UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS / FACULDADE DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL

Tales Lima Fonseca

Redes Neurais Artificiais Adaptativas: Abordagem Evolutiva para Melhoria de Performance e Eficiência

Juiz de Fora
2023

Tales Lima Fonseca

Redes Neurais Artificiais Adaptativas: Abordagem Evolutiva para Melhoria de Performance e Eficiência

Tese apresentada ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Doutor em Modelagem Computacional.

Orientador: Prof. Dr. Leonardo Goliatt da Fonseca

Juiz de Fora

2023

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Lima Fonseca, Tales.
Redes Neurais Artificiais Adaptativas : Abordagem Evolutiva para Melhoria de Performance e Eficiência / Tales Lima Fonseca. – 2023.
335 f. : il.

Orientador: Leonardo Goliatt da Fonseca
Tese (Doutorado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas / Faculdade de Engenharia. Programa de Pós-Graduação em Modelagem Computacional, 2023.

1. Redes Neurais Artificiais. 2. Função de Ativação Adaptativa. 3. Otimização Evolutiva. 4. Pegada de Carbono. I. da Fonseca, Leonardo Goliatt, orient. II. Título.

Tales Lima Fonseca

Redes Neurais Artificiais Adaptativas: Abordagem Evolutiva para Melhoria de Performance e Eficiência

Tese apresentada ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Doutor em Modelagem Computacional. Área de concentração: Modelagem Computacional.

Aprovada em 24 de novembro de 2023.

BANCA EXAMINADORA

Prof. Dr. Leonardo Goliatt da Fonseca - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Afonso Celso de Castro Lemonge
Universidade Federal de Juiz de Fora

Profa. Dra. Luciana Conceição Dias Campos
Universidade Federal de Juiz de Fora

Profa. Dra. Camila Martins Saporetti
Universidade do Estado do Rio de Janeiro

Prof. Dr. Gustavo Rocha da Silva
Universidade Presbiteriana Mackenzie

Juiz de Fora, 20/11/2023.

Dedico este trabalho a todas as pessoas que estiveram ao meu lado durante esta longa jornada. Sem o apoio e incentivo de vocês, não teria sido possível chegar até aqui. Agradeço a cada um de vocês pelas palavras de conforto, pelos conselhos valiosos e pelas horas dedicadas a me ajudar a superar os obstáculos. Este é um momento de celebração, mas também de reflexão sobre o quanto cada um de vocês foi importante em minha vida. A todos, muito obrigado.

AGRADECIMENTOS

Quero agradecer, em primeiro lugar, à força divina, seja ela chamada de Deus, universo, natureza ou qualquer outro nome, que acredito ter me guiado e me dado força e coragem durante toda esta jornada.

Minha querida família, cada um de vocês tem um lugar especial em meu coração. Meus amados pais, Vilma e Jânio, minha bondosa avó/madrinha Íris, meu incrível irmão Douglas (referência para mim em diversos aspectos da vida desde criança), minha querida cunhada Flávia, meu amado sobrinho Joaquim e seu irmãozão Wallynho, minha amada esposa Janaína e nossa adorável filhinha de quatro patas Candyzinha, vocês foram meu porto seguro e meu apoio incondicional durante toda esta jornada. Gostaria também de expressar a minha gratidão a minha sogra Marli, meu cunhado e cunhada Aurélio e Patricia, vocês foram tão acolhedores e estou profundamente grato por fazer parte desta família. Em memória, agradeço aos meus amados avós Joaquim, Antônio Tavares e Maria José. Onde quer que vocês estejam, sinto sua presença e espero que estejam orgulhosos de mim.

Quero expressar minha sincera gratidão ao Professor Leonardo Goliatt da Fonseca. Durante a realização deste trabalho, sua orientação foi inestimável e me permitiu atingir novos patamares em minha carreira acadêmica. Além disso, agradeço por todo o ensinamento precioso que ele me transmitiu durante meus anos de graduação, mestrado e, atualmente, doutorado. O conhecimento e a sabedoria que adquiri ao longo desses anos são frutos do seu grande trabalho como professor e orientador.

A UFJF é uma instituição incansável, com uma estrutura que é verdadeiramente digna de admiração. Porém, minha gratidão vai além da instituição em si e se estende aos verdadeiros pilares do Programa de Pós-Graduação em Modelagem Computacional: seus professores e funcionários dedicados. Eles são fontes inesgotáveis de sabedoria e apoio, compartilhando conhecimentos preciosos e auxiliando em todas as etapas de minha jornada acadêmica. Agradeço imensamente a todos eles pelo apoio incondicional e pelos ensinamentos valiosos.

Gostaria de expressar minha mais profunda gratidão ao Itaú Unibanco pelo apoio e oportunidade concedidos durante minha jornada profissional. Sua compreensão e liberdade para que eu pudesse dedicar tempo à pesquisa e ao aprimoramento de meus conhecimentos foram fundamentais para o meu crescimento. Além disso, a experiência como colaborador da empresa me proporcionou um rico aprendizado e contribuiu significativamente para minha formação. Agradeço imensamente por tudo.

A todos os meus queridos amigos, novos e antigos, agradeço a cada um de vocês por sua amizade, presença e apoio incondicional, que contribuíram de forma significativa para chegar até aqui.

“Tudo aqui é a mais absoluta e devastadora
verdade – tirando os trechos que são mentiras.”
Douglas Adams

RESUMO

Nos últimos anos, as redes neurais artificiais têm se destacado como uma das ferramentas mais poderosas no campo da inteligência artificial, oferecendo soluções inovadoras em uma variedade de aplicações. No entanto, o desenvolvimento dessas redes muitas vezes enfrenta desafios significativos relacionados à eficiência computacional e ao impacto ambiental. Esta pesquisa tem como objetivo investigar esses desafios, em particular no processo de criação de redes neurais artificiais, e propor uma solução para melhorar sua eficiência em termos de qualidade preditiva, restrições de hardware e redução dos impactos ambientais. Partindo da hipótese de que as funções de ativação adaptativas são capazes de criar neurônios mais complexos na arquitetura, reduzindo a necessidade de abordagens de aprendizado profundo que resultam em modelos mais pesados e complexos. O foco principal deste estudo é propor o desenvolvimento de uma estratégia evolutiva para a criação de redes neurais artificiais que utilizem funções de ativação adaptativas, com o objetivo de encontrar modelos com maior poder preditivo e com a menor complexidade possível. Isso visa contribuir para a redução das emissões de dióxido de carbono e outros gases associados às mudanças climáticas, decorrentes da execução desses modelos. Foram realizados quatro conjuntos de experimentos numéricos para avaliar a hipótese e a proposta. Os dois primeiros experimentos compararam a eficiência das funções de ativação adaptativas em relação às funções de ativação tradicionais em problemas de classificação e regressão. O terceiro experimento combinou abordagens evolutivas com funções de ativação adaptativas para encontrar o melhor modelo em termos de métrica e complexidade em problemas de Autoencoder. Por fim, no quarto experimento, estendeu-se a aplicação das abordagens evolutivas combinadas com funções de ativação adaptativas, desta vez voltadas para otimizar o modelo em termos de métrica e complexidade em um contexto real de detecção de tumores cerebrais. Todos os resultados obtidos apresentaram um ganho ao se utilizar as Funções de Ativação Adaptativas na construção das arquiteturas, além de um impacto ambiental comparável com estratégias tradicionais. Esses resultados representam um avanço significativo na busca por redes neurais artificiais mais eficazes e ecologicamente conscientes, alinhando Otimização Evolutiva com a responsabilidade ambiental, especialmente na redução da Pegada de Carbono.

Palavras-chave: Redes Neurais Artificiais. Função de Ativação Adaptativa. Otimização Evolutiva. Pegada de Carbono.

ABSTRACT

In recent years, artificial neural networks have emerged as one of the most powerful tools in the field of artificial intelligence, offering innovative solutions across a variety of applications. However, the development of these networks often faces significant challenges related to computational efficiency and environmental impact. This research aims to investigate these challenges, particularly in the process of creating artificial neural networks, and propose a solution to enhance their efficiency in terms of predictive quality, hardware constraints, and environmental impact reduction. Building on the hypothesis that adaptive activation functions are capable of creating more complex neurons in the architecture, thereby reducing the need for deep learning approaches that result in heavier and more complex models, the primary focus of this study is to propose the development of an evolutionary strategy for creating artificial neural networks using adaptive activation functions. The objective is to find models with higher predictive power and the least possible complexity. This endeavor seeks to contribute to the reduction of carbon dioxide emissions and other gases associated with climate change resulting from the execution of these models. Four sets of numerical experiments were conducted to evaluate the hypothesis and the proposed solution. The first two experiments compared the efficiency of adaptive activation functions against traditional activation functions in classification and regression problems. The third experiment combined evolutionary approaches with adaptive activation functions to find the best model in terms of metrics and complexity in Autoencoder problems. Finally, in the fourth experiment, the application of evolutionary approaches combined with adaptive activation functions was extended, this time focusing on optimizing the model in terms of metrics and complexity in a real context of brain tumor detection. All obtained results demonstrated a gain when using Adaptive Activation Functions in architecture construction, along with an environmental impact comparable to traditional strategies. These findings represent a significant advancement in the pursuit of more effective and environmentally conscious artificial neural networks, aligning Evolutionary Optimization with environmental responsibility, especially in reducing Carbon Footprint.

Keywords: Artificial Neural Networks. Adaptive Activation Function. Evolutionary Optimization. Carbon Footprint.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AAF	*Adaptive Activation Function* (Função de Ativação Adaptativa)
ABC	*Artificial Bee Colony* (Colônia de Abelhas Artificiais)
AF	*Activation Function* (Função de Ativação)
AI	*Artificial Intelligence* (Inteligencia Artificial)
ANN	*Artificial Neural Network* (Rede Neural Artificial)
ANOVA	*One-way Analysis of Variance* (Análise de Variância Simples)
ASF	*Achievement Scalarizing Function* (Função Escalar de Realização)
Auto-CVE	*Automated Coevolutionary Voting Ensemble* (Ensemble de Votação Coevolutiva Automatizado)
AutoML	*Automated Machine Learning* (Automatização de Aprendizado de Máquina)
BLU	*Bendable Linear Unit* (Unidade Linear Flexível)
BN	*Batch Normalization* (Normalização em Lote)
CC	*Crowded-Comparison* (Comparação Aglomerada)
CD	*Crowding Distance* (Distância de Aglomeração)
CFGP	*Context-Free Genetic Programming* (Programação Genética Livre de Contexto)
CNN	*Convolutional Neural Network* (Rede Neural Convolucional)
DE	*Differential Evolution* (Evolução Diferencial)
EA	*Evolutionary Algorithm* (Algoritmo Evolutivo)
ELM	*Extreme Learning Machine* (Máquina de Aprendizado Extremo)
ESP	*Enforced SubPopulations* (SubPopulações Forçadas)
FLOPs	*Floating Point Operations per Second* (Operações de Ponto Flutuante por Segundo)
FN	*False Negative* (Falso Negativo)
FP	*False Positive* (Falso Positivo)
GA	*Genetic Algorithm* (Algoritmo Genético)
GAP	*Global Average Pooling* (Média Global de Agrupamento)
GD	*Gradient Descent* (Gradiente Descendente)
GP	*Genetic Programming* (Programação Genética)
GPT-3	*Generative Pretrained Transformer 3* (Transformador Gerador Pré-Treinado 3)
GS	*Grid Search* (Busca em Grade)
GSNN	*Generalized Sigmoidal Neural Network* (Rede Neural Sigmoidal Generalizada)
HOAG	*Hyperparameter Optimization with Approximate Gradient* (Otimização de Hiperparâmetros com Gradiente Aproximado)
IC	*Independent Component* (Componente Independente)
IGS	*Improved Grid Search* (Busca em Grade Aprimorada)

KNAS	*Kernel-based Neural Architecture Search* (Busca de Arquitetura Neural baseado em *kernel*)
KNN	*Kronecker Neural Network* (Rede Neural Kronecker)
LEMONADE	*Lamarckian Evolutionary Algorithm for Multi-Objective Neural Architecture DEsign* (Algoritmo Evolutivo Lamarckiano para Projeto de Arquitetura Neural Multiobjetivo)
LOO	*Leave One Out* (Deixar Um De Fora)
LSTM	*Long short-term memory* (Rede de Memória de Longo-Curto Prazo)
ML	*Machine Learning* (Aprendizado de Máquina)
MLP	*Multi-Layer Perceptron* (Redes Perceptron Multicamadas)
MSE	*Mean Squared Error* (Erro Quadrático Médio)
NAS	*Neural Architecture Search* (Busca de Arquitetura Neural)
NEAT	*NeuroEvolution of Augmenting Topologies* (Neuroevolução de Topologias de Aumento)
NERO	*Neuroevolving Robotic Operatives* (Operativos Robóticos Neuroevoluindo)
NSGA-I	*Nondominated Sorting Genetic Algorithm I* (Algoritmo Genético de Classificação por Não Dominância I)
NSGA-II	*Nondominated Sorting Genetic Algorithm II* (Algoritmo Genético de Classificação por Não Dominância II)
NSGA-III	*Nondominated Sorting Genetic Algorithm III* (Algoritmo Genético de Classificação por Não Dominância III)
ONU	Organização das Nações Unidas
OvO	*One-vs-One* (Um-contra-Um)
OvR	*One-vs-Rest* (Um-contra-Todos)
PCA	*Principal Component Analysis* (Análise de Componentes Principais)
PIB	Produto Interno Bruto
PMLB	*Penn Machine Learning Benchmarks* (Bancos de Teste de Aprendizado de Máquina da Universidade da Pensilvânia)
PO	Problema de Otimização
PRW	*Pattern Recognition Workbench* (Bancada de Reconhecimento de Padrões)
PSO	*Particle Swarm Optimization* (Enxame de Partículas)
R^2	*Coefficient of Determination* (Coeficiente de Determinação)
RL	*Reinforcement Learning* (Aprendizado por Reforço)
RMSE	*Root Mean Squared Error* (Raiz do Erro Quadrático Médio)
RNN	*Recurrent Neural Network* (Rede Neural Recorrente)
ROC	*Receiver Operating Characteristic* (Característica Operacional do Receptor)
ROC AUC	*Area Under an ROC Curve* (Área sob a Curva ROC)
RS	*Random Search* (Busca Aleatória)

rtNEAT	*real-time NeuroEvolution of Augmenting Topologies* (Neuroevolução em tempo real de Topologias de Aumento)
SBX	*Simulated Binary Crossover* (Recombinação por Simulação Binária)
SNN	*Spiking Neural Networks* (Modelos de Disparos Neuronais)
SVM	*Support Vector Machine* (Máquina de Vetores de Suporte)
SVR	*Support Vector Regression* (Máquina de Vetores Suporte para Regressão)
TL	*Transfer Learning* (Transferência de Conhecimento)
TPOT	*Tree-based Pipeline Optimization Tool* (Ferramenta de Otimização de Pipeline baseada em Árvore)
TN	*True Negative* (Verdadeiro Negativo)
TP	*True Positive* (Verdadeiro Positivo)
Uout	*Uncertainty Out* (Incerteza Fora)
WEKA	*Waikato Environment for Knowledge Analysis* (Ambiente Waikato para Análise de Conhecimento)

SUMÁRIO

1 INTRODUÇÃO

As redes neurais artificiais são modelos computacionais inspirados no cérebro humano, composto por neurônios artificiais que estão organizados em camadas interconectadas (RUSSELL; NORVIG, 2021). Através de um processo de aprendizado a partir de exemplos de dados, estas redes são capazes de realizar um conjunto de tarefas complexas, como processamento de linguagem natural (GOLDBERG, 2017), reconhecimento de imagens (LECUN et al., 2015), veículos autônomos (LIU et al., 2017), diagnósticos médico (MALHOTRA et al., 2020) e até mesmo tomadas de decisão em investimento (JIANG et al., 2017).

Grandes empresas da tecnologia utilizam redes neurais artificiais em seus produtos hoje em dia. Um exemplo comumente comentado é a Netflix (STECK et al., 2021), que utiliza de redes neurais artificiais para realizar recomendações de filmes e séries aos seus assinantes, aumentando desta forma o engajamento e a retenção de seus clientes.

1.1 MOTIVAÇÃO

Apesar disso, o processo de criação de uma rede neural artificial ainda é um grande desafio, pois requer que o usuário escolha vários parâmetros diferentes, como o número de camadas e neurônios, a função de ativação por camada, seus parâmetros de treinamento, entre outros (LECUN et al., 2015). Este processo, quando realizado manualmente e sem muito cuidado, pode gerar modelos semi-ótimos ou até mesmo ruins, o que pode afetar significativamente a performance de predição, o tempo gasto de treinamento, os recursos computacionais utilizados e até mesmo o impacto ambiental, devido ao consumo de energia elevado durante o processo de treinamento e teste das redes (STRUBELL et al., 2019).

Estes problemas podem ser endereçados através de melhores práticas de seleção de parâmetros, automatização do processo de criação de redes neurais e pesquisas em técnicas de otimização de redes neurais. Ao proporcionar um algoritmo capaz de encontrar os melhores modelos de redes neurais artificiais para qualquer problema supervisionado, e que seja possível ser implantado em qualquer *hardware*, devido a sua simplicidade, espera-se contribuir para o campo de estudo e ao mesmo tempo ajudar o meio ambiente com menos pegadas de carbono [1].

1.2 JUSTIFICATIVA

Atualmente, no Brasil, existem iniciativas para atingir os Objetivos de Desenvolvimento Sustentável da Organização das Nações Unidas (ONU) (Nações Unidas no Brasil,

[1] É a medida do impacto ambiental causado pela Inteligencia Artificial em termos de emissões de dióxido de carbono (CO_2) e outros gases que contribuem para as mudanças climáticas (SCHWARTZ et al., 2020).

2022). Este projeto apresenta 17 objetivos que visam combater os principais desafios de desenvolvimento enfrentados por pessoas no Brasil e no mundo, como acabar com a pobreza, proteger o meio ambiente e o clima, e garantir que as pessoas, em todos os lugares, possam desfrutar de paz e prosperidade. A Figura 1, extraída de Nações Unidas no Brasil (2022), apresenta quais são os objetivos para os quais a ONU está contribuindo, a fim de que possamos atingi-los até 2030 no Brasil.

Figura 1 – Objetivos de Desenvolvimento Sustentável da ONU.

Fonte: Extraída de Nações Unidas no Brasil (2022).

O trabalho proposto aqui está relacionado com o 9º objetivo da ONU, denominado "Indústria, Inovação e Infraestrutura". Este objetivo visa construir infraestruturas resilientes, promover a industrialização inclusiva e sustentável, e fomentar a inovação. Dentro deste objetivo, podemos listar os seus objetivos específicos, também definidor pela ONU, nos quais a presente proposta busca contribuir de forma efetiva, sendo eles:

1. Desenvolver infraestrutura de qualidade, confiável, sustentável e resiliente, incluindo infraestrutura regional e transfronteiriça, para apoiar o desenvolvimento econômico e o bem-estar humano, com foco no acesso equitativo e a preços acessíveis para todos;

2. Promover a industrialização inclusiva e sustentável e, até 2030, aumentar significativamente a participação da indústria no setor de emprego e no PIB, de acordo com as circunstâncias nacionais, e dobrar sua participação nos países menos desenvolvidos;

3. Aumentar o acesso das pequenas indústrias e outras empresas, particularmente em países em desenvolvimento, aos serviços financeiros, incluindo crédito acessível, e sua integração em cadeias de valor e mercados;

4. Até 2030, modernizar a infraestrutura e reabilitar as indústrias para torná-las sustentáveis, com eficiência aumentada no uso de recursos e maior adoção de tecnologias e

processos industriais limpos e ambientalmente corretos, com todos os países atuando de acordo com suas respectivas capacidades;

5. Fortalecer a pesquisa científica, melhorar as capacidades tecnológicas de setores industriais em todos os países, particularmente os países em desenvolvimento, inclusive, até 2030, incentivando a inovação e aumentando substancialmente o número de trabalhadores de pesquisa e desenvolvimento por milhão de pessoas, além dos gastos público e privado em pesquisa e desenvolvimento;

6. Facilitar o desenvolvimento de infraestrutura sustentável e resiliente em países em desenvolvimento, por meio de maior apoio financeiro, tecnológico e técnico aos países africanos, aos países menos desenvolvidos, aos países em desenvolvimento sem litoral e aos pequenos Estados insulares em desenvolvimento;

7. Apoiar o desenvolvimento tecnológico, a pesquisa e a inovação nacionais nos países em desenvolvimento, inclusive garantindo um ambiente político propício para, entre outras coisas, a diversificação industrial e a agregação de valor às commodities;

8. Aumentar significativamente o acesso às tecnologias de informação e comunicação e se empenhar para oferecer acesso universal e a preços acessíveis à internet nos países menos desenvolvidos, até 2020 [2].

A otimização do processo de criação de redes neurais artificiais pode contribuir para o fortalecimento da pesquisa científica e melhoria das capacidades tecnológicas de setores industriais. Isso é alcançado através da criação de modelos mais eficientes e precisos, o que aumenta a capacidade de inovar e aplicar esses modelos em diversas áreas, como saúde, commodities e outras. Além disso, a otimização do processo de criação pode contribuir para a redução do desperdício de recursos computacionais e energia, o que pode ser importante para países em desenvolvimento.

1.3 OBJETIVOS

Nesta seção, será revelado o Objetivo Geral e os Objetivos Específicos elaborados que guiaram o desenvolvimento do projeto proposto.

1.3.1 Objetivo Geral

O objetivo principal deste trabalho é desenvolver uma estratégia evolutiva de criação de Redes Neurais Artificiais que utilizam Funções de Ativação Adaptativas, com o intuito de encontrar as melhores arquiteturas em dois critérios de otimização: maximizar o

[2] Apesar de já ter passado de 2020, aqui esta apenas listado quais são os objetivos da ONU, podendo ser considerado, com o objetivo geral, a data de 2030 para atingi-lo.

poder preditivo e minimizar a complexidade. As Funções de Ativação Adaptativas por si só não são capazes de influenciar a arquitetura da rede já definida, mas a sua combinação com um processo evolutivo visa encontrar arquiteturas mais enxutas. Esta proposta busca encontrar modelos que possam ser implantados em diferentes tipos de *Hardware*, incluindo aqueles com menor recurso computacional, e que reduzam a pegada de carbono.

1.3.2 Objetivos Específicos

A proposta consiste em cinco objetivos específicos, onde o processo de construção de cada um depende diretamente dos seus antecessores. Os objetivos são:

O1 - Estudar e definir as funções objetivos necessárias para maximizar o poder preditivo e minimizar a complexidade de uma Rede Neural Artificial;

O2 - Definição e mapeamento do espaço de busca para um modelo de Rede Neural Artificial, responsável por transformar uma solução do problema de otimização em um modelo;

O3 - Estudar e propor funções de ativação que possam se adaptar para ajudar no processo de maximização do poder preditivo e minimização da complexidade de uma Rede Neural Artificial;

O4 - Avaliar o desempenho de métodos evolutivos de otimização para o problema definido;

O5 - Identificar qual processo é capaz de obter os resultados desejáveis.

1.4 ORGANIZAÇÃO DO TEXTO

Esta tese está organizada em oito capítulos, sendo esta introdução o primeiro deles. O Capítulo 2 é destinado à revisão bibliográfica, apresentando os principais trabalhos de automatização em aprendizado de máquina e as diferenças entre as técnicas propostas. Além disso, são explorados trabalhos relacionados a funções de ativação adaptativas.

No Capítulo 3, é apresentado o que é otimização, a formulação de um problema de otimização, a classificação dos algoritmos existentes e um detalhamento de todos os métodos de otimização utilizados para o desenvolvimento do trabalho.

O Capítulo 4 tem como objetivo discutir o conceito de aprendizado de máquina, incluindo os diferentes tipos de aprendizado de máquina, as categorias de aprendizado supervisionado, a generalização dos modelos, o dilema do viés e variância, e, finalmente, o processo de avaliação dos modelos.

O Capítulo 5 tem como objetivo proporcionar uma visão geral do campo de redes neurais, uma técnica de aprendizado de máquina baseada em modelos inspirados na estrutura e função do cérebro humano. Este capítulo começa com uma introdução ao

conceito básico de redes neurais, incluindo o *Perceptron* e as redes *Perceptron* multicamadas. Em seguida, é discutido a aprendizagem de redes neurais, incluindo as funções de ativação tradicionais e sua variação adaptativa. Finalmente, são exploradas as tendências e estratégias atuais que trazem ganhos na arquitetura de redes neurais.

