema de minimização ora definido, foi aplicado o método de Evolução Diferencial conforme apresentado na Seção 2.9*. A modelagem realizada do problema e a aplicação da Evolução Diferencial para a Fase 3 do EMHOSIR são capazes de implementar o autoajuste da faixa intermediária previsto pelo método, minimizando a quantidade de exemplos a serem classificados por $C2$ (menos performático em termos de tempo do que $C1$) ao passo que maximizam a métrica m considerada. Na implementação do EMHOSIR realizada, ao fim desta fase, o que se tem é um modelo $E^{P_h}_{P_l}$*(MLP, K-NN)* com h e l autoajustados.

Como previamente mencionado, na condição de método, o EMHOSIR prevê em sua terceira fase a avaliação do modelo $E^{P_h}_{P_l}$*(C1, C2)* gerado. A implementação realizada instanciou a referida avaliação utilizando validação cruzada de k *folds*, com valor de $k = 10$†. A partir dos *folds* estratificados, são extraídas as métricas de desempenho as quais se tem interesse. Dito isso, é importante destacar que todos os mecanismos descritos para a atual fase do EMHOSIR são realizados *fold* à *fold* de modo que para cada um dos *folds* são definidos os limites de faixa intermediária e obtidas as métricas de desempenho que se tem interesse. Em termos práticos, na implementação realizada do EMHOSIR, o que se tem ao final da Fase 3 são, na verdade, 10 modelos $E^{P_{h_f}}_{P_{l_f}}$*(MLP, K-NN)*, com $f = \{1, 2, ..., 10\}$, representando os vários valores de h e l, um por *fold*. A Figura 4.5 sumariza a Fase 3 do EMHOSIR. Por meio da mesma, são sumarizados

*A inclusão de restrições à minimização é facilmente incorporada na Evolução Diferencial, definindo o escopo dos parâmetros e impondo penalizações aos indivíduos que quebram as restrições.

†O número de *folds* pode ser alterado via GCP *folds* (conforme visto na Subseção 4.2.1), mas na implementação que ser fez, esse número foi fixado em 10.

todos os processos a essa fase inerentes. Como pode ser visto na referida figura, a validação ocorre utilizando a base de dados de eventos completa a partir do classificador $C1$ otimizado e do classificador $C2$ também definido a partir da totalidade da base de dados de eventos.

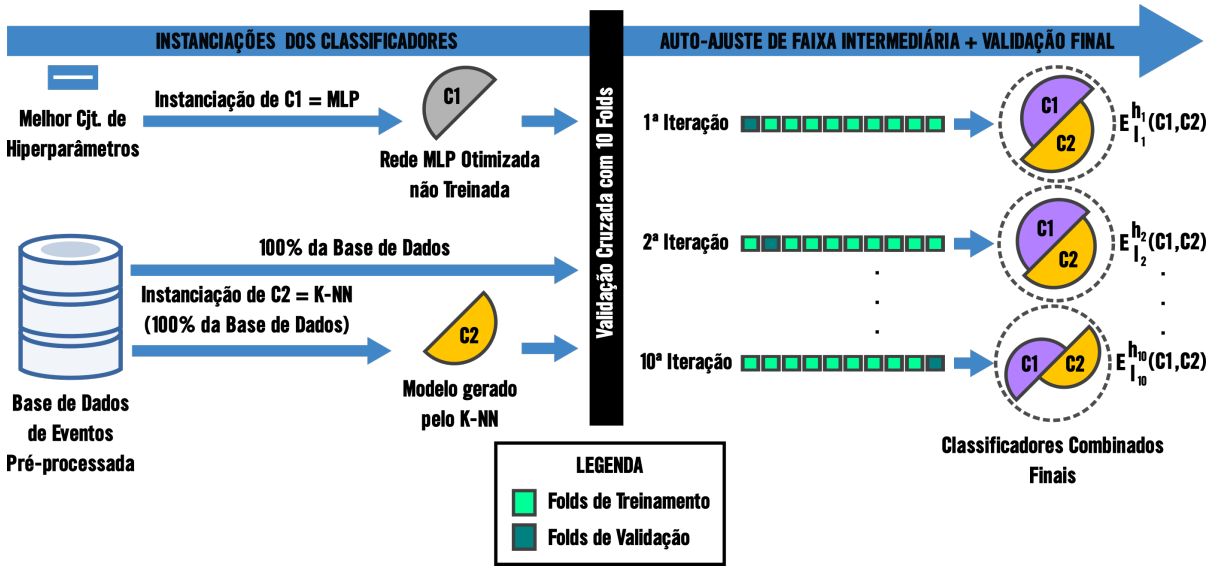

Figura 4.5: Detalhamento da Fase 3 da Implementação do EMHOSIR Realizada

Para todos os *folds*, o classificador $C1 = MLP$ conta com uma mesma hiperparametrização: aquela obtida por meio da Fase 2 com uso da técnica de otimização especificada via GCP *ho_method* (Subseção 4.2.1). Em se tratando de redes neurais, além dos pesos, obviamente diferentes, a diferença entre os 10 modelos é que, para cada um, de acordo com o *fold* que o originou, há um par de valores para h e l, correspondente aos limites das partes da faixa intermediária superior e inferior daquele *fold*. Obtidos pelo autoajuste de faixa intermediária via Evolução Diferencial, esses valores de fronteira, como o próprio nome sugere, serão únicos para cada *fold* e cada execução, uma vez que, conforme a etapa de treinamento, se ajustam aos dados de entrada sendo classificados. Obtidos ao fim do processo da fase corrente, os relatórios com as métricas de desempenho consideradas associados aos *folds* permitem estimar o desempenho de modelos gerados pela implementação do EMHOSIR, realizada por meio de média aritmética simples do desempenho do modelo de cada *fold*.

Compreendidas as especificidades da Fase 3 do EMHOSIR, faz-se necessária uma melhor compreensão a respeito das atividades de pré-processamento realizadas na base de dados de eventos NSL-KDD e das duas bases pré-processadas dela derivadas, utilizadas em todos os experimentos relatados neste trabalho. A apresentação desse assunto é tratada na seção subsequente.

4.3 Pré-processamento da Base NSL-KDD

Não há como compreender o EMHOSIR em sua totalidade sem conhecer as bases de dados de eventos para as quais o método fora concebido e implementado. Tendo em vista a prévia apresentação já realizada da base de dados de eventos NSL-KDD na Seção 2.3, a presente seção se ocupa em detalhar as tarefas de pré-processamento realizadas sobre a mesma, de modo que, sejam conhecidas as novas bases de dados de eventos resultantes do pré-processamento em questão. As referidas bases foram alvo das tarefas empíricas descritas neste trabalho.

É válido lembrar que, conforme apresentado na Seção 2.3, a base de dados de eventos NSL-KDD é distribuída à comunidade científica de modo a ser utilizada em validações do tipo *holdout*, com a prévia separação entre conjunto de treinamento e testes. No entanto, o uso de validação cruzada de k *folds*, feito na implementação realizada por esse trabalho, dispensa a prévia separação de um conjunto de testes como no caso da técnica de *holdout* que, apesar de poderosa, tende a superestimar o desempenho dos classificadores. Sendo assim, da mesma forma que no trabalho de Souza (2018), foi utilizado neste trabalho apenas a parte da NSL-KDD correspondente a parte de treinamento. A parte da NSL-KDD disponibilizada para testes composta por 22.544 instâncias fora descartada. Conforme visto na Seção 2.3, a base de dados NSL-KDD possui 40 classes correspondentes a tipos de conexões (intrusivas e não intrusivas) das quais 17 estão presentes exclusivamente na porção da base de dados destinada a testes, descartá-la fez com que os valores distintos de classes fossem reduzidos para 23.

Á porção da base de dados NSL-KDD selecionada para a implementação deste trabalho, correspondente exclusivamente ao conjunto de treinamento previamente existente foi atribuído o nome de NSL-KDD restrita; em contrapartida, à base de dados NSL-KDD composta pela combinação de conjunto de treinamento e testes em uma única base de dados foi dado o nome de NSL-KDD estendida. A opção por uso da NSL-KDD restrita em detrimento da NSL-KDD estendida se deu para que se pudesse realizar uma comparação mais coerente entre os resultados obtidos neste trabalho e no trabalho de Souza (2018). Deste ponto em diante do texto, a não ser que enunciado de forma explícita o oposto, toda menção à base de dados NSL-KDD diz respeito à sua vertente restrita.

No arcabouço da ciência de dados, muitas são as técnicas de pré-processamento disponíveis para aplicação sobre bases de dados. As referidas técnicas servem aos mais variados propósitos, entre os quais se destacam a melhoria da precisão de classificadores a elas aplicados e a redução dos custos de processamento dessas bases em tarefas que, de alguma maneira, as consomem. É válido lembrar que, conforme apresentado na Seção 2.3, a base de dados de eventos NSL-KDD é formada por registros selecionados da base de dados KDD-99, situação que afasta a necessidade da realização de pré-processamentos relativos à limpeza e garantia da qualidade dos dados (remoção de ruídos, remoção de duplicados e etc.). Sendo assim, neste trabalho os pré-processamentos realizados limitaram-se à processos para melhoria do desempe-

nho dos classificadores utilizados e para compatibilização da base de dados NSL-KDD a esses classificadores.

Antes de detalhar os pré-processamentos realizados sobre a NSL-KDD no escopo deste trabalho, é importante retomar o que fora, nesse mesmo sentido, realizado sobre a base de dados NSL-KDD no trabalho de Souza (2018) para efeito de comparação. No referido trabalho, por meio do agrupamento das 22 classes correspondentes a tipos de ataque em um rótulo único e antagônico à classe de conexões normais/não-intrusivas, o pré-processamento serviu ao propósito de transformar o problema original de múltiplas classes em um problema de classificação binária. Além desse importante pré-processamento, Souza (2018) submeteu a base de dados de eventos NSL-KDD à vários outros pré-processamentos, com destaque para a seleção de atributos. A partir dela, Souza (2018) gerou diversas bases de dados derivadas, cada uma com um subconjunto dos atributos originais da NSL-KDD. Seu método fora aplicado a cada uma dessas bases derivadas. Como pré-processamento comum a todas as bases de dados geradas, além da binarização do problema, houve, sempre que possível, a transformação dos atributos nominais discretos em atributos inteiros.

No entanto, a transformação aplicada não se deu de forma arbitrária: para todos os atributos nominais discretos e cada ocorrência de valor, o referido valor fora substituído por um valor inteiro sem qualquer critério que justificasse cada substituição. Aplicado dessa maneira, apesar de compatibilizar a base de dados de eventos NSL-KDD à utilização de uma rede MLP (entradas numéricas são pré-requisito para uso da mesma), o pré-processamento pode atenuar a capacidade de classificadores aplicados à base resultante obterem os melhores resultados possíveis em termos das métricas que se tiver interesse. Isso ocorre pelo fato da substituição feita a esmo, impor uma ordem de grandeza não natural e inexistente entre os valores dos atributos pré-processados, situação que induz os classificadores a identificarem proporcionalidades, desproporcionalidades e demais correlações entre os dados, todas artificiais, decorrentes exclusivamente da transformação arbitrária.

Conforme apresentado na Seção 2.5, as redes MLP, utilizadas como primeiro classificador[†] tanto no trabalho de Souza (2018) quanto neste trabalho, aceitam exclusivamente entradas de dados numéricas. Desse modo, tendo em vista a presença de atributos nominais discretos na base de dados NSL-KDD, da mesma forma que no trabalho de Souza (2018), fizeram-se necessarios, para este trabalho, pré-processamentos de compatibilização da base de dados com a redes MLP. Para esse fim, optou-se, no entanto, pela aplicação de uma técnica que minimiza os problemas já relatados no pré-processamento realizado por Souza (2018), chamada de binarização de atributos. Na binarização de atributos, cada valor possível de cada atributo nominal discreto se torna um novo atributo; o valor desse novo atributo é 1 apenas nas instâncias em que o valor que ele representa ocorria originalmente, sendo 0 nos demais casos. Desse modo, cada atributo f ao ser binarizado, é substituído por d novos atributos correspondentes à quantidade de valores distintos que haviam em f antes do pré-processamento. Sendo assim, no caso de

[†]Classificador $C1$ do modelo combinado $E^{P_h}_{P_l}$*(C1, C2)* gerado em ambos os trabalhos

haver n atributos nominais discretos, serão gerados $\prod_{i=1}^{n} d_i$ atributos no lugar dos mesmos.

Na base de dados de eventos NSL-KDD, por exemplo, o atributo *protocol_type* pode ter os valores *icmp*, *tcp* ou *udp* ($d = 3$) o que gera, pela binarização, três atributos no lugar do original. Nas instâncias em que *protocol_type* for igual à *icmp*, o atributo gerado correspondente terá o valor 1, enquanto que os atributos gerados correspondentes à *tcp* e *udp* serão 0. Como se pode supor, dependendo das características da base de dados original, a binarização de atributos pode aumentar consideravelmente a dimensão dos dados de entrada. No caso específico da base de dados NSL-KDD, os 42 atributos originais, que incluem a classe, resultaram em 123 atributos após a binarização de atributos. O aumento na dimensão dos dados de entrada implica em um aumento proporcional na dimensão da rede MLP em termos do número de neurônios em suas camadas, situação que torna o treinamento da mesma mais moroso quando em comparação com uma rede MLP menor.

Por fim, tendo em vista a grande diferença nas escalas de valores de atributos da NSL-KDD original, realizou-se também um pré-processamento referente à normalização dos dados. Após a normalização, todos os atributos foram mantidos em uma escala comum, variando entre 0 e 1. A normalização mitiga a ocorrência de situações em que, em razão de sua escala, atributos possam ser interpretados pelos classificadores como sendo mais ou menos influentes para a obtenção do resultado da classificação, fato que pode prejudicar o desempenho do classificador em termos gerais.

Conforme já visto nas Seções 4.2.1, 4.2.2 e 4.2.3, o EMHOSIR não prevê qualquer etapa de pré-processamento nas fases que o compõe, de modo que todo o pré-processamento necessário deve ser realizado antes da aplicação do método. Na implementação do EMHOSIR deste trabalho, ao fim da aplicação das tarefas de pré-processamento realizadas sobre a base de dados de eventos NSL-KDD, foram geradas duas novas bases de dados de eventos derivadas para aplicação do EMHOSIR propriamente dita. Parte dos pré-processamentos realizados foi comum a ambas, de modo que a diferença mais marcante de uma base de dados para a outra está no pré-processamento referente à compatibilização delas com a rede MLP. Em comum, ambas foram alvo do agrupamento de rótulos de classes para transformação do problema original em um problema binário (binarização do problema).

No entanto, uma das base de dados resultantes, chamada de *NOM_TO_INT*, fora gerada pela transformação dos atributos nominais discretos em atributos inteiros, da mesma forma que descrito para Souza (2018), sendo idêntica à base de dados utilizada no trabalho em questão. Por sua vez, na base de dados resultante de nome *NOM_TO_BIN*, optou-se por aplicar, além da binarização de atributos para o pré-processamento de compatibilização da mesma com a rede MLP, a normalização desses atributos[†]. Apesar das preocupações já expostas a respeito da transformação de atributos nominais discretos em atributos inteiros, a justificativa para manutenção e uso da base de dados *NOM_TO_INT* está no potencial inerente a sua utilização no que

[†]Na base de dados *NOM_TO_BIN*, o atributo *num_outbound_cmds* também fora removido por ser igual a 0 em todas as instâncias

diz respeito a comparação entre os resultados finais obtidos por este trabalho e pelo trabalho de Souza (2018). A Tabela 4.2 sumariza os pré-processamentos realizados sobre a NSL-KDD no presente trabalho, comparando as bases de dados *NOM_TO_INT* e *NOM_TO_BIN*. Um excerto da base de dados NSL-KDD junto aos pré-processamentos realizados sobre a mesma na geração das bases de dados pré-processadas pode ser visto na Figura 4.6. Na referida figura, as reticências indicam os demais atributos não exibidos que podem ser vistos na íntegra no Apêndice A. Eventualmente, mesmo nos atributos não exibidos, dependendo de seu tipo e da base de dados pré-processada em questão, um pré-processamento correspondente é aplicado.

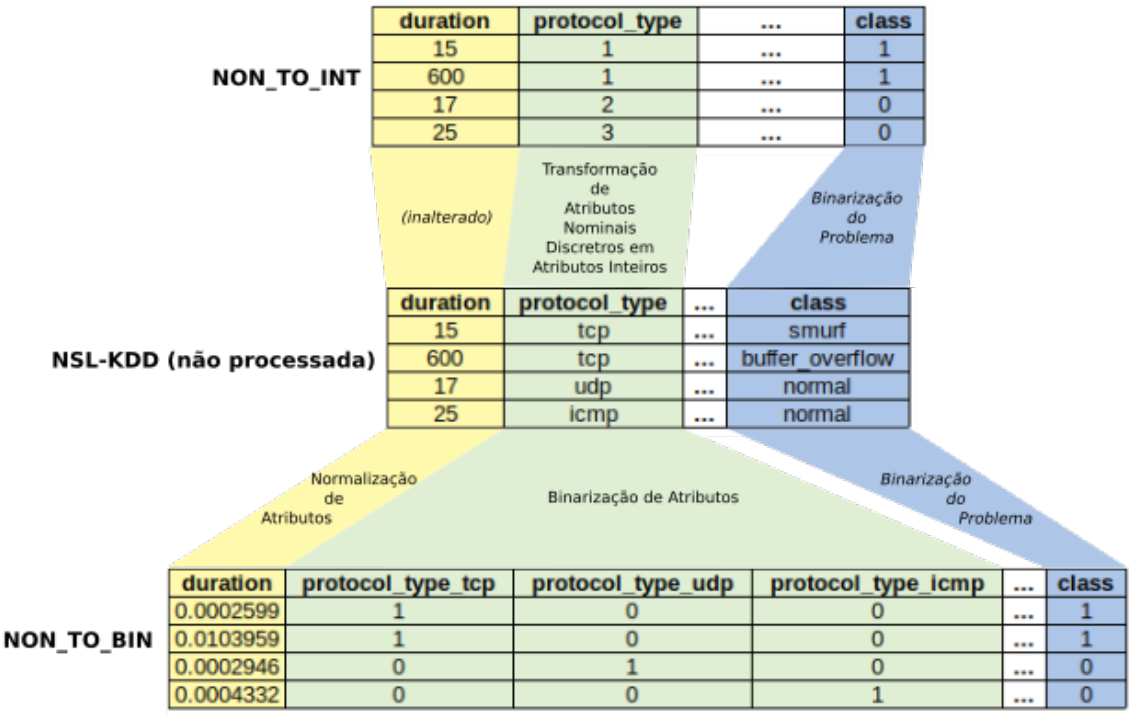

Figura 4.6: Pré-processamentos Realizados sobre Excertos da NSL-KDD

Tabela 4.2: Sumário de Pré-processamentos Aplicados às Bases de Dados *NOM_TO_INT* e *NOM_TO_BIN*

Pré-processamento Aplicado	*NOM_TO_INT*	*NOM_TO_BIN*
Agrupamento de Rótulos (Binarização do Problema)	X	X
Transformação de Atributos Nominais Discretos em Atributos Inteiros	X	
Binarização de Atributos		X
Normalização de Atributos		X

Uma vez conhecidas as atividades de pré-processamento realizadas sobre a base de dados de eventos NSL-KDD, bem como as novas bases de dados de eventos derivadas resultantes de tais pré-processamentos, a próxima seção trata de abordar a redução de dimensionalidade do espaço de busca de otimização. A importância desse assunto se deve ao fato de reduções dessa natureza terem impacto direto no desempenho do EMHOSIR em termos de tempo, principalmente para a Fase 2 em razão da otimização de hiperparâmetros.

4.4 Redução de Dimensionalidade do Espaço de Busca de Otimização

Independentemente do método de otimização utilizado, a etapa de otimização de hiperparâmetros do EMHOSIR tem seu tempo de execução proporcional ao espaço de busca em que a mesma é realizada. Longe de ser desprezível, esse tempo é o que torna a Fase 2 do EMHOSIR a mais longa entre as três fases do método. Faz, portanto, sentido, a busca por mecanismos que permitam tornar a execução dessa fase mais rápida, valorizando a exploração de espaços de busca mais promissores em detrimento de espaços de busca que se mostram menos interessantes em termos de capacidade de maximização das métricas que se tem interesse. Conforme visto na Subseção 4.2.1, os OSSSP são os parâmetros que, ao serem configurados, estabelecem a dimensão do espaço de busca da otimização de hiperparâmetros, sendo que, na implementação do EMHOSIR realizada, cada OSSSP está diretamente associado à um hiperparâmetro da rede MLP sobre a qual a otimização de hiperparâmetros ocorre. A presente seção apresenta a metodologia utilizada neste trabalho para reduzir a dimensionalidade em que a otimização de hiperparâmetros fora realizada, direcionando a atribuição de valores aos OSSSP. Trata-se de um método analítico que exige *expertise* de quem faz sua aplicação, imprimindo certa subjetividade ao processo.

Os OSSSP são divididos em duas categorias: OSSSP por Arrolamento de Valores e OSSSP por Intervalo. Ao todo, na implementação realizada do EMHOSIR, considerou-se a existência de nove OSSSP nas duas categorias existentes conforme pode ser visto na Subseção 4.2.1. No entanto, em termos de otimização de hiperparâmetros, foram mantidos fixos os dois OSSSP por Intervalo, relacionados ao *dropout* e a taxa de aprendizagem (*learning rate*). No caso do *dropout*, o mesmo foi mantido em zero. A opção por fazê-lo adveio do fato de, no decorrer de todos os experimentos realizados até o momento em que se optou pela redução de dimensionalidade do espaço de busca de otimização, não foi observada grandes variações nas métricas de desempenho entre conjuntos de teste e validação (uma baixa variância). Para a taxa de aprendizagem, optou-se por mantê-la baixa ($= 0.01$). O valor escolhido vinha se mostrando suficiente para a convergência das soluções a um tempo aceitável. Fixar ambos os valores foi, também, uma forma de se realizar a redução de dimensionalidade no espaço de busca de otimização da qual trata a presente seção.

Para se ter um vislumbre do potencial dimensional que os sete OSSSP restantes (os OSSSP por Arrolamento de Valores já descritos na a Subseção 4.2.1) confere à otimização de hiperparâmetros, faz-se necessário apresentar os valores que se considerou nas etapas incipientes de experimentação para cada um. Para eles, considerou-se, inicialmente, o seguinte:

- ***optimizer***: Estimação Adaptativa de Momentos, Otimizador de Nesterov e Adam e Raiz do Quadrado da Propagação Média;
- ***loss_function***: Logaritmo do Cosseno Hiperbólico e Entropia Cruzada Binária;
- ***input_activation***, ***internal_activation*** e ***output_activation***: Função Tangente Hiperbólica, Função Sigmoide, Unidade Linear Retificada e Unidade Linear Exponencial;
- ***internal_units***: 4, 8, 16, 32, 41, 64, 128 e 256;
- ***n_hidden_layers***: 1, 2 e 3.

Os OSSSP e valores apresentados delineiam $3 \times 2 \times 4^3 \times 8 \times 3 = 9216$ combinações possíveis de hiperparâmetros, o que representa o tamanho do espaço de busca original a partir do qual se iniciaram os experimentos deste trabalho. Para se ter uma ideia do quão extenso esse espaço de busca é, no caso da utilização do método de Busca em Grade (Subseção 2.8.1), para a implementação do EMHOSIR que se realizou, seria necessário treinar 9216 redes MLP, cada qual com uma hiperparametrização existente entre todas as combinações possíveis do espaço de busca de otimização, algo extremamente custoso em termos de tempo. Sendo assim, os valores de OSSSP apresentados foram alvo do refinamento ora apresentado, de modo que, ao final do processo de redução de dimensionalidade do espaço de busca de otimização, foram mantidos apenas aqueles com melhor desempenho em termos das métricas para as quais se tinha interesse. Tendo em vista que a otimização de hiperparâmetros da Fase 2 fora feita tendo como alvo a acurácia do modelo sendo treinado, foi dada especial atenção ao impacto dos valores de OSSSP na acurácia sobre o conjunto de validação.

Conforme apresentado na seção anterior, a implementação do EMHOSIR deste trabalho fora feita considerando duas bases de dados pré-processadas originadas da base de eventos NSL-KDD: a base *NOM_TO_INT* e a base *NOM_TO_BIN*. Em razão de suas características distintas, tal como o número de atributos de cada uma das bases de dados, fez-se necessária a realização de dois procedimentos distintos de redução de dimensionalidade do espaço de busca de otimização, um para cada uma das bases.

A redução de dimensionalidade fora realizada manualmente após a análise de dados estatísticos extraídos de aplicações preliminares do EMHOSIR às bases de dados pré-processadas. Para as análises, o EMHOSIR fora parametrizado em sua Fase 1 de modo a utilizar o método de otimização de hiperparâmetros Otimização Bayesiana sobre uma amostra estratificada de 10% de cada uma das bases de dados. Foram executados 10 experimentos com as mesmas configurações. A ideia por trás da escolha de uma amostra está em agilizar a descoberta dos espaços de busca mais promissores antes de fazer a aplicação sobre a base de dados completa. A escolha

do método de otimização de hiperparâmetros Otimização Bayesiana se deve ao fato de ser o mais performático em termos de tempo entre os três métodos implementados. Os dados para análise foram extraídos a partir das parametrizações do EMHOSIR expostas na Tabela 4.3[†].

Tabela 4.3: Parametrização do EMHOSIR para Análise de Redução de Dimensionalidade do Espaço de Busca para cada Base de Dados Pré-processada

GCP	*Base de Dados NOM_TO_INT*	*Base de Dados NOM_TO_BIN*
folds	10	
k	1	
restore_best_weights	Não	
epochs	500	
OSSSP	*Base de Dados NOM_TO_INT*	*Base de Dados NOM_TO_BIN*
internal_units	4, 8, 16, 32, 41 e 64	4, 8, 16, 32, 64, 128 e 256
n_hidden_layers	1, 2 e 3	
optimizer	Estimação Adaptativa de Momentos (*Adam*), Otimizador de Nesterov e Adam (*Nadam*) e Raiz do Quadrado da Propagação Média (*RMSprop*)	
loss_function	Logaritmo do Cosseno Hiperbólico (*logcosh*) e Entropia Cruzada Binária (*binary_crossentropy*)	
input_activation, *internal_activation* e *output_activation*	Função Tangente Hiperbólica (*tanh*), Função Sigmoide (*sigmoid*), Unidade Linear Retificada (*relu*) e Unidade Linear Exponencial (*elu*)	

Um detalhe relevante, observado na Tabela 4.3 com relação aos experimentos, está na diferença entre os valores de OSSSP considerados para o número de neurônios nas camadas ocultas em cada uma das bases de dados. A diferença se deve à discrepância entre a dimensão das duas bases em termos do número de atributos, sendo que a causa dessa situação está nos diferentes pré-processamentos realizados na geração de cada uma das bases de dados. Com relação a esse mesmo OSSSP, no caso da base de dados *NOM_TO_INT*, também chama atenção o fato do número de neurônios progredir em potência de 2 com exceção do valor 41 que parece estar deslocado entre os demais números. Na verdade, a escolha desse valor se deve ao fato de, no trabalho de Souza (2018), ter sido esse o número de neurônios utilizado na camada oculta de sua rede MLP. A inclusão do valor no experimento teve o objetivo de possibilitar à otimização

[†]Para simplificação, foram omitidos valores de OSSSP e GCP menos relevantes para a redução de dimensionalidade do espaço de busca de otimização

de hiperparâmetros escolhe-lo caso de fato se tratasse de um valor ótimo para o número de neurônios na camada oculta da rede MLP.