Já o Capítulo 6 expõe o objetivo do método de automatização proposto, com detalhes da construção de sua estrutura com base em todos os conhecimentos técnicos descritos nos capítulos anteriores.

No Capítulo 7, são apresentados os experimentos numéricos realizados durante o desenvolvimento deste trabalho e, por fim, as conclusões são apresentadas no Capítulo 8 com algumas sugestões de trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

A revisão bibliográfica deste capítulo foca em uma síntese dos trabalhos de automatização em aprendizado de máquina e de funções de ativação adaptativas, através de uma cronologia que contempla as publicações pioneiras e o estado da arte. Além disso, foi feito o levantamento dos principais algoritmos utilizados para este propósito e suas características individuais.

2.1 AUTOMATIZAÇÃO DE APRENDIZADO DE MÁQUINA

A Automatização de Aprendizado de Máquina (*Automated Machine Learning* - AutoML) para a criação de modelos teve início em 1995, pela empresa Unica Technologies Inc., onde foi desenvolvido um *software* chamado Bancada de Reconhecimento de Padrões (*Pattern Recognition Workbench* - PRW) que simplificava a tarefa de criar modelos financeiros orientados a dados. Este *software* realizava uma busca exaustiva em um conjunto de possíveis parâmetros, conhecido como Busca em Grade (*Grid Search* - GS) (CHAMBERS; DINSMORE, 2014), de modo a otimizar modelos de Rede Neural Artificial (*Artificial Neural Network* - ANN). Embora a técnica apresente simplicidade no seu processo de busca, muitos trabalhos ainda a utilizam e obtêm resultados satisfatórios (RÄTSCH et al., 2001; LAVALLE et al., 2004; RODI, 2006; BAO; LIU, 2006; DINH; BAAN, 2019).

Em alguns casos, como no trabalho de Sun et al. (2021), o GS pode apresentar muitos intervalos de pesquisa inválidos e sensíveis à etapa de busca. Desta forma, os autores propuseram no trabalho uma variação do GS, chamada de Busca em Grade Aprimorada (*Improved Grid Search* - IGS), para resolver a busca de alguns hiperparâmetros de uma Máquina de Vetores de Suporte para Regressão (*Support Vector Regression* - SVR), alterando automaticamente o intervalo e o passo durante o processo de busca.

Apesar da escolha manual e do GS serem as estratégias mais usadas para otimização de hiperparâmetros, Bergstra e Bengio (2012) propuseram um algoritmo chamado Busca Aleatória (*Random Search* - RS), que realiza exatamente uma busca aleatória, em um conjunto de possíveis parâmetros e realizaram a comparação entre eles, obtendo que o RS encontra melhores modelos na maioria dos casos e exige menos tempo computacional. Assim como o GS, o RS é uma técnica simples capaz de obter resultados satisfatórios na escolha de parâmetros de um modelo, havendo um conjunto de trabalhos que comprovam este benefício (RANJBAR; MARBURG, 2013; OLIINYK et al., 2014; FENG; SHEN, 2015; MANTOVANI et al., 2015; VILLALOBOS-ARIAS; QUESADA-LÓPEZ, 2021).

Chapelle et al. (2002) propuseram em seu trabalho a automatização dos hiperparâmetros de uma Máquina de Vetores de Suporte (*Support Vector Machine* - SVM), usando o algoritmo de otimização Gradiente Descendente (*Gradient Descent* - GD) para pesquisar o espaço dos parâmetros, o que levou a uma melhoria do desempenho e uma redução da

complexidade da solução. Com a mesma ideia, Maclaurin et al. (2015) utilizou o GD para a busca de alguns hiperparâmetros de uma ANN, como a sua arquitetura, esquemas de regularização, inicialização de pesos, etc.

Chen et al. (2004) apresentaram em seu estudo uma outra estratégia, considerada mais eficiente, para a otimização de hiperparâmetros. O trabalho propôs a utilização de um Algoritmo Genético (*Genetic Algorithm* - GA) para a otimização dos parâmetros C e γ de um SVM. Além disso, o trabalho apresentou que uma busca guiada pode trazer melhores resultados do que um GS em um menor intervalo de tempo. No mesmo ano, Samanta (2004) comparou um modelo de ANN contra um modelo SVM para detecção de falha de engrenagem, onde os hiperparâmetros de ambos os modelos e a seleção das variáveis de entrada foram otimizados utilizando um GA.

A busca direcionada de hiperparâmetros utilizando meta-heurísticas de otimização tornou-se cada vez mais frequente devido à sua capacidade de encontrar soluções boas em um pequeno intervalo de tempo. As meta-heurísticas que mais se destacaram neste processo de busca foram: Algoritmo Genético (COHEN et al., 2004; FRIDRICH, 2017; FRANCESCOMARINO et al., 2018; FONSECA et al., 2019; NIKBAKHT et al., 2021), Colônia de Abelhas Artificiais (*Artificial Bee Colony* - ABC) (MUSTAFFA; YUSOF, 2012; ÇAM et al., 2015; NIETO et al., 2017; YULIYONO; GIRSANG, 2019; ERKAN et al., 2022), Enxame de Partículas (*Particle Swarm Optimization* - PSO) (SOUZA et al., 2006; GUO et al., 2008; TOAL et al., 2011; LORENZO et al., 2017; SINGH et al., 2021) e Evolução Diferencial (*Differential Evolution* - DE) (ZHANG et al., 2011; LIU, 2017; PENG et al., 2018; SAPORETTI et al., 2018; SANTOS et al., 2021).

Apesar de existirem diversos trabalhos de otimização de hiperparâmetros através de meta-heurísticas, alguns pesquisadores exploraram e apresentaram excelentes resultados utilizando otimização Bayesiana (BERGSTRA et al., 2011; SNOEK et al., 2012; SWERSKY et al., 2013; EGGENSPERGER et al., 2013; KLEIN et al., 2016; ALIBRAHIM; LUDWIG, 2021), abrindo espaço para outro campo de otimização dentro da busca dos melhores hiperparâmetros.

A otimização de hiperparâmetros traz grandes benefícios na solução de um problema de aprendizado de máquina. Independentemente disso, outras medidas podem e devem ser adotadas para melhorar cada vez mais o modelo final. Uma destas medidas é o processo de seleção e transformação dos dados de entrada, que em sua grande maioria é feito pelo conhecimento do profissional da área e é capaz de melhorar consideravelmente a qualidade do modelo (DOMINGOS, 2012; ZHENG; CASARI, 2018).

Escalante et al. (2009) foram uns dos primeiros a apresentarem uma abordagem completa de seleção de modelos. Sua abordagem consistiu em utilizar o PSO para a seleção de uma combinação de pré-processamento, seleção de variáveis e métodos de aprendizado para obter um classificador final com o menor erro possível. Além de

realizar esta combinação, cada modelo de classificação passa por uma etapa de busca dos hiperparâmetros ótimos. Os resultados obtidos por este trabalho mostraram que, para o estudo realizado, a técnica obteve melhores resultados em comparação com os mesmos algoritmos de aprendizado sem o processo proposto e que a utilização de variáveis mais descritivas pode trazer melhores resultados.

Cheng et al. (2011) apresentaram em seu trabalho uma maneira automática de geração de novos dados de entrada com base nos dados iniciais. A proposta atingiu seu objetivo para um domínio de problema específico, que era o interesse dos autores, mas se mostrou limitada na generalização para outros problemas devido à necessidade de possuir conhecimento do domínio da aplicação. Por outro lado, Kanter e Veeramachaneni (2015) propuseram, de maneira automática, o primeiro processo de seleção e transformação dos dados de entrada sem a necessidade de conhecimento do domínio do problema. Seus resultados mostraram ser competitivos com soluções geradas por humanos (não automáticas) em um conjunto de bases testadas.

Thornton et al. (2013) apresentaram o método Auto-WEKA, um pacote de automatização de modelos que funciona com o *software* "Ambiente Waikato para Análise de Conhecimento" (*Waikato Environment for Knowledge Analysis* - WEKA) (HALL et al., 2009; WITTEN et al., 2016). Neste trabalho, eles demonstraram que a escolha do modelo e seus hiperparâmetros poderia ser feita através de um único problema de otimização hierárquico de hiperparâmetros, onde até mesmo a escolha do modelo é considerada como um hiperparâmetro. Nesta formulação do problema, eles demonstraram que a otimização Bayesiana pode obter modelos com uma alta qualidade preditiva em um tempo razoável e, principalmente, com o mínimo de esforço humano. Outros trabalhos surgiram utilizando o método Auto-WEKA para o seu próprio problema de aprendizado de máquina (SALVADOR et al., 2016; FREITAS et al., 2018; ALSOLAI; ROPER, 2019).

Feurer et al. (2015) desenvolveram um método chamado auto-sklearn que utiliza otimização Bayesiana para encontrar a combinação ideal de pré-processamento dos dados e características, modelos e os hiperparâmetros do mesmo. No processo de otimização do auto-sklearn, o espaço de busca contempla apenas um pré-processador de dados, um pré-processador de características e um modelo. Deste modo, a solução se limita a um *pipeline* funcional, porém pequeno, para solução de problemas de aprendizado. Esta característica pode impactar negativamente a qualidade da solução para problemas mais complexos.

Com o intuito de melhorar os resultados do auto-sklearn, Feurer et al. (2019) propuseram armazenar todos os *pipelines* gerados no processo de busca com o intuito de gerar um *ensemble* dos mesmos. Como descrito por eles, os *pipelines* já são gerados no processo de busca, o que não aumenta a complexidade do método até o momento. O estudo foi desenvolvido com o intuito de verificar a melhor maneira de se combinar os

modelos e observou-se que o ponderamento uniforme de cada *pipeline* não apresentou bons resultados. Deste modo, exploraram *ensembles* do tipo *stacking* e otimização numérica livre de gradiente e observaram que ambos tendem a sobreajustar o conjunto de validação e são computacionalmente caros, enquanto a Seleção de *Ensemble*, proposta por Caruana et al. (2004), apresentou ser robusta e rápida.

A empresa H2O.ai (2016) desenvolveu uma estratégia de automatização, onde diferentes modelos pré determinados são treinados e passam por uma validação cruzada. Alguns destes modelos já possuíam seus hiperparâmetros definidos e outros passaram por uma busca utilizando RS. Após estes treinamentos, dois *ensembles* do tipo *stacking* são treinados. Um dos *ensembles* contém todos os modelos e o segundo contém apenas os modelos com melhor desempenho de cada "classe" de algoritmos. O segundo *ensemble* é otimizado para ser utilizado em produção, pois possui um menor número de modelos combinados. Apesar de ser um *ensemble* mais rápido, ele pode ter um desempenho pior do que o outro *ensemble* devido a sua simplificação. A empresa tem como plano de negócio democratizar a Inteligencia Artificial (*Artificial Intelligence* - AI) para todos e capacitar todas as empresas a serem uma empresa de AI, tendo constantes evoluções nas suas soluções para melhor atender seus clientes. Neste contexto, foi apresentado um retrato das soluções que a empresa desenvolveu até o momento, reconhecendo que a estratégia futura pode seguir outros caminhos.

Pedregosa (2016) apresentou um algoritmo de Otimização de Hiperparâmetros com Gradiente Aproximado (*Hyperparameter Optimization with Approximate Gradient* - HOAG). No trabalho foi apresentada a vantagem que o algoritmo tem de poder atualizar os seus hiperparâmetros antes da convergência completa dos parâmetros do modelo, ou seja, antes de atingir o mínimo do treinamento. Desta forma, o espaço de busca não é fixo, possibilitando encontrar a solução em regiões que não foram mapeadas na inicialização do algoritmo.

Olson et al. (2016a), Olson et al. (2016b), Le et al. (2020) desenvolveram um conceito de otimização de *pipeline* baseado em árvore chamado "Ferramenta de Otimização de Pipeline baseada em Árvore" (*Tree-based Pipeline Optimization Tool* - TPOT) para automatizar o projeto de *pipeline* de um problema de aprendizado de máquina. O TPOT utiliza uma versão de Programação Genética (*Genetic Programming* - GP) (BANZHAF et al., 1998) para projetar e otimizar automaticamente uma série de transformações de dados e modelos de aprendizado de máquina que tentam maximizar a performance para um determinado conjunto de dados de aprendizado supervisionado. Diferente do auto-sklearn, que constrói um *pipeline* com um número delimitado de operadores, o TPOT não é limitado na quantidade de operações que pode realizar nos dados e no modelo.

Chen et al. (2018) apresentaram em seu trabalho a automatização da criação de *ensembles* do tipo *stacking* utilizando Algoritmo Evolutivo (*Evolutionary Algorithm* - EA)

denominado Autostacker. Em sua abordagem, não são utilizados pré-processamento de dados ou seleção de características como a maioria das técnicas apresentadas. Apesar disso, a técnica apresentou resultados competitivos, tanto em relação à métrica quanto em velocidade em relação ao auto-sklearn e ao TPOT.

Larcher e Barbosa (2019) trabalharam em uma abordagem de coevolução, denominada "*Ensemble* de Votação Coevolutiva Automatizado" (*Automated Coevolutionary Voting Ensemble* - Auto-CVE), onde duas populações são evoluídas simultaneamente e interagindo umas com as outras. Uma população é composta por componentes, onde um componente é uma sequência de operações e hiperparâmetros, que é evoluída através da Programação Genética Livre de Contexto (*Context-Free Genetic Programming* - CFGP). A outra população é composta por *ensembles* que são evoluídos através do GA. A estratégia de *ensemble* utilizada é a por votação com voto proporcional e cada indivíduo da população de *ensembles* corresponde a um *ensemble* de votação diferente. Os resultados obtidos nos experimentos numéricos mostraram que o Auto-CVE é capaz de encontrar resultados comparáveis com o atual estado da arte, o TPOT. Além disso, os autores mostraram que o tempo computacional é menor ou igual em quase todas as execuções.

Dando sequência no trabalho, Larcher e Barbosa (2021) propuseram uma alteração no Auto-CVE tradicional, que utiliza validação cruzada k-fold, por um *holdout* dinâmico. A alteração é feita na etapa de medição da performance do *ensemble*, onde na nova solução é realizada a mudança do conjunto a cada geração. Através de um conjunto de experimentos, foi possível observar que a nova estratégia mantém o nível de desempenho de sua antecessora e demanda menos tempo computacional, além de ter uma medida média de *overfitting* mais baixa.

A partir de 2017, várias empresas como Amazon (2017), Golovin et al. (2017), Alibaba Cloude (2018) e Baidu (2018) começaram a lançar soluções comerciais relacionadas ao AutoML, dando maior importância ao assunto no mundo (GOOGLE, 2022).

Por se tratar de um assunto relativamente recente na literatura, muitos trabalhos ainda estão sendo desenvolvidos e a área de AutoML está ganhando significativa força ao longo do tempo. Encontrar o melhor *pipeline* de modelagem, ou conjunto deles, para um determinado problema em um curto intervalo de tempo, sem impactar a sua implantação, ainda é um grande desafio. Apesar disso, novos métodos estão sendo desenvolvidos para abordar esses problemas (HUTTER et al., 2019; KIM et al., 2022a).

2.1.1 Busca de uma arquitetura de redes neurais artificiais

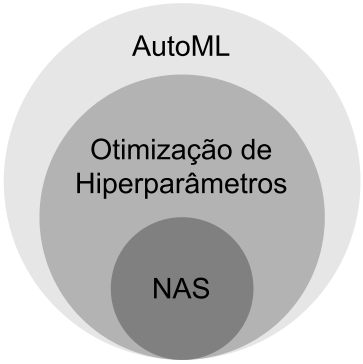
Até o momento, todos os trabalhos apresentados aqui eram agnósticos ao algoritmo de aprendizado de máquina utilizado, tornando assim as soluções genéricas, sem aprofundar em características próprias dos algoritmos. Neste trabalho, focamos na busca de uma arquitetura de ANN (Neural Architecture Search - NAS), uma subárea de AutoML

(HUTTER et al., 2019). Além disso, NAS tem uma grande sobreposição com a área de otimização de hiperparâmetros e até mesmo meta-aprendizagem (HUTTER et al., 2018).

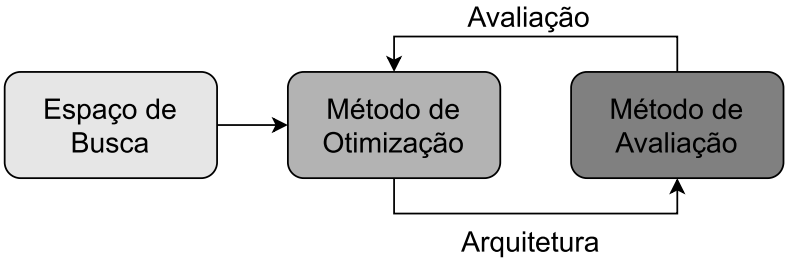
A Figura 2a apresenta a posição da NAS em relação à Otimização de Hiperparâmetros e ao AutoML, enquanto a Figura 2b mostra os componentes da NAS. A Figura 3, adaptada de He et al. (2021), ilustra um pipeline de AutoML abrangendo Preparação de Dados, Engenharia de Características e Geração de Modelo. A Geração de Modelos pode ser subdividida em Espaço de Busca e Métodos de Otimização, e dentro desse bloco se encontra a busca por uma arquitetura de ANN. Assim, prosseguiremos com uma revisão bibliográfica no contexto de NAS.

Figura 2 – Características da NAS.

(a) NAS em relação a Otimização de Hiperparâmetros e AutoML.



(b) Componentes da NAS.

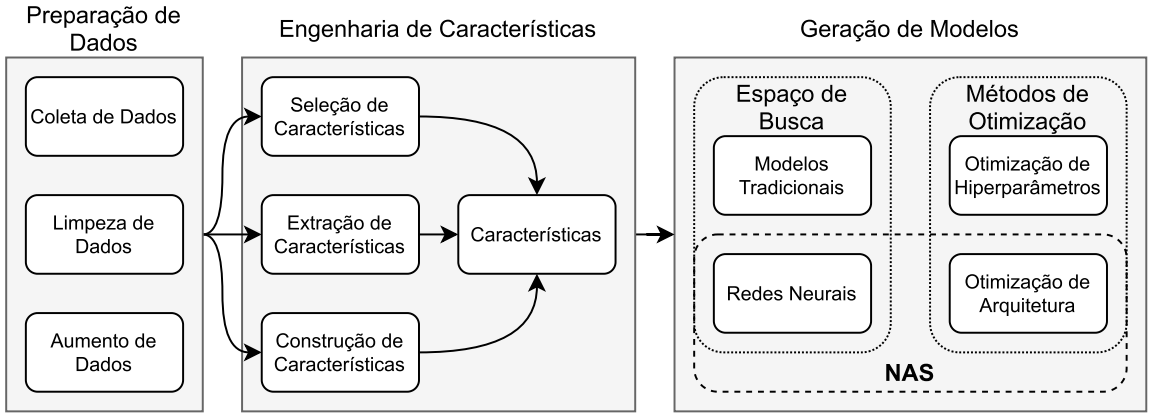


Fonte: Elaborada pelo autor (2023).

Os estudos relacionados à criação automática de ANN tiveram início no final dos anos 80 e início dos anos 90 (HARP et al., 1989; RONALD; SCHOENAUER, 1994), quando estratégias baseadas em GA eram empregadas para encontrar a melhor arquitetura e os pesos das redes (SCHAFFER et al., 1992), inaugurando a área conhecida como Neuroevolução. Vale ressaltar que NAS não se limita estritamente à Neuroevolução por definição, pois não depende necessariamente do uso de EA para encontrar suas soluções.

Wierstra et al. (2005) apresentou um trabalho evolutivo em uma Rede Neural Recorrente (*Recurrent Neural Network* - RNN) chamada Rede de Memória de Longo-Curto Prazo (*Long short-term memory* - LSTM). O processo evolutivo de SubPopulações Forçadas

Figura 3 – Pipeline de AutoML cobrindo Preparação de Dados, Engenharia de Características e Geração de Modelo.



Fonte: Adaptada de He et al. (2021).

(*Enforced SubPopulations* - ESP) serve para descobrir bons pesos para os nós ocultos da LSTM, enquanto se utilizava de uma regressão linear para realizar o mapeamento linear ideal para a camada de saída. Os resultados apresentados no trabalho superaram os métodos estado da arte da época.

Gauci e Stanley (2007) e Risi e Stanley (2012) utilizaram o algoritmo "Neuroevolução de Topologias de Aumento" (*NeuroEvolution of Augmenting Topologies* - NEAT) (STANLEY; MIIKKULAINEN, 2002; STANLEY et al., 2005), uma ANN criada utilizando GA, e obtiveram resultados aceitáveis em seus estudos. No entanto, Verbancsics e Harguess (2013) apresentaram um estudo utilizando o mesmo algoritmo NEAT e observaram que o mesmo não apresentava um desempenho bom como as ANN criadas manualmente por especialistas. Através destes trabalhos, pode-se observar que a técnica de criação de uma ANN utilizando um GA apresentou resultados satisfatórios, mas tem suas limitações em alguns cenários de aplicação.

Stanley et al. (2005) apresentaram em seu trabalho uma modificação do algoritmo NEAT para tempo real, denominado "Neuroevolução em tempo real de Topologias de Aumento" (*real-time NeuroEvolution of Augmenting Topologies* - rtNEAT), para evoluir ANN complexas em tempo real enquanto um jogo está sendo jogado. Um jogo chamado "Operativos Robóticos Neuroevoluindo" (*Neuroevolving Robotic Operatives* - NERO) foi desenvolvido com base no rtNEAT com o intuito de permitir ao jogador treinar uma equipe de robôs virtuais para combater as equipes de outros jogadores. O trabalho mostra que o rtNEAT possibilita a existência de jogos como o NERO, nos quais os agentes evoluem e se adaptam em tempo real para o problema proposto pelo jogo.

Such et al. (2017) utilizaram um GA para evoluir os pesos de uma ANN profunda,

e os resultados apresentam bons desempenhos em problemas de Aprendizado por Reforço (*Reinforcement Learning* - RL), incluindo jogos de Atari e locomoção de humanoides. A solução proposta pelos autores apresentou resultados competitivos com algoritmos populares de RL na literatura.

Real et al. (2017) utilizaram EA no processo de busca da arquitetura ótima em um problema de classificação de imagens com o objetivo de minimizar a participação humana. O trabalho apresentou que é possível desenvolver modelos automáticos com desempenho igual aos publicados no passado por arquiteturas manuais.

Nadkarni e Neves (2018) apresentaram em seu trabalho uma combinação de Análise de Componentes Principais (*Principal Component Analysis* - PCA) com o algoritmo NEAT com o intuito de gerar um sinal de negociação capaz de obter altos retornos e lucros diários com baixo risco associado em negociações nos mercados financeiros.

Apesar de muitos trabalhos utilizarem EA para encontrar a melhor arquitetura, Zoph e Le (2016) e Baker et al. (2016) desenvolveram uma estratégia que utiliza RL para encontrar a arquitetura da rede neural. Zoph e Le (2016) utilizaram RNN para gerar as descrições do modelo de ANN, treinando o mesmo com RL com o objetivo de maximizar a precisão das arquiteturas em um conjunto de validação. Os resultados apresentados superaram os do estado da arte. Baker et al. (2016) apresentaram uma abordagem direcionada a encontrar a melhor arquitetura para uma Rede Neural Convolucional (*Convolutional Neural Network* - CNN) utilizando RL para encontrar cada uma de suas camadas. Os resultados do trabalho mostraram que os modelos gerados superaram as redes existentes projetadas com os mesmos tipos de camada e são competitivos em relação aos métodos estado da arte que utilizam camadas mais complexas em sua arquitetura.

Apesar do desempenho dessas técnicas, muitas delas possuem um custo computacional muito elevado, o que impede a sua utilização em alguns cenários. Consequentemente, novos trabalhos vêm surgindo com o intuito de obter arquiteturas com poder preditivo maior e com um tempo computacional bem reduzido.

Cai et al. (2018) propuseram a reutilização de uma arquitetura e dos pesos como ponto de partida. Assim, utilizando RL, o agente pode tomar duas possíveis ações: aumentar a profundidade da rede ou a largura da camada, com transformações de preservação de função. A rede gerada por esse processo pode ser reutilizada no mesmo contexto ou até mesmo como ponto de partida para outro contexto, economizando assim uma grande quantidade de recursos computacionais.

Pham et al. (2018) apresentaram em seu trabalho um controlador que realiza a busca de um sub-gráfico ideal dentro de um grande gráfico formado por todo o espaço de busca. Cada sub-gráfico se torna uma arquitetura de rede. O controlador é treinado por RL com o objetivo de encontrar o sub-gráfico que maximize a recompensa esperada em uma base de validação, enquanto o modelo gerado pelo sub-gráfico é treinado para

minimizar um erro. A técnica proposta compartilha os parâmetros entre os modelos, reduzindo o tempo de busca em até 1000 vezes em relação a uma busca tradicional de NAS.

Um dos primeiros trabalhos de NAS com multiobjetivo foi o de Elsken et al. (2018). O trabalho propõe o Algoritmo Evolutivo Lamarckiano para Projeto de Arquitetura Neural Multiobjetivo (*Lamarckian Evolutionary algorithm for Multi-Objective Neural Architecture DEsign* - LEMONADE), que busca maximizar o desempenho preditivo da arquitetura e, ao mesmo tempo, minimizar seu número de parâmetros. Além disso, os autores desenvolveram um mecanismo baseado na teoria Lamarckista, onde a geração das ANN filhas herda o desempenho preditivo de seus pais treinados. A combinação desta busca multiobjetivo com a teoria Lamarckista possibilitou que a técnica apresentasse desempenhos competitivos, tanto em eficiência como em performance, em problemas de visão computacional.

Similar ao trabalho anterior, Baldeon-Calisto e Lai-Yuen (2020) buscaram trabalhar com uma abordagem multiobjetivo para minimizar o número de parâmetros de um modelo CNN, conhecido como U-Net, e maximizar a medida de similaridade para um problema de segmentação de imagens médicas de ressonância magnética. Os benefícios foram tanto em performance do modelo quanto em redução do tempo de treinamento e de predição

Lu et al. (2019) seguiram uma abordagem similar, onde se buscava minimizar o erro de uma métrica e a complexidade computacional medida por Operações de Ponto Flutuante por Segundo (*Floating Point Operations per Second* - FLOPs). Para este trabalho, os autores utilizaram um Algoritmo Genético de Classificação por Não Dominância II (*Nondominated Sorting Genetic Algorithm II* - NSGA-II) (DEB et al., 2002), onde a busca se inicializa através de uma população construída por arquiteturas artesanais, uma etapa de exploração composta por operadores de recombinação e mutação de arquiteturas e, por fim, a etapa de explotação que utiliza todo conhecimento histórico obtido até o momento em uma rede Bayesiana. A técnica foi comparada com arquiteturas artesanais e apresentou resultados significativamente melhores, tanto em performance quanto em FLOPs. Quando comparada com outras abordagens de NAS de apenas um objetivo em um problema de classificação para a base de dados CIFAR-10, ela se mostrou mais favorável.

Xu et al. (2021) propuseram um NAS baseado em *kernel* de gradiente, abreviado como KNAS. O trabalho busca atuar no contexto de IA verde (*Green AI*) (SCHWARTZ et al., 2020), visando eficiência nas técnicas para diminuir a pegada de carbono (*carbon footprint*). Para atingir esse objetivo, os autores buscaram uma alternativa de avaliação das arquiteturas sem a necessidade de realizar a etapa de treinamento, utilizando *kernel* de gradiente que, por meio de um estudo teórico, demonstrou uma boa correlação com a perda de treinamento e o desempenho de validação. Foram realizados experimentos que mostraram que a técnica é competitiva em relação a outras abordagens de NAS, apresentando uma velocidade de busca mais rápida do que as técnicas que dependem de

treinamento.

Chen et al. (2021) realizaram um dos primeiros estudos sobre o uso de NAS para encontrar uma arquitetura baseada em *Transformer* (VASWANI et al., 2017) na tarefa de reconhecimento de imagem. O processo de busca é capaz de identificar uma variante do *Transformer* que alcançou melhor desempenho do que soluções tradicionais utilizando a mesma configuração de treinamento. Além disso, os autores apresentaram um novo módulo local para conduzir a busca do *Transformer* no contexto de visão computacional, reduzindo o custo computacional e permitindo a modelagem de correlações locais. Para alcançar esses resultados, os autores utilizaram um NAS hierárquico que lida de maneira eficiente com espaços de busca extensos no *Transformer*, melhorando os resultados da busca. A abordagem proposta demonstrou melhores resultados em comparação com *Transformers* tradicionais aplicados à visão computacional.

Chen et al. (2021), simultaneamente ao trabalho de outros autores Chen et al. (2021), também conduziram um estudo sobre o uso de NAS para encontrar uma arquitetura baseada em *Transformer* para tarefas de visão computacional. Em sua abordagem, foi proposto uma técnica de NAS de única oportunidade (*one-shot*) denominada AutoFormer. A proposta envolve o treinamento de *Transformers* de SuperNets (conjunto de ANNs mescladas na última camada), onde sem ajustes adicionais ou re-treinamento é possível gerar milhares de *Transformers* de alta qualidade herdando os pesos. Essa abordagem alcançou o estado da arte na base de dados ImageNet em comparação com *Transformers* tradicionais aplicados à visão computacional.

Kim et al. (2022b) conduziram seu estudo utilizando Modelos de Disparos Neuronais (*Spiking Neural Networks* - SNN) em vez de ANNs devido ao seu paradigma promissor para inteligência de baixo consumo. O estudo destaca que, apesar das SNNs apresentarem vantagens impressionantes em eficiência energética, ainda ficam atrás das ANNs em termos de precisão. Assim, o estudo concentra-se em utilizar NAS para buscar as melhores arquiteturas de SNN capazes de representar diversos padrões de ativação de disparos em diferentes amostras de dados sem a necessidade de treinamento. Além disso, para aprimorar ainda mais as informações temporais entre os disparos, o processo de otimização busca conexões para frente e para trás entre as camadas, demonstrando que as arquiteturas obtiveram melhor desempenho com conexões para trás. Os experimentos realizados em três problemas de reconhecimento de imagem mostraram que a abordagem encontrada alcança o estado da arte.

A revisão bibliográfica apresentada destaca os principais trabalhos no campo da automatização na busca das arquiteturas ideais de ANNs. No entanto, atualmente, a área de automatização está em rápido desenvolvimento e várias novas técnicas estão sendo propostas, tornando difícil estar a par de todas elas. Assim, o conteúdo apresentado aqui cumpre seu objetivo de apresentar diferentes abordagens e os benefícios de cada uma,

permitindo o entendimento e a contribuição para a área de pesquisa.

2.2 FUNÇÕES DE ATIVAÇÃO ADAPTATIVAS

Em outra vertente de trabalhos relacionados a ANNs, diferente do que já foi apresentado anteriormente em automatização, surge a importância de estudar as funções de ativação e seus impactos no modelo/solução final.

No princípio, os trabalhos relacionados a ANNs comumente utilizavam as funções de ativação de Etapa Binária, Linear, Sigmoide e Tangente Hiperbólica (HAYKIN, 1998). Essas funções, separadas ou combinadas através de diferentes camadas, eram e ainda são capazes de resolver diversos problemas de aprendizado de máquina com qualidade satisfatória. Além disso, a escolha da função de ativação adequada faz uma diferença significativa no processo de criação e aprendizado de uma ANN (BISHOP et al., 1995), levando até a estudos para determinar qual função de ativação ou conjunto delas se adapta melhor ao problema (KARLIK; OLGAC, 2011; RAMACHANDRAN et al., 2017).

Uma ANN com camada de saída linear e pelo menos uma camada oculta, usando qualquer função de ativação limitada (por exemplo, Sigmoide e Tangente Hiperbólica), pode aproximar funções contínuas em subconjuntos compactos de $\mathbb{R}^n$, desde que a rede tenha neurônios suficientes em suas camadas ocultas para a tarefa (CYBENKO, 1989; CSÁJI et al., 2001; GOODFELLOW et al., 2016). Dessa forma, com essas características, uma ANN pode ser considerada um aproximador universal.

Entretanto, muitas ANN's podem demandar um grande número de camadas e neurônios para realizar a aproximação da função ou do problema, o que leva a modelos cada vez maiores e computacionalmente mais lentos, tanto em sua etapa de treinamento quanto na etapa de predição (CANZIANI et al., 2016). Além disso, esses modelos têm impactos no meio ambiente durante o treinamento e a predição (LI et al., 2016; STRUBELL et al., 2019; SCHWARTZ et al., 2019), e também são inviáveis para computação em borda simples (WANG et al., 2020). Dessa maneira, em uma direção oposta ao aumento do número de neurônios e camadas, surgiram trabalhos relacionados ao poder de aprendizado dos neurônios.

Piazza et al. (1992) apresentaram em seu trabalho a utilização de uma função de ativação adaptativa baseada em polinômios. No trabalho, é mostrado que a função de ativação pode ser representada como uma expansão truncada e restrita de uma Série de Volterra (VOLTERRA, 1887; VOLTERRA, 1959), permitindo assim a capacidade de capturar memórias nos dados. Os resultados apresentados mostraram que a proposta obteve excelentes resultados com uma quantidade significativamente menor de parâmetros do que uma ANN tradicional.

Chen e Chang (1996) propuseram em seu trabalho, com o intuito de fornecer mais

flexibilidade e capacidade não linear para as ANN's, uma função de ativação tangente hiperbólica adaptativa/generalizada. Através do método GD, desenvolveram um algoritmo capaz de ajustar automaticamente a função de ativação tangente hiperbólica para um formato que minimizava a função objetivo do problema. Além disso, a adaptatividade da função de ativação evita a saturação dos neurônios não-lineares e, consequentemente, reduz a necessidade de um processo de normalização. Os resultados apresentados na simulação mostraram que as propriedades de convergência são superiores e a função adaptativa traz melhores resultados do que uma ANN com função tradicional, mesmo com menos neurônios.

Depois disso, vários trabalhos surgiram explorando a função de ativação adaptativa utilizando Splines (FERGUSON, 1964; AHLBERG et al., 2016). Campolucci et al. (1996) propôs uma nova arquitetura de ANN, denominada Rede Neural Sigmoidal Generalizada (*Generalized Sigmoidal Neural Network* - GSNN), que possui função de ativação adaptativa com base na spline Catmull–Rom (CATMULL; ROM, 1974). Os seus resultados, em sistemas dinâmicos, demonstraram uma melhor performance na etapa de aprendizado e na sua generalização. Na sequência, foi desenvolvido pelo mesmo grupo de pesquisa as evoluções e os benefícios de se utilizar função de ativação adaptativa baseada em Splines (VECCI et al., 1997; VECCI et al., 1998; GUARNIERI et al., 1999), além de trabalhos que surgiram baseados nos mesmos (SOLAZZI; UNCINI, 2000a; SOLAZZI; UNCINI, 2000b).

Tezel e Özbay (2007) apresentaram um estudo aplicado de funções de ativação adaptativas no processo de classificação de arritmias eletrocardiográficas. No seu estudo de comparação de desempenho, foram utilizadas uma Sigmoide Básica, uma Sigmoide Adaptativa e uma Sigmoide Adaptativa agregada com uma função Seno Adaptativa. Os resultados mostraram que as arquiteturas que utilizaram função de ativação adaptativa foram mais adequadas para o problema de classificação e o tempo de treinamento foi muito mais rápido do que a arquitetura com função básica.

Com o avanço do Aprendizado Profundo (*Deep Learning*), novas funções de ativação foram desenvolvidas com o intuito de melhorar o treinamento dos modelos. As funções ReLU (HAHNLOSER et al., 2000; HAHNLOSER; SEUNG, 2001) e *Leaky* ReLU (MAAS et al., 2013) permitiam um melhor desempenho no treinamento do que outras funções amplamente usadas, como por exemplo a Sigmoide (GLOROT et al., 2011). He et al. (2015) apresentaram então uma nova função de ativação adaptativa, inspirada diretamente na ReLU e na *Leaky* ReLU, denominada PReLU. Os resultados apresentados pelos autores superaram o desempenho em nível humano em um desafio de reconhecimento visual.

Agostinelli et al. (2014) apresentaram uma proposta de função de ativação linear por partes que é aprendida independentemente para cada neurônio usando GD. No estudo, os autores comentaram que a função é capaz de representar funções convexas e não convexas dos dados de entrada. A função possui um hiperparâmetro S definido

previamente que determina a quantidade de dobradiças que a função adaptativa terá, permitindo assim que ela tenha maior capacidade de ajuste conforme o valor de S aumenta. Apesar disso, um valor grande de S pode aumentar consideravelmente a complexidade da função adaptativa. A conclusão do trabalho, devido ao observado nos experimentos, apresentou que funções adaptativas podem levar a um desempenho significativo nas redes sem aumentar consideravelmente o número de parâmetros.

Logo em sequência, Godfrey e Gashler (2015) apresentaram uma nova função de ativação adaptativa chamada *Soft Exponential*, uma função que pode assumir a forma de funções logarítmicas, lineares e exponenciais de acordo com seu parâmetro adaptativo. Os autores informaram que é necessária uma validação empírica das propriedades da função de ativação que, até o momento, não existe na literatura.

Qian et al. (2018) investigaram três estratégias para o aprendizado da função de ativação ReLU e suas variantes (*Leaky ReLU*, ELU, PReLU e PELU) para arquiteturas de CNN. Duas das estratégias realizavam combinações lineares e não lineares das funções de ativação básicas e, em seguida, a última estratégia realizava a combinação das funções de ativação básicas de maneira hierárquica. Os experimentos descritos pelos autores apresentavam uma melhor performance em relação à função de ativação básica ReLU e suas variantes.

Godfrey (2018) desenvolveu em sua tese um estudo comparativo entre três funções de ativação adaptativas: uma função de transferência adaptativa que aprende operações de lógica difusa por gradiente descendente, a função *Soft Exponential* e a função Unidade Linear Flexível (*Bendable Linear Unit* - BLU). Em seus resultados, os modelos adaptativos possuíram uma acurácia maior, convergiram mais rápido do que os não adaptativos e conseguiram aproximar a função identidade. Além disso, ele desenvolveu um estudo voltado para a utilização de ANN sem realimentação como alternativa para trabalhar em problemas de séries temporais.

Kunc e Kléma (2019) apresentaram um estudo direcionado à inferência de expressão gênica, onde uma função hiperbólica foi substituída por uma função Sigmoide adaptativa. Além disso, no trabalho, eles propuseram uma nova família de funções de ativação adaptativa que permite o escalonamento e a tradução de qualquer outra função de ativação.

Jagtap et al. (2020) apresentaram em seu trabalho o uso das funções de ativação adaptativas para regressão em redes neurais profundas e baseadas na física com o intuito de aproximar funções suaves e descontínuas, assim como soluções de equações diferenciais parciais lineares e não lineares. Diferentes experimentos foram realizados neste contexto, e foi observado que a introdução de um parâmetro escalável nas funções de ativação aumenta o poder de convergência da arquitetura e também a sua métrica de desempenho.

Panda e Panda (2020) propuseram uma variação do algoritmo de Retropropagação (*Backpropagation*) para encontrar, durante a fase de treino da arquitetura, os parâmetros

de uma função de ativação adaptativa. O algoritmo de Retropropagação proposto foi capaz de proporcionar uma convergência mais rápida e levou a erros menores na predição em comparação com a versão tradicional do algoritmo, chegando até a superar os resultados de outros trabalhos.

ZahediNasab e Mohseni (2020) apresentaram um estudo focado em funções de ativação adaptativas em arquiteturas CNN para imagens geradas por tomografia computadorizada do crânio e para a base de dados de classificação MNIST (LECUN; CORTES, 2010). Este foi o primeiro e único trabalho encontrado com estudos relacionados à utilização de Neuroevolução, utilizando um GA, para realizar a combinação de funções básicas de forma não linear. Todos os parâmetros gerados na combinação das funções básicas são aprendidos durante o processo de treinamento da arquitetura. O GA ficou responsável também por encontrar o melhor otimizador e a taxa de aprendizado durante o treino, mostrando que o otimizador depende diretamente da função de ativação adaptativa gerada. Os resultados obtidos implicaram que as funções de ativação adaptativas melhoram a capacidade das CNNs sem a necessidade de aumentar a profundidade da rede ou seus parâmetros.

Jagtap et al. (2022) propuseram, em seu trabalho, um novo tipo de ANN, denominado "Rede Neural Kronecker" (*Kronecker Neural Network* - KNN), que forma uma estrutura geral para ANN com funções de ativação adaptativas. Como o próprio nome indica, o KNN utilizou o produto de Kronecker, que é um tipo de produto de matrizes, na construção das matrizes de peso. Isso torna a rede ampla e, ao mesmo tempo, mantém o número de parâmetros baixo. Uma função de ativação adaptativa geral do trabalho segue um somatório de funções, onde normalmente a primeira função é uma função tradicional como ReLU, Tanh, Swish e Softplus, e as demais funções são funções harmônicas senoidais. Nos experimentos, comparando com ANN e versões adaptativas desenvolvidas no trabalho, foi possível observar uma melhora substancial na velocidade de treinamento, bem como na precisão preditiva do modelo final encontrado.

2.3 CONTRIBUIÇÃO DO TRABALHO

Os estudos apresentados anteriormente, relacionados à automatização de modelos e às funções de ativação adaptativas, mostram seu impacto positivo nos resultados obtidos. Até o momento presente, desconhecem-se estudos relacionados à combinação de ambas as abordagens, NAS e função de ativação adaptativa, para encontrar arquiteturas com boa qualidade preditiva e, ao mesmo tempo, com baixa complexidade ou quantidade de parâmetros.

Com essa inspiração, esta tese busca explorar a automatização da construção de ANN's com funções de ativação adaptativas, com o intuito de encontrar ANN's com neurônios mais complexos e, consequentemente, uma arquitetura menor e mais simples,

sem perder o seu poder preditivo. Arquiteturas menores tendem a ocupar menos espaço em memória, permitindo a sua utilização em diferentes dispositivos de borda, além de possuírem um tempo de inferência inferior, possibilitando a aplicação em problemas que exigem uma rápida resposta do modelo. Além disso, no contexto de AI verde, esses modelos acabam reduzindo a pegada de carbono, devido à busca por modelos menos complexos.

É importante destacar que existem diversos novos modelos que possuem uma complexidade enorme, mas que entregam inovações fenomenais para a humanidade, como o ChatGPT, que utiliza o "Transformador Gerador Pré-Treinado 3" (*Generative Pretrained Transformer 3* - GPT-3) (OpenAI, 2020). Esse modelo possui cerca de 175 milhões de parâmetros ajustáveis e, por se tratar de um modelo de linguagem natural de grande porte, pode ter levado dias, semanas ou até mesmo meses para ser treinado.

Por outro lado, existem trabalhos como o de Willsey et al. (2022), que observou que uma ANN foi responsável por acelerar em 45% a velocidade dos dedos robóticos em comparação com os métodos tradicionais não baseados em redes neurais. Isso acaba contradizendo a crença gerada pela era do aprendizado profundo de que são necessárias ANN complexas, como o GPT-3, para alcançar um determinado grau de melhora de desempenho.

Desejamos que, futuramente, este trabalho seja capaz de inspirar e/ou ajudar na busca por modelos que sejam, assim como o GPT-3, uma inovação para a humanidade, sem perder o foco na necessidade de sermos cada vez mais conscientes com os impactos ao nosso meio ambiente.

3 OTIMIZAÇÃO

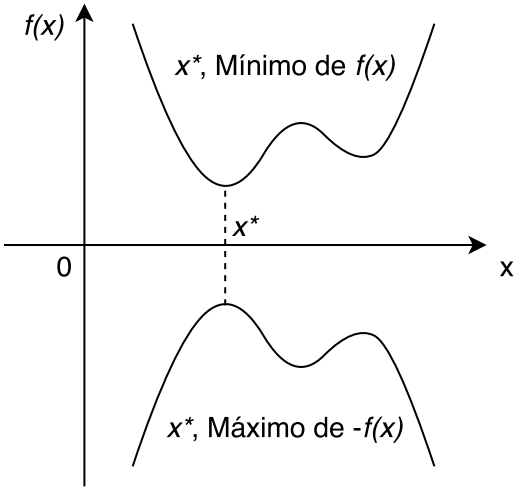
3.1 INTRODUÇÃO

A otimização é uma área da matemática que busca encontrar os valores ideais de determinadas grandezas de interesse. Diversas disciplinas aplicam a otimização para resolver seus problemas de maneira eficiente (VATIN et al., 2017; PHAM et al., 2019; NASERI; KOFFAS, 2020). No contexto deste trabalho, a otimização é utilizada como uma etapa do processo de Aprendizado de Máquina com o intuito de encontrar os melhores modelos sob determinada(s) métrica(s).

Em termos gerais, estabelecemos o processo de otimização como a busca pelos melhores valores de uma ou mais funções, podendo ser esses valores mínimos ou máximos, de modo que as variáveis de busca atendam a um conjunto de restrições de igualdade ou desigualdade (ARORA, 1989; ZöRNIG, 2014; FONSECA, 2017). Os conjuntos de variáveis que satisfaçam esses critérios são conhecidos como soluções ótimas do problema de otimização.

De acordo com o princípio da dualidade (GALE et al., 1948), podemos converter o problema de minimização de uma função $f(\mathbf{x})$ em um problema de maximização de outra função $k(\mathbf{x})$, onde definimos $k(\mathbf{x}) = -f(\mathbf{x})$. Deste modo, realizar a minimização de uma função $f(\mathbf{x})$ é o mesmo que realizar a maximização da função $-f(\mathbf{x})$. Denominamos essas funções como "funções objetivo" e elas quantificam o que precisamos minimizar ou maximizar no problema abordado. A Figura 4, adaptada de Rao e Rao (2009), exemplifica uma situação em que o ponto ótimo $\mathbf{x}^*$ obtém o melhor valor para a função $f(\mathbf{x})$ e, ao mesmo tempo, o melhor valor para a função $-f(\mathbf{x})$.

Figura 4 – A minimização $f(\mathbf{x})$ e a maximização de $-f(\mathbf{x})$.



Fonte: Adaptada de Rao e Rao (2009).

3.2 UM PROBLEMA DE OTIMIZAÇÃO

De modo geral, podemos elaborar formalmente um Problema de Otimização (PO) conforme apresentado na Equação 3.1.

$$\begin{aligned} &\min_{\mathbf{x}} / \max \quad f_i(\mathbf{x}) \qquad i = 1, 2, \ldots, n_f \\ &\text{Sujeito a} \\ &\qquad g_j(\mathbf{x}) \leq 0, \qquad j = 1, 2, \ldots, n_g \\ &\qquad h_k(\mathbf{x}) = 0, \qquad k = 1, 2, \ldots, n_h \\ &\qquad x_l^L \leq x_l \leq x_l^U, \qquad l = 1, 2 \ldots, n \end{aligned} \tag{3.1}$$

onde o vetor $\mathbf{x}$ é denominado como incógnitas do problema ou variáveis de projeto, $f_i : \mathbb{R}^n \to \mathbb{R}$ é a i-ésima função objetivo e $g_j : \mathbb{R}^n \to \mathbb{R}$ e $h_k : \mathbb{R}^n \to \mathbb{R}$ são as j-ésima e k-ésima restrições de desigualdade e igualdade, respectivamente. As funções f_i, g_j e h_k podem ser lineares ou não lineares em relação ao vetor de variáveis de projeto. O limite da l-ésima variável é determinado por x_l^L e x_l^U, representando respectivamente o limite inferior e o limite superior da variável x_l. Nas expressões, o valor de n, n_f, n_g e n_h representam o número de variáveis de projeto, número de funções objetivo, número de restrições de desigualdade e igualdade, respectivamente (FONSECA, 2017).

Em alguns cenários, faz-se necessário realizar a transformação das restrições de igualdade e desigualdade, convertendo uma forma na outra. Uma das maneiras de efetuar a transformação de uma restrição de igualdade é através da criação de duas restrições de desigualdade, conforme apresentado na Equação 3.2 (BAZARAA et al., 2004).

$$h_k(\mathbf{x}) = 0 \to \begin{cases} h'_k(\mathbf{x}) \leq 0 \\ h''_k(\mathbf{x}) \geq 0 \end{cases}, \; k = 1, 2, \ldots, n_h \tag{3.2}$$

Apesar disso, podemos realizar outra transformação na restrição de igualdade, introduzindo uma tolerância ε suficientemente pequena que não seja capaz de afetar significativamente a qualidade da solução. Essa transformação em restrição de desigualdade é feita conforme apresentado na Equação 3.3.

$$h_k(\mathbf{x}) = 0 \to |h_k(\mathbf{x})| - \varepsilon \leq 0, \; k = 1, 2, \ldots, n_h \tag{3.3}$$

Já a transformação de uma restrição de desigualdade em uma de igualdade é feita através da criação de novas variáveis de projeto, conforme apresentado na Equação 3.4 ou 3.5 (BAZARAA et al., 2004).