A análise realizada para redução de dimensionalidade do espaço de busca de otimização consistiu na identificação dos valores de hiperparâmetros com uma correlação mais associada a bons valores de acurácia, no conjunto de validação (próximos à 1), do que com valores de acurácia considerados ruins sobre esse mesmo conjunto (muito distantes de 1). Um detalhe relevante da análise é que os valores de acurácia sobre o conjunto de validação que se considerou não são aqueles do modelo final $E^{p}_{P_l^h}$*(C1, C2)* gerado pelo EMHOSIR e sim o valor parcial obtido apenas pela rede MLP após a otimização de hiperparâmetros. Além de mais rápido, usar essa acurácia ao invés da final do modelo afasta a possibilidade do valor estar sob influência do autoajuste da faixa intermediária. Como apenas hiperparâmetros discretos estavam em análise para a redução da dimensionalidade, optou-se pela realização da referida análise visualmente, a partir dos gráficos *boxplots* correlacionando os valores desses hiperparâmetros aos valores de acurácia obtidos no conjunto de validação. A análise fora realizada separadamente para cada base de dados pré-processada, de acordo com os experimentos realizados sobre a mesma conforme a Tabela 4.3. Os *boxplots* serão exibidos e analisados na Subseção 5.2.1.

A partir dos referidos experimentos, foram, para cada base de dados, criados três gráficos *boxplots*. Para melhor visualização desses gráficos e consequente melhoria na capacidade de intepretação dos mesmos, os hiperparâmetros foram agrupados três à três em cada gráfico. A escolha por essa forma de apresentação gerou a repetição em mais de um gráfico de informações a respeito dos hiperparâmetros referentes ao número de camadas ocultas e número de neurônios na camada interna da rede MLP. Apesar de ser dispensável para a análise proposta, tal repetição, incidente em razão da quantidade de hiperparâmetros relacionados aos OSSSP serem sete, ajuda na racionalização a respeito do comportamento dos demais hiperparâmetros em relação à acurácia no conjunto de validação.

A apresentação dos resultados referentes a redução de dimensionalidade do espaço de busca de otimização pode ser vista na Seção 5.2. A seção a seguir apresenta os experimentos realizados, com nível de profundidade suficiente à sua plena compreensão, juntos às parametrizações [†] realizadas sobre o EMHOSIR para execução dos mesmos.

4.5 Experimentos Realizados

Os experimentos realizados para as bases de dados pré-processadas *NOM_TO_INT* e *NOM_TO_BIN* são apresentados nesta seção. São ao todo seis experimentos, cada um responsável pela geração de um modelo combinado de classificador $E^{p}_{P_l^h}$*(MLP, K-NN)*, sendo três deles para cada uma das bases de dados pré-processadas que se gerou. Muitos foram os experimentos

[†] A Subseção 4.2.1 pode ser consultada para que sejam retomados os conhecimentos referentes à parametrização via GCP e OSSSP

realizados e descartados antes que se chegasse ao conciso conjunto da etapa empírica relatada nesta seção. As intuições adquiridas nos experimentos não reportados, no entanto, moldaram, de certa forma os experimentos que aqui se apresentam.

Antes de se falar dessas intuições, é válido retomar algumas informações a respeito do EMHOSIR e do artefato por ele produzido. Conforme apresentado neste capítulo, o EMHOSIR produz como artefato um modelo combinado de classificadores $E^{P_h}_{P_l}$*(MLP, K-NN)* em que h e l correspondem a valores em termos de percentil correspondentes aos limites da faixa intermediária em suas porções superior e inferior, respectivamente. Ademais, em razão da Fase 2 do EMHOSIR, o referido artefato conta com a hiperparametrização de seu classificador $C1 = MLP$ otimizada. Tendo sido retomadas tais informações, cabe dizer que os experimentos realizados não se concentraram exclusivamente na obtenção das métricas de desempenho consideradas pelo modelo $E^{P_h}_{P_l}$*(MLP, K-NN)* gerado, mas também na comparação dessas com variações desse modelo com a mesma hiperparametrização para a rede MLP, mas considerando faixas intermediárias diversas.

A razão para isso pode ser compreendida retomando o assunto a respeito das intuições adquiridas com experimentos preliminares; uma delas foi a observação de uma tendência dos melhores resultados de acurácia serem encontrados com percentis h e l que tornam as faixas intermediárias mais estreitas. Em razão disso, para todos os experimentos que se realizou em que se obteve um modelo $E^{P_h}_{P_l}$*(MLP, K-NN)*, foram avaliadas também as métricas de desempenho sobre o conjunto de validação para a variação do modelo encontrado em que a faixa intermediária possui percentil 1 para a parte superior e 1 para a parte inferior (modelos no formato $E^{P_l}_{P_l}$*(MLP, K-NN)* pela Notação 4.1). A ideia por trás dessa avaliação é a de comparar os resultados obtidos por meio da Fase 3 do EMHOSIR com uma abordagem trivial que faça uso de uma faixa intermediária fixa estreita.

Um outro dado que se decidiu extrair em todos os experimentos de ambas as bases de dados pré-processadas foi o desempenho da variação do modelo obtido pelo EMHOSIR sobre o conjunto de validação mas com o uso da faixa intermediária definida pelos limites $h = 25$ e $l = 99$ (modelos no formato $E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*). A escolha desse padrão de faixa intermediária específico se justifica pela mesma razão que o fizera ser utilizado como restrição à minimização da função z' (Seção 4.2.3): trata-se do tamanho de faixa intermediária que obteve melhor desempenho da métrica acurácia sobre o conjunto de validação nos experimentos realizados por Souza (2018) em seu trabalho. A extração de dados de métricas de desempenho para essa faixa intermediária específica potencializa a comparação da solução implementada neste trabalho com a solução de Souza (2018).

Foi ainda, para todos os experimentos, verificado também o desempenho da variação do modelo obtido pelo EMHOSIR em que se adota uma faixa intermediária autoajustada sobre o conjunto de validação, por meio da realização da minimização da função z' (Seção 4.2.3) sobre esse conjunto; a realização de tal autoajuste permitiu a identificação dos valores h' e l' ótimos em termos de faixa intermediária para o conjunto de validação. Apesar de se estar gerando um

modelo $E^{P_h}_{P_{l'}}$*(MLP, K-NN)* não factível em termos práticos, a ideia por traz de fazê-lo está na possibilidade de compará-lo ao modelo $E^{P_h}_{P_l}$*(MLP, K-NN)* obtido, respondendo perguntas como *"quão bom é o desempenho de um modelo cuja faixa intermediária fora autoajustada sobre o conjunto de testes em relação a um modelo com o melhor autoajuste de faixa intermediária possível, realizado sobre o conjunto de validação?"*. Responder a essa questão auxilia no entendimento a respeito da capacidade da faixa intermediária autoajustada sobre um conjunto de dados poder ser generalizada para outros conjuntos distintos.

Por fim, verificou-se ainda os desempenhos em termos de métricas obtidos pelos classificadores individuais da combinação de classificadores utilizada na implementação do EMHOSIR deste trabalho. Deu-se o nome de modelos concorrentes a esses classificadores em conjunto aos modelos com faixas intermediárias variadas apresentados, obtidos à partir do modelo gerado pelo EMHOSIR. Todas as comparações entre o modelo gerado pelo EMHOSIR e seus modelos concorrentes foram realizadas em termos das métricas acurácia, sensibilidade e especificidade. Conforme visto na Subseção 4.2.3, a validação que ocorre na Fase 3 é realizada por meio de validação cruzada de 10 *folds*. Sendo assim, cada uma das métricas de desempenho é extraída 10 vezes: uma por *fold*. Desse modo, os resultados apresentados na seção 5.3 se referem à média da métrica correspondente (Acurácia = AC, Sensibilidade = SE, Especificidade = ES) nos 10 folds ($\overline{x}$), bem como ao desvio padrão referente a essa média (σ). As comparações $\Delta\overline{x}$ correspondem à diferença absoluta da média da métrica obtida pelo modelo concorrente em relação à essa mesma média obtida pelo modelo gerado pelo EMHOSIR. Nas comparações, buscou-se identificar diferenças significativas do ponto de vista estatístico entre os modelos. A análise de significância estatística foi realizada sem assumir qualquer distribuição particular para o conjunto de dados analisado, por meio do teste não-paramétrico de Kruskal-Wallis (Kruskal & Wallis, 1952) e do pós-teste de Dunn (Jobson, 2012), todos utilizando um nível de significância de 95%.

Conforme visto na Seção 4.2.1, o EMHOSIR conta, em sua Fase 1, com um processo de parametrização por meio dos GCP e OSSSP[†] que determinam seu comportamento para a sessão de experimentos que se quer realizar. As seções subsequentes detalham as parametrizações realizadas para cada experimento, junto aos valores de hiperparâmetros obtidos na execução dos mesmos e às métricas de desempenho extraídas da validação.

4.5.1 Experimentos Realizados na Base de Dados *NOM_TO_INT*

A presente subseção contém as informações relacionadas aos experimentos realizados para a base de dados pré-processada *NOM_TO_INT*. A Tabela 4.4 apresenta a parametrização básica em termos de GCP e OSSSP realizada para todos os experimentos referente à base de dados *NOM_TO_INT*. Os únicos GCP diferentes entre os três experimentos realizados para a base de dados *NOM_TO_INT* são os relacionados à otimização de hiperparâmetros (*ho_method*

[†]As definições referentes à GCP irrelevantes para a compreensão dos experimentos realizados foram omitidas

e *random_search_percent*, esse último, onde aplicável). É válido lembrar que, conforme visto na Seção 4.4, foram aplicados procedimentos de redução de dimensionalidade do espaço de busca de otimização (resultado em 5.2) para que se chegasse ao conjunto de valores para OSSSP aqui apresentados.

Tabela 4.4: Parametrização Básica do EMHOSIR para Otimização de Hiperparâmetros para a Base de Dados *NOM_TO_INT*

GCP	
folds	10
k	1
metrics	Acurácia, Sensitividade, Especificidade
monitor	Função de Perda (*loss*)
patience	20
model2_full_set_training	Verdadeiro
OSSSP	
optimizer	Otimizador de Nesterov e Adam (*Nadam*)
loss_function	Logaritmo do Cosseno Hiperbólico (logcosh)
input_activation	Função Tangente Hiperbólica (*tanh*), Função Sigmoide (*sigmoid*), Unidade Linear Retificada (*relu*) e Unidade Linear Exponencial (*elu*)
internal_activation	Função Tangente Hiperbólica (*tanh*), Unidade Linear Retificada (*relu*) e Unidade Linear Exponencial (*elu*)
output_activation	Função Tangente Hiperbólica (*tanh*), Função Sigmoide (*sigmoid*), Unidade Linear Retificada (*relu*) e Unidade Linear Exponencial (*elu*)
internal_units	64, 128 e 256
n_hidden_layers	1
dropout	0
learning_rate	$0,01$

No capítulo 5, os resultados decorrentes dos experimentos apresentados nesta seção podem ser vistos na seção 5.3. A subseção a seguir apresenta os experimentos que se realizou sobre a base de dados *NOM_TO_BIN*.

4.5.2 Experimentos Realizados na Base de Dados *NOM_TO_BIN*

Da mesma forma que a Subseção 4.5.2, a presente subseção contém as informações relacionadas aos experimentos realizados para a base de dados pré-processada *NOM_TO_BIN*. A Tabela 4.5 apresenta a parametrização básica em termos de GCP e OSSSP realizada para todos os experimentos referente à base de dados *NOM_TO_BIN*. Da mesma forma que para a base de dados *NOM_TO_INT*, os únicos GCP diferentes entre os três experimentos realizados para a base de dados *NOM_TO_BIN* são os relacionados à otimização de hiperparâmetros, de modo que os OSSSP provém do procedimento apresentado na Seção 4.4.

Tabela 4.5: Parametrização Básica do EMHOSIR para Otimização de Hiperparâmetros para a Base de Dados *NOM_TO_BIN*

GCP	
folds	10
k	1
metrics	Acurácia, Sensitividade, Especificidade
monitor	Função de Perda (*loss*)
patience	20
model2_full_set_training	Verdadeiro
OSSSP	
optimizer	Otimizador de Nesterov e Adam (*Nadam*)
loss_function	Logaritmo do Cosseno Hiperbólico (logcosh)
input_activation	Função Tangente Hiperbólica (*tanh*), Unidade Linear Retificada (*relu*) e Unidade Linear Exponencial (*elu*)
internal_activation	Função Tangente Hiperbólica (*tanh*), Unidade Linear Retificada (*relu*) e Unidade Linear Exponencial (*elu*)
output_activation	Função Tangente Hiperbólica (*tanh*), Função Sigmoide (*sigmoid*), Unidade Linear Retificada (*relu*) e Unidade Linear Exponencial (*elu*)
internal_units	256, 512 e 1024
n_hidden_layers	1
dropout	0
learning_rate	$0,01$

Da mesma forma que mencionado na subseção anterior, os resultados dos experimentos apresentados na presente seção podem ser vistos na Seção 5.3. A seção a seguir apresenta os materiais referentes a *software* e *hardware* utilizados na execução das tarefas empíricas deste trabalho e implementação do EMHOSIR.

4.6 Material Utilizado

A implementação do EMHOSIR realizada utilizou como ferramental de *software* a linguagem de programação Python versão 3. Na referida implementação, foi feito uso das seguintes bibliotecas:

- Implementação dos Métodos de Otimização de Hiperparâmetros Busca em Grade e Busca Aleatória: *talos*[†];
- Implementação do Método de Otimização Bayesiana: *bayes_opt*[‡];
- Implementação da Evolução Diferencial para o autoajuste da Faixa Intermediária: *scipy*[§];
- Implementações dos Classificadores e Validações: *keras*[¶], *numpy*[‖], *pandas*[**], *scipy*, *scikit-learn*[††] e *tensorflow*[‡‡];
- Implementação de Processamento de Parâmetros de Execução (Fase 1 do EMHOSIR): *canonicaljson*[§§], *fire*[¶¶], *hasher*[***] e *marshmallow*[†††];
- Implementações de *Debugging* e Análise para Redução de Dimensionalidade do Espaço de Busca de Otimização: *matplotlib*[‡‡‡] e *seaborn*[§§§];

Além das ferramentas de *software* já citadas, nas fases iniciais de aproximação do problema, foi utilizado, para a construção de protótipos e validação, a ferramenta Weka em sua versão 3[¶¶¶].

No decorrer do desenvolvimento deste trabalho, foram diversos os *hardwares* utilizados para a implementação do EMHOSIR. Os referidos *hardwares*, todos domésticos, tiveram seu

[†] https://github.com/autonomio/talos
[‡] https://github.com/fmfn/BayesianOptimization
[§] https://www.scipy.org/
[¶] https://keras.io/
[‖] https://numpy.org/
[**] https://pandas.pydata.org/
[††] https://scikit-learn.org/
[‡‡] https://www.tensorflow.org/
[§§] https://pypi.org/project/canonicaljson/
[¶¶] https://github.com/google/python-fire
[***] https://pypi.org/project/hasher/
[†††] https://marshmallow.readthedocs.io/
[‡‡‡] https://matplotlib.org/
[§§§] https://seaborn.pydata.org/
[¶¶¶] https://www.cs.waikato.ac.nz/ml/weka/

uso viabilizado pela utilização de pequenas amostras das bases de dados. No entanto, uma vez finalizadas as etapas relativas à desenvolvimento e codificação, toda a experimentação fora realizada em *hardware* de alto desempenho com as configurações mais adequadas à grandes volumes de dados, como é o caso das bases de dados derivadas geradas à partir da base de dados de eventos NSL-KDD. Os experimentos foram realizados em sua totalidade no *cluster* de alto desempenho do Centro de Computação Científica e Software Livre da Universidade Federal do Paraná (C3SL/UFPR), cujo uso fora gentilmente cedido para o desenvolvimento deste trabalho. Chamado de C3HPC[†], as especificações de seu *hardware* podem ser vistas a seguir:

- Computador de alto desempenho C3HPC (6 nodos de processamento, cada um com 4 *sockets* Intel Xeon E5-4627 v2 @ 3.30GHz com 8 núcleos por *sockets*;
- 256 GB de RAM;
- Rede de alta velocidade *InfiniBand ConnectX-3* 56GbE;
- Placas de vídeo NVIDIA com suporte à plataforma CUDA e GPUDirect.

Conhecidos os materiais em termos de *software* e *hardware* utilizados para desenvolvimento deste trabalho, a próxima Seção finaliza o Capítulo 4 com considerações referentes ao que foi apresentado.

4.7 Considerações Finais

Conforme visto no Capítulo que se se encerra, tendo como referência o problema de detecção de intrusão em redes de computadores, o presente trabalho propõe um método de combinação de classificadores abstrato/genérico chamado EMHOSIR com base no método de combinação de classificadores de Souza (2018). Além de ter sido apresentado em nível conceitual, o método proposto, composto por três fases, foi implementado na forma de uma instância cujos detalhes de funcionamento foram discutidos em minúcia.

Foram também apresentadas as bases de dados pré-processadas obtidas a partir da base de dados de eventos original NSL-KDD, com a apropriada discussão a respeito dos porquês de cada procedimento executado para sua geração. As referidas bases de dados, na condição de alvos da implementação do EMHOSIR realizada, servem ao propósito de construção e avaliação do modelo gerado pelo método proposto.

Levantou-se ainda uma importante discussão a respeito da dimensionalidade do espaço de busca da otimização de hiperparâmetros e seu impacto em termos de tempo na segunda fase que compõe o EMHOSIR. Nessa discussão, foram apontadas as vantagens de ser reduzir esse

[†]Dados adicionais e detalhes relacionados à plataforma de *clusterização* e sistema operacional do C3HPC podem ser conferidos diretamente em seu site: https://www.c3sl.ufpr.br/c3hpc

espaço de busca bem como o que fora feito no caso específico deste trabalho para se reduzir a dimensionalidade em questão.

Apresentou-se então os experimentos realizados de modo a se possibilitar uma análise do EMHOSIR em termos de desempenho nas métricas consideradas, além das verificações inerentes aos hiperparâmetros selecionados pela Fase 2 do método.

Discutidos todos os atributos técnicos referentes a métodos empregados neste trabalho, foram, por fim, apresentados os materiais em termos de *hardware* e *software* que subsidiaram a implementação do EMHOSIR conforme proposto e descrito.

Para dar continuidade à apresentação deste trabalho, o capítulo subsequente trata de reportar os resultados obtidos nas etapas experimentais realizadas durante sua execução, com um discussão crítica a respeito de tais resultados frente aos objetivos que foram propostos.

Capítulo 5

Resultados e Discussão

5.1 Considerações Iniciais

No decorrer dos capítulos desta dissertação, foram apresentados os conhecimentos necessários à compreensão do capítulo que ora se inicia. De posse de tais informações, o leitor está capacitado a avaliar criticamente os resultados obtidos, bem como acompanhar a discussão que se segue a sua apresentação.

O capítulo que se inicia apresenta primeiramente as análises e resultados referentes à redução de dimensionalidade do espaço de busca de otimização para ambas as bases de dados pré-processadas que se gerou (Seção 5.2). Em seguida, na Seção 5.3, os resultados em termos de otimização de hiperparâmetros são apresentados. O capítulo segue com uma seção em que se realiza a discussão acerca dos resultados obtidos (Seção 5.4). Por fim, na Seção 5.5, o desfecho do presente capítulo é apresentado, com comentários finais sobre os resultados obtidos e análises realizadas.

5.2 Análise para Redução de Dimensionalidade do Espaço de Busca de Otimização

Ainda que as análises realizadas para cada uma das bases de dados sejam semelhantes, optou-se por realizá-las em subseções próprias para que as discussões a respeito das especificidades de cada uma possam ser melhor desenvolvidas individualmente. A Subseção a seguir trata da análise para redução de dimensionalidade do espaço de busca de otimização para a base de dados *NOM_TO_INT*.

5.2.1 Análise para Redução de Dimensionalidade do Espaço de Busca de Otimização para a Base de Dados *NOM_TO_INT*

Na presente subseção é apresentada a análise realizada para redução de dimensionalidade do espaço de busca de otimização da base de dados *NOM_TO_INT*. O primeiro gráfico *boxplot* analisado para essa base, apresentado na Figura 5.1 correlaciona os valores dos hiperparâmetros referentes a todas as funções de ativação da rede MLP (*input_activation*, *internal_activation* e *output_activation*) à acurácia sobre o conjunto de validação. Por meio da análise desse gráfico, percebe-se a nítida ocorrência de variações de acurácia sobre o conjunto de validação para a maioria dos casos em que se utiliza como função de ativação interna da rede MLP a Função Sigmoide (*internal_activation = sigmoid*). O desempenho obtido com o uso da Função Sigmoide como função de ativação interna da rede MLP só é comparável, e apenas comparável, não melhor, ao desempenho nos demais casos em condições muito específicas, tais como:

- Quando a função de ativação de saída da rede MLP é também a Função Sigmoide (*output_activation = sigmoid*) e a função de ativação de entrada é ou a Unidade Linear Retificada ou a Unidade Linear Exponencial (*input_activation = relu* ou *input_activation = elu*, respectivamente);
- Quando a função de ativação de saída da rede MLP é a Função Tangente Hiperbólica (*output_activation = tanh*) e a função de ativação de entrada é a Unidade Linear Exponencial (*input_activation = elu*);

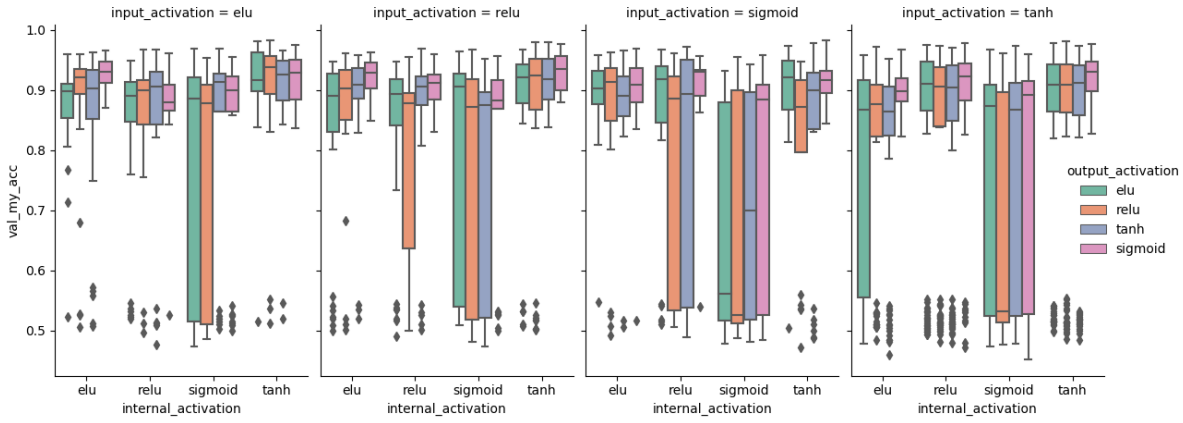

Figura 5.1: Gráfico *Boxplot* da Correlação entre as Funções de Ativação da Rede MLP e a Acurácia no Conjunto de Validação (base de dados *NOM_TO_INT*)

No gráfico analisado em seguida, apresentado na Figura 5.2, as correlações existentes entre a acurácia sobre o conjunto de validação da base de dados *NOM_TO_INT* e os valores de hiperparâmetros da rede MLP referentes ao número de camadas ocultas, ao número de neurônios nas camadas ocultas e a função de perda utilizada para treinamento da mesma (*n_hidden_layers*, *internal_units* e *loss_function* respectivamente) são apresentadas. Por meio da análise do mesmo, percebe-se que:

- A acurácia sobre o conjunto de validação é diretamente proporcional ao número de neurônios nas camadas ocultas;
- Não há diferenças significativas entre a utilização de 1, 2 ou 3 camadas ocultas na rede MLP em termos de acurácia sobre o conjunto de validação;
- A partir de 16 neurônios nas camadas ocultas, a função de perda Logaritmo do Cosseno Hiperbólico apresenta, de forma consistente, a melhor correlação com valores desejáveis da acurácia sobre o conjunto de validação;

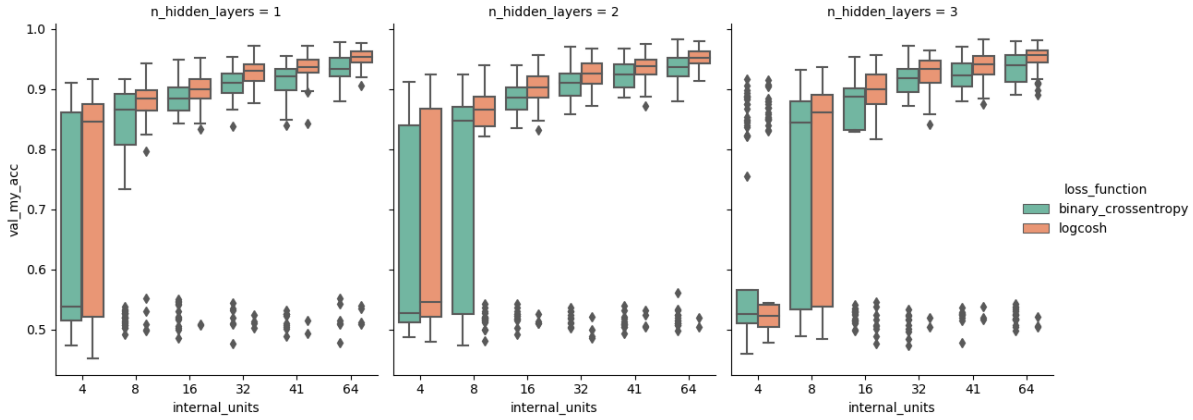

Figura 5.2: Gráfico *Boxplot* da Correlação entre Número de Camadas Ocultas, Número de Neurônios nas Camadas Ocultas e Função de Perda da Rede MLP e a Acurácia no Conjunto de Validação (base de dados *NOM_TO_INT*)

Por fim, na Figura 5.3 em que o último gráfico *boxplot* gerado para a base de dados *NOM_TO_INT* é apresentado, é possível analisar as correlações existentes entre a acurácia sobre o conjunto de validação da base de dados *NOM_TO_INT* e os valores de hiperparâmetros da rede MLP referentes ao número de camadas ocultas, ao número de neurônios nas camadas ocultas e a função de otimização utilizada para treinamento do referido classificador (*n_hidden_layers*, *internal_units* e *optimizer* respectivamente). Além das intuições já adquiridas na análise anterior, referentes a falta de sensibilidade do classificador ao número de camadas ocultas e à melhoria proporcional de seu desempenho diretamente relacionada ao número de neurônios na camada oculta, observa-se também a dominância do Otimizador de Nesterov e Adam quando utilizado como função de otimização (*optimizer* = *nadam*) em comparação com as demais opções inicialmente consideradas.