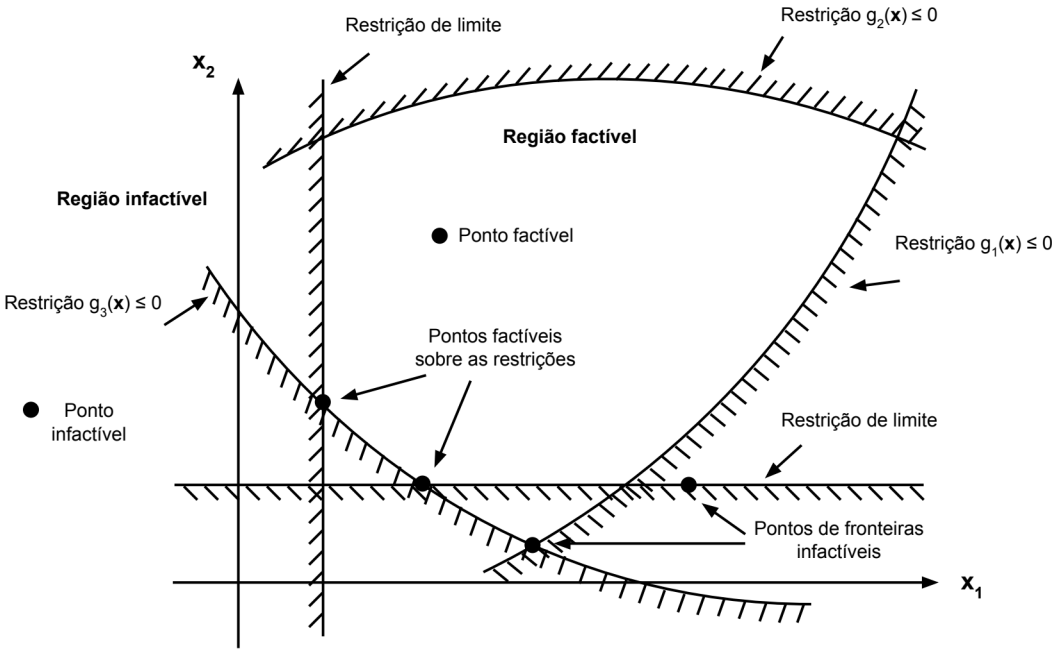
$$g_j(\mathbf{x}) \leq 0 \to g_j(\mathbf{x}) + x_{n+j} = 0, \; j = 1, 2, \ldots, n_g \tag{3.4}$$

$$g_j(\mathbf{x}) \geq 0 \rightarrow g_j(\mathbf{x}) - x_{n+j} = 0,\ j = 1, 2, \ldots, n_g \tag{3.5}$$

onde x_{n+j} deve ser maior ou igual a zero e são conhecidas como variáveis de folga (na transformação de desigualdades do tipo $\leq$) ou variáveis de excesso (na transformação de desigualdades do tipo $\geq$).

Os POs que possuem restrições, seja de igualdade ou desigualdade, apresentam, na maioria das vezes, uma complexidade maior para a obtenção da solução ótima, comparado com os POs sem restrições. Além disso, esses POs costumam necessitar de estratégias particulares do algoritmo de otimização, para que as condições das restrições sejam satisfeitas durante a busca. Denominamos como solução factível do problema o vetor $\mathbf{x}$, que atende todas as restrições do problema, e chamamos de região factível o conjunto composto por todas essas soluções factíveis. A Figura 5, extraída e adaptada de Rao e Rao (2009), apresenta um espaço de busca com duas regiões delimitadas pelas restrições de desigualdade, sendo uma região composta pelas soluções factíveis e a outra pelas soluções infactíveis. A função principal das restrições, tanto de igualdade quanto de desigualdade, é determinar durante o processo de otimização quais são os limites de factibilidade do projeto abordado, conseguindo separar as regiões factíveis das infactíveis (RAO; RAO, 2009).

Figura 5 – Regiões factíveis e infactíveis do espaço de busca com restrições



Fonte: Extraída e adaptada de Rao e Rao (2009).

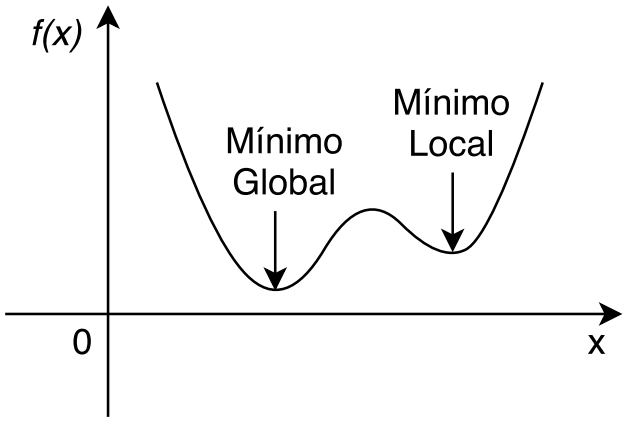
Denominamos um PO com apenas uma função objetivo ($n_f = 1$) como mono-objetivo. Quando o PO possui mais de uma função objetivo ($n_f > 1$), denominamos

de multiobjetivo. Em alguns casos, podemos ter um PO com nenhuma função objetivo ($n_f = 0$) e denominamos simplesmente como otimização sem objetivo.

Quando estamos trabalhando com um problema de otimização sem um objetivo explícito, o foco principal pode ser encontrar soluções que satisfaçam um conjunto específico de restrições. Em vez de buscar otimizar uma métrica particular, o objetivo é identificar as regiões no espaço de solução que atendem a todas as restrições impostas. Esses problemas são conhecidos como problemas de "viabilidade" ou "satisfação de restrições".

Ao trabalharmos com um PO mono-objetivo, seja de minimização ou maximização, definimos como solução ótima o conjunto de variáveis de projeto que proporciona o melhor valor, entre todos os possíveis, para a função objetivo. No caso de minimização, essa solução representa o mínimo global da função objetivo; para maximização, o máximo global. Se o problema abordado tiver mais de uma solução ótima, dizemos que essas são soluções ótimas alternativas e o problema apresenta mais de uma região que leva ao valor mínimo ou máximo global. A comprovação de que um determinado valor é global é uma tarefa difícil devido à complexidade de verificar, no espaço de busca, se existem outras regiões, locais ou globais, que assumem o mesmo valor. Na prática, observa-se que a minimização/maximização global determinística frequentemente é extremamente custosa, a ponto de se tornar inviável (HARTKE, 2011). Normalmente, examinamos uma sub-região do espaço de busca, nas proximidades do ponto analisado, e determinamos se o valor da função objetivo é o menor ou maior entre todos. A Figura 6 apresenta um exemplo de mínimo local e global em um PO mono-objetivo.

Figura 6 – Pontos de mínimo global e local.



Fonte: Elaborada pelo autor (2023).

Um PO mono-objetivo diferencia-se principalmente de um PO multiobjetivo pela particularidade da definição de otimalidade de cada um (DEB, 2001). No cenário de um PO multiobjetivo, a caracterização da otimalidade de uma solução não é elementar, uma

vez que as funções objetivo são, na maioria das vezes, conflitantes entre si. Assim, torna-se necessário a definição de um critério, diferente do utilizado em um PO mono-objetivo, capaz de realizar a comparação de possíveis soluções de um PO multiobjetivo. A dominância de Pareto é o critério de comparação de possíveis soluções de um PO multiobjetivo mais difundido na literatura (DEB, 2001).

Considerando um PO multiobjetivo de minimização, composto por n_f funções objetivo, uma solução $\mathbf{x_1}$ domina uma solução $\mathbf{x_2}$, conforme a dominância de Pareto, se (DEB, 2001):

- $\mathbf{x_1}$ não é pior do que $\mathbf{x_2}$ em nenhuma das funções objetivo, ou seja, $f_i(\mathbf{x_1}) \leq f_i(\mathbf{x_2})$, onde $i = 1, 2, \ldots, n_f$;
- $\mathbf{x_1}$ é estritamente superior a $\mathbf{x_2}$ em pelo menos uma das funções objetivo, ou seja, $f_i(\mathbf{x_1}) < f_i(\mathbf{x_2})$, onde $i = 1, 2, \ldots, n_f$.

Ao utilizar essa definição, três possíveis situações podem surgir ao analisar as duas soluções $\mathbf{x_1}$ e $\mathbf{x_2}$ (DEB, 2001):

1. $\mathbf{x_1} \prec \mathbf{x_2}$, ou seja, $\mathbf{x_1}$ domina $\mathbf{x_2}$;
2. $\mathbf{x_2} \prec \mathbf{x_1}$, ou seja, $\mathbf{x_2}$ domina $\mathbf{x_1}$;
3. $\mathbf{x_1}$ e $\mathbf{x_2}$ são soluções não-dominadas entre si.

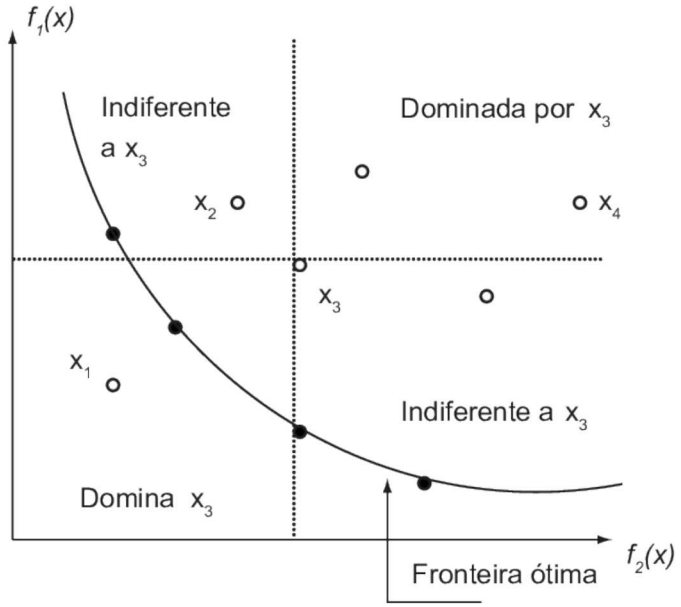
Chamamos de Soluções Pareto-Ótimas ou Conjunto Pareto-Ótimo, a todas as soluções factíveis e não-dominadas de um PO multiobjetivo. Ao utilizar as Soluções Pareto-Ótimas, conseguimos gerar, no espaço das funções objetivo do problema, a Frente de Pareto-Ótima (CHANKONG; HAIMES, 2008; COELLO et al., 2007; RUSSO, 2017; LIMA, 2019). A Figura 7, extraída de Iannoni e Morabito (2006), exemplifica os conceitos da dominância de Pareto, onde o conjunto de soluções eficientes determina a curva de *trade-off* chamada curva Pareto-Ótima, representada pela linha sobre os pontos em negrito.

De acordo com Deb (2001), existem duas metas em um processo de otimização de um PO multiobjetivo, sendo elas:

- **Convergência:** Busca-se encontrar um conjunto de soluções que estão na frente de Pareto-Ótima;
- **Diversidade:** Busca-se encontrar um conjunto de soluções que sejam suficientemente diversificadas para representar toda a gama da frente de Pareto-Ótima;

Deste modo, ao realizar a otimização do PO multiobjetivo, deseja-se encontrar um conjunto de possíveis soluções distintas e, ao mesmo tempo, o mais próximo da Frente

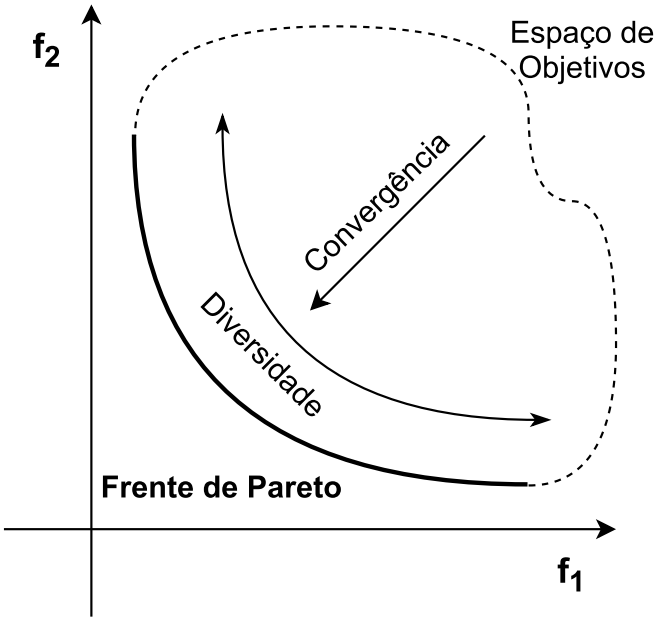
Figura 7 – Dominância de Pareto.



Fonte: Extraída de Iannoni e Morabito (2006).

de Pareto-Ótima. Dessa forma, em uma análise *a posteriori*, é possível que um tomador de decisão escolha a melhor solução para o seu problema (RUSSO, 2017). A Figura 8, adaptada de Russo (2017), ilustra as duas metas definidas por Deb (2001).

Figura 8 – Convergência e diversidade ilustradas em um PO com dois objetivos, f_1 e f_2.



Fonte: Adaptada de Russo (2017).

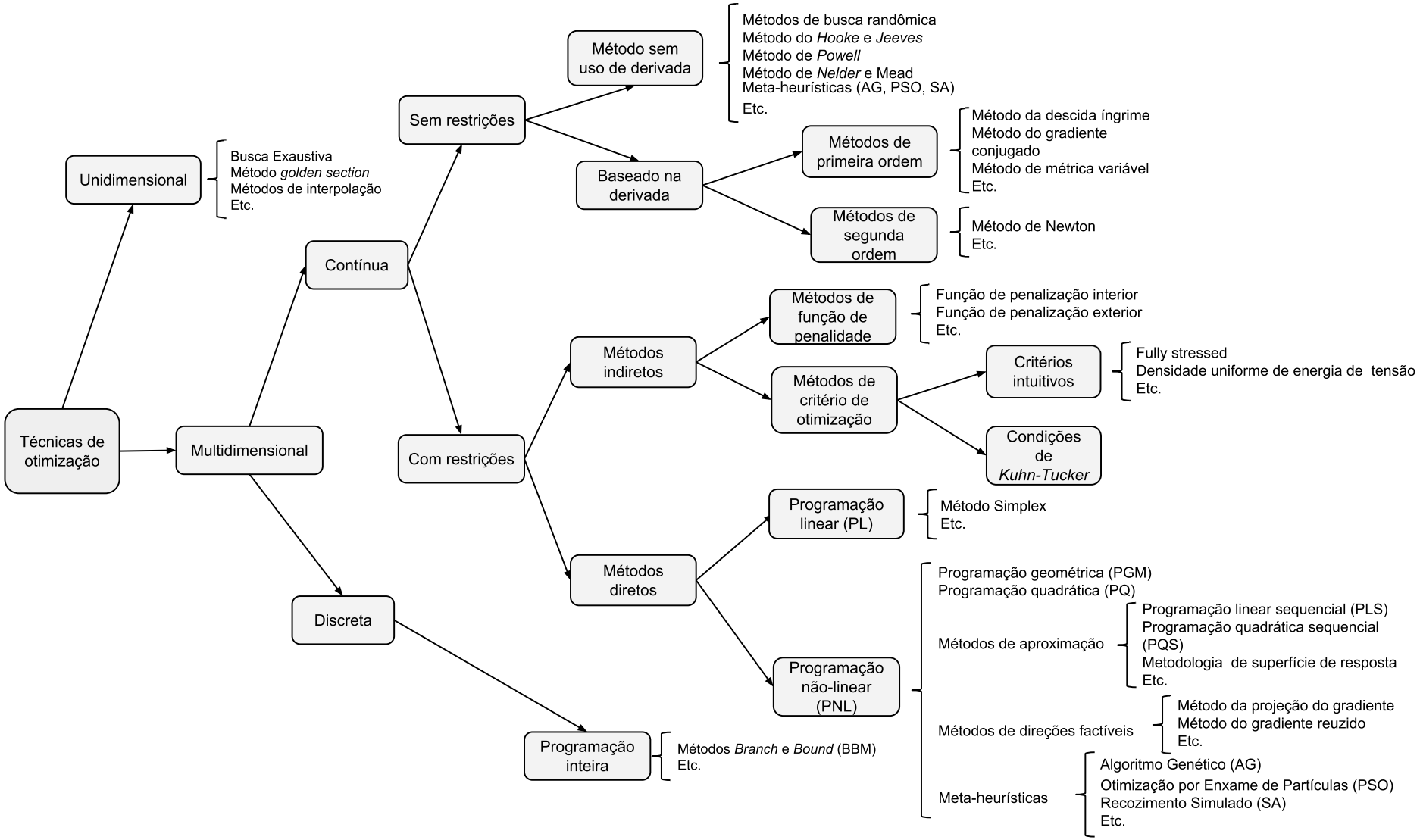
Quando construímos um PO, acabamos definindo quais serão as suas funções objetivo, as restrições que o problema possui e quais são as variáveis com as quais podemos trabalhar durante o processo de otimização. O conjunto dessas informações sobre o PO pode ser classificado. A Tabela 1, extraída e adaptada de Gandomi et al. (2013), contém algumas das classificações mais comuns existentes dentro de um PO. Além disso, na Figura 9, extraída e adaptada de Gandomi et al. (2013), é apresentado um conjunto de técnicas de otimização já consolidadas na literatura e suas classificações entre os diferentes grupos.

Tabela 1 – Classificação dos problemas de otimização.

Base da Classificação	Categoria
Quantidade de Variáveis	Monovariável Multivariável
Quantidade de Funções Objetivo	Mono-objetivo Multiobjetivo Nenhum
Presença de Restrições	Sem Restrições Com Restrições
Forma das Funções Objetivo e das Restrições	Linear Não Linear
Classificação das Variáveis	Discreta Contínua Mista
Natureza das Variáveis e dos Dados de Entrada	Determinística Probabilística

Fonte: Extraída e adaptada de Gandomi et al. (2013).

Figura 9 – Classificação das técnicas de otimização numérica.



Fonte: Extraída e adaptada de Gandomi et al. (2013).

3.3 META-HEURÍSTICAS

As meta-heurísticas são estruturas algorítmicas genéricas, frequentemente inspiradas na natureza, projetadas para resolver problemas complexos de otimização (BIANCHI et al., 2009). Esses algoritmos fazem poucas suposições sobre o problema de otimização que está sendo resolvido e, por esse motivo, podem ser utilizados em uma variedade de problemas (BLUM; ROLI, 2003). Além disso, esses algoritmos utilizam seleções aleatórias e informações obtidas durante sua execução para se guiarem no processo de busca, evitando paradas prematuras em soluções ótimas locais (GOLDBARG et al., 2017).

Os algoritmos descritos a seguir são utilizados, diretamente ou indiretamente, como base para o desenvolvimento deste trabalho.

3.3.1 Algoritmo Genético

O Algoritmo Genético (*Genetic Algorithm* - GA) é um algoritmo de otimização inspirado na Teoria da Evolução (DARWIN, 1859), que foi proposto e desenvolvido por John Holland (HOLLAND, 1975; HOLLAND, 1995). John Holland buscava desenvolver um algoritmo capaz de resolver problemas complexos de otimização por meio da evolução. Durante a concepção do algoritmo, John Holland buscou garantir dois pontos fundamentais: conseguir abstrair e explicar precisamente os processos de adaptação dos sistemas naturais e, além disso, conseguir projetar uma metodologia para a simulação de sistemas artificiais, mantendo todos os procedimentos fundamentais dos sistemas naturais (GOLDBERG, 1989).

Os GAs são definidos como um procedimento de busca inspirado na genética e também na seleção natural das espécies (KOZA, 1992). O algoritmo adota uma concepção de sobrevivência dos indivíduos através de testes de qualificação dentro de um sistema artificial que simula o problema que se deseja resolver. Os testes de qualificação, realizados utilizando todas as informações de aptidões dos indivíduos, têm como único objetivo identificar de forma quantitativa os melhores e os piores indivíduos dentro de uma determinada população. Desta forma, são desenvolvidos dentro do GA os operadores genéticos, todos inspirados na natureza, com o objetivo de formar um processo de busca evolutiva.

Durante o desenvolvimento da estratégia, Holland (1975) decidiu adotar em seu algoritmo uma semântica da biologia. Dessa maneira, ele listou três elementos fundamentais para o desenvolvimento de um GA, sendo eles: o cromossomo, o indivíduo e a população. Seu principal objetivo era tentar corresponder cada possível solução a um cromossomo que representasse um indivíduo. Portanto, uma população é um conjunto de possíveis soluções do problema de otimização.

A primeira representação de um cromossomo, considerada como sendo a versão clássica, foi utilizando a codificação binária. Essa codificação é composta por 0's e 1's,

representando artificialmente um "código genético" (LEMONGE, 1999). No entanto, trabalhos foram surgindo mostrando a qualidade da utilização de codificação Real em um GA (GOLDBERG et al., 1990; ESHELMAN; SCHAFFER, 1993).

No processo evolutivo do GA, cada um dos indivíduos é codificado por seus cromossomos e é submetido à avaliação com o objetivo de medir seu desempenho no problema de otimização. Em seguida, os indivíduos são qualificados por um critério baseado em seus desempenhos, e três operações básicas são efetuadas sobre os indivíduos, sendo elas: a seleção dos indivíduos, a recombinação entre os indivíduos e a mutação individual.

Inicialmente, aplicamos o operador de seleção para escolher os indivíduos "genitores" e, por meio deles, aplicamos o operador de recombinação para gerar novos indivíduos. Por fim, aplicamos um operador de mutação nos indivíduos "proles". Assim como na Teoria da Evolução, considerando que os indivíduos selecionados como "genitores" sejam aqueles que apresentam as melhores aptidões na população atual, esperamos que os indivíduos gerados no processo de recombinação também apresentem as melhores aptidões em sua geração. Dessa maneira, os indivíduos da população vão evoluindo a cada geração, gerando soluções cada vez melhores.

Em resumo, um GA pode ser especificado por quatro componentes essenciais (MITCHELL, 1998), sendo eles:

1. Populações de cromossomos (possíveis soluções);
2. Seleção dos cromossomos conforme a aptidão;
3. Recombinação de cromossomos, produzindo novos cromossomos;
4. Mutação aleatória nos novos cromossomos;

O Algoritmo 1 apresenta um pseudocódigo de um GA com elitismo, com o objetivo de reintroduzir os indivíduos mais bem avaliados de uma geração para a seguinte, evitando a perda de informação dos melhores indivíduos durante as gerações.

Os GAs se tornam muito atrativos pelo fato de serem capazes de encontrar as soluções globais do problema em seu processo evolutivo. Além disso, não possuem nenhum tipo de restrição em relação ao domínio da aplicação e são capazes de solucionar problemas de otimização que envolvem diferentes tipos de variáveis, como: contínuas, discretas e mistas. As funções objetivo do problema não precisam ser contínuas ou diferenciáveis para o algoritmo funcionar, e não é necessário informar uma solução inicial para iniciar a busca. Como a avaliação dos indivíduos de uma população não é dependente, podemos utilizar programação paralela para agilizar a busca da solução ótima.

Embora os GAs tenham suas vantagens, eles não foram criados com o objetivo de lidar com restrições, sejam elas de igualdade ou desigualdade. No entanto, existem

Algoritmo 1: Pseudocódigo de um Algoritmo Genético.

```
1  Seja N_POP o tamanho da população e N_G o número de gerações;
2  i ← 0;
3  Gera uma população inicial P_i com N_POP indivíduos;
4  Avalia a população P_i;
5  enquanto i < N_G faça
6      Seleciona N_POP indivíduos em P_i, formando a população G_i;
7      Aplica operador de recombinação (probabilidade p_c) em G_i;
8      Aplica operador de mutação(taxa p_m) em G_i;
9      Avalia a população G_i;
10     Seleciona as k melhores soluções de P_i formando K_i;
11     Combina K_i com G_i, formando uma população mista Q_i;
12     Ordena a população Q_i;
13     Seleciona os N_POP melhores indivíduos de Q_i e copia para P_{i+1};
14     i ← i + 1;
```

soluções que adaptam os GAs para otimizar esses desafios. Como o foco deste trabalho não aborda problemas com restrições, essas técnicas não serão exploradas.

As seções 3.3.1.1 até 3.3.1.5 descrevem com mais detalhes os componentes de um GA.

3.3.1.1 Codificação

A primeira etapa antes de se utilizar um GA é conseguir transformar as possíveis soluções do problema em indivíduos. Esses indivíduos então devem ser codificados e essa codificação vai depender diretamente das propriedades do problema de otimização. Existem vários *"alfabetos"* capazes de realizar a codificação em um GA, destacando-se principalmente a codificação binária e a real.

Quando estamos trabalhando com problemas que possuem variáveis discretas, o uso de uma codificação binária é mais comum e recomendada. Por outro lado, quando possuímos variáveis contínuas no problema, o uso de uma codificação real é mais comum e recomendada. Em alguns problemas em que existem variáveis discretas e contínuas, podemos utilizar a combinação dessas codificações. A Tabela 2 apresenta um exemplo da codificação binária, real e mista. Uma possível solução do problema é formada pela combinação das variáveis (x_1, x_2, x_3, x_4 e x_5) de uma codificação.

Quando trabalhamos com a codificação binária, precisamos realizar um processo de decodificação para a base 10, onde obtemos os valores inteiros ou reais das variáveis. Esse processo de decodificação é realizado por um mapeamento do valor binário para um valor inteiro e/ou real, não necessariamente representando a conversão tradicional entre a base 2 para a base 10. A Tabela 3 apresenta exemplos da decodificação entre a codificação binária e os valores inteiros e reais.

Tabela 2 – Exemplo da codificação binária, real e mista.

Codificação Binária				
x_1	x_2	x_3	x_4	x_5
110	1111	101	1010	110001

Codificação Real				
x_1	x_2	x_3	x_4	x_5
46.1	28.9	70.0	7.2	30.6

Codificação Mista				
x_1	x_2	x_3	x_4	x_5
$65,8$	$94,35$	010	1110	01011

Fonte: Elaborada pelo autor (2023).

Tabela 3 – Exemplos da decodificação do código binário.

Decodificação: Binário para Inteiro								
Binário	000	001	010	011	100	101	110	111
Inteiro	-5	-3	-1	1	3	5	7	9

Decodificação: Binário para Real								
Binário	000	001	010	011	100	101	110	111
Real	-2.6	-2.0	-1.4	-0.8	-0.2	0.4	1.0	1.6

Decodificação: Binário para Inteiro e Real								
Binário	000	001	010	011	100	101	110	111
Inteiro e Real	0.01	0.05	0.1	0.5	1	10	100	1000

Fonte: Elaborada pelo autor (2023).

A definição da decodificação e a quantidade de bits utilizados para cada uma das variáveis depende diretamente do problema abordado. Algumas variáveis do problema podem requerer um maior número de bits representativos.

Michalewicz (1992) apresentou um estudo detalhado referente à utilização da codificação binária e real, indicando que a utilização da codificação real é mais rápida, mais consistente nas execuções e possibilita uma maior precisão das variáveis, especialmente nos problemas de otimização que possuem um grande domínio. No estudo é apresentado que a codificação binária pode exigir uma representação muito longa, sendo em alguns casos impeditiva para a execução do algoritmo.

Por estar mais próximo do espaço real do problema, a utilização da codificação real torna mais simples o processo de absorção do conhecimento específico do mesmo nos

operadores de seleção, recombinação e mutação.

3.3.1.2 População inicial

Considerando que já temos uma codificação adequada, o primeiro procedimento dentro do GA é a formação de uma população inicial. Esta população inicial é composta por possíveis soluções do problema abordado.

A etapa de inicialização de uma população inicial corretamente é fundamental para o processo de busca do algoritmo e, em alguns casos, pode influenciar na solução final encontrada. Se o GA possui uma boa inicialização da população inicial, então a probabilidade de encontrar uma solução ótima para o problema aumenta (ZITZLER et al., 2000; BURKE et al., 2004).

Um dos processos mais comuns de inicialização de uma população é de modo aleatório, utilizando as funções pseudo-aleatórias. De acordo com o problema abordado, esse tipo de inicialização pode proporcionar vantagens ou desvantagens (BACK, 1996). A vantagem de realizar uma inicialização aleatória é proporcionar ao algoritmo a exploração do espaço de busca nas primeiras gerações. Por outro lado, como desvantagem, as soluções geradas podem ser inviáveis em problemas com restrições, dificultando no processo de busca encontrar a região factível.

Outra maneira de realizar a inicialização de uma população é através do conhecimento de um especialista do problema. O especialista pode informar algumas possíveis soluções previamente descobertas e até mesmo regiões com maior probabilidade de se obter a solução ótima. Em alguns casos, esta abordagem pode ser benéfica se a região conter a solução ótima do problema; caso contrário, pode induzir o algoritmo a encontrar uma solução local.

O tamanho da população é um hiperparâmetro do GA que deve ser informado antes do início do processo de busca. É importante que esse hiperparâmetro seja definido de maneira apropriada para cada problema (GOLDBERG et al., 1991; GOLDBERG et al., 2014).

3.3.1.3 Operador de seleção

Considerando que temos uma população, o GA possui uma etapa de seleção dos indivíduos para serem “genitores” em um processo de recombinação. A questão mais importante neste operador de seleção é como esses indivíduos devem ser selecionados.

Levando em consideração a Teoria da Evolução, os indivíduos mais aptos de uma população devem ser selecionados para sobreviver e gerar novos indivíduos. Isso se dá principalmente pelo fato de que a espécie se adapta melhor ao ambiente em que vive e evita a sua extinção.

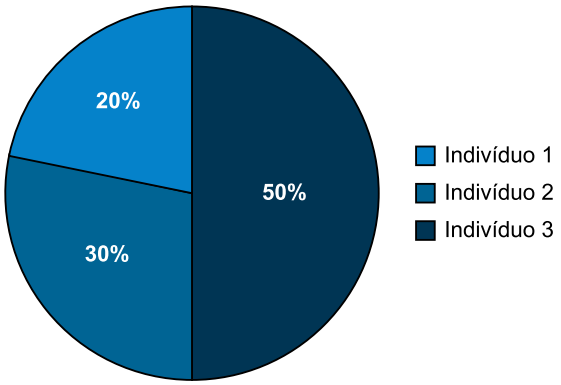
A seleção dos indivíduos de uma população pode ser feita de diversas formas, sendo os operadores a seguir os mais utilizados na literatura (MITCHELL, 1998):

- Seleção por Roleta:

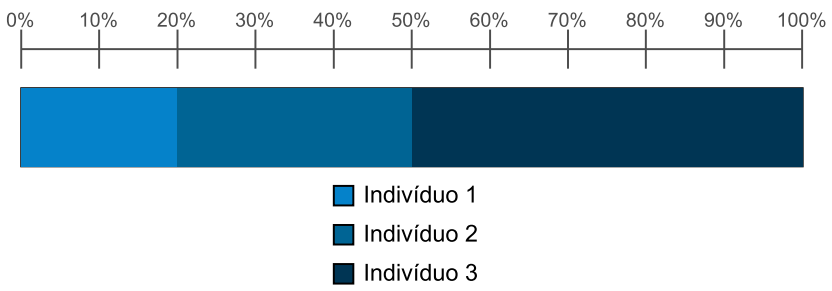
  Os indivíduos são representados por uma probabilidade de seleção proporcional à sua aptidão. Desta maneira, os indivíduos com um alto valor de aptidão vão obter uma probabilidade maior de serem selecionados, enquanto aqueles indivíduos com um baixo valor de aptidão vão obter uma probabilidade menor. Por exemplo, quando queremos maximizar uma função $f(x)$ e existem três indivíduos na população com os seguintes valores de $f(x)$: 60, 90 e 150, temos que suas probabilidades são 20%, 30% e 50%, respectivamente. A Figura 10 apresenta o exemplo descrito anteriormente em relação à distribuição de probabilidades dos indivíduos na "roleta", onde a Figura 10a apresenta ela como um gráfico de setores e a Figura 10b apresenta ela como um gráfico de barras segmentadas.

Figura 10 – Seleção por Roleta.

(a) Roleta em formato de gráfico de setores.



(b) Roleta em formato de gráfico de barras segmentadas.



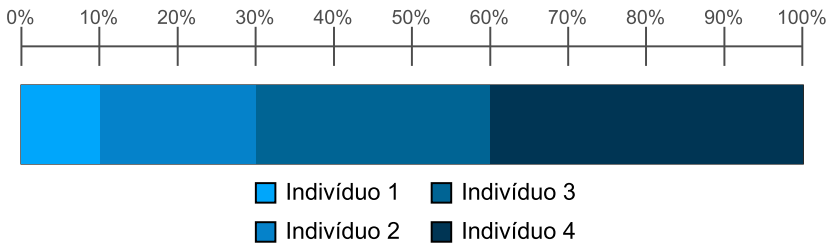
Fonte: Elaborada pelo autor (2023).

- Seleção por Classificação (*Ranking*):

  A primeira etapa deste operador é realizar a ordenação dos indivíduos do pior para o melhor, de acordo com a sua aptidão, e então é realizada uma atribuição de

classificação determinada por sua posição. Atribuímos para o pior indivíduo da população o valor de 1, ao indivíduo seguinte o valor de 2 e assim por diante. Desta forma, o melhor indivíduo da população terá um valor igual a N, onde N é o tamanho da população. Esses valores são então convertidos em probabilidades, dividindo cada um dos valores de classificação pela soma dos termos da progressão aritmética $S_t = ((1 + N) * N/2)$, ou seja, a probabilidade do melhor indivíduo da população será igual a N/S_t e do pior indivíduo será igual a $1/S_t$. Por exemplo, quando queremos maximizar uma função $f(x)$ e existem quatro indivíduos na população com os seguintes valores de 10, 45, 86 e 120, temos as classificações 1, 2, 3 e 4, e suas probabilidades são 10%, 20%, 30% e 40%, respectivamente. A Figura 11 apresenta o exemplo descrito anteriormente em relação à distribuição de probabilidades dos indivíduos por "classificação".

Figura 11 – Seleção por Classificação.



Fonte: Elaborada pelo autor (2023).

- Seleção por Torneio:

  Uma subpopulação é formada através da escolha aleatória de n indivíduos da população, com $n \geq 2$. Seleciona-se o indivíduo da subpopulação que possui o maior valor de adequação. Neste operador de seleção, o usuário deve informar o hiperparâmetro n.

O processo de seleção tem como finalidade encontrar o conjunto de indivíduos que vai se recombinar para produzir os novos indivíduos. Esses novos indivíduos vão compor a população da geração seguinte. Se a geração seguinte for composta apenas por esses novos indivíduos, podemos perder os indivíduos da geração atual que apresentaram bons desempenhos.

Para evitar esse problema, podemos escolher uma parcela dos melhores indivíduos da geração atual para serem mantidos na geração seguinte. Esse processo é conhecido como Elitismo (DAVIS, 1996). Dessa maneira, durante todo o processo de busca, teremos na população atual a melhor solução conhecida até o momento.

3.3.1.4 Operador de recombinação

Entre todos os operadores de um GA, o operador de recombinação é considerado o predominante. Através do processo de recombinação, conseguimos gerar novos indivíduos combinando os atributos dos indivíduos "genitores". Partes da codificação de um indivíduo "genitor" são trocadas ou combinadas com partes equivalentes do outro "genitor" e, como resultado, conseguimos criar um ou mais indivíduos "proles". Espera-se que, ao realizar esta recombinação, os indivíduos "proles" possam herdar os melhores atributos dos indivíduos "genitores" e aumentar a probabilidade de obter um desempenho superior. Os operadores de recombinação mais utilizados são (GOLDBERG, 1989):

- Recombinação de um Ponto:

  Operadores de recombinação mais clássicos para codificação binária: Seleciona-se aleatoriamente um ponto de corte dentro do intervalo dos cromossomos escolhidos como "genitores" e realiza a recombinação através da troca dos valores entre eles a partir do ponto de corte. Este processo gera dois novos indivíduos "proles". O exemplo a seguir apresenta esta operação:

  $$\begin{array}{llll} Genitor_1 & 1111|1111 & Prole_1 & 1111|0000 \\ Genitor_2 & 0000|0000 & Prole_2 & 0000|1111 \end{array}$$

- Recombinação de dois Pontos:

  Recombinação análoga à recombinação de um ponto. Seleciona-se aleatoriamente dois pontos de corte dentro do intervalo dos cromossomos escolhidos como "genitores" e realiza a recombinação através da troca dos valores dentro do intervalo formado pelos dois pontos de corte. Este processo gera dois novos indivíduos "proles". O exemplo a seguir apresenta esta operação:

  $$\begin{array}{llll} Genitor_1 & 100|001|0000 & Prole_1 & 100|110|0000 \\ Genitor_2 & 110|110|0110 & Prole_2 & 110|001|0110 \end{array}$$

- Recombinação de n Pontos:

  Recombinação que generaliza a recombinação de um e dois pontos. Seleciona-se aleatoriamente n pontos de corte dentro do intervalo dos cromossomos escolhidos como "genitores" e realiza a recombinação através da troca dos valores dentro dos intervalos formados pelos n pontos de corte. Para este operador de recombinação, o usuário deve informar o hiperparâmetro n.

- Recombinação Uniforme:

  Neste caso, são gerados 0's e 1's aleatórios uniformemente, do tamanho do cromossomo, que indicarão a posição em que se deve trocar o valor entre os indivíduos "genitores". A combinação destes valores é chamada de "máscara" e haverá troca entre os indivíduos "genitores" sempre que o valor da posição for igual a 1 e, caso contrário, nada acontecerá. Este operador possui um hiperparâmetro que representa a taxa de recombinação p_r, definida pelo usuário. Quanto maior este valor p_r, maior será a troca entre os indivíduos "genitores". O exemplo a seguir apresenta esta operação:

$$\begin{array}{cccc} Genitor_1 & 1011011010 & Prole_1 & 1001011100 \\ & Máscara & 0011100110 & \\ Genitor_2 & 1001011101 & Prole_2 & 1011011011 \end{array}$$

- Recombinação por Simulação Binária:

  A Recombinação por Simulação Binária (*Simulated Binary Crossover* - SBX) (DEB; AGRAWAL, 1995) é um operador de recombinação desenvolvido para a codificação real. Este operador foi desenvolvido com base nas características apresentadas pela operação de recombinação binária de um ponto. Suas principais características são:

  - **Média:** Após a realização da operação de recombinação, a média dos indivíduos "proles" é similar à dos indivíduos "genitores".
  - **Fator de Propagação β :** As operações de recombinação tendem, na sua maioria, a gerar um fator de propagação de $\beta \approx 1$. Isso significa que os indivíduos "proles" tendem a ser similares aos indivíduos "genitores".

  Para que a propriedade da média seja satisfeita, os valores dos indivíduos "proles" são obtidos conforme as Equações 3.6 e 3.7.

$$ch_1 = \bar{q} - \frac{1}{2}\beta(p_2 - p_1) \tag{3.6}$$

$$ch_2 = \bar{q} + \frac{1}{2}\beta(p_2 - p_1) \tag{3.7}$$

  onde $\bar{q} = \frac{1}{2}(p_1 + p_2)$, $p_2 > p_1$, p_1 e p_2 são os indivíduos "genitores", e ch_1 e ch_2 são os indivíduos "proles". Podemos observar que as principais características do operador são satisfeitas através destas fórmulas.

  Para que o operador SBX tenha um comportamento similar ao da recombinação binária de um ponto, precisamos garantir que a distribuição probabilística de β seja

semelhante em ambos os operadores. Para isso, Deb e Agrawal (1995) apresentam a seguinte função de distribuição probabilística através da Equação 3.8.

$$f(\beta) = \begin{cases} 0.5(\eta+1)\beta^{\eta}, & \text{se } \beta \leq 1, \\ 0.5(\eta+1)\frac{1}{\beta^{\eta+2}} & \text{caso contrário} \end{cases} \tag{3.8}$$

onde o valor do hiperparâmetro η controla o quão similares os indivíduos "proles" serão em relação aos indivíduos "genitores". A atribuição de pequenos valores para η faz com que os indivíduos sejam mais parecidos, enquanto valores maiores geram indivíduos menos parecidos.

Podemos definir o Fator de Propagação $\bar{\beta}$ através de um valor aleatório que siga a função de distribuição probabilística proposta por Deb e Agrawal (1995). Esse valor pode ser calculado seguindo as seguintes etapas:

- Obtém um valor aleatório r no intervalo de [0,1];
- Busca qual valor de $\bar{\beta}$ que resulta em uma área sob a curva igual a r.

Com o valor de $\bar{\beta}$ encontrado, os indivíduos "proles" podem ser calculados conforme a Equações 3.9 e 3.10.

$$ch_1 = \bar{q} - \frac{1}{2}\bar{\beta}(p_2 - p_1) \tag{3.9}$$

$$ch_2 = \bar{q} + \frac{1}{2}\bar{\beta}(p_2 - p_1) \tag{3.10}$$

Os itens apresentados nesta seção não representam todos os operadores de recombinação para um GA, sendo os listados apenas as recombinações mais utilizadas na literatura.

3.3.1.5 Operador de mutação

O operador de mutação é comumente utilizado após a aplicação de um operador de recombinação. O principal objetivo de se utilizar este tipo de operador é introduzir diversidade na nova geração. Assim como nos operadores de seleção e recombinação, existem diversas maneiras de realizar a mutação nos novos indivíduos, sendo os operadores a seguir os mais utilizados na literatura:

- Mutação de Inversão de Bits

  Operador de mutação utilizado para codificação binária. Este operador percorre cada *bit* do indivíduo e realiza a troca do seu valor com base em uma determinada taxa de mutação p_m, que é definida pelo usuário. Se o *bit* analisado possui o valor 1 e foi selecionado para troca, seu valor é alterado para 0, e vice-versa. Os exemplos a seguir esclarecem esta operação:

$$1|0|11|1|01|0|01 \quad \rightarrow \quad 1|1|11|0|01|1|01$$
$$|0|1010|1|101|0| \quad \rightarrow \quad |1|1010|0|101|1|$$

- Mutação Aleatória

  Operador de mutação que pode ser utilizado tanto para codificação binária quanto para codificação real. Neste operador, uma parte do cromossomo é selecionada aleatoriamente e o seu valor atual é substituído por um novo valor. O novo valor é obtido igualmente de maneira aleatória, dentro do conjunto ou intervalo suportado pela variável em questão. Os exemplos a seguir esclarecem esta operação:

$$0110|0|110 \quad \rightarrow \quad 0110|1|110 \;\; \textit{conjunto} \; \{0,\ 1\}$$
$$22.5 \; |1.34| \; 47.89 \quad \rightarrow \quad 22.5 \; |4.34| \; 47.89 \;\; \textit{intervalo} \; [0.2,\ 6.0]$$

- Mutação Polinomial

  Operador de mutação utilizado para codificação real proposto por Deb e Goyal (1996). Este operador funciona realizando uma perturbação nas proximidades do indivíduo por meio de uma distribuição de probabilidade polinomial. Seja x_i a variável i de um determinado indivíduo. Definimos os valores $x_i^{(L)}$ e $x_i^{(U)}$ como seus limitantes inferiores e superiores, respectivamente. A mutação de x_i, que denotamos como x_i', é gerada utilizando um valor aleatório $r \in [0, 1]$ e apresentada pela Equação 3.11.

$$x_i' = \begin{cases} x_i + \bar{\delta}_L(x_i - x_i^{(L)}), & \text{para } r \leq 0.5, \\ x_i + \bar{\delta}_R(x_i^{(R)} - x_i) & \text{para } r > 0.5 \end{cases} \tag{3.11}$$

  onde os valores $\bar{\delta}_L$ e $\bar{\delta}_R$ podem ser obtidos conforme as Equações 3.12 e 3.13:

$$\bar{\delta}_L = (2r)^{(1/1+\eta_m)} - 1, para\ r \leq 0.5 \tag{3.12}$$

$$\bar{\delta}_R = 1 - (2(1-r))^{(1/1+\eta_m)}, para\ r > 0.5 \tag{3.13}$$

  O hiperparâmetro η_m é informado pelo usuário e é responsável por controlar a similaridade do indivíduo mutado com o original. Valores pequenos geram indivíduos mais similares ao original, enquanto valores grandes geram indivíduos dissimilares. Um importante detalhe deste operador é que ele não gera nenhum valor fora dos limitantes das variáveis do intervalo.

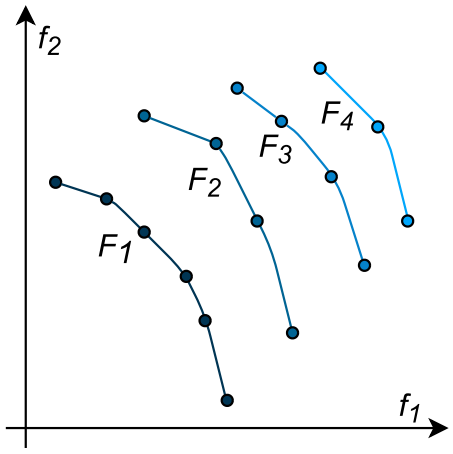
Assim como os itens apresentados na seção de recombinação, os itens apresentados na etapa de mutação não representam todos os operadores existentes para um GA. Os listados são apenas as mutações mais utilizadas na literatura.

3.3.2 Algoritmo Genético de Classificação por Não Dominância I

O Algoritmo Genético de Classificação por Não Dominância I (*Nondominated Sorting Genetic Algorithm I* - NSGA-I) (SRINIVAS; DEB, 1994) foi desenvolvido para funcionar em problemas multiobjetivo. O desenvolvimento do algoritmo foi baseado em uma sugestão de Goldberg (1989), onde ele propunha a classificação por não dominância e o Compartilhamento, que estão descritos nesta seção.

A classificação por não dominância é um procedimento que atribui soluções de uma população a diferentes frentes de Pareto com base em suas relações de dominância (LONG et al., 2021). Considerando uma população $\mathcal{P}$ como exemplo, a primeira etapa da classificação por não dominância consiste em selecionar todas as soluções não dominadas da população $\mathcal{P}$ e atribuí-las à primeira frente de Pareto, denominada como $\mathcal{F}_1$; em seguida, todas as soluções não dominadas na população $\mathcal{P} - \mathcal{F}_1$ são atribuídas à segunda frente de Pareto, denominada como $\mathcal{F}_2$; esse processo se repete até que todas as soluções da população $\mathcal{P}$ sejam atribuídas a alguma frente de Pareto. A Figura 12, extraída e adaptada de Long et al. (2021), exemplifica a classificação por não dominância.

Figura 12 – Exemplo de quatro frentes de Pareto seguindo a classificação por não dominância.



Fonte: Extraída e adaptada de Long et al. (2021).

O algoritmo NSGA-I difere de um GA simples apenas no modo como o operador de seleção funciona. Srinivas e Deb (1994) descrevem o processo do NSGA-I da seguinte forma: A primeira etapa consiste em realizar a classificação por não dominância da população. Em seguida, a primeira frente de Pareto é obtida e é atribuído a cada indivíduo um grande valor de aptidão fictício. O mesmo valor de aptidão é atribuído com o intuito de proporcionar um potencial reprodutivo igual a todos os indivíduos não dominados. Para manter a diversidade na população, esses indivíduos da primeira frente de Pareto são então Compartilhados (GOLDBERG et al., 1987; GOLDBERG; DEB, 1991), sendo-lhes

atribuídos valores fictícios de aptidão. O Compartilhamento é realizado executando a operação de seleção usando valores de aptidão degradados, obtidos pela divisão do valor de aptidão original de um indivíduo por uma quantidade proporcional ao número de indivíduos ao seu redor. Isso permite a coexistência de vários pontos ótimos na população.

O Compartilhamento em cada frente é obtido calculando um valor de função de compartilhamento entre dois indivíduos na mesma frente, conforme a Equação 3.14.

$$Sh(d_{ij}) = \begin{cases} 1 - (\frac{d_{ij}}{\sigma_{compar}})^2, & \text{se } d_{ij} < \sigma_{compar} \\ 0, & \text{caso contrário} \end{cases} \tag{3.14}$$

onde o parâmetro d_{ij} representa a distância fenotípica entre dois indivíduos i e j na frente atual, e σ_{compar} é a distância fenotípica máxima permitida entre quaisquer dois indivíduos para se tornarem membros de um nicho. Uma contagem de nicho é obtida somando os valores da função de compartilhamento acima para todos os indivíduos na frente atual. Por fim, o valor compartilhado de aptidão de cada indivíduo é calculado dividindo seu valor fictício de aptidão pela contagem de nicho.