Resultado da Análise e Redução de Dimensionalidade Realizada

As análises realizadas respaldaram a realização da redução de dimensionalidade do espaço de busca de otimização para a base de dados *NOM_TO_INT*. Após sua realização, optou-se por readequar os valores dos OSSSP a serem considerados para os experimentos finais sobre

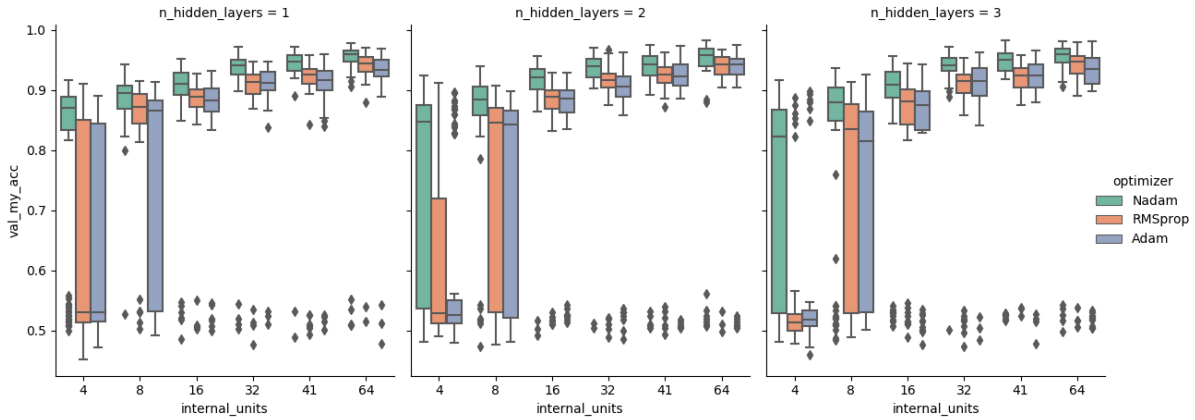

Figura 5.3: Gráfico *Boxplot* da Correlação entre Número de Camadas Ocultas, Número de Neurônios nas Camadas Ocultas e Função de Otimização da Rede MLP e a Acurácia no Conjunto de Validação (base de dados *NOM_TO_INT*)

o EMHOSIR nessa base, conforme abaixo apresentado:

- ***optimizer***: Otimizador de Nesterov e Adam (descartados os otimizadores Estimação Adaptativa de Momentos e Raiz do Quadrado da Propagação Média);
- ***loss_function***: Logaritmo do Cosseno Hiperbólico (descartada a função Entropia Cruzada Binária);
- ***input_activation*** e ***output_activation***: Função Tangente Hiperbólica, Função Sigmoide, Unidade Linear Retificada e Unidade Linear Exponencial;
- ***internal_activation***: Função Tangente Hiperbólica, Unidade Linear Retificada e Unidade Linear Exponencial (descartado o uso da Função Sigmoide);
- ***internal_units***: 64, 128 e 256 (descartadas as opções de se usar 4, 8, 16, 32 e 41 neurônios);
- ***n_hidden_layers***: 1 (descartado o uso de 2 e 3 camadas).

Apesar de, em um primeiro momento, aparentar ser um contrassenso a constatação obtida pela análise realizada de que o número de camadas ocultas da rede MLP não tem influência sobre a acurácia obtida no conjunto de validação, tal situação encontra respaldo no teorema da aproximação universal (Hornik, 1991). O teorema em questão prova que redes neurais de múltiplas camadas alimentadas para frente (como a rede MLP utilizada neste trabalho) são, sob condições gerais relacionadas a função de ativação da camada oculta, aproximadores universais desde que nela existam neurônios suficientes para tal. Tendo em vista o custo computacional para o treinamento de redes neurais com mais camadas, o referido teorema e os resultados experimentais obtidos e analisados, optou-se por fixar em 1 o número de camadas ocultas da rede MLP utilizada com a base de dados *NOM_TO_INT*.

A observação de que a acurácia sobre o conjunto de validação aumenta de acordo com o número de neurônios na camada oculta, também fez com que se optasse por apostar na oti-

mização considerando valores para os hiperparâmetros correspondentes ainda maiores do que os utilizados nos experimentos para redução de dimensionalidade do espaço de busca de otimização da base de dados. Conforme pode ser visto na Tabela 4.3, o valor 64, que era limite superior do OSSSP *internal_units* nos experimentos para redução de dimensionalidade, passou a ser limite inferior desse OSSSP, que foi acrescido dos valores 128 e 256.

Ao fim da redução de dimensionalidade do espaço de busca de otimização para base de dados *NOM_TO_INT*, o espaço de busca de otimização foi reduzido de 6912 para $1 \times 1 \times 4^2 \times 3 \times 3 \times 1 = 144$, equivalente a $\approx 2\%$ do espaço de busca que se tinha inicialmente.

A subseção a seguir trata da análise realizada para redução de dimensionalidade do espaço de busca de otimização para base de dados *NOM_TO_BIN*.

5.2.2 Análise para Redução de Dimensionalidade do Espaço de Busca de Otimização para a Base de Dados *NOM_TO_BIN*

Para a análise da redução de dimensionalidade do espaço de busca de otimização para base de dados *NOM_TO_BIN*, a Figura 5.4 mostra o gráfico *boxplot* que correlaciona os valores dos hiperparâmetros referentes a todas as funções de ativação (*input_activation*, *internal_activation* e *output_activation*) com a acurácia sobre conjunto de validação gerado a partir da base de dados *NOM_TO_BIN*. Pela análise do gráfico, fica bastante claro que a Função Sigmoide, ao ser utilizada como função de ativação das camadas internas da rede MLP (*internal_activation* = *sigmoid*), gera valores de acurácia sobre o conjunto de validação bastante dispersos, com os primeiros quartis bem próximos de $0,5$ na maioria dos casos. Como após o pré-processamento de binarização do problema as bases de dados de evento resultantes passaram à uma proporção aproximada de 50% de cada classe, pode-se dizer que um valor de hiperparâmetro cuja ocorrência está frequentemente relacionada a acurácias sobre o conjunto de validação próximas à $0,5$ não está propiciando um desempenho muito melhor que o erro da classe majoritária.

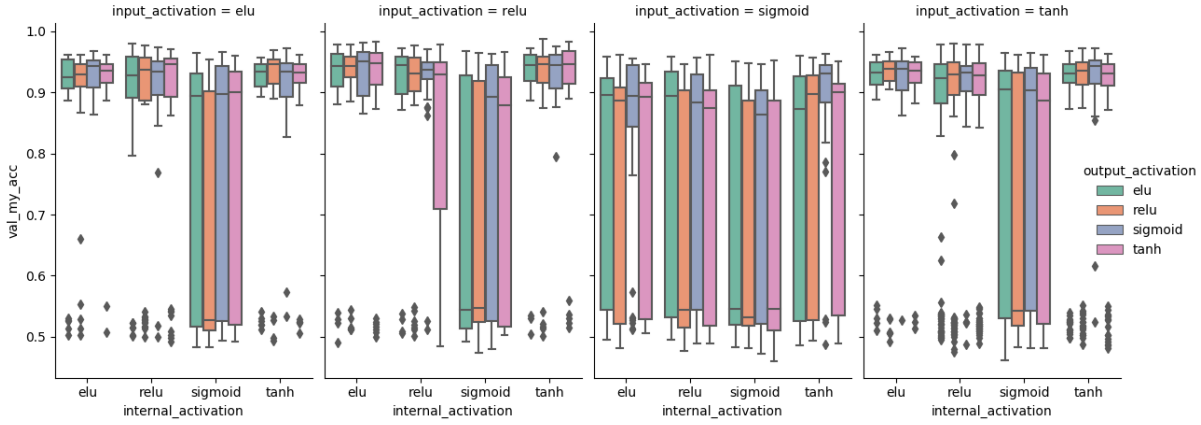

Figura 5.4: Gráfico *Boxplot* da Correlação entre as Funções de Ativação da Rede MLP e a Acurácia no Conjunto de Validação (base de dados *NOM_TO_BIN*)

Analisando o mesmo gráfico na Figura 5.4, também é possível perceber que a Função Sigmoide, ao ser utilizada como função de ativação da camada de entrada da rede MLP (*input_activation* = *sigmoid*), possui um desempenho não muito diferente de quando usada como função de ativação interna, com exceção de quando utilizada também como função de ativação de saída (*output_activation* = *sigmoid*) em conjunto ou com a função Unidade Linear Exponencial ou com a Função Tangente Hiperbólica como funções de ativação interna (*internal_activation* = *elu* ou *internal_activation* = *tanh*, respectivamente). No entanto, mesmo nas ocorrências em que a Função Sigmoide apresenta desempenho aceitável como função de ativação de entrada, não se observa nenhuma hegemonia no referido desempenho quando em comparação com o uso de outras funções em seu lugar. Considerou-se, portanto, que o uso da Função Sigmoide como função de ativação tanto na camada de entrada quanto como função de ativação interna não se mostra mais promissor que outras funções disponíveis para a obtenção de bons desempenhos em termos de acurácia sobre o conjunto de validação para a base de dados *NOM_TO_BIN*.

Prosseguindo com a análise, seguiu-se com a observação das correlações existentes entre a acurácia sobre o conjunto de validação da base de dados *NOM_TO_BIN* e os valores de hiperparâmetros da rede MLP referentes ao número de camadas ocultas, ao número de neurônios nas camadas ocultas e a função de perda utilizada para treinamento da mesma (*n_hidden_layers*, *internal_units* e *loss_function* respectivamente). As referidas correlações podem ser vistas no gráfico *boxplot* da Figura 5.5. Por meio desse gráfico, é possível perceber que:

- A acurácia sobre o conjunto de validação é diretamente proporcional ao número de neurônios nas camadas ocultas;
- Não há diferenças significativas entre a utilização de 1, 2 ou 3 camadas ocultas na rede MLP em termos de acurácia sobre o conjunto de validação;
- A partir de 64 neurônios nas camadas ocultas, a função de perda Logaritmo do Cosseno Hiperbólico apresenta, de forma consistente, a melhor correlação com valores desejáveis da acurácia sobre o conjunto de validação;

De forma semelhante, no gráfico *boxplot* da Figura 5.6, é possível observar novamente as correlações existentes entre a acurácia sobre o conjunto de validação da base de dados *NOM_TO_BIN* e os valores de hiperparâmetros da rede MLP referentes ao número de camadas ocultas, ao número de neurônios nas camadas ocultas e a função de otimização utilizada para treinamento do referido classificador (*n_hidden_layers*, *internal_units* e *optimizer* respectivamente). Conforme esperado, por ser um gráfico gerado por parte dos dados da Figura 5.5, a relação de proporcionalidade direta entre o número de neurônios nas camadas ocultas da rede MLP e a acurácia sobre o conjunto de validação, assim como a indiferença dessa métrica ao número de camadas ocultas da rede MLP também pode ser observada. No entanto, a análise adicional a se fazer para o gráfico *boxplot* é com relação ao hiperparâmetro correspondente a função de otimização (*optimizer*). Conforme pode ser visto em relação a esse hiperparâmetro,

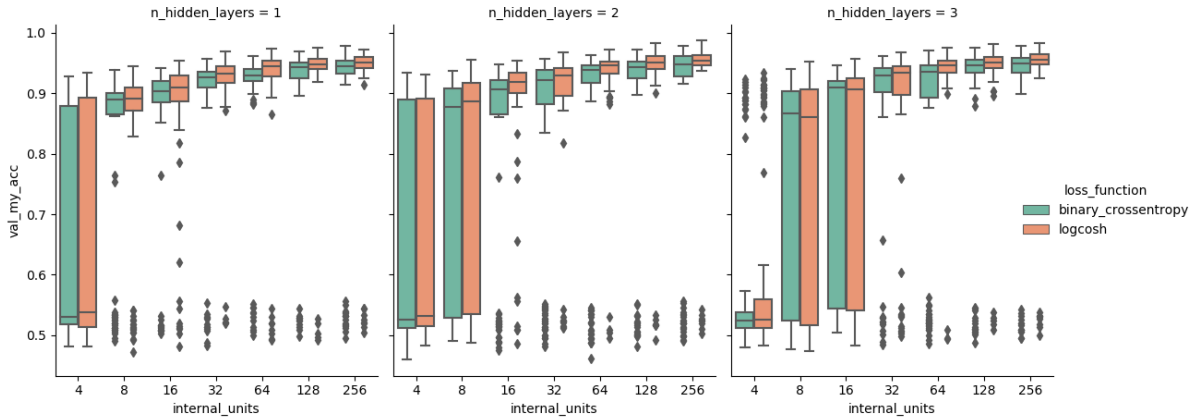

Figura 5.5: Gráfico *Boxplot* da Correlação entre Número de Camadas Ocultas, Número de Neurônios nas Camadas Ocultas e Função de Perda da Rede MLP e a Acurácia no Conjunto de Validação (base de dados *NOM_TO_BIN*)

o Otimizador de Nesterov e Adam (*nadam*) apresentam desempenho consistentemente melhor na maioria dos casos, sendo a melhor entre as funções de otimização para redes MLP com uma única camada oculta.

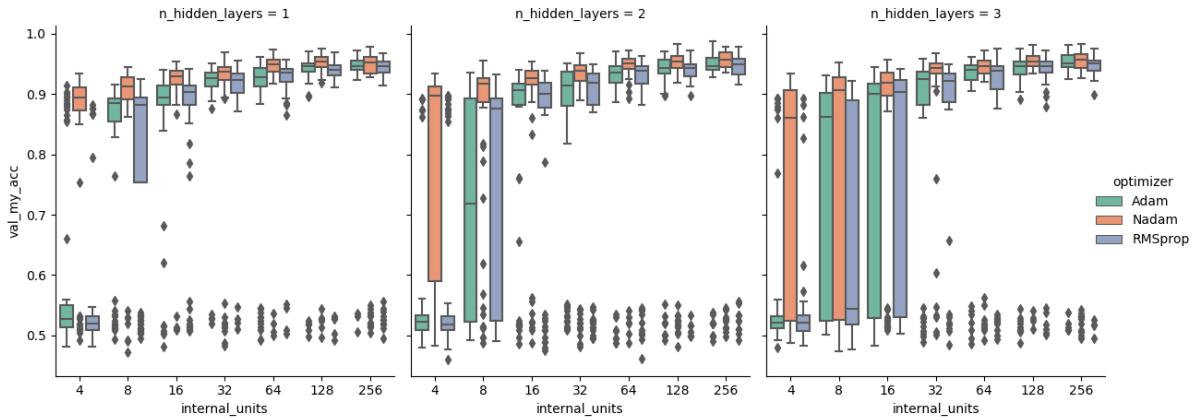

Figura 5.6: Gráfico *Boxplot* da Correlação entre Número de Camadas Ocultas, Número de Neurônios nas Camadas Ocultas e Função de Otimização da Rede MLP e a Acurácia no Conjunto de Validação (base de dados *NOM_TO_BIN*)

Resultado da Análise e Redução de Dimensionalidade Realizada

Da mesma maneira que realizado para a base de dados *NOM_TO_INT*, ao fim da análise de correlações de valores de hiperparâmetros com a acurácia sobre o conjunto de validação da rede MLP, foi realizada a redução de dimensionalidade do espaço de busca de otimização para a base de dados *NOM_TO_BIN*. Os valores dos OSSSP readequados, a serem considerados para os experimentos finais sobre o EMHOSIR nessa base, são abaixo apresentados:

- ***optimizer***: Otimizador de Nesterov e Adam (descartados os otimizadores Estimação

Adaptativa de Momentos e Raiz do Quadrado da Propagação Média);

- ***loss_function***: Logaritmo do Cosseno Hiperbólico (descartada a função Entropia Cruzada Binária);
- ***input_activation*** e ***internal_activation***: Função Tangente Hiperbólica, Unidade Linear Retificada e Unidade Linear Exponencial (descartado o uso da Função Sigmoide);
- ***output_activation***: Função Tangente Hiperbólica, Função Sigmoide, Unidade Linear Retificada e Unidade Linear Exponencial;
- ***internal_units***: 256, 512, 1024 (descartadas as opções de se usar 4, 8, 16, 32, 64 e 128 neurônios);
- ***n_hidden_layers***: 1 (descartado o uso de 2 e 3 camadas).

Da mesma forma que para a base de dados *NOM_TO_INT*, optou-se por fixar em 1 o número de camadas ocultas da rede MLP. Também de forma semelhante, conforme pode ser visto na Tabela 4.3, o valor 256, que era limite superior do OSSSP *internal_units* nos experimentos para redução de dimensionalidade, passou a ser limite inferior desse OSSSP referente ao número de neurônios na camada oculta da rede MLP. Os valores adicionais considerados para o referido hiperparâmetro (512 e 1024) também partiram da observação de melhoria na acurácia sobre o conjunto de validação proporcional ao crescimento do número de neurônios na camada oculta da rede MLP.

Finalizada a redução de dimensionalidade do espaço de busca de otimização para base de dados *NOM_TO_BIN*, o espaço de busca de otimização cujo tamanho inicial era 8064 fora reduzido para $1 \times 1 \times 3^2 \times 4 \times 3 \times 1 = 108$, equivalente a $\approx 1.33\%$ do espaço de busca que se tinha inicialmente.

Apresentados os procedimentos que se realizou em ambas as bases de dados para redução de dimensionalidade do espaço de busca de otimização, a subseção a seguir finaliza as discussões sobre o tópico discutindo brevemente os impactos da aplicação da técnica.

5.2.3 Impacto da Redução de Dimensionalidade do Espaço de Busca de Otimização

Como pôde ser visto no decorrer da presente seção, a redução de dimensionalidade do espaço de busca de otimização, além de exigir análise criteriosa, manual e, até certo ponto, subjetiva, trata-se de um processo moroso que se recomenda fazer antes da aplicação propriamente dita do EMHOSIR. Não se pode esquecer, no entanto, que reduzir o espaço de busca de otimização é deixar de considerar como opções alguns valores de hiperparâmetros, talvez até aqueles que, sobre circunstâncias bem específicas, poderiam levar o modelo final ao seu máximo desempenho em termos das métricas consideradas. Realizar a redução de dimensionalidade do

espaço de busca de otimização de forma diligente minimiza bastante a chance disso ocorrer, mas não afasta a possibilidade por completo.

Apesar de não ser obrigatória, a redução de dimensionalidade do espaço de busca de otimização se mostra importante por economizar recursos computacionais, principalmente na Fase 2 do EMHOSIR. Pelo fato de tais recursos serem frequentemente escassos em pesquisas de cunho acadêmico, a redução de dimensionalidade do espaço de busca de otimização pode ser uma opção para, ao preço da subjetividade inerente ao processo e outros problemas citados, viabilizar experimentos para os quais não haveriam disponíveis recursos computacionais suficientes. A opção pelo uso da técnica neste trabalho se deu justamente por não ter sido possível, com o recurso computacional disponível para experimentação apresentado na Seção 4.6, finalizar a execução das tarefas empíricas utilizando todo o espaço de busca de otimização definido pelos OSSSP conforme inicialmente implementado.

5.3 Resultados em Termos de Otimização de Hiperparâmetros

Conforme apresentado na Seção 4.5, os resultados dos seis experimentos realizados para as bases de dados pré-processadas *NOM_TO_INT* e *NOM_TO_BIN* em termos de otimização de hiperparâmetros são relacionados nesta seção. Cada experimento diz respeito a uma técnica de otimização de hiperparâmetros aplicada sobre uma das bases de dados pré-processadas.

5.3.1 Otimização de Hiperparâmetros por Busca em Grade para a Base de Dados *NOM_TO_INT*

Conforme apresentado na Subseção 4.5.1, a parametrização básica do experimento de otimização de hiperparâmetros por Busca em Grade para a base de dados *NOM_TO_INT* está contido na Tabela 4.4. Com relação ao GCP *ho_method*, o experimento aqui apresentado atribuiu a ele o valor correspondente ao método de otimização de hiperparâmetros em questão (Busca em Grade). No referido experimento, durante a Fase 2 do EMHOSIR, fora encontrada a hiperparametrização constante na Tabela 5.1.

A partir dos hiperparâmetros identificados na Tabela 5.1 durante a Fase 2 do EMHOSIR, fora construído um modelo $E^{P_h}_{P_i}$*(MLP, K-NN)* com autoajuste de faixa intermediária ao final da Fase 3. Na Tabela 5.2, o desempenho do referido modelo em termos das métricas acurácia, sensibilidade e especificidade é apresentado em comparação com o dos modelos concorrentes (

Tabela 5.1: Hiperparametrização Encontrada na Fase 2 do EMHOSIR por meio da Busca em Grade para a Base de Dados *NOM_TO_INT*

Hiperparâmetros Otimizados	
input_activation	Função Tangente Hiperbólica (*tanh*)
internal_activation	Unidade Linear Retificada (*relu*)
output_activation	Função Tangente Hiperbólica (*tanh*)
internal_units	256

Tabela 5.2: Comparativo: Métricas de Desempenho do Modelo $E^{P_h}_{P_l}$*(MLP, K-NN)* Obtido por Busca em Grade sobre a Base de Dados *NOM_TO_INT* com Demais Modelos Concorrentes

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
AC	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,75	$6{,}33 \times 10^{-4}$			
	MLP	**99,67**	$\mathbf{7{,}69 \times 10^{-4}}$	$\mathbf{8{,}10 \times 10^{-4}}$	$\mathbf{1{,}71 \times 10^{-2}}$	$\mathbf{1{,}71 \times 10^{-2}}$
	K-NN	99,74	$4{,}32 \times 10^{-4}$	$1{,}51 \times 10^{-4}$	$2{,}26 \times 10^{-1}$	
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,75	$7{,}08 \times 10^{-4}$	$4{,}76 \times 10^{-5}$	$7{,}91 \times 10^{-1}$	
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,74	$4{,}20 \times 10^{-4}$	$1{,}19 \times 10^{-4}$	$2{,}72 \times 10^{-1}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,78	$3{,}81 \times 10^{-4}$	$2{,}30 \times 10^{-4}$	$4{,}93 \times 10^{-1}$	
SE	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,69	$1{,}02 \times 10^{-3}$			
	MLP	**99,55**	$\mathbf{1{,}25 \times 10^{-3}}$	$\mathbf{1{,}36 \times 10^{-3}}$	$\mathbf{1{,}88 \times 10^{-2}}$	$\mathbf{1{,}88 \times 10^{-2}}$
	K-NN	99,74	$5{,}57 \times 10^{-4}$	$5{,}29 \times 10^{-4}$	$1{,}58 \times 10^{-1}$	
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,66	$1{,}06 \times 10^{-3}$	$3{,}24 \times 10^{-4}$	$1{,}59 \times 10^{-1}$	
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,75	$5{,}36 \times 10^{-4}$	$5{,}46 \times 10^{-4}$	$1{,}37 \times 10^{-1}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,72	$8{,}37 \times 10^{-4}$	$2{,}73 \times 10^{-4}$	$9{,}39 \times 10^{-1}$	
ES	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,81	$5{,}62 \times 10^{-4}$			
	MLP	99,78	$8{,}27 \times 10^{-4}$	$3{,}27 \times 10^{-4}$	$3{,}82 \times 10^{-1}$	
	K-NN	**99,73**	$\mathbf{7{,}19 \times 10^{-4}}$	$\mathbf{7{,}42 \times 10^{-4}}$	$\mathbf{1{,}24 \times 10^{-2}}$	$\mathbf{1{,}24 \times 10^{-2}}$
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,83	$6{,}87 \times 10^{-4}$	$1{,}93 \times 10^{-4}$	$2{,}25 \times 10^{-1}$	
	$\boldsymbol{E^{P_{25}}_{P_{99}}}$***(MLP, K-NN)***	**99,74**	$\mathbf{7{,}21 \times 10^{-4}}$	$\mathbf{6{,}98 \times 10^{-4}}$	$\mathbf{1{,}25 \times 10^{-2}}$	$\mathbf{1{,}25 \times 10^{-2}}$
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,83	$8{,}29 \times 10^{-4}$	$1{,}93 \times 10^{-4}$	$2{,}39 \times 10^{-1}$	

5.3.2 Otimização de Hiperparâmetros por Busca Aleatória para a Base de Dados *NOM_TO_INT*

Tendo como parametrização básica do experimento de otimização de hiperparâmetros por Busca Aleatória para a base de dados *NOM_TO_INT* os valores de GCP e OSSSP contidos na Tabela 4.4, o experimento que aqui se apresenta teve, como se pode presumir, atribuído o valor correspondente à busca aleatória ao GCP *ho_method*.

No detalhamento realizado a respeito desse método de otimização na Subseção 2.8.2, foi discutido seu potencial de encontrar soluções melhores que a Busca em Grade, tendo em vista sua capacidade de explorar hiperparâmetros contínuos. O fato dos OSSSP considerados para realização da otimização de hiperparâmetros serem todos discretos mitigou tal vantagem.

No entanto, uma especificidade da Busca Aleatória, conforme implementada no escopo deste trabalho, foi a possibilidade de se buscar apenas sobre uma amostra do espaço de busca de otimização, minimizando o esforço do método durante a Fase 2 do EMHOSIR em termos computacionais e de tempo. A adoção dessa prática se mostra interessante pois nem todo o grande espaço de busca de otimização dos hiperparâmetros é igualmente promissor no sentido de maximizar as métricas de desempenho que se tem interesse. No entanto, várias questões podem emergir da adoção de tal prática e a mais natural é a dúvida de qual tamanho de amostra do espaço de busca de otimização de hiperparâmetros deve ser utilizado. Conforme dito no início da Seção 4.5, vários experimentos não reportados e descartados precederam os que se encontram descritos no presente capítulo. Tanto por meio da análise desses experimentos, quando por meio dos experimentos feitos especificamente para a redução do espaço de busca de otimização de hiperparâmetros (Seção 4.4), identificou-se que uma amostra de 50% do espaço de busca de otimização teria boas chances de conter instâncias de hiperparametrização capazes de alcançar bons resultados em termos de acurácia sobre o conjunto de validação, com uma pequena margem de segurança para tal. Em razão disso, o presente experimento foi feito com o GCP *random_search_percent* $= 50$. No experimento que aqui se apresenta, durante a Fase 2 do EMHOSIR, fora encontrada a hiperparametrização constante na Tabela 5.3.