Após realizar o Compartilhamento, esses indivíduos da primeira frente de Pareto são temporariamente ignorados para processar o restante das frentes, da maior dominância para a menor. Novamente, é realizada a etapa de atribuir um valor de aptidão fictício, que é menor do que a aptidão mínima Compartilhada da frente anterior. Esse processo se repete até que todas as frentes passem por esse processo. A população é então reproduzida de acordo com os valores de aptidão fictícia.

Como descrito anteriormente, o NSGA-I difere de um GA apenas na forma como a etapa de seleção funciona. Antes de aplicar qualquer operador de seleção, o algoritmo realiza a criação de um único valor de aptidão que resume a qualidade das soluções. Isso implica que após o processo de criação do valor de aptidão, a etapa de seleção, recombinação e mutação que ocorriam em um GA permanecem iguais. O critério de convergência (normalmente o número máximo de gerações) é verificado e, caso não seja satisfeito, o algoritmo refaz todo o processo com a nova população.

As soluções não dominadas durante o processo de busca do NSGA-I são então consideradas como sendo as soluções ótimas do problema.

3.3.3 Algoritmo Genético de Classificação por Não Dominância II

Deb et al. (2002) apresentaram uma evolução no NSGA-I, chamada de Algoritmo Genético de Classificação por Não Dominância II (*Nondominated Sorting Genetic Algorithm II* - NSGA-II), com o objetivo de melhorar alguns pontos negativos da primeira versão. No trabalho, foram apresentadas as principais críticas à abordagem do NSGA-I, sendo elas:

- **Alta complexidade computacional de classificação por não dominância:**

O algoritmo de ordenação por não dominância utilizado tem uma complexidade computacional de $\mathcal{O}(MN^3)$, em que M representa o número de funções objetivo do problema e N é o tamanho da população. Isso torna o NSGA-I computacionalmente dispendioso para grandes tamanhos de população, onde essa alta complexidade decorre do procedimento de classificação por não dominância em cada geração.

- **Falta de elitismo:** Alguns trabalhos (ZITZLER et al., 2000; RUDOLPH, 2001) apresentam que o elitismo pode acelerar significativamente o desempenho do GA, o que também pode ajudar a evitar a perda de soluções boas uma vez obtidas.

- **Especificação do hiperparâmetro de compartilhamento σ_{compar}:** Os mecanismos tradicionais para garantir a diversidade em uma população, a fim de obter uma ampla variedade de soluções equivalentes, têm se baseado principalmente no conceito de Compartilhamento. O principal problema com o Compartilhamento é que requer a especificação de um parâmetro de compartilhamento σ_{compar} pelo usuário ou especialista. Embora exista algum trabalho sobre dimensionamento dinâmico do parâmetro de compartilhamento (FONSECA; FLEMING, 1998), um mecanismo de preservação de diversidade sem necessidade de parâmetro é desejável.

A classificação por não dominância do NSGA-I utilizava uma abordagem simples, onde para realizar a detecção das soluções presentes na primeira frente não dominada em uma população de tamanho N, cada solução era comparada com todas as outras soluções da população para determinar se ela é ou não dominada. Isso requer $\mathcal{O}(MN)$ comparações para cada solução, onde temos M funções objetivo. Quando esse processo se repete para localizar todos os membros do primeiro nível não dominado da população, a complexidade total se torna $\mathcal{O}(MN^2)$. Considerando o pior cenário, onde apenas um elemento pertence a cada frente não dominada, o custo computacional seria de $\mathcal{O}(MN^3)$.

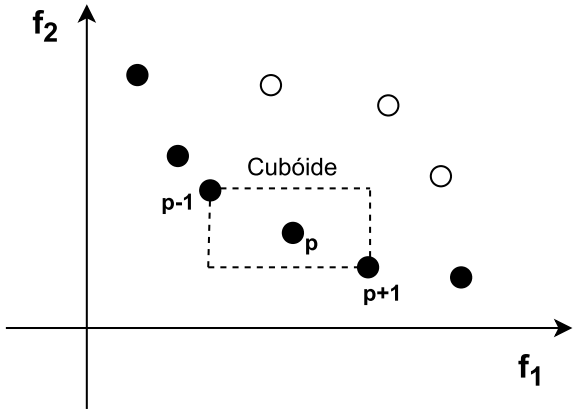
Com o objetivo de tornar este processo mais eficiente, o NSGA-II traz uma forma mais rápida de classificação por não dominância. Primeiramente, para cada uma das soluções, calculamos duas informações: n_p, que é o número de soluções que dominam a solução p, e S_p, um conjunto de soluções que a solução p domina. Este processo requer $\mathcal{O}(MN^2)$ comparações. Todas as soluções da primeira frente não dominada terão o número de soluções que a dominam igual a zero. Como próximo passo, para cada solução p com o valor de $n_p = 0$, visitamos cada membro q do seu conjunto S_p e reduzimos sua contagem de dominação em um elemento. Durante esse processo, se qualquer solução q tiver $n_q = 0$, ela é colocada em uma lista separada Q, que se tornará a segunda frente não dominada. Esse processo se repete até identificar todas as frentes. É importante observar que, embora a complexidade de tempo tenha sido reduzida para $\mathcal{O}(MN^2)$, ainda é necessário armazenar $\mathcal{O}(N^2)$ informações.

Além da melhoria na classificação por não dominância, o NSGA-II substituiu a etapa de Compartilhamento do NSGA-I, que apresentava os seguintes problemas: o desempenho da função de compartilhamento em manter uma variedade de soluções depende muito do valor σ_{compar} escolhido, e cada solução deve ser comparada com todas as outras soluções na população, tornando a complexidade geral da abordagem igual a $\mathcal{O}(N^2)$. O NSGA-II substituiu o Compartilhamento por uma abordagem de comparação aglomerada que elimina ambas as dificuldades apresentadas.

A primeira etapa é estimar a densidade das soluções em torno de uma solução específica p na população. O valor, denotado como $p_{distancia}$, é obtido através do cálculo da distância média entre os dois pontos adjacentes de p, para cada um dos objetivos. Este valor serve como uma estimativa do perímetro do maior cubóide que envolve esse ponto, chamado de Distância de Aglomeração (*Crowding Distance* - CD).

Para realizar o cálculo da CD, é necessário ordenar a população de acordo com cada valor da função objetivo em ordem crescente de magnitude. As soluções de contorno recebem um valor de CD infinito devido ao fato de não possuírem um ponto adjacente. Já as soluções intermediárias receberão um valor de CD igual à diferença absoluta normalizada dos valores das funções objetivo entre duas soluções adjacentes. Cada função objetivo é normalizada e o valor geral da CD é calculado como sendo a soma das distâncias individuais correspondentes a cada objetivo.

Figura 13 – Cálculo da Distância de Aglomeração.



Fonte: Extraída e adaptada de Deb et al. (2002).

A Figura 13, extraída e adaptada de Deb et al. (2002), exemplifica o processo de cálculo da CD. É possível observar que soluções que se encontram em regiões com um menor número de pontos recebem valores maiores de CD do que as soluções localizadas em regiões com um número maior de pontos.

Através destas definições, o trabalho apresenta o operador Comparação Aglomerada (*Crowded-Comparison* - CC), denotado como $\prec_n$, que orienta o processo de seleção nos

vários estágios do algoritmo. Este operador ajuda a guiar a busca para uma frente de Pareto ótima uniformemente distribuída. Assumindo que cada indivíduo p da população possui dois atributos, sendo a sua classificação por não dominância (p_{rank}) e a sua CD ($p_{distancia}$), definimos:

$$p \prec_n q \quad Se\ (p_{rank} < q_{rank})\ ou\ ((p_{rank} = q_{rank})\ e\ (p_{distancia} > q_{distancia}))$$

ou seja, entre duas soluções p e q com classificações de não-dominação diferentes, preferimos a solução com a classificação mais baixa. Caso contrário, se ambas as soluções pertencem à mesma frente, preferimos a solução que está localizada em uma região menos populosa. Este operador é utilizado no algoritmo como o critério de seleção.

Com essas três inovações (um novo procedimento rápido de classificação por não dominância, um procedimento de estimativa de CD e o operador CC), podemos descrever as etapas do algoritmo NSGA-II. Inicialmente, uma população aleatória P_0 é criada com tamanho N_{pop}. A população P_0 é classificada com base na não-dominação. Cada uma das soluções recebe um valor de aptidão igual ao seu nível de não dominância, sendo 1 o melhor valor e valores maiores piores, considerando um problema de minimização. Os operadores de seleção, recombinação e mutação são usados para criar uma população Q_0 de descendentes. A cada geração t, começando em $t = 0$, as populações P_t e Q_t são unidas para formar uma população R_t de tamanho $2N_{pop}$. A nova classificação por não dominância é aplicada à população R_t. Em seguida, selecionam-se as i melhores fronteiras com o objetivo de formar a população P_{t+1}. Caso ocorra de as fronteiras selecionadas não formarem um conjunto de tamanho N_{pop}, ordenam-se os indivíduos da fronteira $i + 1$ de acordo com o operador CC para escolher as melhores soluções necessárias para preencher todas as vagas da população. Aplicam-se os operadores de cruzamento e mutação na população P_{t+1} com o objetivo de gerar a população Q_{t+1} e o processo se repete até o critério de parada estabelecido. A Figura 14, extraída e adaptada de Deb et al. (2002), exemplifica o procedimento realizado dentro do algoritmo NSGA-II.

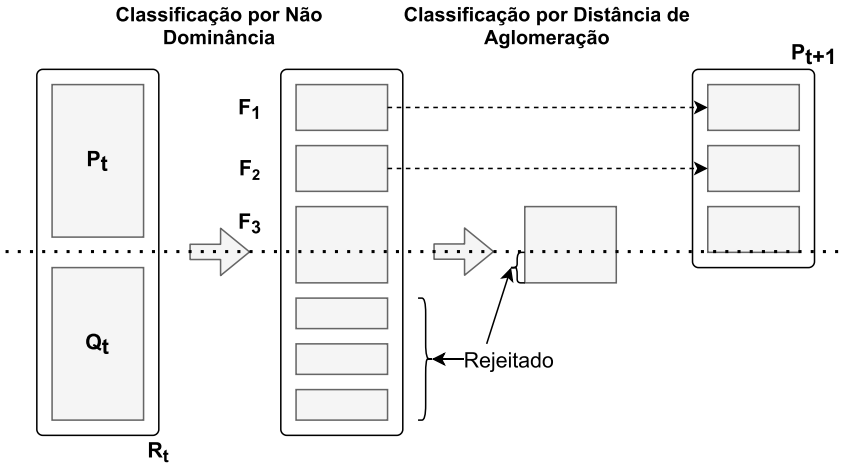
É importante ressaltar que o NSGA-II utiliza a seleção por Torneio e que o operador CC é utilizado para definir qual solução é melhor do que a outra.

3.3.4 Algoritmo Genético de Classificação por Não Dominância III

Deb e Jain (2013) apresentaram uma evolução no NSGA-II, chamado de Algoritmo Genético de Classificação por Não Dominância III (*Nondominated Sorting Genetic Algorithm III* - NSGA-III), com o objetivo principal de lidar com um número elevado de funções objetivo, sendo considerado elevado a partir de quatro funções ou mais.

No trabalho, eles começam a listar as dificuldades, mapeadas por outros trabalhos (DEB et al., 2006; GARZA-FABRE et al., 2009), que as metodologias enfrentam ao

Figura 14 – Procedimento do NSGA-II.



Fonte: Extraída e adaptada de Deb et al. (2002).

trabalhar no princípio de dominância em muitos objetivos. Sendo elas:

- **Uma grande fração da população é não-dominada:** Com o aumento do número de objetivos, uma fração cada vez maior da população gerada aleatoriamente torna-se não-dominada. Como a maioria dos algoritmos evolutivos para problemas multiobjetivos enfatiza soluções não-dominadas em uma população, não há muito espaço para criar novas soluções em uma geração, tornando o processo de busca lento e ineficiente.

- **A avaliação da medida de diversidade torna-se computacionalmente cara:** A identificação dos vizinhos em problemas com muitos objetivos é computacionalmente custosa, tornando o processo de determinar a aglomeração das soluções em uma população também custoso. Neste processo, para agilizar os cálculos, pode ser necessário fazer concessões na estimativa da diversidade, mas isso pode levar a uma distribuição insatisfatória das soluções no final.

- **A operação de recombinação pode ser ineficiente:** Se apenas um pequeno grupo de soluções for encontrado em um espaço de alta dimensão, então essas soluções provavelmente estarão muito distantes umas das outras. O processo de recombinação, considerado o operador essencial para algoritmos evolutivos em problemas multiobjetivos, torna-se questionável em tal população. Como esperado, os descendentes de duas soluções que estão separadas provavelmente também estarão separados. Desta forma, operadores especiais de recombinação podem ser necessários para lidar eficientemente com problemas com um alto número de objetivos.

Os autores apresentam duas opções para lidar com os três problemas em um processo evolutivo baseado em dominação. Essas opções são:

- **Uso de um princípio de dominação especial:** Utilizar um princípio de dominação especial que discretize de forma adaptativa a frente ótima de Pareto e encontre um conjunto de pontos bem distribuído, abordando assim o primeiro problema descrito. O segundo desafio da preservação da diversidade também é mitigado por essa abordagem. Quanto ao terceiro problema, ele pode ser abordado empregando um esquema de restrição de recombinação ou um método de recombinação único que enfatiza soluções próximas aos genitores.

- **Uso de uma pesquisa segmentada múltipla predefinida:** Ao procurar soluções Pareto-ótimas, um algoritmo pode ser programado para realizar várias pesquisas direcionadas em vez de uma pesquisa global. O primeiro problema de lidar com um grande conjunto não dominado também é reduzido, pois são determinados os pontos ideais que correspondem a cada uma das pesquisas direcionadas. Usando uma estratégia de restrição de recombinação, na qual duas soluções de alvos vizinhos estão envolvidas no procedimento de criação de novas soluções, pode-se abordar o terceiro problema.

O NSGA-III é baseado no segundo princípio, utilizando uma pesquisa segmentada múltipla predefinida. Especificamente, o trabalho propõe duas estratégias distintas para colocar em prática a noção de busca direcionada múltipla, sendo elas:

1. É possível especificar várias direções de pesquisa antecipadamente, com o escopo dessas pesquisas abrangendo toda a frente Pareto-ótima. Dada a natureza dispersa das direções de busca, as soluções para a maioria dos problemas provavelmente serão espalhadas por toda a frente de Pareto-ótimo. Outro trabalho, desenvolvido por Zhang e Li (2007), utiliza esse conceito.

2. Pontos de referência predefinidos podem ser fornecidos em vez de várias direções de pesquisa. Posteriormente, os pontos que correspondem a cada ponto de referência podem ser enfatizados para localizar um grupo de pontos Pareto-ótimos com uma ampla faixa de valores. Algumas implementações foram propostas (ZHOU et al., 2009; SINDHYA et al., 2012; DEB; JAIN, 2012), e um método alternativo estendendo os algoritmos propostos por Deb e Jain (2012) é desenvolvido pelo NSGA-III.

Embora apresente um novo operador de seleção, a estrutura proposta do NSGA-III é similar à do NSGA-II original. No NSGA-III, no entanto, um grande número de pontos de referência amplamente dispersos é fornecido e atualizado de forma adaptativa, o que torna mais fácil preservar a variedade da população do que no NSGA-II.

No NSGA-III, o operador CD é substituído pelas seguintes abordagens:

- **Classificação da população em níveis de não dominância:** Como passo inicial, S_t inclui todas as frentes não dominadas de 1 a l, onde novamente S_t representa a criação de uma nova população da geração t. Se $|S_t| = N$, a geração atual está completa e a nova é iniciada como $P_{t+1} = S_t$. Caso contrário, se $|S_t| > N$, os indivíduos das $l-1$ frentes são selecionados para P_{t+1} e o restante, formado por $N - |P_{t+1}|$ indivíduos, é selecionado da última frente F_l. À medida que avançamos no restante da abordagem, continuaremos a detalhar a seleção desta última frente F_l.

- **Determinação de pontos de referência em um hiperplano:** Conforme mencionado anteriormente, o NSGA-III utiliza um conjunto fixo de pontos de referência para verificar se as soluções que ele gera são distintas entre si. O algoritmo permite escolher entre usar um conjunto predefinido de pontos de referência ou fornecer seu próprio conjunto preferido de pontos. Sem conhecer as preferências do usuário, o NSGA-III pode utilizar várias estruturas predefinidas para posicionar os pontos de referência. No entanto, por padrão, ele emprega a abordagem sistemática de Das e Dennis (1998), que posiciona os pontos em um hiperplano normalizado (um simplex unitário de dimensão $(M-1)$, onde M é a quantidade de funções objetivo), o qual é perpendicular a todos os eixos objetivos e intercepta cada eixo com um valor unitário. Em um problema com M objetivos e p divisões ao longo de cada objetivo, podemos calcular o número total de pontos de referência (H) conforme a Equação 3.15.

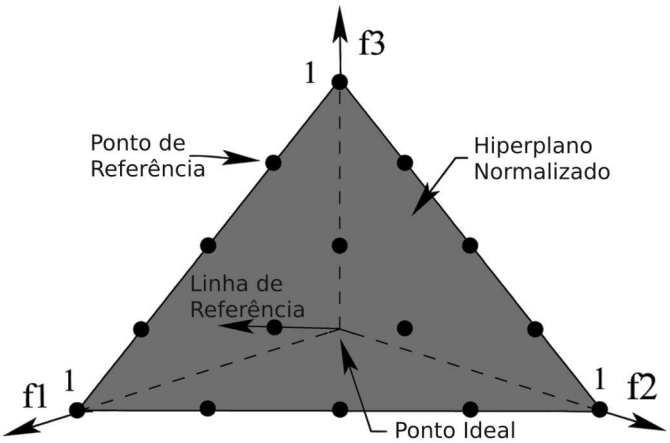
$$H = \binom{M+p-1}{p} \tag{3.15}$$

  A Figura 15, extraída e adaptada de Deb e Jain (2013), exemplifica os pontos de referência a serem criados para quatro divisões $(p = 4)$ em um problema com 3 objetivos.

  O algoritmo prioriza não apenas soluções não dominadas, mas também membros da população que estão conectados a cada um desses pontos de referência. Assim como os pontos de referência gerados estão espalhados por todo o hiperplano normalizado, as soluções obtidas provavelmente também estarão amplamente distribuídas na frente de Pareto-ótimo ou próximo a ela.

- **Normalização adaptativa dos membros da população:** Primeiramente, o ponto ideal da população S_t é determinado identificando, para cada função objetivo $i = 1, 2, ..., M$, o valor mínimo z_i^{min} em $\bigcup_{\tau=0}^{t} S_\tau$. Desta forma, o ponto ideal $\bar{z} = (z_1^{min}, z_2^{min}, ..., z_M^{min})$ para a população S_t é construído. Para transformar $\bar{z}$ de S_t em um vetor de zeros, primeiro subtraímos cada função objetivo i pelo valor mínimo encontrado, indicando uma função de tradução como $f_i^{'}(\mathbf{x}) = f_i(\mathbf{x}) - z_i^{min}$.

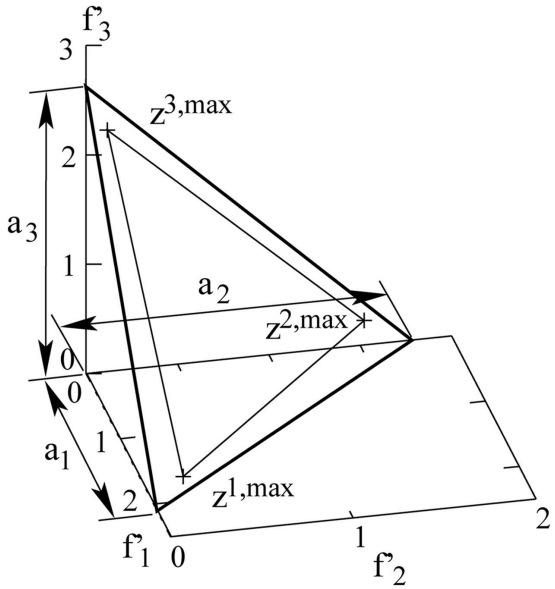
Figura 15 – Quinze pontos de referência estruturados exibidos em um plano de referência normalizado para um problema de três objetivos com $p = 4$.



Fonte: Extraída e adaptada de Deb e Jain (2013).

Em seguida, determinamos o ponto extremo $\mathbf{z}^{i,max}$ em cada eixo i-objetivo como sendo a solução em S_t que minimiza a Função Escalar de Realização (*Achievement Scalarizing Function* - ASF) (WIERZBICKI, 1980) apropriada, formada com $f_i'(\mathbf{x})$ e um vetor de peso próximo ao eixo i-ésimo objetivo. Esses M vetores extremos são, então, usados para constituir um hiperplano M-dimensional e, assim, obtemos o intercepto a_i do eixo i-ésimo objetivo. A Figura 16, extraída de Deb e Jain (2013), exemplifica o processo de cálculo dos interceptos e a formação do hiperplano a partir de pontos extremos para um problema de três objetivos.

Figura 16 – O procedimento para computar interceptos e, em seguida, formar o hiperplano a partir de pontos extremos para um problema de três objetivos.



Fonte: Extraída de Deb e Jain (2013).

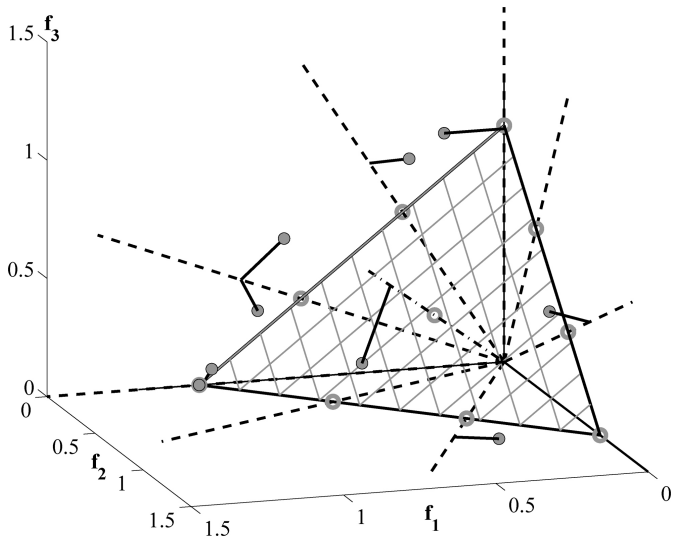
Casos degenerados e interceptações não negativas são tratados com cuidado especial. Após essa etapa, as funções objetivo podem ser normalizadas conforme a Equação 3.16.

$$f_i^n(\mathbf{x}) = \frac{f_i'(\mathbf{x})}{a_i},\ para\ i = 1, 2, ..., M \tag{3.16}$$

O processo do NSGA-III preserva de forma adaptativa a diversidade no espaço abrangido pelos membros de S_t, realizando o procedimento de normalização e criando o hiperplano a cada geração usando os pontos extremos já obtidos desde o início do processo de busca. Isso permite a resolução de problemas com uma frente ótima de Pareto cujos valores objetivos podem estar em escalas diferentes.

- **Operação da Associação:** Após realizar o processo de normalização, a etapa subsequente consiste em associar cada indivíduo da população atual a um ponto de referência. Para isso, definimos uma linha de referência que conecta a origem com cada ponto de referência no hiperplano. Em seguida, determinamos a distância de cada membro da população S_t perpendicularmente às linhas de referência. Associar um ponto de referência a um indivíduo da população significa atribuir à linha de referência desse ponto a distância mais próxima ao indivíduo da população no espaço objetivo normalizado. A Figura 17, extraída de Deb e Jain (2013), exemplifica o processo de associação.

Figura 17 – Associação dos indivíduos da população com pontos de referência.



Fonte: Extraída de Deb e Jain (2013).

É importante notar que um ponto de referência pode ter um ou mais indivíduos da população associados a ele, ou não ter nenhum.

- **Operação de Preservação de Nicho:** Denotamos a contagem de nicho como ρ_j, que representa a contagem de indivíduos de uma população P_{t+1} associados ao j-ésimo ponto de referência. Por meio dessa contagem, o NSGA-III define um novo operador de preservação de nicho.

  Primeiramente, buscamos o conjunto de pontos de referência com mínimo ρ_j, ou seja, $J_{min} = \{j : \text{argmin}_j\, \rho_j\}$. Em casos onde o conjunto tem mais de um elemento, escolhemos aleatoriamente um deles e o denominamos de $\bar{j}$. A mesma denominação é feita para o único elemento do conjunto em caso contrário.