Tabela 5.3: Hiperparametrização Encontrada na Fase 2 do EMHOSIR por meio da Busca Aleatória para a Base de Dados *NOM_TO_INT*

Hiperparâmetros Otimizados	
input_activation	Função Tangente Hiperbólica (*tanh*)
internal_activation	Unidade Linear Retificada (*relu*)
output_activation	Função Tangente Hiperbólica (*tanh*)
internal_units	256

A partir dos hiperparâmetros identificados na Tabela 5.3 durante a Fase 2 do EMHOSIR, fora obtido o modelo $E^{P_h}_{P_l}$*(MLP, K-NN)* com autoajuste de faixa intermediária ao final da Fase 3. Na Tabela 5.4, o desempenho do referido modelo em termos das métricas acurácia, sensibilidade e especificidade é apresentado em comparação com o dos modelos concorrentes: modelos concorrentes cujas métricas apresentaram diferenças significativas do ponto de vista estatístico em relação às métricas do modelo gerado pelo EMHOSIR são apresentados em negrito.

Tabela 5.4: Comparativo: Métricas de Desempenho do Modelo $E^{P_h}_{P_l}$*(MLP, K-NN)* Obtido por Busca Aleatória sobre a Base de Dados *NOM_TO_INT* com Demais Modelos Concorrentes

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
AC	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,77	$4{,}86 \times 10^{-4}$			
	MLP	**99,70**	$\mathbf{7{,}73 \times 10^{-4}}$	$\mathbf{7{,}30 \times 10^{-4}}$	$\mathbf{2{,}10 \times 10^{-2}}$	$\mathbf{2{,}10 \times 10^{-2}}$
	K-NN	99,74	$4{,}58 \times 10^{-4}$	$3{,}49 \times 10^{-4}$	$7{,}49 \times 10^{-2}$	
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,77	$4{,}64 \times 10^{-4}$	$7{,}94 \times 10^{-6}$	1,00	
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,74	$4{,}27 \times 10^{-4}$	$2{,}78 \times 10^{-4}$	$1{,}85 \times 10^{-1}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,79	$4{,}44 \times 10^{-4}$	$2{,}14 \times 10^{-4}$	$2{,}26 \times 10^{-1}$	
SE	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,73	$8{,}89 \times 10^{-4}$			
	MLP	**99,60**	$\mathbf{1{,}13 \times 10^{-3}}$	$\mathbf{1{,}38 \times 10^{-3}}$	$\mathbf{1{,}21 \times 10^{-2}}$	$\mathbf{1{,}21 \times 10^{-2}}$
	K-NN	99,74	$7{,}08 \times 10^{-4}$	$8{,}53 \times 10^{-5}$	$8{,}49 \times 10^{-1}$	
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,70	$9{,}82 \times 10^{-4}$	$3{,}41 \times 10^{-4}$	$4{,}26 \times 10^{-1}$	
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,74	$7{,}12 \times 10^{-4}$	$8{,}53 \times 10^{-5}$	$8{,}49 \times 10^{-1}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,74	$7{,}67 \times 10^{-4}$	$5{,}12 \times 10^{-5}$	$8{,}49 \times 10^{-1}$	
ES	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,80	$4{,}89 \times 10^{-4}$			
	MLP	99,79	$7{,}57 \times 10^{-4}$	$1{,}63 \times 10^{-4}$	$6{,}48 \times 10^{-1}$	
	K-NN	**99,73**	$\mathbf{7{,}97 \times 10^{-4}}$	$\mathbf{7{,}28 \times 10^{-4}}$	$\mathbf{2{,}31 \times 10^{-2}}$	$\mathbf{2{,}31 \times 10^{-2}}$
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,83	$3{,}06 \times 10^{-4}$	$2{,}82 \times 10^{-4}$	$8{,}07 \times 10^{-2}$	
	$\boldsymbol{E^{P_{25}}_{P_{99}}}$***(MLP, K-NN)***	**99,74**	$\mathbf{7{,}03 \times 10^{-4}}$	$\mathbf{5{,}94 \times 10^{-4}}$	$\mathbf{1{,}53 \times 10^{-2}}$	$\mathbf{1{,}53 \times 10^{-2}}$
	$\boldsymbol{E^{P_{h'}}_{P_{l'}}}$***(MLP, K-NN)***	**99,84**	$\mathbf{3{,}49 \times 10^{-4}}$	$\mathbf{3{,}56 \times 10^{-4}}$	$\mathbf{4{,}00 \times 10^{-2}}$	$\mathbf{4{,}00 \times 10^{-2}}$

5.3.3 Otimização de Hiperparâmetros por Otimização Bayesiana para a Base de Dados *NOM_TO_INT*

Parametrizado conforme a Tabela 4.4, o experimento referente à otimização de hiperparâmetros por Otimização Bayesiana para a base de dados *NOM_TO_INT* teve atribuído ao GCP *ho_method* o valor que lhe correspondesse. No experimento desta subsubseção, durante a Fase 2 do EMHOSIR, fora encontrada a hiperparametrização constante na Tabela 5.5.

Tabela 5.5: Hiperparametrização Encontrada na Fase 2 do EMHOSIR por meio da Otimização Bayesiana para a Base de Dados *NOM_TO_INT*

Hiperparâmetros Otimizados	
input_activation	Função Tangente Hiperbólica (*tanh*)
internal_activation	Unidade Linear Retificada (*relu*)
output_activation	Unidade Linear Exponencial (*elu*)
internal_units	256

A partir dos hiperparâmetros identificados na Tabela 5.5 durante a Fase 2 do EMHOSIR, fora obtido o modelo $E^{P_h}_{P_l}$*(MLP, K-NN)* com autoajuste de faixa intermediária ao final da Fase 3. Na Tabela 5.6, o desempenho do referido modelo em termos das métricas acurácia, sensibilidade e especificidade é apresentado em comparação com o dos modelos concorrentes: modelos concorrentes cujas métricas apresentaram diferenças significativas do ponto de vista estatístico em relação às métricas do modelo gerado pelo EMHOSIR são apresentados em negrito.

Tabela 5.6: Comparativo: Métricas de Desempenho do Modelo $E^{P_h}_{P_l}$*(MLP, K-NN)* Obtido por Otimização Bayesiana sobre a Base de Dados *NOM_TO_INT* com Demais Modelos Concorrentes

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
AC	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,76	$4{,}95 \times 10^{-4}$			
	MLP	**99,67**	$\mathbf{4{,}32 \times 10^{-4}}$	$\mathbf{8{,}97 \times 10^{-4}}$	$\mathbf{1{,}28 \times 10^{-3}}$	$\mathbf{1{,}28 \times 10^{-3}}$
	K-NN	99,74	$6{,}48 \times 10^{-4}$	$1{,}98 \times 10^{-4}$	$5{,}44 \times 10^{-1}$	
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,73	$6{,}02 \times 10^{-4}$	$2{,}54 \times 10^{-4}$	$2{,}72 \times 10^{-1}$	
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,74	$5{,}79 \times 10^{-4}$	$1{,}35 \times 10^{-4}$	$5{,}44 \times 10^{-1}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,78	$4{,}23 \times 10^{-4}$	$2{,}06 \times 10^{-4}$	$2{,}11 \times 10^{-1}$	
SE	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,70	$8{,}74 \times 10^{-4}$			
	MLP	**99,48**	$\mathbf{1{,}13 \times 10^{-3}}$	$\mathbf{2{,}15 \times 10^{-3}}$	$\mathbf{1{,}27 \times 10^{-3}}$	$\mathbf{1{,}27 \times 10^{-3}}$
	K-NN	99,72	$9{,}10 \times 10^{-4}$	$1{,}88 \times 10^{-4}$	$5{,}44 \times 10^{-1}$	
	$E^{P_l}_{P_l}$(MLP, K-NN)	**99,59**	$\mathbf{1{,}20 \times 10^{-3}}$	$\mathbf{1{,}06 \times 10^{-3}}$	$\mathbf{3{,}61 \times 10^{-2}}$	$\mathbf{3{,}61 \times 10^{-2}}$
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,73	$8{,}41 \times 10^{-4}$	$2{,}73 \times 10^{-4}$	$3{,}42 \times 10^{-1}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,72	$7{,}85 \times 10^{-4}$	$1{,}88 \times 10^{-4}$	$4{,}47 \times 10^{-1}$	
ES	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,81	$5{,}92 \times 10^{-4}$			
	MLP	99,82	$5{,}93 \times 10^{-4}$	$1{,}93 \times 10^{-4}$	$4{,}03 \times 10^{-1}$	
	K-NN	**99,75**	$\mathbf{7{,}77 \times 10^{-4}}$	$\mathbf{5{,}35 \times 10^{-4}}$	$\mathbf{4{,}83 \times 10^{-2}}$	$\mathbf{4{,}83 \times 10^{-2}}$
	$E^{P_l}_{P_l}$(MLP, K-NN)	**99,85**	$\mathbf{4{,}17 \times 10^{-4}}$	$\mathbf{4{,}45 \times 10^{-4}}$	$\mathbf{2{,}50 \times 10^{-2}}$	$\mathbf{2{,}50 \times 10^{-2}}$
	$E^{P_{25}}_{P_{99}}$(MLP, K-NN)	**99,76**	$\mathbf{7{,}28 \times 10^{-4}}$	$\mathbf{4{,}90 \times 10^{-4}}$	$\mathbf{4{,}83 \times 10^{-2}}$	$\mathbf{4{,}83 \times 10^{-2}}$
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,83	$4{,}71 \times 10^{-4}$	$2{,}23 \times 10^{-4}$	$2{,}53 \times 10^{-1}$	

5.3.4 Otimização de Hiperparâmetros por Busca em Grade para a Base de Dados *NOM_TO_BIN*

Para o primeiro experimento apresentado para a base de dados *NOM_TO_BIN*, referente a otimização de hiperparâmetros por Busca em Grade, fora, conforme informado no início da seção, utilizada a parametrização constante na Tabela 4.5, com o GCP *ho_method* atribuído com valor correspondente. No experimento em questão fora encontrada, durante a Fase 2 do EMHOSIR, a hiperparametrização constante na Tabela 5.7.

Tabela 5.7: Hiperparametrização Encontrada na Fase 2 do EMHOSIR por meio da Busca em Grade para a Base de Dados *NOM_TO_BIN*

Hiperparâmetros Otimizados	
input_activation	Função Tangente Hiperbólica (*tanh*)
internal_activation	Unidade Linear Retificada (*relu*)
output_activation	Unidade Linear Exponencial (*elu*)
internal_units	1024

A partir dos hiperparâmetros identificados na Tabela 5.7 durante a Fase 2 do EMHOSIR, foi obtido o modelo $E^{P_h}_{P_l}$*(MLP, K-NN)* com autoajuste de faixa intermediária ao final da Fase 3. Na Tabela 5.8, o desempenho do referido modelo em termos das métricas acurácia, sensibilidade e especificidade é apresentado em comparação com o dos modelos concorrentes: da mesma forma que nos demais experimentos, modelos concorrentes cujas métricas apresentaram diferenças significativas do ponto de vista estatístico em relação às métricas do modelo gerado pelo EMHOSIR são apresentados em negrito.

Tabela 5.8: Comparativo: Métricas de Desempenho do Modelo $E^{P_h}_{P_l}$*(MLP, K-NN)* Obtido por Busca em Grade sobre a Base de Dados *NOM_TO_BIN* com Demais Modelos Concorrentes

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
AC	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,77	$4{,}48 \times 10^{-4}$			
	MLP	99,74	$4{,}50 \times 10^{-4}$	$2{,}78 \times 10^{-4}$	$3{,}84 \times 10^{-1}$	
	K-NN	99,74	$4{,}95 \times 10^{-4}$	$2{,}86 \times 10^{-4}$	$1{,}98 \times 10^{-1}$	
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,78	$4{,}03 \times 10^{-4}$	$7{,}15 \times 10^{-5}$	$5{,}44 \times 10^{-1}$	
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,75	$4{,}84 \times 10^{-4}$	$2{,}06 \times 10^{-4}$	$3{,}62 \times 10^{-1}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,78	$4{,}19 \times 10^{-4}$	$1{,}27 \times 10^{-4}$	$3{,}63 \times 10^{-1}$	
SE	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,75	$4{,}76 \times 10^{-4}$			
	MLP	**99,68**	$\mathbf{5{,}95 \times 10^{-4}}$	$\mathbf{6{,}99 \times 10^{-4}}$	$\mathbf{1{,}81 \times 10^{-2}}$	$\mathbf{1{,}81 \times 10^{-2}}$
	K-NN	99,72	$5{,}97 \times 10^{-4}$	$3{,}41 \times 10^{-4}$	$2{,}68 \times 10^{-1}$	
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,73	$5{,}39 \times 10^{-4}$	$2{,}22 \times 10^{-4}$	$4{,}47 \times 10^{-1}$	
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,73	$5{,}28 \times 10^{-4}$	$1{,}71 \times 10^{-4}$	$5{,}67 \times 10^{-1}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,75	$4{,}58 \times 10^{-4}$	$3{,}41 \times 10^{-5}$	$8{,}48 \times 10^{-1}$	
ES	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,79	$6{,}49 \times 10^{-4}$			
	MLP	99,80	$6{,}94 \times 10^{-4}$	$8{,}91 \times 10^{-5}$	$7{,}91 \times 10^{-1}$	
	K-NN	99,77	$7{,}28 \times 10^{-4}$	$2{,}38 \times 10^{-4}$	$3{,}24 \times 10^{-1}$	
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,82	$5{,}46 \times 10^{-4}$	$3{,}27 \times 10^{-4}$	$3{,}06 \times 10^{-1}$	
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,77	$7{,}45 \times 10^{-4}$	$2{,}38 \times 10^{-4}$	$3{,}06 \times 10^{-1}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,82	$5{,}41 \times 10^{-4}$	$2{,}67 \times 10^{-4}$	$5{,}17 \times 10^{-1}$	

5.3.5 Otimização de Hiperparâmetros por Busca Aleatória para a Base de Dados *NOM_TO_BIN*

Utilizando-se dos parâmetros definidos na Tabela 4.5, o experimento de otimização de hiperparâmetros por Busca Aleatória para a base de dados *NOM_TO_BIN* teve o valor do GCP *ho_method* configurado de forma correspondente ao método da experimentação, com o GCP *random_search_percent* atribuído o valor 50 pelas mesmas intuições expressadas na aplicação desse método de otimização sobre a base de dados *NOM_TO_INT*. No presente experimento, durante a Fase 2 do EMHOSIR, fora encontrada a hiperparametrização constante na Tabela 5.9.

Tabela 5.9: Hiperparametrização Encontrada na Fase 2 do EMHOSIR por meio da Busca Aleatória para a Base de Dados *NOM_TO_BIN*

Hiperparâmetros Otimizados	
input_activation	Função Tangente Hiperbólica (*tanh*)
internal_activation	Unidade Linear Retificada (*relu*)
output_activation	Unidade Linear Exponencial (*elu*)
internal_units	512

A partir dos hiperparâmetros identificados na Tabela 5.9 durante a Fase 2 do EMHOSIR, fora obtido o modelo $E^{P_i}_{P_i}$*(MLP, K-NN)* com autoajuste de faixa intermediária ao final da Fase 3. Na Tabela 5.10, o desempenho do referido modelo em termos das métricas acurácia, sensibilidade e especificidade é apresentado em comparação com o dos modelos concorrentes: são apresentados em negrito os modelos concorrentes cujas métricas apresentaram diferenças significativas do ponto de vista estatístico em relação às métricas do modelo gerado pelo EMHOSIR.

5.3.6 Otimização de Hiperparâmetros por Otimização Bayesiana para a Base de Dados *NOM_TO_BIN*

Para este último experimento, conforme previamente enunciado, também fora utilizada a parametrização presente na Tabela 4.5. Esse experimento, relacionado à otimização de hiperparâmetros por Otimização Bayesiana para a base de dados *NOM_TO_BIN*, teve a configuração de GCP *ho_method* realizada de modo a expressar o método de otimização que se queria utilizar. No experimento aqui apresentado, durante a Fase 2 do EMHOSIR, fora encontrada a hiperparametrização constante na Tabela 5.11.

A partir dos hiperparâmetros identificados na Tabela 5.11 durante a Fase 2 do EMHOSIR, fora obtido o modelo $E^{P_i}_{P_i}$*(MLP, K-NN)* com autoajuste de faixa intermediária ao final da Fase 3. Na Tabela 5.12, o desempenho do referido modelo em termos das métricas acurácia, sensibilidade e especificidade é apresentado em comparação com o dos modelos concorrentes:

Tabela 5.10: Comparativo: Métricas de Desempenho do Modelo $E^{P_h}_{P_l}$*(MLP, K-NN)* Obtido por Busca Aleatória sobre a Base de Dados *NOM_TO_BIN* com Demais Modelos Concorrentes

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
AC	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,76	$5{,}58 \times 10^{-4}$			
	MLP	99,73	$6{,}49 \times 10^{-4}$	$2{,}62 \times 10^{-4}$	$2{,}26 \times 10^{-1}$	
	K-NN	99,74	$6{,}95 \times 10^{-4}$	$2{,}38 \times 10^{-4}$	$2{,}89 \times 10^{-1}$	
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,77	$5{,}30 \times 10^{-4}$	$9{,}53 \times 10^{-5}$	$5{,}18 \times 10^{-1}$	
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,74	$6{,}81 \times 10^{-4}$	$2{,}06 \times 10^{-4}$	$2{,}89 \times 10^{-1}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,77	$5{,}77 \times 10^{-4}$	$1{,}35 \times 10^{-4}$	$3{,}43 \times 10^{-1}$	
SE	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,74	$7{,}72 \times 10^{-4}$			
	MLP	**99,65**	$\mathbf{9{,}45 \times 10^{-4}}$	$\mathbf{8{,}19 \times 10^{-4}}$	$\mathbf{4{,}86 \times 10^{-2}}$	$\mathbf{4{,}86 \times 10^{-2}}$
	K-NN	99,72	$9{,}06 \times 10^{-4}$	$2{,}05 \times 10^{-4}$	$5{,}96 \times 10^{-1}$	
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,72	$7{,}38 \times 10^{-4}$	$1{,}88 \times 10^{-4}$	$4{,}47 \times 10^{-1}$	
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,73	$8{,}60 \times 10^{-4}$	$6{,}82 \times 10^{-5}$	$7{,}61 \times 10^{-1}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,73	$8{,}69 \times 10^{-4}$	$5{,}12 \times 10^{-5}$	$7{,}90 \times 10^{-1}$	
ES	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,78	$5{,}32 \times 10^{-4}$			
	MLP	99,80	$6{,}42 \times 10^{-4}$	$2{,}23 \times 10^{-4}$	$4{,}71 \times 10^{-1}$	
	K-NN	99,75	$7{,}80 \times 10^{-4}$	$2{,}67 \times 10^{-4}$	$3{,}41 \times 10^{-1}$	
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,81	$6{,}11 \times 10^{-4}$	$3{,}42 \times 10^{-4}$	$8{,}06 \times 10^{-2}$	
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,75	$7{,}41 \times 10^{-4}$	$3{,}27 \times 10^{-4}$	$1{,}60 \times 10^{-1}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,81	$6{,}60 \times 10^{-4}$	$2{,}97 \times 10^{-4}$	$1{,}72 \times 10^{-1}$	

Tabela 5.11: Hiperparametrização Encontrada na Fase 2 do EMHOSIR por meio da Otimização Bayesiana para a Base de Dados *NOM_TO_BIN*

Hiperparâmetros Otimizados	
input_activation	Unidade Linear Retificada (*relu*)
internal_activation	Unidade Linear Retificada (*relu*)
output_activation	Unidade Linear Exponencial (*elu*)
internal_units	1024

em negrito, são apresentados os modelos concorrentes cujas métricas apresentaram diferenças significativas do ponto de vista estatístico em relação às métricas do modelo gerado pelo EMHOSIR.

Tabela 5.12: Comparativo: Métricas de Desempenho do Modelo $E^{P_h}_{P_l}$*(MLP, K-NN)* Obtido por Otimização Bayesiana sobre a Base de Dados *NOM_TO_BIN* com Demais Modelos Concorrentes

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
AC	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,78	$2{,}99 \times 10^{-4}$			
	MLP	**99,75**	$\mathbf{3{,}41 \times 10^{-4}}$	$\mathbf{3{,}18 \times 10^{-4}}$	$\mathbf{3{,}73 \times 10^{-2}}$	$\mathbf{3{,}73 \times 10^{-2}}$
	K-NN	**99,75**	$\mathbf{3{,}75 \times 10^{-4}}$	$\mathbf{3{,}33 \times 10^{-4}}$	$\mathbf{1{,}71 \times 10^{-2}}$	$\mathbf{1{,}71 \times 10^{-2}}$
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,78	$2{,}95 \times 10^{-4}$	$7{,}93 \times 10^{-6}$	$8{,}20 \times 10^{-1}$	
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,76	$3{,}21 \times 10^{-4}$	$2{,}22 \times 10^{-4}$	$9{,}51 \times 10^{-2}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,79	$3{,}25 \times 10^{-4}$	$5{,}56 \times 10^{-5}$	$4{,}94 \times 10^{-1}$	
SE	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,74	$6{,}58 \times 10^{-4}$			
	MLP	**99,68**	$\mathbf{5{,}62 \times 10^{-4}}$	$\mathbf{5{,}97 \times 10^{-4}}$	$\mathbf{4{,}35 \times 10^{-2}}$	$\mathbf{4{,}35 \times 10^{-2}}$
	K-NN	99,72	$7{,}09 \times 10^{-4}$	$1{,}54 \times 10^{-4}$	$7{,}90 \times 10^{-1}$	
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,72	$7{,}24 \times 10^{-4}$	$1{,}71 \times 10^{-4}$	$4{,}66 \times 10^{-1}$	
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*	99,74	$6{,}92 \times 10^{-4}$	0,00	$9{,}70 \times 10^{-1}$	
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,74	$7{,}63 \times 10^{-4}$	$9{,}00 \times 10^{-9}$	$9{,}70 \times 10^{-1}$	
ES	$E^{P_h}_{P_l}$*(MLP, K-NN)*	99,82	$3{,}64 \times 10^{-4}$			
	MLP	99,81	$4{,}33 \times 10^{-4}$	$7{,}43 \times 10^{-5}$	$7{,}04 \times 10^{-1}$	
	K-NN	**99,77**	$\mathbf{3{,}43 \times 10^{-4}}$	$\mathbf{4{,}90 \times 10^{-4}}$	$\mathbf{1{,}22 \times 10^{-2}}$	$\mathbf{1{,}22 \times 10^{-2}}$
	$E^{P_l}_{P_l}$*(MLP, K-NN)*	99,84	$3{,}24 \times 10^{-4}$	$1{,}34 \times 10^{-4}$	$2{,}88 \times 10^{-1}$	
	$\boldsymbol{E^{P_{25}}_{P_{99}}}$***(MLP, K-NN)***	**99,78**	$\mathbf{3{,}27 \times 10^{-4}}$	$\mathbf{4{,}16 \times 10^{-4}}$	$\mathbf{2{,}06 \times 10^{-2}}$	$\mathbf{2{,}06 \times 10^{-2}}$
	$E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*	99,83	$3{,}64 \times 10^{-4}$	$1{,}04 \times 10^{-4}$	$4{,}70 \times 10^{-1}$	

5.4 Discussão sobre Resultados dos Experimentos

A seção anterior apresentou os experimentos realizados, suas parametrizações, os hiperparâmetros otimizados encontrados, bem como os resultados extraídos pela etapa de validação inerente à Fase 3 do EMHOSIR. Na seção que se inicia, é realizada uma discussão analítica a respeito do que fora apresentado na seção anterior, comparando as métricas de desempenho obtidas pelos modelos gerados pelo EMHOSIR em cada experimento tanto com as obtidas por seus modelos concorrentes correspondentes, quanto com as obtidas pelo método de combinação de classificadores de Souza (2018). Também fora realizada uma análise comparativa no mesmo sentido entre os seis modelos gerados em cada um dos experimentos. Por fim, uma comparação entre o desempenho do EMHOSIR em relação a demais técnicas de detecção de intrusão em redes de computadores constantes no estado da arte da área foi apresentada.

As referidas discussões são apresentadas em subseções próprias, cada uma delas tratando de uma das comparações realizadas. Na Subseção 5.4.1 , é realizada a discussão e comparação correspondente aos modelos gerados pelo EMHOSIR nos experimentos e os modelos concor-

rentes respectivos em termos das métricas de desempenho extraídas na validação da Fase 3 do método. Na Subseção 5.4.2, em que se realiza a comparação entre os modelos gerados pelo EMHOSIR e o método de combinação de classificadores de Souza (2018), além das métricas acurácia, sensibilidade e especificidade já apresentadas, apresenta-se também uma nova métrica correspondente a quantidade média de exemplos enviados ao classificador $C2$ por $fold$ em termos percentuais em cada experimento (métrica rotulada como C2 nas tabelas). A escolha de uma métrica relativa ao número de instâncias enviadas à $C2$ em detrimento dos valores nominais de h e l respectivos diz respeito ao fato desses últimos variarem $fold$ à $fold$, além de não darem a real dimensão do quanto o classificador $C2$ precisa ser acionado para que as métricas de desempenho do modelo gerado sejam as obtidas. Na Subseção 5.4.3, as métricas extraídas pela validação da Fase 3 do EMHOSIR, junto à nova métrica correspondente a quantidade média de exemplos enviados ao classificador $C2$, são utilizadas para comparar entre si todos os seis modelos gerados pelo EMHOSIR neste trabalho. Finalmente, na Subseção 5.4.4, o EMHOSIR é comparado com outras técnicas de detecção de intrusão em redes de computadores constantes no estado da arte apresentadas no Capítulo 3.