  Nos casos em que temos $\rho_{\bar{j}} = 0$, ou seja, não existem indivíduos associados ao ponto de referência $\bar{j}$, podem ocorrer dois cenários com $\bar{j}$ no conjunto F_l. No primeiro cenário, existem um ou mais indivíduos na frente F_l que estão associados ao ponto de referência $\bar{j}$. Nesse caso, aquele que tiver a menor distância perpendicular à linha de referência é adicionado a P_{t+1}. A contagem de $\rho_{\bar{j}}$ é, então, incrementada em uma unidade. No segundo cenário, a frente F_l não possui nenhum indivíduo associado ao ponto de referência $\bar{j}$. Nesse outro caso, o ponto de referência é excluído das considerações adicionais para a geração atual.

  No caso de $\rho_{\bar{j}} \geq 0$, um indivíduo escolhido aleatoriamente (ou o ponto mais próximo do ponto de referência, ou qualquer outro critério de preservação da diversidade), se existir, da frente F_l, que é associado com o ponto de referência $\bar{j}$, é adicionado a P_{t+1}. A contagem de $\rho_{\bar{j}}$ é então incrementada em uma unidade. Depois que as contagens de nicho são atualizadas, o processo é repetido K vezes até que toda a população P_{t+1} tenha sido preenchida.

Como supracitado, este processo substitui o operador CD do NSGA-II e apresenta uma nova abordagem que realiza uma seleção elitista cuidadosa de soluções, tentando manter a diversidade entre elas, enfatizando as soluções mais próximas da linha de referência de cada ponto de referência.

Selecionando genitores aleatoriamente de P_{t+1}, a população Q_{t+1} é construída utilizando os operadores de recombinação e mutação, igual ao que é apresentado no NSGA-II. No entanto, os autores recomendam a escolha de um valor um pouco maior para o índice de distribuição no operador SBX, que produzirá soluções descendentes próximas de seus genitores (para enfrentar o terceiro problema de ineficiência na recombinação).

3.3.5 Métricas de Avaliação

As métricas de avaliação são um componente vital de todo problema de otimização, pois permitem quantificar o desempenho do algoritmo e determinar se ele atende aos objetivos exigidos. Elas nos permitem comparar diferentes algoritmos e suas configurações, identificar áreas de melhoria e tomar decisões informadas sobre o desempenho. Nesta

subseção, examinaremos métricas de avaliação que são utilizadas nesta tese para problemas de otimização e como elas podem nos ajudar a tomar decisões fundamentadas.

3.3.5.1 Cobertura

A métrica Cobertura (*Coverage*), proposta por Zitzler et al. (2000), é utilizada para analisar a capacidade de convergência de um algoritmo em comparação com outro. Seu objetivo é indicar a proporção de soluções de uma Frente de Pareto que são mais prevalentes do que as de outra Frente de Pareto. Considere a existência de duas Frentes de Pareto, A e B, como dois conjuntos diferentes de soluções que são representados no espaço das funções objetivo. Definimos a métrica Cobertura pela a Equação 3.17.

$$C(A,B) = \frac{|b \in B; \exists a \in A : a \preceq b|}{|B|} \tag{3.17}$$

A métrica pode assumir valores no intervalo entre zero e um. Quando o valor é igual a um, significa que todas as soluções da Frente de Pareto A dominam as soluções da Frente de Pareto B, enquanto quando for zero representa o oposto.

3.3.5.2 Espaçamento

A métrica Espaçamento (*Spacing*), proposta por Schott (1995), foi proposta para avaliar a diversidade de soluções em torno da Frente de Pareto. O valor da métrica é obtido através do cálculo da distância relativa entre as soluções adjacentes da Frente de Pareto. Podemos definir a métrica através da Equação 3.18.

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2} \tag{3.18}$$

onde

$$d_i = \min_j(\sum_{k=1}^{n_f} |f_k^i \overrightarrow{x} - f_k^j \overrightarrow{x}|),$$

f_k^i e f_k^j são a k-ésima função objetivo da solução i e j, respectivamente. A solução j é a solução adjacente à direita da solução i. Aqui, n representa o número de soluções não dominadas, n_f representa a quantidade de funções objetivo, e $\bar{d}$ é a distância média entre as soluções adjacentes.

Quando o valor é igual a zero, significa que todas as soluções da Frente de Pareto estão igualmente espaçadas.

3.3.5.3 Hipervolume

A métrica Hipervolume (*Hypervolume*), proposta por Zitzler (1999), busca medir, simultaneamente, a convergência e o espalhamento da Frente de Pareto analisada. A

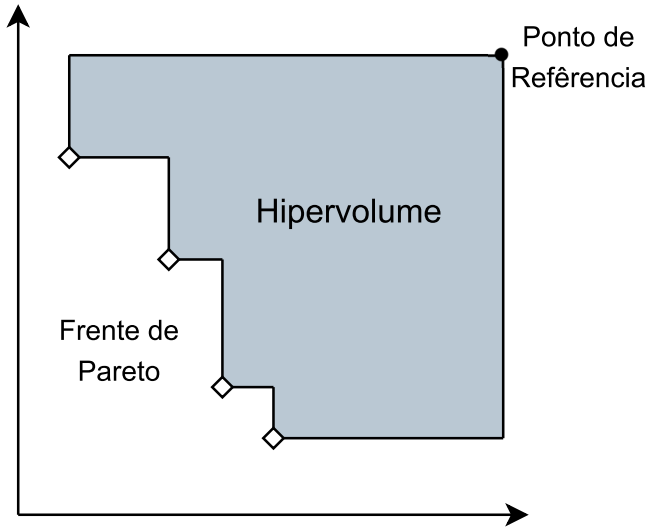
métrica é calculada avaliando o espaço especificado pelos objetivos do problema de otimização e medindo o volume desse espaço que é dominado pelas soluções presentes na Frente de Pareto. Quanto maior for o volume do espaço dominado pelas soluções, melhor o conjunto de soluções é considerado. A métrica pode ser escrita conforme a Equação 3.19 (GUERREIRO et al., 2020):

$$HV = \Lambda(q \in \mathbb{R}^{n_f} | \exists p \in S : p \leq q \; e \; q \leq W) \tag{3.19}$$

onde $\Lambda(.)$ representa a medida de Lebesgue (RUDIN, 1987), S é o conjunto de soluções da Frente de Pareto e $W \in \mathbb{R}^{n_f}$ é o ponto de referência. W é utilizado para definir quais são os limites do espaço das funções objetivo e, normalmente, é definido como o ponto em que todos os objetivos estão em seu valor máximo possível.

A Figura 18 apresenta um exemplo da métrica Hipervolume, demonstrando o ponto de referência e a Frente de Pareto. Neste caso, o Hipervolume é representado pela área em cinza.

Figura 18 – Exemplo da métrica Hipervolume.



Fonte: Elaborada pelo autor (2023).

3.3.5.4 Perfis de desempenho

A métrica/ferramenta gráfica Perfis de Desempenho (*Performance Profiles*), proposta por Dolan e Moré (2002), tem como objetivo simplificar a interpretação dos resultados de experimentos que envolvem grandes quantidades de dados, tornando-os mais acessíveis e compreensíveis.

De modo geral, para avaliar a eficácia de diferentes soluções em resolver problemas específicos, é comum realizar uma análise comparativa entre vários algoritmos diferentes, com o objetivo de identificar aqueles que apresentam o melhor desempenho.

Seja um conjunto de problemas teste $P = \{p_j | \forall j = 1, 2, \ldots, n_p\}$, um conjunto de algoritmos $S = \{s_i | \forall i = 1, 2, \ldots, n_s\}$ e $t_{p,s} > 0$ uma métrica de desempenho (tradicionalmente utilizada é o tempo). Podemos definir a razão de desempenho através da Equação 3.20.

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}, \tag{3.20}$$

onde $\min\{t_{p,s} : s \in S\}$ representa o valor $t_{p,s}$ do algoritmo em S que consegue a melhor métrica de desempenho (a com menor valor) para resolver o problema p. Faz-se necessário existir um valor $r_w \geq r_{p,s}$ para todos os $r_{p,s}$, tal que $r_{p,s} = r_w$ se e somente se o algoritmo s não consegue resolver o problema p. Desta forma, garantimos que, no pior cenário, o valor de r_w é no mínimo igual ao maior valor existente.

Com a razão de desempenho definida, podemos estabelecer a métrica de Perfil de Desempenho de um algoritmo pela Equação 3.21.

$$\rho_s(\tau) = \frac{1}{n_p} |p \in P : r_{p,s} \leq \tau| \tag{3.21}$$

onde $\rho_s(\tau)$ é a probabilidade de que a razão de desempenho $r_{p,s}$ do algoritmo $s \in S$ esteja dentro de um fator $\tau \in [1, r_w]$ do melhor desempenho, considerando todos os algoritmos em S.

O valor obtido quando temos $\rho_s(1)$ representa a porção de problemas em que o algoritmo s apresentou o melhor desempenho de acordo com a medida definida. Quando desejamos encontrar qual algoritmo é mais robusto, buscamos aquele que possui o menor valor de τ tal que $\rho_s(\tau) = 1$.

Como supracitado, conseguimos ter uma visão de desempenho para cada valor de τ, podendo variá-lo entre $[1, r_w]$. Uma forma de avaliar a performance global de um algoritmo utilizando os Perfis de Desempenho, como apresentado por Barbosa et al. (2010), é calcular a área sob a curva. Algoritmos que apresentam o maior valor dessa área indicam que são os mais eficientes para resolver os problemas analisados.

4 APRENDIZADO DE MÁQUINA

O Aprendizado de Máquina (*Machine Learning* - ML) é um sub-ramo da Inteligência Artificial (*Artificial Intelligence* - AI) que proporciona a capacidade dos sistemas melhorarem automaticamente, através da otimização de um determinado critério de desempenho, utilizando os dados de exemplos ou experiências anteriores para este fim (ALPAYDIN, 2020). Através do ML, podemos desenvolver modelos que aprendem através dos dados, sem a necessidade de se programarem as regras de negócio como nos sistemas especialistas.

Cada um dos modelos gerados através do ML possui alguns parâmetros que são otimizados utilizando os dados de exemplos ou experiências anteriores (FRIEDMAN et al., 2001). Deste modo, o modelo extrai os conhecimentos dos dados e pode ser utilizado para fazer previsões no futuro (outro conjunto de dados) ou descritivo para obter conhecimento dos dados, ou até mesmo para ambas as coisas.

4.1 TIPOS DE APRENDIZADO DE MÁQUINA

Dentro do ML, existem algumas variações da definição dos algoritmos, sendo elas geralmente divididas de acordo com sua finalidade. O problema a ser resolvido e o tipo de entrada e de saída influenciam na categorização desses algoritmos. As principais categorias são as seguintes (RUSSELL; NORVIG, 2002):

- **Supervisionado**:

  Possui dados rotulados, ou seja, para cada dado de entrada, tem-se o dado de saída correspondente. O modelo aprende com os dados históricos e pode ser aplicado a novos dados para prever eventos futuros. Assim, a partir de um conjunto de dados de treinamento conhecido e rotulado, o algoritmo de ML supervisionado é capaz de produzir uma função para fazer previsões sobre os valores de saída em dados nunca vistos antes.

- **Não Supervisionado**:

  Não possui dados rotulados, ou seja, para cada dado de entrada não se tem o dado de saída correspondente. O modelo aprende através dos dados históricos os padrões dos dados, sem a utilização de dados rotulados ou de supervisão humana. Assim, a partir de um conjunto de dados de treinamento não rotulado, o algoritmo de ML não supervisionado é capaz de produzir uma função para descrever uma estrutura oculta presente nos dados.

- **Aprendizado por Reforço**:

  Possui dados rotulados obtidos através da interação com o ambiente, ou seja, para cada dado de entrada (ação), tem-se o dado de saída correspondente (recompensa).

Um agente interage com seu ambiente, produzindo ações e recebendo recompensas (positivas ou negativas) e, através disso, o modelo aprende com esses dados e pode ser aplicado para descobrir a recompensa para cada possível ação no seu estado atual. Assim, a partir de um conjunto de dados formado pela experiência de tentativa e erro, o algoritmo de ML por reforço é capaz de produzir uma função que determina automaticamente o comportamento ideal dentro de um contexto específico para maximizar o desempenho do agente.

- **Semi-Supervisionado**:

  Possui um conjunto de dados rotulados e outros não rotulados, ou seja, para cada dado de entrada de um subconjunto, tem-se o dado de saída correspondente e para o outro conjunto não. Normalmente, apenas uma pequena quantidade destes dados possui os seus rótulos. Os algoritmos de ML semi-supervisionados estão entre o aprendizado supervisionado e o não supervisionado. Assim, a partir de um conjunto de dados de treinamento não rotulado e rotulado, o algoritmo de ML semi-supervisionado é capaz de produzir uma função para fazer previsões sobre os valores de saída e uma outra função para descrever uma estrutura oculta presente nos dados. Ambas as funções podem ser combinadas de diversas maneiras para melhorar o desempenho no problema abordado.

Este trabalho tem como foco a proposta de um método de automatização para modelos supervisionados. Deste modo, as demais não serão exploradas no decorrer do trabalho.

4.2 CATEGORIAS EM APRENDIZADO SUPERVISIONADO

Como mostrado anteriormente, dentro do ML supervisionado geramos um modelo preditivo que pode ser descrito como um problema matemático de aproximação de uma função que mapeia as variáveis de entrada às variáveis de saída. Ao treinar este modelo com dados históricos, desejamos encontrar a melhor função de mapeamento. Geralmente, as aproximações de funções em problemas supervisionados podem ser classificadas em duas categorias, sendo elas (ALPAYDIN, 2020):

- **Classificação**:

  Aproxima uma função de mapeamento de variáveis de entrada para variáveis de saída discreta, normalmente chamadas de categorias, rótulos ou classes. Esta função é capaz de prever a classe para uma determinada observação ou, em alguns casos, a predição são valores contínuos que representam as probabilidades ou pontuações (*scores*) de uma dada observação pertencer a cada uma das classes. Um problema

de classificação, por exemplo, é a tentativa de identificar se uma foto contém um cachorro ou um gato (e as suas respectivas probabilidades ou pontuações).

- **Regressão**:

  Aproxima uma função de mapeamento de variáveis de entrada para variáveis de saída contínuas (valor real), ou seja, um valor inteiro ou ponto flutuante. Normalmente, a saída contínua é uma quantidade. Em um caso especial, onde temos as variáveis de entrada ordenadas pelo tempo, consideramos como um problema de regressão de previsão de séries temporais. Um problema de regressão, por exemplo, é a tentativa de identificar o preço de um imóvel de acordo com suas características (quantidade de quartos, quantidade de banheiros, área útil, vagas na garagem, etc.). Existe um caso especial, tradicionalmente utilizado em ANN, onde as variáveis de saída são iguais às variáveis de entrada, denominado *Autoencoder* (HINTON; SALAKHUTDINOV, 2006), e tem como objetivo reduzir a dimensionalidade dos dados internamente na ANN.

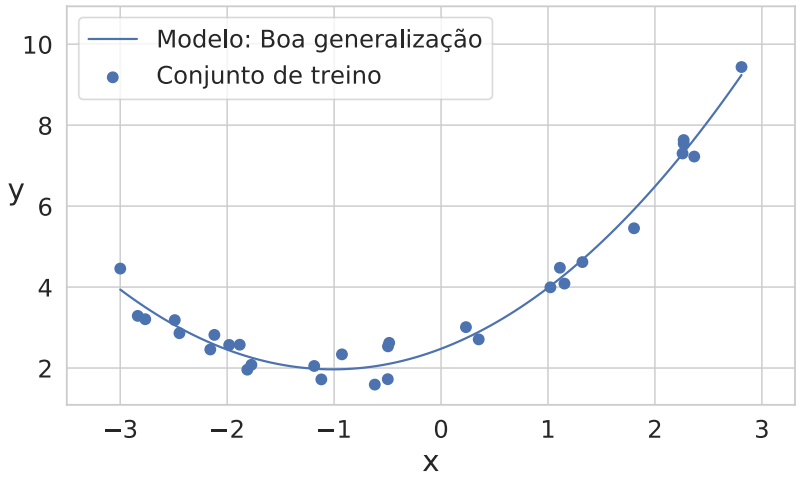
4.3 GENERALIZAÇÃO

Ao se construir um bom modelo de ML, o objetivo é que o mesmo generalize bem os dados de treinamento para quaisquer outros dados do domínio do problema. É fundamental lembrar que qualquer conjunto de dados utilizado para criar um modelo de ML é apenas uma amostra da população do problema em questão. Portanto, a principal preocupação deve ser se a amostra é representativa, ou seja, se reflete a população (COCHRAN, 1977; BOLFARINE; BUSSAB, 2005). Se isso não for verdade, estaremos criando um modelo com base em uma amostra que representa apenas uma parte da população. Nesse caso, o modelo não seria capaz de generalizar bem para a população em sua totalidade.

O conceito de generalização descreve o quão bem um modelo de ML, durante sua etapa de aprendizado, consegue extrair as características dos dados de treinamento e, posteriormente, ser aplicado com bom desempenho a exemplos específicos nunca vistos antes (BROWNLEE, 2016). Deste modo, um modelo que possui uma boa generalização pode ser utilizado para realizar previsões de dados futuros que o modelo nunca viu. A Figura 19 apresenta o conjunto de treino e um modelo com boa generalização.

Ao apresentar um baixo desempenho de generalização em uma base de dados nova, que ainda representa o mesmo domínio em que o modelo foi treinado, o modelo de ML pode estar sofrendo de sobreajuste (*Overfitting*) ou subajuste (*Underfitting*). O sobreajuste e o subajuste são as duas maiores causas de baixo desempenho dos algoritmos de ML.

Figura 19 – Conjunto de treino e modelo com boa generalização.

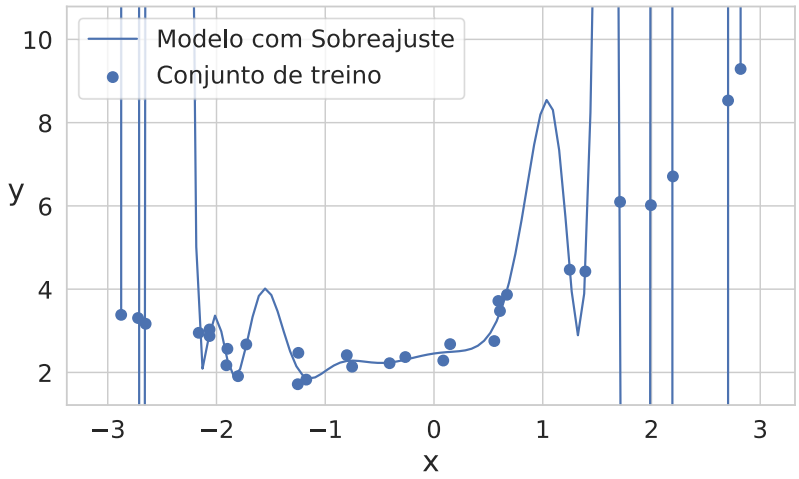


Fonte: Elaborada pelo autor (2023).

4.3.1 Sobreajuste

Um modelo de ML que sofre de sobreajuste se ajusta muito bem aos dados de treinamento, aprendendo inclusive os ruídos dos dados, o que influencia de maneira negativa o desempenho do modelo ao ser aplicado em novos dados (BROWNLEE, 2016). A Figura 20 apresenta o conjunto de treino e um modelo com sobreajuste.

Figura 20 – Conjunto de treino e modelo com Sobreajuste.

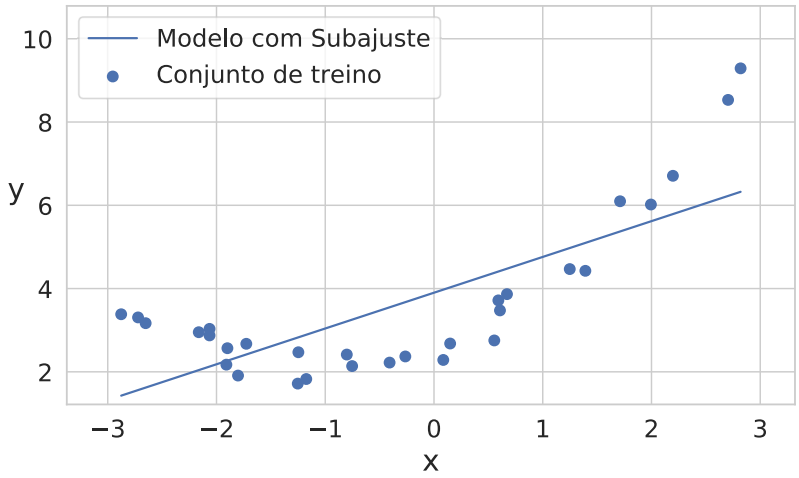


Fonte: Elaborada pelo autor (2023).

4.3.2 Subajuste

Um modelo de ML que sofre de subajuste não é capaz de se ajustar aos dados de treinamento e muito menos generalizar para um novo conjunto de dados (BROWNLEE, 2016). A Figura 21 apresenta o conjunto de treino e um modelo com subajuste.

Figura 21 – Conjunto de treino e modelo com Subajuste.



Fonte: Elaborada pelo autor (2023).

4.4 VIÉS E VARIÂNCIA

Ao desenvolver um modelo supervisionado de ML, estamos buscando o melhor mapeamento de função da variável de saída através das variáveis de entrada. Deste modo, um modelo supervisionado de ML é uma função aproximada de um determinado domínio, construída através dos dados de treinamento. Como se trata de uma função aproximada, o modelo pode apresentar erros em suas previsões. Esses erros podem ser divididos em três partes: irredutível, viés e variância (KOHAVI et al., 1996; LUXBURG; SCHÖLKOPF, 2011).

Independentemente da técnica que se utiliza, o erro irredutível não pode ser reduzido através de um modelo que generaliza bem. Esse erro refere-se a fatores desconhecidos que influenciam o mapeamento das variáveis de entrada para as de saída, como a inclusão de variáveis irrelevantes, a falta de variáveis relevantes ou o próprio ruído nos dados (BROWNLEE, 2016).

O viés é a diferença entre a predição média do modelo de ML e o valor correto que se deseja prever. Deste modo, o viés representa as suposições feitas pelo modelo com o intuito de facilitar o aprendizado da função aproximada (BROWNLEE, 2016). O nível do viés pode ser determinado de acordo com as suposições, sendo:

- **Baixo Viés**: Menores suposições sobre a forma da função aproximada;
- **Alto Viés**: Maiores suposições sobre a forma da função aproximada.

Normalmente, os algoritmos de ML paramétricos possuem um alto viés, sendo rápidos para treinar e fáceis de entender. Apesar disso, esses algoritmos são menos flexíveis e possuem desempenho preditivo menor em problemas complexos que não atendem às suas suposições (BROWNLEE, 2016).

Já a variância representa o espalhamento dos dados nas previsões. Deste modo, a variância representa o quanto o modelo vai se alterar para uma base de treinamento diferente (BROWNLEE, 2016). O nível da variância pode ser determinado através das especificidades dos dados de treinamento, sendo:

- **Baixa Variância**: Alterações nos dados de treinamento alteram insignificativamente a função aproximada;
- **Alta Variância**: Alterações nos dados de treinamento alteram significativamente a função aproximada.

Normalmente, os modelos de ML não paramétricos que têm muita flexibilidade possuem uma alta variância e, por este motivo, tendem a não generalizar bem a dados nunca vistos antes (BROWNLEE, 2016). Em um cenário ideal, um modelo de ML deveria ser bom em identificar um mapeamento oculto entre as variáveis de entrada e as de saída.

A Figura 22, extraída e adaptada de Gudivada et al. (2017), apresenta a diferença entre o baixo e alto viés e a baixa e alta variância. O centro de cada alvo representa um modelo capaz de realizar as previsões perfeitamente. Modelos que apresentam alto viés e baixa variância sofrem de subajuste, enquanto modelos que apresentam baixo viés e alta variância sofrem de sobreajuste.