5.4.1 Comparação entre Modelos Gerados pelo EMHOSIR e Modelos Concorrentes Correspondentes

Conforme apresentado na Seção 4.5, foram realizados seis experimentos resultantes da combinação das duas bases de dados *NOM_TO_INT* e *NOM_TO_BIN* com os três métodos de otimização de hiperparâmetros implementados (Busca em Grade, Busca Aleatória e Otimização Bayesiana). Cada um dos seis experimentos, por meio do EMHOSIR, gerou um modelo combinado de classificador $E^{P_h}_{P_l}$*(MLP, K-NN)*; os referidos modelos foram então, cada um, comparados à cinco modelos chamados de concorrentes em termos das métricas de desempenho acurácia, sensibilidade e especificidade. Os primeiros dois modelos concorrentes com os quais a comparação fora realizada nada mais são do que os classificadores $C1$ e $C2$, correspondentes respectivamente à rede MLP e ao modelo gerado pelo K-NN, que compõe o modelo combinado de classificador $E^{P_h}_{P_l}$*(MLP, K-NN)*; os demais três modelos concorrentes são variações dos modelos $E^{P_h}_{P_l}$*(MLP, K-NN)* gerados pela Fase 2 do EMHOSIR, mas com faixas intermediárias diversas à prescrição da Fase 3 do método, sendo duas faixas intermediárias fixas ($E^{P_l}_{P_l}$*(MLP, K-NN)* e $E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*) e uma autoajustada sobre o conjunto de dados de validação ($E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)*). A comparação se deu em termos das médias das métricas obtidas em cada um dos 10 $folds$ da validação cruzada realizada na Fase 3 do EMHOSIR. Para simplificação textual da análise, optou-se por, em um primeiro momento, discorrer apenas a respeito das situações em que, em razão dos *p-valores* do teste não-paramétrico de Kruskal-Wallis (Kruskal & Wallis, 1952) e do pós-teste de Dunn (Jobson, 2012), fora observada diferença estatisticamente significativa entre os desempenhos dos modelos: por haver a necessidade de um olhar mais criterioso, as análises dos demais casos, em que os modelos possuem desempenho equivalente segundo os testes de hipótese, tiveram sua discussão postergada para o fim desta seção.

Ao se iniciar as análises pelos três experimentos realizados sobre a base de dados de eventos derivada *NOM_TO_INT*, é possível ver, na Tabela 5.2, o comparativo entre o modelo $E^{P_h}_{P_l}$*(MLP, K-NN)* gerado com o uso da Busca em Grade para otimização de hiperparâmetros e seus modelos concorrentes em termos das métricas desempenho consideradas. Conforme se observa nesse experimento, tendo em vista os *p-valores* calculados e a hiperparametrização obtida na Fase 2 do EMHOSIR (Tabela 5.1), o modelo combinado de classificador $E^{P_h}_{P_l}$*(MLP, K-NN)* gerado pelo EMHOSIR fora capaz de obter um desempenho médio em termos das métricas Acurácia e Sensibilidade superior ao do modelo concorrente correspondente ao uso isolado de uma rede MLP. Também é possível observar que a abordagem proposta supera os desempenhos médios em termos de Especificidade obtidos tanto pelo modelo gerado pelo K-NN quando pelo modelo correspondente com faixa intermediária fixa em que $h = 25$ e $l = 99$.

Comparado a seus modelos concorrentes, situação semelhante é observada no experimento envolvendo a base de dados *NOM_TO_INT* utilizando Busca Aleatória, com a diferença de que, no caso desse experimento, o modelo concorrente $E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)* obteve melhor desempenho em termos de Especificidade do que o modelo combinado de classificador gerado pelo EMHOSIR (Tabela 5.4). Observa-se que, no referido experimento, os hiperparâmetros encontrados pela Fase 2 do EMHOSIR (Tabela 5.3) foram os mesmos que os obtidos no experimento anteriormente analisado.

O terceiro experimento realizado com a base de dados *NOM_TO_INT* fez uso do método de Otimização Bayesiana para a otimização de hiperparâmetros. Observando a Tabela 5.5, é possível ver uma hiperparametrização semelhante as já analisadas, variando apenas em relação à função de ativação na camada de saída da rede MLP que compõe o classificador $E^{P_h}_{P_l}$*(MLP, K-NN)*. Conforme pode ser visto na Tabela 5.6, novamente o modelo combinado de classificador $E^{P_h}_{P_l}$*(MLP, K-NN)* gerado pelo EMHOSIR apresentou melhor desempenho médio em termos de Acurácia e Sensibilidade quando comparado ao modelo concorrente correspondente à rede MLP; ainda em termos de Sensibilidade, o EMHOSIR gerou um modelo capaz de superar o modelo combinado de classificador de faixa intermediária fixa $E^{P_l}_{P_l}$*(MLP, K-NN)*. Em termos da métrica Especificidade, o modelo gerado superou ainda os modelos concorrentes correspondentes ao gerado pelo K-NN e o modelo combinado de classificador $E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*; o referido modelo, no entanto, fora superado pelo modelo combinado concorrente $E^{P_l}_{P_l}$*(MLP, K-NN)*. Nesse experimento, é visível o *trade-off* de desempenho do modelo concorrente $E^{P_l}_{P_l}$*(MLP, K-NN)*: sua capacidade de superar o modelo combinado de classificador gerado pelo EMHOSIR em termos de Especificidade se reflete em uma deficiência em termos de Sensibilidade que o leva a ser, por ele, superado.

O que se observou de modo recorrente nos modelos combinados de classificador $E^{P_h}_{P_l}$*(MLP, K-NN)* gerados pelo EMHOSIR por qualquer um dos métodos de otimização sobre a base de dados *NOM_TO_INT* foi sua capacidade de superar os modelos concorrentes correspondentes ao uso isolado de redes MLP em termos da métrica de desempenho Acurácia. O mesmo acabara acontecendo em relação aos modelos concorrentes gerados pelo uso exclusivo do K-NN

em termos da métrica Especificidade. Tendo em vista que o autoajuste da faixa intermediária da Fase 3 do EMHOSIR, realizado por meio da minimização da função z', incorpora, entre outros, mecanismos que bonificam a melhoria da Acurácia do modelo sendo gerado em relação ao uso isolado da rede MLP, a situação observada era algo, não apenas esperado, mas também buscado. Outras melhorias acabam sendo colaterais, como é o caso da melhor Sensibilidade dos modelos gerados frente ao uso da rede MLP em todos os experimentos dessa base de dados: o que o ocorre é que uma melhor Acurácia em termos de significância estatística está frequentemente associada à melhoria, sob o mesmo aspecto, de ao menos uma das demais métricas aferidas.

Partindo para a análise dos experimentos feitos com a base de dados *NOM_TO_BIN*, o que se observa, no primeiro deles, é que, em comparação com seus modelos concorrentes, o modelo combinado de classificador $E^{P_h}_{P_l}$*(MLP, K-NN)*, gerado pelo EMHOSIR utilizando Busca em Grade na Fase 2, obteve melhor Sensibilidade do que o uso individual da rede MLP. Situação análoga ocorrera com o uso de Busca Aleatória sobre a base de dados *NOM_TO_BIN*. Em termos de hiperparâmetros encontrados na Fase 2 do EMHOSIR, as Tabelas 5.7 e 5.9 mostram que, comparando-se os resultados obtidos nos dois experimentos, a diferença está exclusivamente na quantidade de neurônios nas camadas internas da rede MLP em cada um deles.

No experimento da base de dados *NOM_TO_BIN* com o uso de Otimização Bayesiana na Fase 2 do EMHOSIR, o modelo combinado de classificador $E^{P_h}_{P_l}$*(MLP, K-NN)* superou o modelo concorrente correspondente à rede MLP em termos das métrica Acurácia e Sensibilidade; esse mesmo modelo fora ainda superior ao modelo concorrente correspondente ao gerado pelo K-NN em termos de Acurácia e Especificidade. Por fim, o modelo gerado nesse experimento teve ainda melhor desempenho que o modelo concorrente que considera o uso da faixa intermediária fixa $h = 25$ e $l = 99$ (modelo combinado de classificador $E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*).

Em comum a todos os experimentos realizados para a base de dados *NOM_TO_BIN*, pode-se observar a melhoria da métrica Sensibilidade em relação à obtida pelo modelo concorrente correspondente ao uso individual da rede MLP. No entanto, a referida melhoria só se refletiu em uma melhor Acurácia em relação a esse mesmo modelo concorrente no experimento em que se utilizou a Otimização Bayesiana como método de otimização de hiperparâmetros.

Um fato que não pode passar despercebido é que a interpretação das situações de equivalência de modelos em termos de métricas de desempenho varia de acordo com os modelos sendo comparados. Isso é fato, principalmente, na comparação entre os modelos combinados de classificador $E^{P_h}_{P_l}$*(MLP, K-NN)* e seus modelos concorrentes. Para os modelos que se gerou por meio dos experimentos apresentados na Seção 4.5, pode ser mais vantajoso ou menos vantajoso a equivalência em relação à cada um de seus concorrentes, de modo que as análises devem ser individuais.

Quando, por exemplo, o modelo combinado de classificador $E^{P_h}_{P_l}$*(MLP, K-NN)* gerado pelo EMHOSIR é incapaz de superar as métricas de desempenho obtidas pelo uso isolado da rede MLP, pode-se estar diante de uma das duas situações a seguir: ou o autoajuste realizado fora

inócuo/mal-realizado, ou não há margem para melhoria por meio do autoajuste da faixa intermediária, o que significa que a otimização de hiperparâmetros da Fase 2 do EMHOSIR foi tão assertiva que obteve uma rede MLP capaz de obter desempenho comparável ao da combinação dos classificadores que se propõe combinar na implementação realizada. A comparação dos modelos gerados com o modelo concorrente $E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)* (cuja faixa intermediária fora autoajustada sobre o conjunto de validação) ajuda a identificar diante de qual das duas situações se está diante.

Os únicos experimentos em que os modelos combinados de classificador $E^{P_h}_{P_l}$*(MLP, K-NN)* gerados pelo EMHOSIR não apresentaram melhor desempenho em relação a métrica Acurária quando em comparação ao modelo concorrente correspondente ao uso individual da rede MLP foram os realizados sobre a base de dados *NOM_TO_BIN*, com o uso dos métodos de otimização de hiperparâmetros Busca em Grade e Busca Aleatória. Na verdade, nos referidos experimentos, o que se observou foi uma equivalência quase que total de métricas dos modelos gerados em relação a seus concorrentes, algo que, conforme adiantado em discussão anterior, não figura necessariamente problema ou disfunção do modelos $E^{P_h}_{P_l}$*(MLP, K-NN)* gerados e/ou do autoajuste de faixa intermediária aos mesmos inerentes.

Nos referidos casos, além de superarem a Sensibilidade do uso individual da rede MLP, os modelos $E^{P_h}_{P_l}$*(MLP, K-NN)* gerados equivalem em termos das métricas de desempenho consideradas aos valores obtidos pelos modelos concorrentes gerados pelo uso individual do K-NN que, conforme visto na Seção 2.6, possui desempenho reconhecidamente interessante para o problema em questão; além disso, esses modelos também possuem desempenho equivalente ao modelo $E^{P_{h'}}_{P_{l'}}$*(MLP, K-NN)* que representa o que melhor seria possível obter em termos teóricos no que diz respeito à autoajuste de faixa intermediária considerando a técnica de *pipeline* de fluxo condicional. Mediante essa análise, tratam-se, portanto, de oportunas e desejáveis as equivalências identificadas. A ideia aqui apresentada é ainda reforçada pelas análises presentes na Subseção 5.4.3.

Por fim, é válido destacar que as intuições adquiridas em experimentos preliminares que apontavam um bom desempenho de modelos de combinação de classificadores com hiperparâmetros otimizados do tipo $E^{P_l}_{P_l}$*(MLP, K-NN)* se mostraram assertivas: de fato, boa parte das vezes, os modelos do tipo $E^{P_h}_{P_l}$*(MLP, K-NN)*, com autoajuste da faixa intermediária, tiveram o mesmo desempenho que os seus concorrentes de faixa intermediária fixa $h = 1$ e $l = 1$, com uma ocasião específica de vantagem do último em relação ao primeiro em termos da métrica Especificidade (Tabela 5.6). No outro extremo, na maioria das vezes os modelos concorrentes de combinação de classificadores com hiperparâmetros otimizados do tipo $E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* não lograram êxito em superar ou se igualar a suas contra-partes de faixa intermediária autoajustada.

Concluídas as análises propostas nesta subseção, a subseção que se inicia a seguir apresenta as análises comparativas correspondentes aos modelos combinados de classificadores gerados pelo EMHOSIR em relação à combinação de classificadores proposta por Souza (2018).

5.4.2 Comparação entre Modelos Gerados pelo EMHOSIR e o Método de Combinação de Classificadores de Souza (2018)

A presente subseção apresenta uma análise comparativa entre os modelos de combinação de classificadores gerados pelo EMHOSIR nos experimentos da Seção 4.5 com o método de combinação de classificadores de Souza (2018). Conforme adiantado no início desta seção, a comparação realizada se dá em termos das métricas de desempenho extraídas ao final da Fase 3 do EMHOSIR, acrescida, onde aplicável, da métrica correspondente à quantidade média de exemplos enviados ao classificador $C2$, rotulada apenas como "C2"em todas as tabelas comparativas.

Em seu trabalho, Souza (2018) extraiu métricas de desempenho tanto de seu modelo de combinação de classificadores $E^{P_{25}}_{P_{99}}$*(MLP, K-NN)*, quanto do uso individual da rede MLP. No caso da rede MLP individual, apenas a métrica Acurácia fora extraída; as métricas extraídas serviram para que fosse possível mensurar a melhoria de sua técnica em detrimento do uso exclusivo da rede MLP. Dada a disponibilidade de métricas relatada, as comparações realizadas nesta subseção foram realizadas da seguinte forma:

- O modelo de combinação de classificadores $E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* de Souza (2018) foi comparado em termos de quantidade média de exemplos enviados ao classificador $C2$, acurácia, sensibilidade e especificidade aos modelos de combinação de classificadores $E^{P_h}_{P_i}$*(MLP, K-NN)* gerados pelo EMHOSIR em cada experimento realizado. Esse mesmo modelo também foi comparado, apenas em termos de Acurácia, ao uso individual das redes MLP de hiperparâmetros otimizados geradas pelo EMHOSIR;
- O modelo individual de rede MLP do trabalho de Souza (2018), com os hiperparâmetros fixos, foi comparado ao uso individual das redes MLP de hiperparâmetros otimizados geradas pelo EMHOSIR em termos da métrica Acurácia - a única métrica disponível em comum a ambos;

As tabelas presentes no decorrer desta subseção, se organizam de modo que os modelos comparados, oriundos do trabalho de Souza (2018), se encontram sempre sobre os modelos a ele contrapostos, em linhas com cor de fundo diferenciada; por sua vez, sob esses modelos, dispõe-se os modelos gerados pelos experimentos realizados neste trabalho. A ocorrência de diferença estatisticamente significante entre as métricas dos modelos, observável pelos *p-valores* do teste não-paramétrico de Kruskal-Wallis e do pós-teste de Dunn da referida comparação, fora destacada em negrito.

A Tabela 5.13 apresenta a comparação entre os modelos de rede MLP individual e combinação de classificadores de Souza (2018) à rede MLP individual e combinação de classificadores do EMHOSIR para Busca em Grade sobre a base de dados *NOM_TO_INT*.

Nesse experimento, a única métrica que o modelo de combinação de classificadores ge-

Tabela 5.13: Comparativo: Métricas de Modelos de Souza (2018) x Métricas de Modelos do EMHOSIR com Busca em Grade sobre a Base de Dados *NOM_TO_INT*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	66,12	$1{,}24 \times 10^{-2}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	**4,76**	$\mathbf{1{,}74 \times 10^{-2}}$	**61,37**	$\mathbf{1{,}57 \times 10^{-4}}$	$\mathbf{1{,}57 \times 10^{-4}}$
AC	MLP | Combinação de Classificadores de Souza (2018)	97,36	$8{,}22 \times 10^{-3}$			
	MLP | Busca em Grade - *NOM_TO_INT*	**99,67**	$\mathbf{7{,}69 \times 10^{-4}}$	**2,31**	$\mathbf{1{,}57 \times 10^{-4}}$	$\mathbf{1{,}57 \times 10^{-4}}$
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,70	$5{,}13 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	**99,75**	$\mathbf{6{,}33 \times 10^{-4}}$	**0,05**	$\mathbf{2{,}82 \times 10^{-2}}$	$\mathbf{2{,}82 \times 10^{-2}}$
	MLP | Busca em Grade - *NOM_TO_INT*	99,67	$7{,}69 \times 10^{-4}$	0,03	$4{,}50 \times 10^{-1}$	
SE	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,53	$5{,}45 \times 10^{-3}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,69	$1{,}02 \times 10^{-3}$	0,16	$8{,}20 \times 10^{-1}$	
ES	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,70	$6{,}93 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	**99,81**	$\mathbf{5{,}62 \times 10^{-4}}$	**0,10**	$\mathbf{4{,}04 \times 10^{-3}}$	$\mathbf{4{,}04 \times 10^{-3}}$

rado pelo EMHOSIR não foi capaz de superar em relação à combinação de classificadores de Souza (2018) foi a Sensibilidade: apesar dos valores nominais do modelo gerado pelo EMHOSIR superarem os obtidos pela abordagem de Souza (2018), não se observou significância estatística na diferença obtida, de modo que ambas as soluções se mostraram equivalentes em termos da métrica em questão. Na comparação entre modelos sobre métricas realizada, destaca-se ainda a superioridade em termos de Acurácia da rede MLP com parâmetros otimizados gerada pelo EMHOSIR, quando comparada à rede MLP de hiperparâmetros fixos utilizada por Souza (2018). Conforme pode ser visto, a diferença é de $2,31\%$ em média, o que faz com que o uso exclusivo da rede MLP gerada pelo EMHOSIR tenha desempenho comparável ao da combinação de classificadores de Souza (2018) em termos de Acurácia (a primeira com $99,7\%$ e a última com $99,67\%$). Como último destaque dessa análise, pode-se observar que, para superar em quase todas as métricas o desempenho da abordagem proposta por Souza (2018), o modelo $E^{P_h}_{P_l}$*(MLP, K-NN)*, gerado pelo EMHOSIR, enviou em média apenas $4,76\%$ das instâncias/exemplos classificados à $C2$: um diferença média da ordem de $61,37\%$ à enviada pelo modelo $E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* gerado pelo método de Souza (2018).

Dando prosseguimento à análise, na Tabela 5.14 é possível ver a comparação entre os modelos de rede MLP individual e combinação de classificadores de Souza (2018) à rede MLP individual e combinação de classificadores do EMHOSIR para Busca Aleatória sobre a base de dados *NOM_TO_INT*.

Em termos de comparação, a mudança no método de otimização de hiperparâmetros realizada sobre a mesma base de dados fez com que os fenômenos observados na análise anterior se repetissem, o que significa dizer que as ocasiões de superioridade em termos de métricas comparando modelos também se fazem presentes, com iguais destaques relacionados à equivalência em termos de Sensibilidade entre os modelos de combinações de classificadores de ambas as abordagens, à superioridade da Acurácia do uso exclusivo da rede MLP de hiperparâmetros otimizados do EMHOSIR em relação ao uso exclusivo da rede MLP de hiperparâmetros fixos de Souza (2018), à equivalência entre a rede MLP de hiperparâmetros otimizados do EMHO-

Tabela 5.14: Comparativo: Métricas de Modelos de Souza (2018) x Métricas de Modelos do EMHOSIR com Busca Aleatória sobre a Base de Dados *NOM_TO_INT*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	66,12	$1{,}24 \times 10^{-2}$			
	$\boldsymbol{E^{P_{99}}_{P_1}}$***(MLP, K-NN)*** **| Busca Aleatória -** ***NOM_TO_INT***	**8,32**	$\mathbf{5{,}87 \times 10^{-2}}$	**57,81**	$\mathbf{1{,}57 \times 10^{-4}}$	$\mathbf{1{,}57 \times 10^{-4}}$
AC	MLP | Combinação de Classificadores de Souza (2018)	97,36	$8{,}22 \times 10^{-3}$			
	MLP | Busca Aleatória - ***NOM_TO_INT***	**99,70**	$\mathbf{7{,}73 \times 10^{-4}}$	**2,34**	$\mathbf{1{,}56 \times 10^{-4}}$	$\mathbf{1{,}56 \times 10^{-4}}$
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,70	$5{,}13 \times 10^{-4}$			
	$\boldsymbol{E^{P_{99}}_{P_1}}$***(MLP, K-NN)*** **| Busca Aleatória -** ***NOM_TO_INT***	**99,77**	$\mathbf{4{,}86 \times 10^{-4}}$	**0,07**	$\mathbf{1{,}55 \times 10^{-2}}$	$\mathbf{1{,}55 \times 10^{-2}}$
	MLP | Busca Aleatória - *NOM_TO_INT*	99,70	$7{,}73 \times 10^{-4}$	0,00	$8{,}20 \times 10^{-1}$	
SE	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,53	$5{,}45 \times 10^{-3}$			
	$E^{P_{99}}_{P_1}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,73	$8{,}89 \times 10^{-4}$	0,21	$2{,}26 \times 10^{-1}$	
ES	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,70	$6{,}93 \times 10^{-4}$			
	$\boldsymbol{E^{P_{99}}_{P_1}}$***(MLP, K-NN)*** **| Busca Aleatória -** ***NOM_TO_INT***	**99,80**	$\mathbf{4{,}89 \times 10^{-4}}$	**0,10**	$\mathbf{3{,}17 \times 10^{-3}}$	$\mathbf{3{,}17 \times 10^{-3}}$

SIR à abordagem de combinação de classificadores de Souza (2018) em termos de Acurácia e à diferença notória em favor do EMHOSIR em termos da quantidade de instâncias enviadas à $C2$ pelos modelos de combinação de classificadores propostos por cada trabalho.

Finalizando as análises relativas aos três experimentos realizados sobre a base de dados *NOM_TO_INT*, na Tabela 5.15 é possível ver a comparação entre os modelos de rede MLP individual e combinação de classificadores de Souza (2018) à rede MLP individual e combinação de classificadores do EMHOSIR para Otimização Bayesiana sobre a base de dados *NOM_TO_INT*.

Tabela 5.15: Comparativo: Métricas de Modelos de Souza (2018) x Métricas de Modelos do EMHOSIR com Otimização Bayesiana sobre a Base de Dados *NOM_TO_INT*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	66,12	$1{,}24 \times 10^{-2}$			
	$\boldsymbol{E^{P_{99}}_{P_1}}$***(MLP, K-NN)*** **| Otimização Bayesiana -** ***NOM_TO_INT***	**6,49**	$\mathbf{6{,}20 \times 10^{-2}}$	**59,63**	$\mathbf{1{,}57 \times 10^{-4}}$	$\mathbf{1{,}57 \times 10^{-4}}$
AC	MLP | Combinação de Classificadores de Souza (2018)	97,36	$8{,}22 \times 10^{-3}$			
	MLP | Otimização Bayesiana - ***NOM_TO_INT***	**99,67**	$\mathbf{4{,}32 \times 10^{-4}}$	**2,31**	$\mathbf{1{,}56 \times 10^{-4}}$	$\mathbf{1{,}56 \times 10^{-4}}$
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,70	$5{,}13 \times 10^{-4}$			
	$\boldsymbol{E^{P_{99}}_{P_1}}$***(MLP, K-NN)*** **| Otimização Bayesiana -** ***NOM_TO_INT***	**99,76**	$\mathbf{4{,}95 \times 10^{-4}}$	**0,05**	$\mathbf{2{,}81 \times 10^{-2}}$	$\mathbf{2{,}81 \times 10^{-2}}$
	MLP | Otimização Bayesiana - *NOM_TO_INT*	99,67	$4{,}32 \times 10^{-4}$	0,03	$1{,}12 \times 10^{-1}$	
SE	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,53	$5{,}45 \times 10^{-3}$			
	$E^{P_{99}}_{P_1}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,70	$8{,}74 \times 10^{-4}$	0,17	$7{,}62 \times 10^{-1}$	
ES	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,70	$6{,}93 \times 10^{-4}$			
	$\boldsymbol{E^{P_{99}}_{P_1}}$***(MLP, K-NN)*** **| Otimização Bayesiana -** ***NOM_TO_INT***	**99,81**	$\mathbf{5{,}92 \times 10^{-4}}$	**0,10**	$\mathbf{4{,}04 \times 10^{-3}}$	$\mathbf{4{,}04 \times 10^{-3}}$

Nesse último experimento realizado sobre a base de dados *NOM_TO_INT*, todas as relações observadas nos experimentos anteriores entre métricas e modelos sobre essa mesma base de dados também foram mantidas, sendo dispensável repetir as considerações já tecidas ao propósito de poupar este texto de demasiada repetição. Em tempo, é válido dizer que o mesmo ocorreu em outros dois experimentos apresentados em sequência: o experimento com Busca em Grade e o experimento com Busca Aleatória, ambos para a base de dados *NOM_TO_BIN*. Sendo assim, para os referidos experimentos, apenas as tabelas comparativas são apresenta-

das, servindo ao propósito de relatar a grandeza em que se comparam os modelos gerados pelo EMHOSIR nos mesmos em relação aos modelos da abordagem proposta por Souza (2018) em termos das métricas aferidas.

No que diz respeito às análises referentes aos experimentos realizados sobre a base de dados *NOM_TO_BIN*, a Tabela 5.16 apresenta a comparação entre os modelos de rede MLP individual e combinação de classificadores de Souza (2018) à rede MLP individual e combinação de classificadores do EMHOSIR para Busca em Grade sobre a base de dados *NOM_TO_BIN*.

Tabela 5.16: Comparativo: Métricas de Modelos de Souza (2018) x Métricas de Modelos do EMHOSIR com Busca em Grade sobre a Base de Dados *NOM_TO_BIN*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	66,12	$1{,}24 \times 10^{-2}$			
	$\boldsymbol{E^{P}_{P_i}}$***(MLP, K-NN)*** **| Busca em Grade -** ***NOM_TO_BIN***	**2,81**	$\mathbf{8{,}40 \times 10^{-3}}$	**63,32**	$\mathbf{1{,}57 \times 10^{-4}}$	$\mathbf{1{,}57 \times 10^{-4}}$
AC	MLP | Combinação de Classificadores de Souza (2018)	97,36	$8{,}22 \times 10^{-3}$			
	MLP | Busca em Grade - ***NOM_TO_BIN***	**99,74**	$\mathbf{4{,}50 \times 10^{-4}}$	**2,39**	$\mathbf{1{,}54 \times 10^{-4}}$	$\mathbf{1{,}54 \times 10^{-4}}$
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,70	$5{,}13 \times 10^{-4}$			
	$\boldsymbol{E^{P}_{P_i}}$***(MLP, K-NN)*** **| Busca em Grade -** ***NOM_TO_BIN***	**99,77**	$\mathbf{4{,}48 \times 10^{-4}}$	**0,07**	$\mathbf{1{,}01 \times 10^{-2}}$	$\mathbf{1{,}01 \times 10^{-2}}$
	MLP | Busca em Grade - *NOM_TO_BIN*	99,74	$4{,}50 \times 10^{-4}$	0,04	$9{,}58 \times 10^{-2}$	
SE	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,53	$5{,}45 \times 10^{-3}$			
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,75	$4{,}76 \times 10^{-4}$	0,22	$1{,}12 \times 10^{-1}$	
ES	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,70	$6{,}93 \times 10^{-4}$			
	$\boldsymbol{E^{P_h}_{P_i}}$***(MLP, K-NN)*** **| Busca em Grade -** ***NOM_TO_BIN***	**99,79**	$\mathbf{6{,}49 \times 10^{-4}}$	**0,09**	$\mathbf{1{,}55 \times 10^{-2}}$	$\mathbf{1{,}55 \times 10^{-2}}$

Ainda com respeito às análises referentes aos experimentos realizados sobre a base de dados *NOM_TO_BIN*, tem-se, na Tabela 5.17, a comparação entre os modelos de rede MLP individual e combinação de classificadores de Souza (2018) à rede MLP individual e combinação de classificadores do EMHOSIR para Busca Aleatória sobre a base de dados *NOM_TO_BIN*.

Tabela 5.17: Comparativo: Métricas de Modelos de Souza (2018) x Métricas de Modelos do EMHOSIR com Busca Aleatória sobre a Base de Dados *NOM_TO_BIN*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	66,12	$1{,}24 \times 10^{-2}$			
	$\boldsymbol{E^{P_h}_{P_i}}$***(MLP, K-NN)*** **| Busca Aleatória -** ***NOM_TO_BIN***	**3,25**	$\mathbf{1{,}28 \times 10^{-2}}$	**62,88**	$\mathbf{1{,}57 \times 10^{-4}}$	$\mathbf{1{,}57 \times 10^{-4}}$
AC	MLP | Combinação de Classificadores de Souza (2018)	97,36	$8{,}22 \times 10^{-3}$			
	MLP | Busca Aleatória - ***NOM_TO_BIN***	**99,73**	$\mathbf{6{,}49 \times 10^{-4}}$	**2,37**	$\mathbf{1{,}57 \times 10^{-4}}$	$\mathbf{1{,}57 \times 10^{-4}}$
	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,70	$5{,}13 \times 10^{-4}$			
	$\boldsymbol{E^{P_h}_{P_i}}$***(MLP, K-NN)*** **| Busca Aleatória -** ***NOM_TO_BIN***	**99,76**	$\mathbf{5{,}58 \times 10^{-4}}$	**0,06**	$\mathbf{4{,}91 \times 10^{-2}}$	$\mathbf{4{,}91 \times 10^{-2}}$
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,73	$6{,}49 \times 10^{-4}$	0,03	$2{,}26 \times 10^{-1}$	
SE	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,53	$5{,}45 \times 10^{-3}$			
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,74	$7{,}72 \times 10^{-4}$	0,21	$2{,}26 \times 10^{-1}$	
ES	$E^{P_{25}}_{P_{99}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,70	$6{,}93 \times 10^{-4}$			
	$\boldsymbol{E^{P_h}_{P_i}}$***(MLP, K-NN)*** **| Busca Aleatória -** ***NOM_TO_BIN***	**99,78**	$\mathbf{5{,}32 \times 10^{-4}}$	**0,08**	$\mathbf{1{,}25 \times 10^{-2}}$	$\mathbf{1{,}25 \times 10^{-2}}$

Por fim, na Tabela 5.18, é apresentada a comparação entre os modelos de rede MLP individual e combinação de classificadores de Souza (2018) à rede MLP individual e combinação de classificadores do EMHOSIR para Otimização Bayesiana sobre a base de dados *NOM_TO_BIN*.

Tabela 5.18: Comparativo: Métricas de Modelos de Souza (2018) x Métricas de Modelos do EMHOSIR com Otimização Bayesiana sobre a Base de Dados *NOM_TO_BIN*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E_{P_{99}^{25}}^{P_{25}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	66,12	$1{,}24 \times 10^{-2}$			
	$E_{P_l}^{P_h}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	**2,36**	$\mathbf{1{,}42 \times 10^{-2}}$	**63,76**	$\mathbf{1{,}56 \times 10^{-4}}$	$\mathbf{1{,}56 \times 10^{-4}}$
AC	MLP | Combinação de Classificadores de Souza (2018)	97,36	$8{,}22 \times 10^{-3}$			
	MLP | Otimização Bayesiana - *NOM_TO_BIN*	**99,75**	$\mathbf{3{,}41 \times 10^{-4}}$	**2,39**	$\mathbf{1{,}57 \times 10^{-4}}$	$\mathbf{1{,}57 \times 10^{-4}}$
	$E_{P_{99}^{25}}^{P_{25}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,70	$5{,}13 \times 10^{-4}$			
	$E_{P_l}^{P_h}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	**99,78**	$\mathbf{2{,}99 \times 10^{-4}}$	**0,08**	$\mathbf{8{,}73 \times 10^{-4}}$	$\mathbf{8{,}73 \times 10^{-4}}$
	MLP | Otimização Bayesiana - *NOM_TO_BIN*	**99,75**	$\mathbf{3{,}41 \times 10^{-4}}$	**0,05**	$\mathbf{2{,}33 \times 10^{-2}}$	$\mathbf{2{,}33 \times 10^{-2}}$
SE	$E_{P_{99}^{25}}^{P_{25}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,53	$5{,}45 \times 10^{-3}$			
	$E_{P_l}^{P_h}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,74	$6{,}58 \times 10^{-4}$	0,21	$1{,}12 \times 10^{-1}$	
ES	$E_{P_{99}^{25}}^{P_{25}}$*(MLP, K-NN)* | Combinação de Classificadores de Souza (2018)	99,70	$6{,}93 \times 10^{-4}$			
	$E_{P_l}^{P_h}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	**99,82**	$\mathbf{3{,}64 \times 10^{-4}}$	**0,12**	$\mathbf{6{,}63 \times 10^{-4}}$	$\mathbf{6{,}63 \times 10^{-4}}$

Nesse experimento, além das relações entre métricas e modelos já discutidas e apresentadas até o momento, observou-se que o uso exclusivo da rede MLP com hiperparâmetros otimizados na Fase 2 do EMHOSIR fora capaz, não apenas de equivaler ao modelo combinado de classificadores de Souza (2018), mas de superá-lo em termos da métrica Acurácia.

Salvo pontuais exceções de equivalência, que por vezes, inclusive, revelam superioridade do EMHOSIR em relação aos modelos gerados no trabalho de Souza (2018)[†], o que se observa, nas comparações apresentadas nesta subseção, é uma clara e recorrente ocorrência de superioridade dos modelos gerados pelo EMHOSIR em comparação com os gerados no trabalho de Souza (2018). No entanto, não é possível, por meio do que foi até então apresentado, fazer quaisquer constatações relacionadas a de que modo os modelos gerados pelo EMHOSIR em cada experimento se relacionam entre si. A subseção apresentada a seguir traz os dados e análises direcionados a tais comparações.

5.4.3 Modelos Gerados pelo EMHOSIR Comparados entre Si

Conforme apresentado de forma recorrente nesta seção, foram realizados seis experimentos sendo que cada um gerou um modelo $E_{P_l}^{P_h}$*(MLP, K-NN)* segundo a base de dados considerada e a otimização de hiperparâmetros correspondente. Combinados dois a dois, são necessárias 15 comparações[‡] ao todo para que seja possível realizar uma análise completa de todos os modelos entre si. Dado seu quantitativo, para uma melhor fluidez textual, as tabelas que embasam à análise realizada nesta subseção encontram-se no Apêndice B do trabalho.

Ao comparar os modelos gerados pelos seis experimentos, o que se observa, em termos gerais, é a equivalência dos mesmos em relação à maioria das métricas, com uma sutil vantagem dos modelos gerados à partir da base de dados *NOM_TO_BIN* em relação aos gerados

[†]É o caso do uso exclusivo da rede MLP com hiperparâmetros otimizados se equiparando, em termos de Acurácia, ao modelo de combinação de classificadores $E_{P_{99}^{25}}^{P_{25}}$*(MLP, K-NN)* de Souza (2018)

[‡]Relativo à fórmula de combinações $C_{(n,p)} = \frac{n!}{p! \cdot (n-p)!}$, com $n = 6$ e $p = 2$

pela *NOM_TO_INT*, independentemente da técnica de otimização de hiperparâmetros empregada (vide tabelas B.3, B.4, B.5, B.7, B.8, B.9, B.10, B.12). A vantagem marginal mencionada diz respeito à quantidade média de instâncias/exemplos enviados ao classificador $C2$, frequentemente decorrente de melhores métricas (Acurácia e/ou Sensibilidade) das redes MLP individuais obtidas nos experimentos sobre a base de dados *NOM_TO_BIN* em relação a essas redes MLP geradas nos experimentos sobre a base de dados *NOM_TO_INT*. Excepcionalmente, tem-se a comparação entre o modelo obtido com Otimização Bayesiana sobre a base de dados *NOM_TO_INT* e o modelo obtido com Busca Aleatória sobre a base de dados *NOM_TO_BIN* (Tabela B.11), em que, apesar da diferença nominal expressiva em termos da quantidade média de instâncias/exemplos enviados ao classificador $C2$, não se verificou vantagem de um sobre o outro em termos de significância estatística. Ainda assim, mesmo para esse caso, o uso individual da rede MLP obtida por meio da Busca Aleatória sobre a base de dados *NOM_TO_BIN* foi capaz de obter melhor Acurácia e Sensibilidade que o uso exclusivo da rede MLP obtida por Otimização Bayesiana sobre a base de dados *NOM_TO_INT*. Quaisquer outras diferenças percebidas entre métricas e modelos destacadas nas tabelas do Apêndice B foram entendidas como ocasionais.

Dada a consistência das situações em que os modelos gerados sobre a base de dados *NOM_TO_BIN* superam os gerados sobre a base de dados *NOM_TO_INT*, entende-se que, em relação a essa última, há um indicativo de vantagens relativas aos pré-processamentos realizados sobre a base de dados NSL-KDD restrita na geração da base de dados pré-processada *NOM_TO_BIN*. Ao que tudo indica, a referida base de dados, quando comparada a base de dados *NOM_TO_INT*, possibilita a obtenção de redes MLP de melhor desempenho independentemente da técnica de otimização de hiperparâmetros empregada. Com redes MLP mais assertivas, minimiza-se a necessidade de acionamento do modelo gerado pelo K-NN (classificador $C2$ na implementação do EMHOSIR realizada).

Em contrapartida, com o uso da base de dados *NOM_TO_INT*, as redes MLP geradas pela otimização de hiperparâmetros possuem maior margem de melhoria; o papel do autoajuste da faixa intermediária se mostra mais proeminente, identificando a menor faixa intermediária possível para se compensar a imprecisão das redes MLP geradas e obter desempenho comparável ao dos modelos de combinação de classificadores $E^{P_h}_{P'_h}$*(MLP, K-NN)* da base de dados *NOM_TO_BIN* ao preço de mais exemplos classificados pelo K-NN.

Findada essa análise, a seção a seguir apresenta uma comparação dos resultados experimentais obtidos com o EMHOSIR comparados aos resultados obtidos por outras técnicas apresentadas no estado da arte presentes no Capítulo 3.

5.4.4 EMHOSIR Comparado a Outros Métodos do Estado da Arte

Apesar de aspectos que dificultam a comparação direta com outras técnicas apresentadas no estado da arte presente no Capítulo 3, tais como método de validação e bases de dados

de *benchmark* distintas, num esforço de situar o EMHOSIR ante o estado da arte de detecção de intrusão em redes de computadores por meio de aprendizagem de máquina, na Tabela 5.19 pode-se verificar o posicionamento da implementação do EMHOSIR realizada em termos de acurácia comparada a outras soluções identificadas no estado da arte. As métricas apresentadas na referida tabela dizem respeito aos melhores casos em termos de desempenho obtido pelos métodos, uma vez que parte dos mesmos apresentou diferentes experimentos, variando em termos de bases de dados ou de tipo de detecção (anomalia, mal-uso, etc.). É válido dizer que, entre os métodos do estado da arte apresentados que fazem uso da base de dados de *benchmark* NSL-KDD, o EMHOSIR, conforme implementado, obteve a melhor acurácia.

Tabela 5.19: Posição em Termos de Acurácia da Implementação do EMHOSIR em Relação a Diversos Métodos do Estado da Arte

Posição	Método	Acurácia (%)
1	Kanimozhi & Jacob (2019)	99,97
2	**EMHOSIR**	**99,78**
3	Aljawarneh et al. (2018)	99,75
4	Mohan (2017)	99,67
5	Kasongo & Sun (2020)	99,66
6	Ali Alheeti & McDonald-Maier (2016)	99,18
7	Reazul et al. (2017)	98,26
8	Shenfield et al. (2018)	98,00
9	Liu et al. (2020)	96,86
10	Hamed et al. (2018)	92,90
11	Lu et al. (2018)	87,58

Esta subseção conclui a apresentação dos resultados e a análise crítica dos mesmos. A seção seguinte realiza o desfecho deste capítulo, com uma visão geral de tudo que fora apresentado no mesmo.

5.5 Considerações Finais

O presente capítulo se encerra com uma análise final a respeito dos resultados e considerações apresentados em seu decorrer. Nesse sentido, um primeiro comentário válido de ser feito diz respeito aos valores de hiperparâmetros encontrados nos experimentos em razão da Fase 2 do EMHOSIR. As discussões sobre tal questão foram postergadas pois neste trabalho não há qualquer pretensão de se recomendar quaisquer um dos valores de hiperparâmetros como mais ou menos promissores: inerente ao EMHOSIR enquanto método é identificá-los em tempo de

sua aplicação e não *a priori*. Ainda assim, alguns comentários relacionados a esses valores se fazem condizentes.

Ao observar as tabelas 5.1, 5.3, 5.5, referente aos valores de hiperparâmetros encontrados nos experimentos realizados para a base de dados *NOM_TO_INT*, percebe-se que, apesar de ter sido considerada mesmo após a redução do espaço de busca de otimização (Seção 4.4), a função de ativação Sigmóide se fez ausente como hiperparâmetro selecionado na Fase 2 do EMHOSIR. A situação não mudou em termos dos experimentos realizados sobre a base de dados *NOM_TO_BIN*, cujos resultados constam nas tabelas 5.7, 5.7, 5.11. Um outro fenômeno percebido diz respeito a escolha recorrente da função de ativação Unidade Linear Retificada para um dos hiperparâmetros específicos: em todos os experimentos, independente de base de dados e método de otimização de hiperparâmetros, a referida função foi selecionada pelas otimizações de hiperparâmetros para compor as função de ativação das camadas internas da rede MLP. Por fim, salvo o experimento que realizou a Busca Aleatória sobre a base de dados *NOM_TO_BIN*, em todos os demais a otimização de hiperparâmetros escolheu o valor máximo disponível no OSSSP *internal_units* correspondente ao número de neurônios nas camadas internas. A tendência de isso ocorrer já havia sido identificada na Seção 4.4.

Em termos das análises comparativas dos resultados encontrados, basicamente, os modelos combinados de classificadores $E_{P_l}^{P_h}$*(MLP, K-NN)* gerados pelo EMHOSIR nos seis experimentos relatados foram comparados:

- Com variações dos mesmos em termos de faixa intermediária (Subseção 5.4.1);
- Com a utilização dos classificadores individuais que os compõe (Subseção 5.4.1);
- Com o modelo de combinação de classificadores de Souza (2018) (Subseção 5.4.2);
- Com a utilização da rede MLP individual de hiperparâmetros fixos que compõe o método proposto por Souza (2018) (Subseção 5.4.2);
- Uns aos outros;
- Com outras técnicas encontradas na literatura;

Nas comparações realizadas em relação a variações dos modelos gerados com faixas intermediárias diversas observou-se, em boa parte das ocasiões, equivalência, em termos de métricas de desempenho. Conforme já mencionado neste capítulo, as equivalências possuem interpretações intrínsecas aos modelos que se equivalem, não podendo ser consideradas desejáveis ou indesejáveis exclusivamente em razão de sua ocorrência. Pode-se, contudo, afirmar que a geração de um modelo $E_{P_l}^{P_h}$*(MLP, K-NN)* com desempenhos equivalentes à sua contra-parte ideal $E_{P_{l'}}^{P_{h'}}$*(MLP, K-NN)* é desejável, ao passo que gerar um modelo equivalente, em termos de métricas, ao modelo correspondente $E_{P_l}^{P_l}$*(MLP, K-NN)* põe a prova a necessidade de se realizar o autoajuste da faixa intermediária prescrito pela Fase 3 do EMHOSIR. Todavia, faz-se necessária a condução de uma maior quantidade de experimentos estudando a relação específica entre modelos do tipo $E_{P_l}^{P_h}$*(MLP, K-NN)* comparados a modelos correspondentes $E_{P_l}^{P_l}$*(MLP, K-NN)* para

que se possa afirmar que sob as condições propiciadas pelo EMHOSIR, com destaque para a otimização de hiperparâmetros da Fase 2, a faixa intermediária $h = 1, l = 1$ é tão boa quanto uma faixa intermediária obtida por meio de técnicas de autoajuste como no caso da aplicada.

Um outro fenômeno identificado que independe do método de otimização de hiperparâmetros diz respeito a diferença na margem de melhoria das redes MLP geradas a partir da base de dados *NOM_TO_INT* em relação as geradas a partir da base de dados *NOM_TO_BIN* em termos das métricas de desempenho consideradas. O que se observou foi que as redes MLP geradas a partir da base de dados *NOM_TO_INT* são muito mais suscetíveis a melhorias do que as geradas a partir da base de dados *NOM_TO_BIN*, estando essas últimas mais próximas de um desempenho ideal.

Na comparação dos modelos gerados pelo EMHOSIR com o uso individual dos classificadores que os compõe, obteve-se o resultado que já era esperado: a combinação de classificadores foi capaz de gerar modelos com desempenhos equivalentes ou melhores do que os de seus componentes, corroborando com os resultados encontrados pelo trabalho de Souza (2018).

Ao comparar os modelos gerados pelo EMHOSIR com os modelos do trabalho de Souza (2018) para os quais as métricas de desempenho foram reportadas, na maioria dos casos, os primeiros superaram os últimos; nas escassas situações em que tal hegemonia não se verificou, ao menos a equivalência fora garantida. Ainda assim, em todos os casos de combinação de classificadores, o EMHOSIR precisou enviar ao K-NN um número expressivamente menor de exemplos por meio do *pipeline* de fluxo condicional.

Quando comparados uns aos outros, os modelos gerados pelos EMHOSIR nos seis experimentos,independentemente do método de otimização de hiperparâmetros utilizado, demonstraram uma consistente capacidade dos modelos gerados a partir da base de dados *NOM_TO_BIN* superarem aqueles gerados a partir da base de dados *NOM_TO_INT* em termos de número de exemplos enviados ao K-NN.

Na comparação com outras técnicas de detecção de intrusão em redes de computadores do estado da arte, o EMHOSIR apresentou desempenho superior a maioria em termos da métrica acurácia, sendo o de melhor desempenho nessa métrica entre aqueles que fazem uso da base de dados de *benchmark* NSL-KDD.

O presente capítulo se encerra com as análises apresentadas nesta seção. A seguir, no Capítulo 6, a conclusão do texto desta dissertação é apresentada, com um retrospecto crítico de tudo que até então foi apresentado.

Capítulo 6

Conclusão

A seção precedente (Seção 5.5) antecipa boa parte das conclusões que se teve com o desfecho do trabalho proposto. No entanto, uma palavra final compondo todo o conteúdo apresentado se faz necessária de modo a sumarizar os aprendizados depreendidos ao longo do desenvolvimento da pesquisa ao mesmo inerente.

Conforme apresentado no Capítulo 1, o problema de intrusão em rede de computadores, apesar de não ser novo, mantém-se pungente, atual, emergente e possui potencial de impactar na vida de bilhões de pessoas ao redor do mundo que fazem uso das TIC no dia-a-dia.

Iniciativas, como as apresentadas no Capítulo 3 possuem diferentes níveis de eficácia no tratamento do problema, preocupando-se com aspectos distintos relacionados ao mesmo e empregando as mais diversas técnicas no encalço de uma solução.

A proposta deste trabalho, apresentada detalhadamente no Capítulo 4 faz uso das tecnologias e ferramental apresentado no Capítulo 2, de modo a compor um método chamado EMHOSIR, que combina de forma autoajustada, classificadores baseados em aprendizagem de máquina, com a aplicação prévia de otimização de hiperparâmetros.

Os resultados obtidos pela implementação do trabalho proposto, apresentados no Capítulo 5, permitem comprovar a hipótese que se tinha (Subseção 1.1.2) de que modelos de combinação de classificadores voltados a detecção de intrusão em redes de computadores podem ter sua acurácia, custo computacional e demais métricas de desempenho melhoradas por meio de otimização de hiperparâmetros e autoajuste da faixa intermediária. Em termos de objetivos, conforme enunciado na Sub-subseção 1.1.1, observa-se que:

- A não ser em termos de tempo e custo computacional, com relação a métricas de desempenho, não há grandes diferenças em termos práticos em relação às otimizações realizadas por cada uma das técnicas de otimização de hiperparâmetros, tendo todas elas encontrado resultados equivalentes. No entanto, os pré-processamentos realizados, com a criação das bases de dados pré-processadas, mostraram uma marginal vantagem do método em atingir boas métricas de desempenho sobre a base de dados *NOM_TO_BIN* em relação a base de dados *NOM_TO_INT*. A situação reforça o potencial do pré-processamento na obtenção de boas soluções dentro do domínio, deixando claro o porquê de boa parte dos trabalhos

identificados no estado da arte se concentrarem nessas atividades (Mohan, 2017; Selvakumar & Muneeswaran, 2019; Liu et al., 2020; Vijayanand et al., 2018; Lu et al., 2018; Aljawarneh et al., 2018; Zhang & Zhu, 2018; Reazul et al., 2017; Hamed et al., 2018; Kasongo & Sun, 2020; Elkhadir & Mohammed, 2019; Aljawarneh et al., 2018; Cepheli et al., 2016);

- Em termos de instâncias com a necessidade de serem classificadas por $C2$, considerando o *pipeline* de fluxo condicional, houve uma redução drástica em favor da implementação do EMHOSIR em relação ao método de combinação de classificadores original de Souza (2018). A redução de instâncias a serem classificadas por $C2$ foi da ordem de 60%, com implicação num menor custo computacional da técnica proposta em relação a técnica de Souza (2018). Obviamente, a Fase 2 do EMHOSIR, referente à otimização de hiperparâmetros, acresce em diversas ordens o custo de se obter um modelo pronto para uso em relação ao método de Souza (2018). No entanto, uma vez prontos, a implementação do EMHOSIR realizada se mostra menos custosa em termos de tempo computacional;
- Comparando-se em termos de significância estatística as métricas de desempenho obtidas pelo novo método EMHOSIR em relação a proposta original de Souza (2018), observou-se que apenas a sensibilidade não pôde ser superada, tendo a mesma, no entanto, apresentado valores nominais melhores no EMHOSIR do que no método de Souza (2018). Também se observou que o uso exclusivo de uma rede MLP otimizada em termos de seus hiperparâmetros na Fase 2 do EMHOSIR foi capaz de obter equivalência em termos da métrica de desempenho ao modelo de combinação de classificadores de Souza (2018) ao ponto de, em um dos experimentos, a referida rede chegar ao ponto de, em termos de significância estatística, superar a acurácia alcançada pelo método de combinação de classificadores de Souza (2018);
- Conforme por ser visto na Subseção 5.4.4, o EMHOSIR apresenta desempenho satisfatório quanto comparado a outras técnicas de detecção de intrusão em rede de computadores, figurando entre as de melhor acurácia identificadas.

Feita a referida análise, conclui-se que o presente trabalho fora capaz de cumprir cada um dos objetivos a que se propôs quando em sua concepção, tendo a pesquisa sido satisfatória no que diz respeito a contribuir com o estado da arte da detecção de intrusão em redes de computadores com o uso de aprendizagem de máquina.

Observando o impacto nas métricas de desempenho, trabalhos futuros poderiam explorar implementações diversas do EMHOSIR, variando tanto os classificadores $C1$ e $C2$ quanto o método para autoajuste de faixa intermediária; outra possibilidade seria a utilização de outras técnicas de otimização de hiperparâmetros, ou até mesmo apostar em outros hiperparâmetros para a otimização. Ainda que se mantivesse várias das decisões de projeto da implementação do EMHOSIR que se fez, incluindo aí a Evolução Diferencial como método para autoajuste da faixa intermediária, no caso da minimização da função objetivo $z'(h, l)$ (Subseção 4.2.3),

valores mais agressivos de β_2 e w_1 poderiam ser usados abrindo mão do desempenho em termos de custo computacional para se tentar obter melhores valores nas demais métricas. Não obstante, o uso de uma quantidade ainda maior de neurônios na camada interna da rede MLP também se mostra bastante promissor, tendo em vista os indícios de melhoria da acurácia associados a esse hiperparâmetro descritos na Seção 4.4. Variações em termos de bases de dados de *benchmark* também poderiam corroborar com os resultados por este trabalho identificados, uma vez que existem bases de dados de *benchmark* mais modernas que a NSL-KDD; de fato, até mesmo o uso da base de dados NSL-KDD em sua vertente estendida poderia apresentar resultados distintos a partir dos quais novos *insights* poderiam surgir. Por fim, a aplicação do EMHOSIR a domínios alheios à detecção de intrusão em redes de computadores também se mostra interessante para avaliar a capacidade do método extrapolar o domínio para o o qual foi concebido.

Outras análises específicas voltadas ao desempenho do EMHOSIR em relação ao custo computacional/tempo de execução também se fazem necessárias, tendo em vista que as referidas analises foram apenas tangenciadas no sentido de se observar a quantidades de operações delegadas ao classificador $C2$, sendo, portando, limitadas. Tais análises também podem ser objeto de trabalhos futuros.

Referências Bibliográficas

Ahmad, A., Zainudin, W. S., Kama, M. N., Idris, N. B. & Saudi, M. M. (2018). State of the Art Intrusion Detection System for Cloud Computing, *International Journal of Communication Networks and Information Security (IJCNIS)* **10**(3): 2018. Citado na página 60.

Ali, A., Hamouda, W. & Uysal, M. (2015). Next generation m2m cellular networks: challenges and practical considerations, *IEEE Communications Magazine* **53**(9): 18–24. Citado na página 29.

Ali Alheeti, K. M. & McDonald-Maier, K. (2016). Hybrid intrusion detection in connected self-driving vehicles, *2016 22nd International Conference on Automation and Computing, ICAC 2016: Tackling the New Challenges in Automation and Computing* pp. 456–461. Citado 5 vezes nas páginas 19, 36, 61, 62 e 130.

Aljawarneh, S., Aldwairi, M. & Yassein, M. B. (2018). Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model, *Journal of Computational Science* **25**: 152–160.
URL: *http://dx.doi.org/10.1016/j.jocs.2017.03.006* Citado 6 vezes nas páginas 36, 40, 60, 64, 130 e 134.

Amer, S. H. & Hamilton, J. (2010). Intrusion detection systems (ids) taxonomy-a short review, *Defense Cyber Security* **13**(2): 23–30. Citado na página 36.

Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy, *Technical report*, Technical report. Citado na página 36.

Bace, R. & Mell, P. (2001). Intrusion detection systems, national institute of standards and technology (nist), *Technical Report 800-31* . Citado na página 36.

Bergstra, J. & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization, **13**: 281–305. Citado 4 vezes nas páginas 51, 52, 53 e 79.

Bhuyan, M. H., Bhattacharyya, D. K. & Kalita, J. K. (2014). Network anomaly detection: Methods, systems and tools, *IEEE Communications Surveys and Tutorials* **16**(1): 303–336. Citado na página 60.

Brochu, E., Cora, V. M. & De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, *arXiv preprint arXiv:1012.2599* . Citado 2 vezes nas páginas 54 e 55.

Cain, G. (2016). *Artificial Neural Networks : New Research.*, Computer Science, Technology and Applications, Nova Science Publishers, Inc.
URL: *http://search.ebscohost.com/login.aspx?direct=truedb=e000xwwAN=1440478lang=pt-brsite=eds-livescope=site* Citado na página 45.

Cepheli, Ö., Büyükçorak, S. & Karabulut Kurt, G. (2016). Hybrid Intrusion Detection System for DDoS Attacks, *Journal of Electrical and Computer Engineering* **2016**. Citado 4 vezes nas páginas 36, 60, 64 e 134.

Cover, T. & Hart, P. (1967). Nearest neighbor pattern classification, *IEEE Transactions on*

Information Theory **13**(1): 21–27. Citado na página 48.

Creech, G. & Hu, J. (2013). Generation of a new ids test dataset: Time to retire the kdd collection, pp. 4487–4492. cited By 103. Citado na página 64.

Crosbie, M. & Spafford, E. H. (1995). Defending a computer system using autonomous agents. Citado na página 36.

Dasarathy, B. V. (1991). Nearest neighbor (nn) norms: Nn pattern classification techniques, *IEEE Computer Society Tutorial* . Citado na página 48.

Debar, H., Dacier, M. & Wespi, A. (1999). Towards a taxonomy of intrusion-detection systems, *Computer networks* **31**(8): 805–822. Citado na página 36.

Devijver, P. A. & Kittler, J. (1982). *Pattern recognition: A statistical approach*, Prentice hall. Citado na página 48.

Dhanabal, L. & Shantharajah, S. (n.d.). A study on nsl-kdd dataset for intrusion detection system based on classification algorithms, *International Journal of Advanced Research in Computer and Communication Engineering* **4**(6): 446–452. Citado na página 40.

Dietterich, T. (1995). Overfitting and undercomputing in machine learning, *ACM computing surveys (CSUR)* **27**(3): 326–327. Citado na página 47.

Dyba, T., Kitchenham, B. A. & Jorgensen, M. (2005). Evidence-based software engineering for practitioners, *IEEE Software* **22**(1): 58–65. Citado na página 59.

Elkan, C. (2011). Nearest neighbor classification, *elkan@ cs. ucsd. edu*‖,‖ *January* **11**: 3. Citado na página 48.

Elkhadir, Z. & Mohammed, B. (2019). A cyber network attack detection based on GM Median Nearest Neighbors LDA, *Computers and Security* **86**: 63–74. Citado 5 vezes nas páginas 36, 40, 60, 63 e 134.

Estevez-Tapiador, J. M., Garcia-Teodoro, P. & Diaz-Verdejo, J. E. (2004). Anomaly detection methods in wired networks: a survey and taxonomy, *Computer Communications* **27**(16): 1569–1584. Citado na página 36.

Ethem, A. (2016). *Machine Learning : The New AI.*, MIT Press Essential Knowledge Series, The MIT Press.
URL: *http://search.ebscohost.com/login.aspx?direct=truedb=e000xwwAN=1369420lang=pt-brsite=eds-livescope=site* Citado na página 41.

Fernandes, G., Rodrigues, J. J., Carvalho, L. F., Al-Muhtadi, J. F. & Proença, M. L. (2019). A comprehensive survey on network anomaly detection, *Telecommunication Systems* **70**(3): 447–489.
URL: *https://doi.org/10.1007/s11235-018-0475-8* Citado na página 60.

Ghahramani, Z. (2004). Advanced lectures on machine learning, *Journal of the Royal Statistical Society* . Citado na página 42.

Goncalves, F., Ribeiro, B., Gama, O., Santos, A., Costa, A., Dias, B., MacEdo, J. & Nicolau, M. J. (2019). A Systematic Review on Intelligent Intrusion Detection Systems for VANETs, *International Congress on Ultra Modern Telecommunications and Control Systems and Workshops* **2019-October**. Citado na página 59.

Hadamard, J. (1902). Sur les problèmes aux dérivées partielles et leur signification physique, *Princeton university bulletin* pp. 49–52. Citado na página 41.

Hamed, T., Dara, R. & Kremer, S. C. (2018). Network intrusion detection system based on recursive feature addition and bigram technique, *Computers and Security* **73**: 137–155. **URL:** *https://doi.org/10.1016/j.cose.2017.10.011* Citado 6 vezes nas páginas 36, 60, 61, 64, 130 e 134.

Haykin, S. S. (2001). *Redes neurais: Princípios e Práticas*, Bookman. Citado 2 vezes nas páginas 46 e 47.

Hazan, E., Klivans, A. & Yuan, Y. (2017). Hyperparameter Optimization: A Spectral Approach. **URL:** *http://arxiv.org/abs/1706.00764* Citado na página 52.

Heady, R., Luger, G., Maccabe, A. & Servilla, M. (1990). The architecture of a network level intrusion detection system, *Technical report*, Los Alamos National Lab., NM (United States); New Mexico Univ., Albuquerque .... Citado na página 36.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks, *Neural Networks* **4**(2): 251–257. **URL:** *http://www.sciencedirect.com/science/article/pii/089360809190009T* Citado na página 106.

Hsu, C.-W., Chang, C.-C., Lin, C.-J. et al. (2003). A practical guide to support vector classification. Citado na página 79.

Jobson, J. D. (2012). *Applied multivariate data analysis: volume II: Categorical and Multivariate Methods*, Springer Science & Business Media. Citado 2 vezes nas páginas 97 e 120.

Kanimozhi, V. & Jacob, T. P. (2019). Artificial Intelligence based Network Intrusion Detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing, *ICT Express* **5**(3): 211–214. **URL:** *https://doi.org/10.1016/j.icte.2019.03.003* Citado 3 vezes nas páginas 36, 64 e 130.

Kasongo, S. M. & Sun, Y. (2020). A deep learning method with wrapper based feature extraction for wireless intrusion detection system, *Computers and Security* **92**: 101752. **URL:** *https://doi.org/10.1016/j.cose.2020.101752* Citado 6 vezes nas páginas 36, 60, 62, 63, 130 e 134.

Khraisat, A., Gondal, I., Vamplew, P. & Kamruzzaman, J. (2019). Survey of intrusion detection systems : techniques , datasets and challenges. Citado na página 60.

Kitchenham, B. A., Budgen, D. & Pearl Brereton, O. (2011). Using mapping studies as the basis for further research - a participant-observer case study, *Inf. Softw. Technol.* **53**(6): 638–651. **URL:** *https://doi.org/10.1016/j.infsof.2010.12.011* Citado na página 59.

Kniffin, K. M., Narayanan, J., Anseel, F., Antonakis, J., Ashford, S. P., Bakker, A. B., Bamberger, P., Bapuji, H., Bhave, D. P., Choi, V. K. et al. (2021). Covid-19 and the workplace: Implications, issues, and insights for future research and action., *American Psychologist* **76**(1): 63. Citado na página 30.

Kolias, C., Kambourakis, G., Stavrou, A. & Gritzalis, S. (2015). Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset, *IEEE Communications Surveys & Tutorials* **18**(1): 184–208. Citado na página 63.

Kruskal, W. H. & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis, *Journal of the American statistical Association* **47**(260): 583–621. Citado 2 vezes nas páginas 97 e 120.

Kumar, V., Srivastava, J. & Lazarevic, A. (2006). *Managing cyber threats: issues, approaches, and challenges*, Vol. 5, Springer Science & Business Media. Citado na página 36.

Lallie, H. S., Shepherd, L. A., Nurse, J. R., Erola, A., Epiphaniou, G., Maple, C. & Bellekens, X. (2020). Cyber security in the age of covid-19: a timeline and analysis of cyber-crime and cyber-attacks during the pandemic, *arXiv preprint arXiv:2006.11929* . Citado na página 30.

Lazarevic, A., Ozgur, A., Ertoz, L., Srivastava, J. & Kumar, V. (2003). A comparative study of anomaly detection schemes in network intrusion detection, *In Proceedings of the Third SIAM International Conference on Data Mining* pp. 25–36.
URL: *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.91.2580* Citado na página 39.

Liao, H.-J., Richard Lin, C.-H., Lin, Y.-C. & Tung, K.-Y. (2012). Intrusion detection system: A comprehensive review, *Journal of Network and Computer Applications* **36**(1): 16–24.
URL: *http://www.sciencedirect.com/science/article/pii/S1084804512001944* Citado 4 vezes nas páginas 36, 37, 38 e 39.

Liao, H. J., Richard Lin, C. H., Lin, Y. C. & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review, *Journal of Network and Computer Applications* **36**(1): 16–24.
URL: *http://www.sciencedirect.com/science/article/pii/S1084804512001944* Citado na página 60.

Lima, I. V. M. d. (2005). Uma abordagem simplificada de detecção de intrusão baseada em redes neurais artificiais. Citado na página 47.

Liu, J., Zhang, W., Tang, Z., Xie, Y., Ma, T., Zhang, J., Zhang, G. & Niyoyita, J. P. (2020). Adaptive intrusion detection via GA-GOGMM-based pattern learning with fuzzy rough set-based attribute selection, *Expert Systems with Applications* **139**: 112845.
URL: *https://doi.org/10.1016/j.eswa.2019.112845* Citado 6 vezes nas páginas 36, 40, 60, 63, 130 e 134.

Lu, N., Sun, Y., Liu, H. & Li, S. (2018). Intrusion Detection System Based on Evolving Rules for Wireless Sensor Networks, *Journal of Sensors* **2018**. Citado 6 vezes nas páginas 36, 60, 61, 63, 130 e 134.

Mahmoud, O., Harrison, A., Perperoglou, A., Gul, A., Khan, Z., Metodiev, M. V. & Lausen, B. (2014). A feature selection method for classification within functional genomics experiments based on the proportional overlapping score, *BMC bioinformatics* **15**(1): 274. Citado na página 62.

McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory, *ACM Transactions on Information and System Security (TISSEC)* **3**(4): 262–294. Citado na página 40.

Mitchell, T. M. et al. (1997). Machine learning. 1997, *Burr Ridge, IL: McGraw Hill* **45**(37): 870–877. Citado 3 vezes nas páginas 41, 42 e 48.

Mohan, K. P. (2017). Hybrid Network Intrusion Detection System Based on GA- NN Models, **116**(11): 31–39. Citado 6 vezes nas páginas 36, 40, 60, 61, 130 e 134.

Petersen, K., Vakkalanka, S. & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update, *Information and Software Techno-*

logy **64**: 1 – 18.
URL: *http://www.sciencedirect.com/science/article/pii/S0950584915000646* Citado na página 59.

Ramkumar, J. & Murugeswari, R. (2015). Fuzzy logic approach for detecting black hole attack in hybrid wireless mesh network. Citado 2 vezes nas páginas 36 e 61.

Rasmussen, C. E. (2003). Gaussian processes in machine learning, *Summer School on Machine Learning*, Springer, pp. 63–71. Citado na página 56.

Rasmussen, C. E. & Williams, C. K. (2005). Gaussian processes for machine learning (adaptive computation and machine learning) isbn: 026218253x. Citado na página 55.

Reazul, M., Rahman, A. & Samad, T. (2017). A Network Intrusion Detection Framework based on Bayesian Network using Wrapper Approach, *International Journal of Computer Applications* **166**(4): 13–17. Citado 6 vezes nas páginas 36, 60, 61, 63, 130 e 134.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain., *Psychological review* **65**(6): 386. Citado na página 45.

Russell, S. J. & Norvig, P. (2010). *Artificial intelligence: A Modern Approach*, Pearson Education. Citado na página 48.

Russell, S. & Norvig, P. (2002). Artificial intelligence: a modern approach. Citado na página 41.

Sabahi, F. & Movaghar, A. (2008). Intrusion detection: A survey, *2008 Third International Conference on Systems and Networks Communications*, IEEE, pp. 23–26. Citado na página 36.

Salzberg, S. (1991). A nearest hyperrectangle learning method, *Machine learning* **6**(3): 251–276. Citado na página 48.

Samonas, S. & Coss, D. (2014). The cia strikes back: Redefining confidentiality, integrity and availability in security., *Journal of Information System Security* **10**(3). Citado na página 36.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers, *IBM Journal of research and development* **3**(3): 210–229. Citado na página 41.

Selvakumar, B. & Muneeswaran, K. (2019). Firefly algorithm based feature selection for network intrusion detection, *Computers and Security* **81**: 148–155.
URL: *https://doi.org/10.1016/j.cose.2018.11.005* Citado 6 vezes nas páginas 36, 40, 60, 63, 64 e 134.

Sharafaldin, I., Lashkari, A. H. & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization., *ICISSP*, pp. 108–116. Citado na página 64.

Shashank, K. & Balachandra, M. (2018). Review on Network Intrusion Detection Techniques using Machine Learning, *2018 IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)* pp. 104–109. Citado na página 60.

Shenfield, A., Day, D. & Ayesh, A. (2018). Intelligent intrusion detection systems using artificial neural networks, *ICT Express* **4**(2): 95–99.
URL: *https://doi.org/10.1016/j.icte.2018.04.003* Citado 4 vezes nas páginas 36, 61, 65 e 130.

Snoek, J., Larochelle, H. & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms, *in* F. Pereira, C. J. C. Burges, L. Bottou & K. Q. Weinberger (eds), *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., pp. 2951–2959.
URL: *http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf* Citado na página 79.

Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D. & Nakao, K. (2011). Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation, *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, BADGERS '11, Association for Computing Machinery, New York, NY, USA, p. 29–36.
URL: *https://doi.org/10.1145/1978672.1978676* Citado na página 61.

Souza, C. A. d. (2018). *Método híbrido de detecção de intrusão aplicando inteligência artificial*, Master's thesis, Universidade Estadual do Oeste do Paraná. Citado 46 vezes nas páginas 15, 17, 19, 21, 22, 30, 31, 32, 35, 36, 39, 40, 48, 49, 50, 51, 67, 68, 69, 70, 71, 72, 73, 74, 75, 81, 86, 88, 89, 90, 91, 94, 96, 101, 119, 120, 123, 124, 125, 126, 127, 128, 131, 132, 134 e 145.

Stolfo, S., Fan, W., Lee, W. et al. (1999). Kdd-cup-99 task description. Citado na página 40.

Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A. & Chan, P. K. (2000). Cost-based modeling for fraud and intrusion detection: Results from the jam project, *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, Vol. 2, IEEE, pp. 130–144. Citado na página 40.

Storn, R. & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *Journal of global optimization* **11**(4): 341–359. Citado na página 56.

Tavallaee, M. (2009). A detailed analysis of the KDD CUP 99 data set, (July 2009). Citado na página 40.

Thomas, R. & Pavithran, D. (2019). A Survey of Intrusion Detection Models based on NSL-KDD Data Set, *2018 Fifth HCT Information Technology Trends (ITT)* (Itt): 286–291. Citado na página 60.

Tirronen, V. & Neri, F. (2009). Differential evolution with fitness diversity self-adaptation, *Nature-inspired algorithms for optimisation*, Springer, pp. 199–234. Citado 2 vezes nas páginas 56 e 57.

Tommy Wai-shing, C. & David Siu-yeung, C. (2007). *Neural Networks And Computing: Learning Algorithms And Applications.*, number v. 7 in *Series in Electrical and Computer Engineering*, Imperial College Press.
URL: *http://search.ebscohost.com/login.aspx?direct=truedb=e000xwwAN=516737lang=pt-brsite=eds-livescope=site* Citado 2 vezes nas páginas 44 e 45.

Vijayanand, R., Devaraj, D. & Kannapiran, B. (2018). Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection, *Computers and Security* **77**: 304–314.
URL: *https://doi.org/10.1016/j.cose.2018.04.010* Citado 5 vezes nas páginas 36, 60, 61, 64 e 134.

Wu, S. X. & Banzhaf, W. (2010a). The use of computational intelligence in intrusion detection

systems: A review, *Applied Soft Computing Journal* **10**(1): 1–35.
URL: *http://www.sciencedirect.com/science/article/pii/S1568494609000908* Citado 2 vezes nas páginas 39 e 40.

Wu, S. X. & Banzhaf, W. (2010b). The use of computational intelligence in intrusion detection systems: A review, *Applied Soft Computing* **10**(1): 1 – 35.
URL: *http://www.sciencedirect.com/science/article/pii/S1568494609000908* Citado 2 vezes nas páginas 36 e 37.

Xenakis, C., Panos, C. & Stavrakakis, I. (2011). A comparative evaluation of intrusion detection architectures for mobile ad hoc networks, *Computers & Security* **30**(1): 63–80. Citado na página 36.

Zhang, T. & Zhu, Q. (2018). Distributed Privacy-Preserving Collaborative Intrusion Detection Systems for VANETs, *IEEE Transactions on Signal and Information Processing over Networks* **4**(1): 148–161. Citado 6 vezes nas páginas 36, 39, 40, 60, 63 e 134.

Apêndice A

Detalhamento da Base de Dados de *Benchmark* NSL-KDD

Tabela A.1: Atributos da Base de Dados NSL-KDD (Fonte: Adaptado de Souza (2018))

#	**Atributo**	**Descrição**	**Tipo**	**Valores / Escopo**
1	*duration*	Duração da conexão	Proporcional Discreto	0 - 57.715
2	*protocol_type*	Tipo de protocolo	Nominal Discreto	*tcp, udp, icmp*

3	*service*	Tipo de serviço de destino	Nominal Discreto	*aol, auth, bgp, courier, csnet_s,* *ctf, daytime, discard, domain, domain_,* *echo, eco_, ecr_, efs, exec, finger,* *ftp, ftp_ata, gopher, harvest, hostnames,* *http, http_784, http_43, http_001,* *imap4, IRC, iso_sap, klogin, kshell, ldap,* *link, login, mtp, name, netbios_gm,* *netbios_s, netbios_sn, netstat, nnsp,* *nntp, ntp_, other, pm_ump, pop_, pop_,* *rinter, private, red_, remote_ob, rje, shell, smtp, sql_et, ssh,* *sunrpc, supdup, systat, telnet, tftp_, tim_, time,* *urh_, urp_, uucp, uucp_ath, vmnet, whois, X11, Z39_0*
4	*flag*	Estado da conexão	Nominal Discreto	*OTH, REJ, RSTO, RSTOS0, RSTR,* *S0, S1, S2, S3, SF, SH*
5	*src_bytes*	Números de *bytes* da origem ao destino	Proporcional Discreto	0 - 1.379.963.888
6	*dst_bytes*	Número de *bytes* do destino à origem	Proporcional Discreto	0 - 1.309.937.401

7	*land*	1 se o Host e a porta da origem e destino são os mesmos, 0 caso contrário	Nominal Discreto	0 - 1
8	*wrong_fragment*	Número de fragmentos errados	Proporcional Discreto	0 - 3
9	*urgent*	Número de pacotes urgentes	Proporcional Discreto	0 - 3
10	*hot*	Número de ações "*hot*"em uma conexão, tais como: inserir um diretório do sistema, criar e executar programas	Proporcional Discreto	0 - 101
11	*num_failed_logins*	Número de tentativas de *login* com falha	Proporcional Discreto	0 - 5
12	*logged_in*	1 se o *login* obteve sucesso e 0 caso contrário	Nominal Discreto	0 - 1
13	*num_compromised*	Número de condições comprometidas	Proporcional Discreto	0 - 7.479

14	*root_shell*	1 se o *shell root* é obtido 0 caso contrário	Nominal Discreto	0 - 1
15	*su_attempted*	1 se houver tentativa de conseguir "*su root*", 0 caso contrário	Nominal Discreto	0 - 2*
16	*num_root*	Número de acessos como *root*	Proporcional Discreto	0 - 7.468
17	*num_file_creations*	Número de operações de criação de arquivos	Proporcional Discreto	0 - 100
18	*num_shells*	Número de *shell prompts* abertos	Proporcional Discreto	0 - 5
19	*num_access_files*	Número de operações a arquivos de controle de acesso	Proporcional Discreto	0 - 9
20	*num_outbound_cmds*	Números de comandos externos (sessão FTP)	Proporcional Discreto	0†
21	*is_hot_login*	1 se o *login* pertence à lista "*hot*", 0 caso contrário	Nominal Discreto	0 - 1

*Apesar de sua ocorrência, em termos lógicos, o valor 2 não deveria ser registrado nesse atributo

†Esse atributo possui seu valor constante como sendo 0 em todas as instâncias

22	*is_guest_login*	1 se o *login* é do tipo "*guest*", 0 caso contrário	Nominal Discreto	0 - 1
23	*count*	Número de conexões para o mesmo *host* da conexão atual	Proporcional Discreto	0 - 511
24	*srv_count*	Número de conexões ao mesmo serviço da conexão atual	Proporcional Discreto	0 - 511
25	*serror_rate*	% de conexões que tiveram erros do tipo "SYN", dentre as conexões consideradas em count (23)	Proporcional Contínuo	0 - 1
26	*srv_serror_rate*	% de conexões que tiveram erros "SYN", dentre as conexões consideradas em *srv_count* (24)	Proporcional Contínuo	0 - 1
27	*rerror_rate*	% de conexões que tiveram erros do tipo "REJ", dentre as conexões consideradas em *count* (23)	Proporcional Contínuo	0 - 1

28	*srv_rerror_rate*	% de conexões que tiveram erros "REJ", dentre as conexões consideradas em *srv_count* (24)	Proporcional Contínuo	0 - 1
29	*same_srv_rate*	% de conexões ao mesmo serviço, dentre as conexões consideradas em *count* (23)	Proporcional Contínuo	0 - 1
30	*diff_srv_rate*	% de conexões a diferentes serviços, dentre as conexões consideradas em *count* (23)	Proporcional Contínuo	0 - 1
31	*srv_diff_host_rate*	% de conexões a diferentes hosts, dentre as conexões consideradas em *srv_count* (24)	Proporcional Contínuo	0 - 1
32	*dst_host_count*	Número de conexões para o mesmo *host* da conexão atual dentre as últimas 100 conexões	Proporcional Discreto	0 - 255
33	*dst_host_srv_count*	Número de conexões ao mesmo serviço da conexão atual dentre as últimas 100 conexões	Proporcional Discreto	0 - 255

34	*dst_host_same_srv_rate*	% de conexões ao mesmo serviço, dentre as conexões consideradas em *count* (32)	Proporcional Contínuo	0 - 1
35	*dst_host_diff_srv_rate*	% de conexões a diferentes serviços, dentre as conexões consideradas em *count* (32)	Proporcional Contínuo	0 - 1
36	*dst_host_same_src_port_rate*	% de conexões a mesma porta de origem, dentre as conexões consideradas em *dst_host_srv_count* (33)	Proporcional Contínuo	0 - 1
37	*dst_host_srv_diff_host_rate*	% de conexões a diferentes hosts, dentre as conexões consideradas em *dst_host_srv_count* (33)	Proporcional Contínuo	0 - 1
38	*dst_host_serror_rate*	% de conexões que tiveram erros do tipo "SYN", dentre as conexões consideradas em *dst_host_count* (32)	Proporcional Contínuo	0 - 1

39	*dst_host_srv_serror_rate*	% de conexões que tiveram erros "SYN", dentre as conexões consideradas em *dst_host_count* (33)	Proporcional Contínuo	0 - 1
40	*dst_host_rerror_rate*	% de conexões que tiveram erros do tipo "REJ", dentre as conexões consideradas em *dst_host_count* (32)	Proporcional Contínuo	0 - 1
41	*dst_host_srv_rerror_rate*	% de conexões que tiveram erros do tipo "REJ", dentre as conexões consideradas em *dst_host_srv_count* (33)	Proporcional Contínuo	0 - 1

42	*class*	Tipo de conexão	Nominal Discreto	*apache2, back, buffer_verflow, ftp_rite,* *guess_asswd, httptunnel, imap, ipsweep,* *land, loadmodule, mailbomb, mscan,* *multihop, named, neptune, nmap, normal,* *perl, phf, pod, portsweep, processtable,* *ps, rootkit, saint, satan, sendmail,* *smurf, snmpgetattack, snmpguess, spy, sqlattack,* *teardrop, udpstorm, warezclient, warezmaster, worm,* *xlock, xsnoop, xterm*

Apêndice B

Modelos Gerados pelo EMHOSIR Comparados entre Si

Tabela B.1: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHOSIR: Busca Aleatória sobre *NOM_TO_INT x* Busca em Grade sobre *NOM_TO_INT*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	8,32	$5{,}87 \times 10^{-2}$			
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	4,76	$1{,}74 \times 10^{-2}$	3,56	$9{,}63 \times 10^{-2}$	
AC	MLP | Busca Aleatória - *NOM_TO_INT*	99,70	$7{,}73 \times 10^{-4}$			
	MLP | Busca em Grade - *NOM_TO_INT*	99,67	$7{,}69 \times 10^{-4}$	0,03	$3{,}07 \times 10^{-1}$	
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,77	$4{,}86 \times 10^{-4}$			
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,75	$6{,}33 \times 10^{-4}$	0,02	$6{,}23 \times 10^{-1}$	
	MLP | Busca em Grade - *NOM_TO_INT*	**99,67**	$\mathbf{7{,}69 \times 10^{-4}}$	**0,10**	$\mathbf{9{,}08 \times 10^{-3}}$	$\mathbf{9{,}08 \times 10^{-3}}$
SE	MLP | Busca Aleatória - *NOM_TO_INT*	99,60	$1{,}13 \times 10^{-3}$			
	MLP | Busca em Grade - *NOM_TO_INT*	99,55	$1{,}25 \times 10^{-3}$	0,04	$7{,}05 \times 10^{-1}$	
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,73	$8{,}89 \times 10^{-4}$			
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,69	$1{,}02 \times 10^{-3}$	0,04	$3{,}61 \times 10^{-1}$	
ES	MLP | Busca Aleatória - *NOM_TO_INT*	99,79	$7{,}57 \times 10^{-4}$			
	MLP | Busca em Grade - *NOM_TO_INT*	99,78	$8{,}27 \times 10^{-4}$	0,01	$8{,}20 \times 10^{-1}$	
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,80	$4{,}89 \times 10^{-4}$			
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,81	$5{,}62 \times 10^{-4}$	0,00	$3{,}82 \times 10^{-1}$	

Tabela B.2: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHO-SIR: Busca em Grade sobre *NOM_TO_INT x* Otimização Bayesiana sobre *NOM_TO_INT*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	4,76	$1{,}74 \times 10^{-2}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	6,49	$6{,}20 \times 10^{-2}$	1,73	$7{,}05 \times 10^{-1}$	
AC	MLP | Busca em Grade - *NOM_TO_INT*	99,67	$7{,}69 \times 10^{-4}$			
	MLP | Otimização Bayesiana - *NOM_TO_INT*	99,67	$4{,}32 \times 10^{-4}$	0,01	$7{,}91 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,75	$6{,}33 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,76	$4{,}95 \times 10^{-4}$	0,00	$8{,}20 \times 10^{-1}$	
	MLP | Otimização Bayesiana - *NOM_TO_INT*	**99,67**	$\mathbf{4{,}32 \times 10^{-4}}$	**0,09**	$\mathbf{3{,}92 \times 10^{-3}}$	$\mathbf{3{,}92 \times 10^{-3}}$
SE	MLP | Busca em Grade - *NOM_TO_INT*	99,55	$1{,}25 \times 10^{-3}$			
	MLP | Otimização Bayesiana - *NOM_TO_INT*	99,48	$1{,}13 \times 10^{-3}$	0,07	$2{,}71 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,69	$1{,}02 \times 10^{-3}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,70	$8{,}74 \times 10^{-4}$	0,01	$7{,}90 \times 10^{-1}$	
ES	MLP | Busca em Grade - *NOM_TO_INT*	99,78	$8{,}27 \times 10^{-4}$			
	MLP | Otimização Bayesiana - *NOM_TO_INT*	99,82	$5{,}93 \times 10^{-4}$	0,05	$1{,}50 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,81	$5{,}62 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,81	$5{,}92 \times 10^{-4}$	0,00	$3{,}82 \times 10^{-1}$	

Tabela B.3: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHO-SIR: Busca em Grade sobre *NOM_TO_BIN x* Busca em Grade sobre *NOM_TO_INT*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	2,81	$8{,}40 \times 10^{-3}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	**4,76**	$\mathbf{1{,}74 \times 10^{-2}}$	**1,95**	$\mathbf{8{,}15 \times 10^{-3}}$	$\mathbf{8{,}15 \times 10^{-3}}$
AC	MLP | Busca em Grade - *NOM_TO_BIN*	99,74	$4{,}50 \times 10^{-4}$			
	MLP | Busca em Grade - *NOM_TO_INT*	**99,67**	$\mathbf{7{,}69 \times 10^{-4}}$	**0,07**	$\mathbf{1{,}88 \times 10^{-2}}$	$\mathbf{1{,}88 \times 10^{-2}}$
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,77	$4{,}48 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,75	$6{,}33 \times 10^{-4}$	0,02	$9{,}40 \times 10^{-1}$	
	MLP | Busca em Grade - *NOM_TO_INT*	**99,67**	$\mathbf{7{,}69 \times 10^{-4}}$	**0,10**	$\mathbf{4{,}53 \times 10^{-3}}$	$\mathbf{4{,}53 \times 10^{-3}}$
SE	MLP | Busca em Grade - *NOM_TO_BIN*	99,68	$5{,}95 \times 10^{-4}$			
	MLP | Busca em Grade - *NOM_TO_INT*	**99,55**	$\mathbf{1{,}25 \times 10^{-3}}$	**0,12**	$\mathbf{9{,}94 \times 10^{-3}}$	$\mathbf{9{,}94 \times 10^{-3}}$
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,75	$4{,}76 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,69	$1{,}02 \times 10^{-3}$	0,06	$1{,}09 \times 10^{-1}$	
ES	MLP | Busca em Grade - *NOM_TO_BIN*	99,80	$6{,}94 \times 10^{-4}$			
	MLP | Busca em Grade - *NOM_TO_INT*	99,78	$8{,}27 \times 10^{-4}$	0,03	$5{,}20 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,79	$6{,}49 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,81	$5{,}62 \times 10^{-4}$	0,02	$5{,}44 \times 10^{-1}$	

Tabela B.4: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHO-SIR: Busca em Grade sobre *NOM_TO_INT x* Busca Aleatória sobre *NOM_TO_BIN*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	4,76	$1{,}74 \times 10^{-2}$			
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	**3,25**	$\mathbf{1{,}28 \times 10^{-2}}$	**1,51**	$\mathbf{3{,}76 \times 10^{-2}}$	$\mathbf{3{,}76 \times 10^{-2}}$
AC	MLP | Busca em Grade - *NOM_TO_INT*	99,67	$7{,}69 \times 10^{-4}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,73	$6{,}49 \times 10^{-4}$	0,06	$8{,}86 \times 10^{-2}$	
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,75	$6{,}33 \times 10^{-4}$			
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,76	$5{,}58 \times 10^{-4}$	0,01	$8{,}20 \times 10^{-1}$	
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,73	$6{,}49 \times 10^{-4}$	0,02	$3{,}25 \times 10^{-1}$	
SE	MLP | Busca em Grade - *NOM_TO_INT*	99,55	$1{,}25 \times 10^{-3}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,65	$9{,}45 \times 10^{-4}$	0,10	$1{,}20 \times 10^{-1}$	
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,69	$1{,}02 \times 10^{-3}$			
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,74	$7{,}72 \times 10^{-4}$	0,04	$3{,}23 \times 10^{-1}$	
ES	MLP | Busca em Grade - *NOM_TO_INT*	99,78	$8{,}27 \times 10^{-4}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,80	$6{,}42 \times 10^{-4}$	0,03	$5{,}45 \times 10^{-1}$	
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,81	$5{,}62 \times 10^{-4}$			
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,78	$5{,}32 \times 10^{-4}$	0,03	$8{,}07 \times 10^{-2}$	

Tabela B.5: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHO-SIR: Otimização Bayesiana sobre *NOM_TO_BIN x* Busca em Grade sobre *NOM_TO_INT*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	2,36	$1{,}42 \times 10^{-2}$			
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	**4,76**	$\mathbf{1{,}74 \times 10^{-2}}$	**2,39**	$\mathbf{4{,}06 \times 10^{-3}}$	$\mathbf{4{,}06 \times 10^{-3}}$
AC	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,75	$3{,}41 \times 10^{-4}$			
	MLP | Busca em Grade - *NOM_TO_INT*	**99,67**	$\mathbf{7{,}69 \times 10^{-4}}$	**0,08**	$\mathbf{1{,}72 \times 10^{-2}}$	$\mathbf{1{,}72 \times 10^{-2}}$
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,78	$2{,}99 \times 10^{-4}$			
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,75	$6{,}33 \times 10^{-4}$	0,03	$1{,}97 \times 10^{-1}$	
	MLP | Busca em Grade - *NOM_TO_INT*	**99,67**	$\mathbf{7{,}69 \times 10^{-4}}$	**0,11**	$\mathbf{6{,}67 \times 10^{-4}}$	$\mathbf{6{,}67 \times 10^{-4}}$
SE	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,68	$5{,}62 \times 10^{-4}$			
	MLP | Busca em Grade - *NOM_TO_INT*	**99,55**	$\mathbf{1{,}25 \times 10^{-3}}$	**0,12**	$\mathbf{1{,}10 \times 10^{-2}}$	$\mathbf{1{,}10 \times 10^{-2}}$
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,74	$6{,}58 \times 10^{-4}$			
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,69	$1{,}02 \times 10^{-3}$	0,05	$1{,}83 \times 10^{-1}$	
ES	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,81	$4{,}33 \times 10^{-4}$			
	MLP | Busca em Grade - *NOM_TO_INT*	99,78	$8{,}27 \times 10^{-4}$	0,04	$3{,}83 \times 10^{-1}$	
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,82	$3{,}64 \times 10^{-4}$			
	$E^{P_h}_{P_i}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_INT*	99,81	$5{,}62 \times 10^{-4}$	0,01	$8{,}79 \times 10^{-1}$	

Tabela B.6: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHO-SIR: Busca Aleatória sobre *NOM_TO_INT x* Otimização Bayesiana sobre *NOM_TO_INT*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	8,32	$5{,}87 \times 10^{-2}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	6,49	$6{,}20 \times 10^{-2}$	1,82	$1{,}12 \times 10^{-1}$	
AC	MLP | Busca Aleatória - *NOM_TO_INT*	99,70	$7{,}73 \times 10^{-4}$			
	MLP | Otimização Bayesiana - *NOM_TO_INT*	99,67	$4{,}32 \times 10^{-4}$	0,03	$1{,}50 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,77	$4{,}86 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,76	$4{,}95 \times 10^{-4}$	0,02	$3{,}84 \times 10^{-1}$	
	MLP | Otimização Bayesiana - *NOM_TO_INT*	**99,67**	$\mathbf{4{,}32 \times 10^{-4}}$	**0,11**	$\mathbf{5{,}77 \times 10^{-4}}$	$\mathbf{5{,}77 \times 10^{-4}}$
SE	MLP | Busca Aleatória - *NOM_TO_INT*	99,60	$1{,}13 \times 10^{-3}$			
	MLP | Otimização Bayesiana - *NOM_TO_INT*	**99,48**	$\mathbf{1{,}13 \times 10^{-3}}$	**0,11**	$\mathbf{4{,}07 \times 10^{-2}}$	$\mathbf{4{,}07 \times 10^{-2}}$
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,73	$8{,}89 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,70	$8{,}74 \times 10^{-4}$	0,04	$3{,}42 \times 10^{-1}$	
ES	MLP | Busca Aleatória - *NOM_TO_INT*	99,79	$7{,}57 \times 10^{-4}$			
	MLP | Otimização Bayesiana - *NOM_TO_INT*	99,82	$5{,}93 \times 10^{-4}$	0,04	$4{,}04 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,80	$4{,}89 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,81	$5{,}92 \times 10^{-4}$	0,00	$8{,}49 \times 10^{-1}$	

Tabela B.7: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHO-SIR: Busca Aleatória sobre *NOM_TO_INT x* Busca em Grade sobre *NOM_TO_BIN*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	8,32	$5{,}87 \times 10^{-2}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	**2,81**	$\mathbf{8{,}40 \times 10^{-3}}$	**5,51**	$\mathbf{1{,}15 \times 10^{-3}}$	$\mathbf{1{,}15 \times 10^{-3}}$
AC	MLP | Busca Aleatória - *NOM_TO_INT*	99,70	$7{,}73 \times 10^{-4}$			
	MLP | Busca em Grade - *NOM_TO_BIN*	99,74	$4{,}50 \times 10^{-4}$	0,05	$1{,}85 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,77	$4{,}86 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,77	$4{,}48 \times 10^{-4}$	0,00	$7{,}62 \times 10^{-1}$	
	MLP | Busca em Grade - *NOM_TO_BIN*	99,74	$4{,}50 \times 10^{-4}$	0,03	$3{,}25 \times 10^{-1}$	
SE	MLP | Busca Aleatória - *NOM_TO_INT*	99,60	$1{,}13 \times 10^{-3}$			
	MLP | Busca em Grade - *NOM_TO_BIN*	99,68	$5{,}95 \times 10^{-4}$	0,08	$5{,}82 \times 10^{-2}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,73	$8{,}89 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,75	$4{,}76 \times 10^{-4}$	0,02	$7{,}61 \times 10^{-1}$	
ES	MLP | Busca Aleatória - *NOM_TO_INT*	99,79	$7{,}57 \times 10^{-4}$			
	MLP | Busca em Grade - *NOM_TO_BIN*	99,80	$6{,}94 \times 10^{-4}$	0,01	$9{,}09 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,80	$4{,}89 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,79	$6{,}49 \times 10^{-4}$	0,01	$9{,}70 \times 10^{-1}$	

Tabela B.8: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHO-SIR: Busca Aleatória sobre *NOM_TO_INT x* Busca Aleatória sobre *NOM_TO_BIN*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	8,32	$5{,}87 \times 10^{-2}$			
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	**3,25**	$\mathbf{1{,}28 \times 10^{-2}}$	**5,07**	$\mathbf{2{,}50 \times 10^{-3}}$	$\mathbf{2{,}50 \times 10^{-3}}$
AC	MLP | Busca Aleatória - *NOM_TO_INT*	99,70	$7{,}73 \times 10^{-4}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,73	$6{,}49 \times 10^{-4}$	0,03	$3{,}44 \times 10^{-1}$	
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,77	$4{,}86 \times 10^{-4}$			
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,76	$5{,}58 \times 10^{-4}$	0,01	$6{,}22 \times 10^{-1}$	
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,73	$6{,}49 \times 10^{-4}$	0,04	$3{,}07 \times 10^{-1}$	
SE	MLP | Busca Aleatória - *NOM_TO_INT*	99,60	$1{,}13 \times 10^{-3}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,65	$9{,}45 \times 10^{-4}$	0,06	$2{,}55 \times 10^{-1}$	
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,73	$8{,}89 \times 10^{-4}$			
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,74	$7{,}72 \times 10^{-4}$	0,00	$8{,}79 \times 10^{-1}$	
ES	MLP | Busca Aleatória - *NOM_TO_INT*	99,79	$7{,}57 \times 10^{-4}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,80	$6{,}42 \times 10^{-4}$	0,01	$7{,}62 \times 10^{-1}$	
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,80	$4{,}89 \times 10^{-4}$			
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,78	$5{,}32 \times 10^{-4}$	0,02	$4{,}47 \times 10^{-1}$	

Tabela B.9: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHO-SIR: Busca Aleatória sobre *NOM_TO_INT x* Otimização Bayesiana sobre *NOM_TO_BIN*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	8,32	$5{,}87 \times 10^{-2}$			
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	**2,36**	$\mathbf{1{,}42 \times 10^{-2}}$	**5,95**	$\mathbf{8{,}77 \times 10^{-4}}$	$\mathbf{8{,}77 \times 10^{-4}}$
AC	MLP | Busca Aleatória - *NOM_TO_INT*	99,70	$7{,}73 \times 10^{-4}$			
	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,75	$3{,}41 \times 10^{-4}$	0,05	$5{,}35 \times 10^{-2}$	
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,77	$4{,}86 \times 10^{-4}$			
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,78	$2{,}99 \times 10^{-4}$	0,01	$7{,}91 \times 10^{-1}$	
	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,75	$3{,}41 \times 10^{-4}$	0,02	$3{,}25 \times 10^{-1}$	
SE	MLP | Busca Aleatória - *NOM_TO_INT*	99,60	$1{,}13 \times 10^{-3}$			
	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,68	$5{,}62 \times 10^{-4}$	0,08	$5{,}82 \times 10^{-2}$	
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,73	$8{,}89 \times 10^{-4}$			
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,74	$6{,}58 \times 10^{-4}$	0,01	$8{,}79 \times 10^{-1}$	
ES	MLP | Busca Aleatória - *NOM_TO_INT*	99,79	$7{,}57 \times 10^{-4}$			
	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,81	$4{,}33 \times 10^{-4}$	0,03	$5{,}19 \times 10^{-1}$	
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_INT*	99,80	$4{,}89 \times 10^{-4}$			
	$E^{p_p}_{P_i}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,82	$3{,}64 \times 10^{-4}$	0,02	$4{,}02 \times 10^{-1}$	

Tabela B.10: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHO-SIR: Busca em Grade sobre *NOM_TO_BIN x* Otimização Bayesiana sobre *NOM_TO_INT*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	2,81	$8{,}40 \times 10^{-3}$			
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	**6,49**	$\mathbf{6{,}20 \times 10^{-2}}$	**3,69**	$\mathbf{1{,}91 \times 10^{-2}}$	$\mathbf{1{,}91 \times 10^{-2}}$
AC	MLP | Busca em Grade - *NOM_TO_BIN*	99,74	$4{,}50 \times 10^{-4}$			
	MLP | Otimização Bayesiana - *NOM_TO_INT*	**99,67**	$\mathbf{4{,}32 \times 10^{-4}}$	**0,08**	$\mathbf{5{,}59 \times 10^{-3}}$	$\mathbf{5{,}59 \times 10^{-3}}$
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,77	$4{,}48 \times 10^{-4}$			
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,76	$4{,}95 \times 10^{-4}$	0,02	$3{,}62 \times 10^{-1}$	
	MLP | Otimização Bayesiana - *NOM_TO_INT*	**99,67**	$\mathbf{4{,}32 \times 10^{-4}}$	**0,11**	$\mathbf{5{,}02 \times 10^{-4}}$	$\mathbf{5{,}02 \times 10^{-4}}$
SE	MLP | Busca em Grade - *NOM_TO_BIN*	99,68	$5{,}95 \times 10^{-4}$			
	MLP | Otimização Bayesiana - *NOM_TO_INT*	**99,48**	$\mathbf{1{,}13 \times 10^{-3}}$	**0,20**	$\mathbf{1{,}28 \times 10^{-3}}$	$\mathbf{1{,}28 \times 10^{-3}}$
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,75	$4{,}76 \times 10^{-4}$			
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,70	$8{,}74 \times 10^{-4}$	0,05	$8{,}72 \times 10^{-2}$	
ES	MLP | Busca em Grade - *NOM_TO_BIN*	99,80	$6{,}94 \times 10^{-4}$			
	MLP | Otimização Bayesiana - *NOM_TO_INT*	99,82	$5{,}93 \times 10^{-4}$	0,02	$4{,}71 \times 10^{-1}$	
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,79	$6{,}49 \times 10^{-4}$			
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,81	$5{,}92 \times 10^{-4}$	0,01	$9{,}70 \times 10^{-1}$	

Tabela B.11: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHO-SIR: Otimização Bayesiana sobre *NOM_TO_INT x* Busca em Grade sobre *NOM_TO_BIN*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	6,49	$6{,}20 \times 10^{-2}$			
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	3,25	$1{,}28 \times 10^{-2}$	3,25	$6{,}96 \times 10^{-2}$	
AC	MLP | Otimização Bayesiana - *NOM_TO_INT*	99,67	$4{,}32 \times 10^{-4}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	**99,73**	$\mathbf{6{,}49 \times 10^{-4}}$	**0,07**	$\mathbf{2{,}10 \times 10^{-2}}$	$\mathbf{2{,}10 \times 10^{-2}}$
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,76	$4{,}95 \times 10^{-4}$			
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,76	$5{,}58 \times 10^{-4}$	0,00	$9{,}09 \times 10^{-1}$	
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,73	$6{,}49 \times 10^{-4}$	0,02	$5{,}70 \times 10^{-1}$	
SE	MLP | Otimização Bayesiana - *NOM_TO_INT*	99,48	$1{,}13 \times 10^{-3}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	**99,65**	$\mathbf{9{,}45 \times 10^{-4}}$	**0,17**	$\mathbf{5{,}65 \times 10^{-3}}$	$\mathbf{5{,}65 \times 10^{-3}}$
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,70	$8{,}74 \times 10^{-4}$			
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,74	$7{,}72 \times 10^{-4}$	0,04	$4{,}04 \times 10^{-1}$	
ES	MLP | Otimização Bayesiana - *NOM_TO_INT*	99,82	$5{,}93 \times 10^{-4}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,80	$6{,}42 \times 10^{-4}$	0,02	$3{,}63 \times 10^{-1}$	
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,81	$5{,}92 \times 10^{-4}$			
	$E^{P_0}_{P_1}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,78	$5{,}32 \times 10^{-4}$	0,03	$5{,}43 \times 10^{-1}$	

Tabela B.12: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHOSIR: Otimização Bayesiana sobre *NOM_TO_BIN x* Otimização Bayesiana sobre *NOM_TO_INT*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	2,36	$1{,}42 \times 10^{-2}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	**6,49**	$\mathbf{6{,}20 \times 10^{-2}}$	**4,13**	$\mathbf{1{,}01 \times 10^{-2}}$	$\mathbf{1{,}01 \times 10^{-2}}$
AC	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,75	$3{,}41 \times 10^{-4}$			
	MLP | Otimização Bayesiana - *NOM_TO_INT*	**99,67**	$\mathbf{4{,}32 \times 10^{-4}}$	**0,09**	$\mathbf{2{,}11 \times 10^{-4}}$	$\mathbf{2{,}11 \times 10^{-4}}$
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,78	$2{,}99 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,76	$4{,}95 \times 10^{-4}$	0,03	$2{,}71 \times 10^{-1}$	
	MLP | Otimização Bayesiana - *NOM_TO_INT*	**99,67**	$\mathbf{4{,}32 \times 10^{-4}}$	**0,12**	$\mathbf{1{,}55 \times 10^{-4}}$	$\mathbf{1{,}55 \times 10^{-4}}$
SE	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,68	$5{,}62 \times 10^{-4}$			
	MLP | Otimização Bayesiana - *NOM_TO_INT*	**99,48**	$\mathbf{1{,}13 \times 10^{-3}}$	**0,20**	$\mathbf{1{,}45 \times 10^{-3}}$	$\mathbf{1{,}45 \times 10^{-3}}$
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,74	$6{,}58 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,70	$8{,}74 \times 10^{-4}$	0,04	$1{,}58 \times 10^{-1}$	
ES	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,81	$4{,}33 \times 10^{-4}$			
	MLP | Otimização Bayesiana - *NOM_TO_INT*	99,82	$5{,}93 \times 10^{-4}$	0,01	$7{,}05 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,82	$3{,}64 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_INT*	99,81	$5{,}92 \times 10^{-4}$	0,02	$3{,}23 \times 10^{-1}$	

Tabela B.13: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHOSIR: Busca em Grade sobre *NOM_TO_BIN x* Busca Aleatória sobre *NOM_TO_BIN*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	2,81	$8{,}40 \times 10^{-3}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	3,25	$1{,}28 \times 10^{-2}$	0,44	$4{,}06 \times 10^{-1}$	
AC	MLP | Busca em Grade - *NOM_TO_BIN*	99,74	$4{,}50 \times 10^{-4}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,73	$6{,}49 \times 10^{-4}$	0,01	$4{,}95 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,77	$4{,}48 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,76	$5{,}58 \times 10^{-4}$	0,01	$9{,}09 \times 10^{-1}$	
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,73	$6{,}49 \times 10^{-4}$	0,04	$1{,}40 \times 10^{-1}$	
SE	MLP | Busca em Grade - *NOM_TO_BIN*	99,68	$5{,}95 \times 10^{-4}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,65	$9{,}45 \times 10^{-4}$	0,03	$4{,}47 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,75	$4{,}76 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,74	$7{,}72 \times 10^{-4}$	0,01	$9{,}09 \times 10^{-1}$	
ES	MLP | Busca em Grade - *NOM_TO_BIN*	99,80	$6{,}94 \times 10^{-4}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,80	$6{,}42 \times 10^{-4}$	0,00	$9{,}40 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,79	$6{,}49 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,78	$5{,}32 \times 10^{-4}$	0,01	$4{,}95 \times 10^{-1}$	

Tabela B.14: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHO-SIR: Otimização Bayesiana sobre *NOM_TO_BIN x* Busca em Grade sobre *NOM_TO_BIN*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	2,36	$1{,}42 \times 10^{-2}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	2,81	$8{,}40 \times 10^{-3}$	0,44	$9{,}62 \times 10^{-2}$	
AC	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,75	$3{,}41 \times 10^{-4}$			
	MLP | Busca em Grade - *NOM_TO_BIN*	99,74	$4{,}50 \times 10^{-4}$	0,01	$9{,}40 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,78	$2{,}99 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,77	$4{,}48 \times 10^{-4}$	0,01	$2{,}72 \times 10^{-1}$	
	MLP | Busca em Grade - ***NOM_TO_BIN***	**99,74**	$\mathbf{4{,}50 \times 10^{-4}}$	**0,04**	$\mathbf{2{,}28 \times 10^{-2}}$	$\mathbf{2{,}28 \times 10^{-2}}$
SE	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,68	$5{,}62 \times 10^{-4}$			
	MLP | Busca em Grade - *NOM_TO_BIN*	99,68	$5{,}95 \times 10^{-4}$	0,00	$8{,}79 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,74	$6{,}58 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,75	$4{,}76 \times 10^{-4}$	0,01	$8{,}49 \times 10^{-1}$	
ES	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,81	$4{,}33 \times 10^{-4}$			
	MLP | Busca em Grade - *NOM_TO_BIN*	99,80	$6{,}94 \times 10^{-4}$	0,01	$6{,}77 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,82	$3{,}64 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca em Grade - *NOM_TO_BIN*	99,79	$6{,}49 \times 10^{-4}$	0,03	$4{,}95 \times 10^{-1}$	

Tabela B.15: Comparação entre Modelos Combinados de Classificadores Gerados pelo EMHO-SIR: Otimização Bayesiana sobre *NOM_TO_BIN x* Busca Aleatória sobre *NOM_TO_BIN*

	Modelo	$\overline{x}$ *(%)*	σ *(%)*	*Comparação*		
				$\Delta\overline{x}$ *(%)*	*p-valor*	
					Kruskall-Wallis	*Teste de Dunn*
C2	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	2,36	$1{,}42 \times 10^{-2}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - ***NOM_TO_BIN***	**3,25**	$\mathbf{1{,}28 \times 10^{-2}}$	**0,88**	$\mathbf{4{,}12 \times 10^{-2}}$	$\mathbf{4{,}12 \times 10^{-2}}$
AC	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,75	$3{,}41 \times 10^{-4}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,73	$6{,}49 \times 10^{-4}$	0,02	$7{,}62 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,78	$2{,}99 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,76	$5{,}58 \times 10^{-4}$	0,02	$2{,}71 \times 10^{-1}$	
	MLP | Busca Aleatória - ***NOM_TO_BIN***	**99,73**	$\mathbf{6{,}49 \times 10^{-4}}$	**0,05**	$\mathbf{2{,}56 \times 10^{-2}}$	$\mathbf{2{,}56 \times 10^{-2}}$
SE	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,68	$5{,}62 \times 10^{-4}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,65	$9{,}45 \times 10^{-4}$	0,03	$4{,}03 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,74	$6{,}58 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,74	$7{,}72 \times 10^{-4}$	0,00	$9{,}40 \times 10^{-1}$	
ES	MLP | Otimização Bayesiana - *NOM_TO_BIN*	99,81	$4{,}33 \times 10^{-4}$			
	MLP | Busca Aleatória - *NOM_TO_BIN*	99,80	$6{,}42 \times 10^{-4}$	0,01	$6{,}76 \times 10^{-1}$	
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Otimização Bayesiana - *NOM_TO_BIN*	99,82	$3{,}64 \times 10^{-4}$			
	$E^{P_h}_{P_l}$*(MLP, K-NN)* | Busca Aleatória - *NOM_TO_BIN*	99,78	$5{,}32 \times 10^{-4}$	0,04	$8{,}71 \times 10^{-2}$	