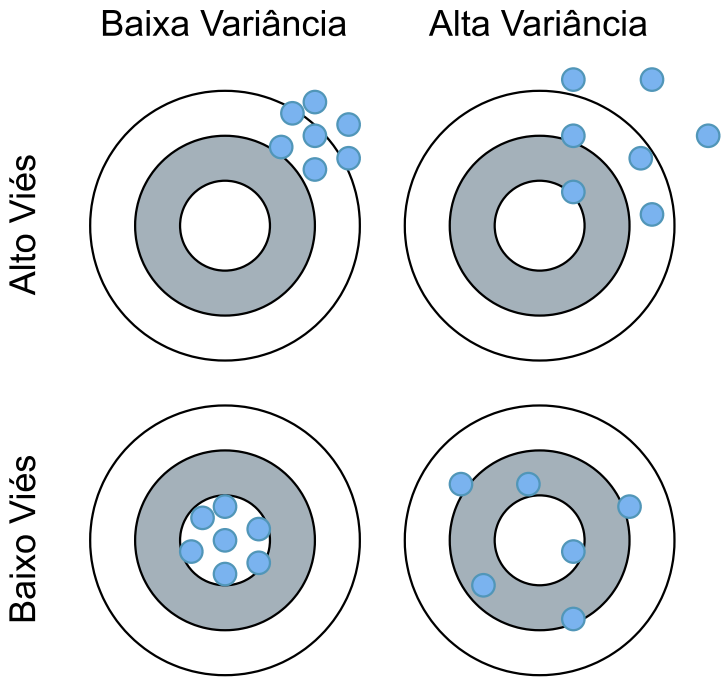
4.4.1 Dilema Viés-Variância

Conforme descrito sobre viés e variância, deseja-se encontrar um modelo de ML que tenha ao mesmo tempo um baixo viés e uma baixa variância (FRIEDMAN et al., 2001; JAMES et al., 2013). Devido a isso, existe um dilema na relação entre viés e variância no ML, que pode ser resumido como:

- O aumento do viés vai diminuir a variância;
- O aumento da variância vai diminuir o viés.

A escolha do algoritmo de ML, juntamente com a escolha dos seus hiperparâmetros, influencia diretamente os valores do viés e da variância. Os modelos de ML são aproximações

Figura 22 – Representação do baixo e alto viés e da baixa e alta variância.



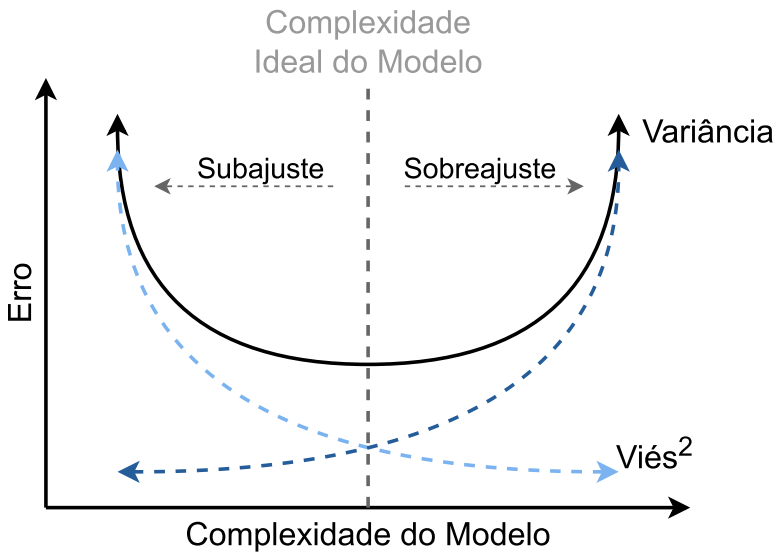
Fonte: Extraída e adaptada de Gudivada et al. (2017).

e não são capazes de prever exatamente os valores verdadeiros. Por esse motivo, não é possível calcular exatamente os erros de viés e variância. No entanto, é possível estimar esses erros usando diferentes técnicas e avaliar o desempenho do modelo com base nessas estimativas (GEMAN et al., 1992; BISHOP, 2006; JAMES et al., 2013).

A Figura 23, extraída e adaptada de Friedman et al. (2001) e James et al. (2013), apresenta o dilema entre viés e variância. Em um cenário onde não existe o erro irredutível, podemos observar que o erro de um modelo é a soma do viés ao quadrado e da variância.

Quanto maior a complexidade do modelo, maior a chance de sobreajuste (alta variância e baixo viés), enquanto quanto menor a complexidade, maior a chance de subajuste (baixa variância e alto viés). É possível observar que existe um nível de complexidade ótima, onde o erro é mínimo, gerando um modelo com uma boa generalização.

Figura 23 – Dilema entre Viés e Variância.



Fonte: Extraída e adaptada de Friedman et al. (2001) e James et al. (2013).

4.5 AVALIAÇÃO DE MODELOS

Como apresentado anteriormente, ao treinar um modelo de ML, buscamos encontrar o modelo que melhor generalize para dados nunca vistos antes, ou seja, para um conjunto de dados diferente do utilizado para treinamento. Para isso, precisamos definir as métricas de desempenho dos modelos e as diferentes formas de dividir a base de dados para realizar o estudo. As subseções seguintes apresentam as métricas e os particionamentos mais comuns.

4.5.1 Métricas de Desempenho

As métricas de desempenho são medidas utilizadas para avaliar o desempenho de um modelo de ML. Elas podem ser divididas, nesta tese, em duas categorias principais: métricas de classificação, que são utilizadas para avaliar modelos de classificação, e métricas de regressão, que são utilizadas para avaliar modelos de regressão, como os próprios nomes indicam.

4.5.1.1 Métricas de classificação

Dentro do contexto da classificação, deparamo-nos com duas abordagens principais: a classificação binária e a multiclasse. Na classificação binária, a tarefa envolve separar os dados em duas categorias distintas, enquanto na classificação multiclasse, lidamos com três classes ou mais. Embora esses enfoques compartilhem semelhanças, eles diferem na forma como as métricas de desempenho são calculadas.

4.5.1.1.1 Classificação binária

Para começar a descrever as métricas de classificação binária, começamos primeiramente detalhando a matriz de confusão, uma tabela que resume os resultados de uma classificação. Esta tabela é utilizada para realizar a comparação da classe prevista pelo modelo com a classe verdadeira para cada exemplo de teste. A matriz de confusão é construída da seguinte maneira para problemas de classificação binária:

		Classe esperada		
		Negativo	Positivo	Total
Classe prevista	Negativo	TN	FN	$TN + FN$
	Positivo	FP	TP	$FP + TP$
	Total	$TN + FP$	$FN + TP$	N

onde temos:

- Verdadeiro Positivo (*True Positive* - TP): são todos os exemplos nos quais o modelo foi capaz de prever corretamente a classe positiva.
- Falso Positivo (*False Positive* - FP): são todos os exemplos nos quais o modelo previu de forma incorreta a classe positiva.
- Falso Negativo (*False Negative* - FN): são todos os exemplos nos quais o modelo previu de forma incorreta a classe negativa.
- Verdadeiro Negativo (*True Negative* - TN): são todos os exemplos nos quais o modelo foi capaz de prever corretamente a classe negativa.
- $N = TP + TN + FP + FN$.

Desta forma, por meio dessa matriz de confusão, conseguimos avaliar a taxa de acerto e erro do modelo em cada classe do problema estudado. A partir da matriz de confusão, conseguimos calcular algumas métricas de desempenho para classificação binária (SOKOLOVA; LAPALME, 2009; POWERS, 2011), tais como:

- Acurácia (*Accuracy*): a porcentagem de exemplos que foram classificados corretamente pelo modelo, conforme Equação 4.1.

$$Acurácia = \frac{TP + TN}{N} \tag{4.1}$$

- Precisão (*Precision*): a proporção de exemplos positivos que foram classificados corretamente pelo modelo, conforme Equação 4.2.

$$Precisão = \frac{TP}{TP + FP} \tag{4.2}$$

- Revocação (*Recall*): a proporção de exemplos positivos que foram corretamente identificados pelo modelo, conforme Equação 4.3.

$$Revocação = \frac{TP}{TP + FN} \tag{4.3}$$

- F1-*score*: uma métrica que combina precisão e revocação através de uma média harmônica entre elas, conforme Equação 4.4. Bastante utilizada quando o desempenho do modelo em ambas as classes é importante.

$$F1\text{-}score = 2 * \frac{Precisão * Revocação}{Precisão + Revocação} \tag{4.4}$$

É possível extrair medidas mais complexas da matriz de confusão, como explorado por Powers (2011), mas definimos os principais da literatura e os que foram utilizados neste trabalho.

Outra métrica utilizada neste trabalho é a *Log Loss* (SCHULER, 2002), uma métrica utilizada para medir a "distância" entre as previsões do classificador e os rótulos verdadeiros. A equação pode ser escrita conforme Equação 4.5.

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \tag{4.5}$$

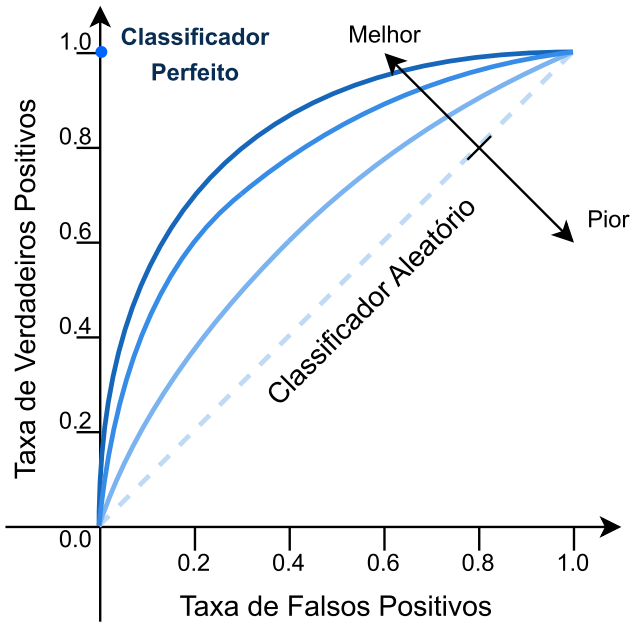
onde n representa o número de amostras analisadas, y_i é o rótulo verdadeiro da i-ésima amostra (0 ou 1 para classificação binária) e p_i é a probabilidade prevista pelo classificador para a i-ésima amostra. Quanto menor o valor apresentado pelo *Log Loss*, melhor é o desempenho do modelo para realizar as previsões. Desta forma, valores próximos de 0 indicam que o modelo está realizando previsões precisas, enquanto valores maiores indicam o inverso.

Todas as métricas apresentadas estão intrinsecamente ligadas à previsão da classe. No entanto, é importante notar que a maioria dos modelos também tem a capacidade de fornecer uma medida de probabilidade ou confiança associada à sua previsão. A decisão de classificação é geralmente feita ao aplicar um limiar de probabilidade, comumente estabelecido em 0.5, onde valores menores do que este limiar são definidos como a classe 0 e valores maiores como a classe 1. No entanto, é relevante destacar que ajustar esse limiar pode ter um impacto significativo nas previsões resultantes, e consequentemente, nas métricas discutidas anteriormente.

Uma forma de analisar um determinado modelo sem definir o limiar de probabilidade é utilizando a Curva Característica Operacional do Receptor (*Receiver Operating Characteristic* - ROC) (FAWCETT, 2006) ou Curva ROC. A Curva ROC é uma representação gráfica que apresenta o desempenho de um determinado modelo de classificação em diferentes limiares. Ela exibe no eixo Y a taxa de verdadeiros positivos ($TP/(TP + FN)$)

e no eixo X a taxa de falsos positivos ($FP/(FP+TN)$). A Figura 24 apresenta a visualização da curva ROC, onde o melhor modelo se aproxima do canto superior esquerdo, representando uma alta taxa de verdadeiros positivos e uma baixa taxa de falsos positivos. Enquanto isso, uma linha diagonal representa o desempenho de uma escolha aleatória do modelo.

Figura 24 – Visualização da Curva ROC.



Fonte: Elaborada pelo autor (2023).

Para quantificar a qualidade de uma curva ROC, utiliza-se a Área sob a Curva ROC (*Area Under an ROC Curve* - ROC AUC) (HANLEY; MCNEIL, 1982; BRADLEY, 1997). Como o próprio nome indica, esta métrica mede a área sob a curva ROC, refletindo a capacidade de um modelo distinguir entre as classes positivas e negativas, independentemente do limiar de probabilidade. Uma ROC AUC de 0.5 representa um modelo aleatório, enquanto valores próximos de 1 indicam classificadores melhores, próximos do modelo perfeito.

4.5.1.1.2 Classificação multiclasse

De maneira similar, para começar a descrever as métricas de classificação multiclasse, começamos primeiramente detalhando a matriz de confusão para K classes, uma tabela que resume os resultados de uma classificação com múltiplas classes. Esta tabela é utilizada para realizar a comparação da classe prevista pelo modelo com a classe verdadeira para cada exemplo de teste. A matriz de confusão é construída da seguinte maneira para problemas de classificação multiclasse:

		Classe esperada			
		Classe 1	Classe 2	...	Classe K
Classe prevista	Classe 1	T_{11}	T_{12}	...	T_{1K}
	Classe 2	T_{21}	T_{22}	...	T_{2K}
	...	...	...	...	...
	Classe K	T_{K1}	T_{K2}	...	T_{KK}

onde temos os elementos T_{ij} representando a contagem de observações em que a classe verdadeira é a i-ésima classe e a classe prevista é a j-ésima classe. Isso possibilita a identificação das correspondências corretas quando $i = j$, bem como das discrepâncias quando $i \neq j$.

Desta forma, por meio dessa matriz de confusão, conseguimos avaliar a taxa de acerto e erro do modelo em cada classe do problema estudado. A partir da matriz de confusão, similar a classificação binária, conseguimos calcular algumas métricas de desempenho para classificação multiclasse, tais como:

- Acurácia (*Accuracy*): a porcentagem de exemplos que foram classificados corretamente pelo modelo, conforme Equação 4.6.

$$Acurácia = \frac{\sum_{i=1}^{K} T_{ii}}{\sum_{i=1}^{K} \sum_{j=1}^{K} T_{ij}} \tag{4.6}$$

- Precisão$_i$ (*Precision*$_i$): a proporção de exemplos positivos que foram classificados corretamente pelo modelo para a classe i, conforme Equação 4.7.

$$Precisão_i = \frac{T_{ii}}{\sum_{j=1}^{K} T_{ji}} \tag{4.7}$$

- Revocação$_i$ (*Recall*$_i$): a proporção de exemplos positivos que foram corretamente identificados pelo modelo para a classe i, conforme Equação 4.8.

$$Revocação_i = \frac{T_{ii}}{\sum_{j=1}^{K} T_{ij}} \tag{4.8}$$

- F1-*score*$_i$: uma métrica que combina precisão e revocação através de uma média harmônica entre elas, conforme Equação 4.9.

$$F1\text{-}score_i = 2\frac{Precisão_i * Revocação_i}{Precisão_i + Revocação_i} \tag{4.9}$$

Para calcular a Precisão, Revocação e F1-score, realizamos o cálculo da média não ponderada das métricas. Deste modo, temos as Equações 4.10, 4.11 e 4.12.

- Precisão (*Precision*):

$$Precisão = \sum_{i=1}^{K} Precisão_i \tag{4.10}$$

- Revocação (*Recall*):

$$Revocação = \sum_{i=1}^{K} Revocação_i \tag{4.11}$$

- F1-*score*:

$$F1\text{-}score = \sum_{i=1}^{K} F1\text{-}score_i \tag{4.12}$$

Existem outras maneiras de se realizar o cálculo destas métricas, como a média ponderada ou até mesmo calculando as métricas globalmente contando o total de verdadeiros positivos, falsos negativos e falsos positivos. Apesar disso, este trabalho adotou a média não ponderada para representar os resultados.

Novamente, podemos definir a métrica *Log Loss* (SCHULER, 2002), onde a principal diferença é que, para a classificação binária, temos apenas uma probabilidade prevista para a classe positiva, enquanto na classificação multiclasse, temos uma probabilidade prevista para cada classe. A equação pode ser escrita conforme a Equação 4.13.

$$LogLoss = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{K} y_{ij} \log(p_{ij}) \tag{4.13}$$

onde n representa o número de amostras analisadas, K é o número de classes, y_{ij} é o rótulo verdadeiro de que a i-ésima amostra pertence à classe j e p_{ij} é a probabilidade de que a i-ésima amostra pertence à classe j. Quanto menor o valor apresentado pelo *Log Loss*, melhor é o desempenho do modelo para realizar as previsões. Desta forma, valores próximos de 0 indicam que o modelo está realizando previsões precisas, enquanto valores maiores indicam o inverso.

Por fim, para a métrica ROC AUC, é necessário realizar algumas adaptações para o seu cálculo, existindo então duas abordagens principais:

- Um-contra-Um (*One-vs-One* - OvO): Calcula a média da ROC AUC de todas as combinações possíveis em pares de classes (HAND; TILL, 2001).
- Um-contra-Todos (*One-vs-Rest* - OvR): Calcula a ROC AUC de cada classe em relação às demais (FAWCETT, 2006).

Neste trabalho, adotamos o uso do cálculo OvR, realizando por fim o cálculo da média não ponderada das métricas para a sumarização do valor final.

4.5.1.2 Métricas de regressão

Aqui descrevemos algumas das métricas de regressão que são mais comuns e que foram utilizadas dentro deste trabalho. Elas são:

- Erro Quadrático Médio (*Mean Squared Error* - MSE) (BROWNLEE, 2011): é calculado através da média dos erros quadrados gerados pela diferença entre os valores preditos pelo modelo de regressão e os valores reais. Desta forma, a métrica mede o quão próximos os valores preditos estão dos valores reais, e valores próximos de zero são desejáveis. A equação pode ser escrita pela Equação 4.14.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.14}$$

  onde n representa o número de observações, $\hat{y}_i$ representa o i-ésimo valor predito pelo modelo e y_i representa o i-ésimo valor real.

- Raiz do Erro Quadrático Médio (*Root Mean Square Error* - RMSE) (KOEHLER; SPRENT, 1996): é calculada tirando a raiz quadrada da métrica MSE, conforme Equação 4.15. É bastante utilizada, pois expressa o erro na unidade do próprio problema, enquanto a MSE apresenta a medida ao quadrado. Além disso, o RMSE acaba sendo mais robusto para problemas que apresentam observações mais extremas, categorizadas como *outliers*, pois o mesmo penaliza os erros maiores de forma mais agressiva.

$$RMSE = \sqrt{MSE} \tag{4.15}$$

- Coeficiente de Determinação (*Coefficient of Determination* - R^2) (GARSON, 1991): é uma métrica que tem como objetivo medir o quão bem o modelo consegue explicar os dados. Para isso, ele mede o quanto da variância dos dados é explicado pelo modelo treinado, conforme Equação 4.16. O seu valor pode ser interpretado como a porcentagem da variância dos dados que o modelo consegue explicar.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.16}$$

  onde n representa o número de observações, $\bar{y}$ representa o valor médio dos valores reais, $\hat{y}_i$ representa o i-ésimo valor predito pelo modelo e y_i representa o i-ésimo valor real. SS_{res} está representando o erro quadrático residual enquanto SS_{tot} representa a variância total dos dados.

4.5.2 Validação Cruzada

A validação cruzada é uma estratégia estatística que consiste em realizar a divisão dos seus dados em conjuntos com o objetivo de realizar a avaliação do desempenho dos modelos de ML (KOHAVI et al., 1995). A estratégia pode ser utilizada para realizar a estimação da performance, comparar diferentes modelos e/ou ajudar a obter os melhores hiperparâmetros.

A seguir, detalhamos os dois tipos de validação cruzada que são utilizados neste trabalho, apesar de existirem outros tipos na literatura (KELLEHER et al., 2015).

4.5.2.1 Separação

A validação cruzada por Separação (*Holdout*) (KOHAVI et al., 1995) é a estratégia mais simples e a mais utilizada em ML. Ela consiste em dividir os dados em dois conjuntos disjuntos: um conjunto de treinamento e outro conjunto de teste. Como os próprios nomes indicam, um destes conjuntos é utilizado para treinar o modelo e, após o treinamento, o conjunto de teste é utilizado para medir a performance.

Apesar de ser simples de utilizar, essa estratégia tende a ser menos precisa do que as outras técnicas de validação, visto que a avaliação é feita em apenas um conjunto de teste. Em contrapartida, devido ao fato de o treinamento e avaliação ocorrerem apenas uma vez, o tempo computacional dessa técnica é o menor entre todas as outras estratégias possíveis.

4.5.2.2 *K-fold*

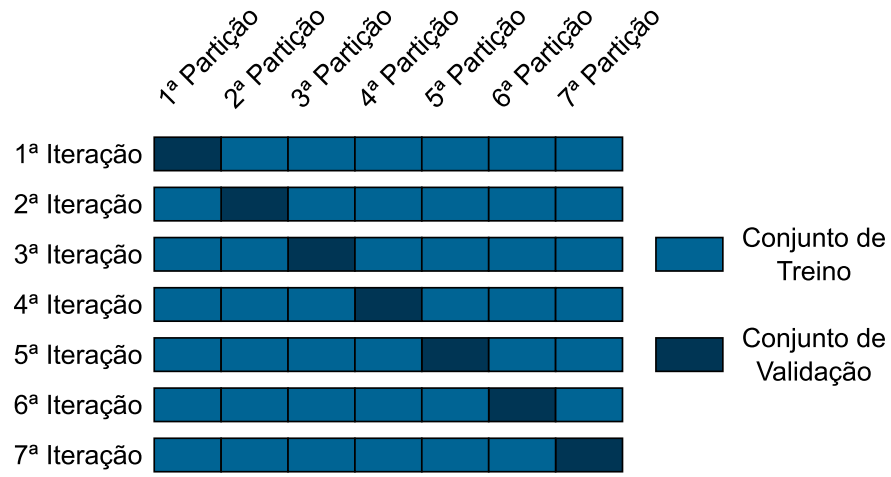
A validação cruzada *K-fold* (JAMES et al., 2013) consiste em realizar divisões aleatórias do conjunto de dados em K partições de tamanhos aproximadamente iguais. Em cada iteração do processo, uma dessas partições é selecionada como o conjunto de validação, enquanto as outras $K-1$ partições são utilizadas como o conjunto de treinamento. Desta forma, conseguimos realizar esse processo K vezes, alterando a cada iteração qual partição é utilizada como o conjunto de validação, onde o modelo é treinado com $K-1$ partições e avaliado na partição restante.

Ao final do processo, é possível obter K métricas de qualidade do modelo, uma para cada iteração. A média dessas métricas é então calculada e utilizada como uma estimativa do desempenho do modelo em dados novos. Ao utilizar a validação cruzada *K-fold*, é possível aproveitar todo o conjunto de observações para treinar e validar o modelo, o que aumenta a confiabilidade da avaliação. A Figura 25 exemplifica a estratégia *K-fold* com $K = 7$.

A validação cruzada *K-fold* possui o K como um hiperparâmetro que deve ser definido pelo usuário, mas não existe uma regra definida para a escolha. Quando definimos o valor de K como sendo igual à quantidade de dados presentes no conjunto de observações, obtemos um caso especial da estratégia chamada Deixar Um De Fora (*Leave One Out* - LOO) (STONE, 1974).

Assim como avaliado na seção anterior, a escolha do valor de K tem um impacto direto no tempo de computação necessário para treinar e avaliar o modelo, uma vez que o processo de treinamento e avaliação é realizado K vezes. Quanto maior o valor de K, maior será o número de iterações e, consequentemente, maior será o tempo computacional. Por outro lado, quanto menor o valor de K, menor será o número de iterações e, consequentemente, menor será o tempo computacional. É extremamente importante, neste caso, considerar o dilema entre o tempo computacional e a qualidade da avaliação ao escolher o

Figura 25 – Exemplificação da estratégia *K-fold* com $K = 7$.



Fonte: Elaborada pelo autor (2023).

valor de K.

Outra questão é que quanto maior o valor de K, menor é o valor do viés da técnica (KUHN; JOHNSON, 2013). Já em relação à variância, o valor de K pode ou não ter influência direta, dependendo de diversos fatores, como o tamanho e a complexidade da base de dados, o algoritmo de aprendizado utilizado e outros aspectos (EFRON, 1983; KOHAVI et al., 1995; BENGIO; GRANDVALET, 2004; ZHANG; YANG, 2015).

De acordo com a pesquisa apresentada por James et al. (2013), valores de K iguais a 5 ou 10 têm sido empiricamente observados para produzir níveis de viés e variância aceitáveis em muitos casos, o que os torna possíveis opções de escolha para a validação cruzada. No entanto, é importante lembrar que esses valores podem não ser ideais em todas as situações e que é necessário avaliar cuidadosamente todos os fatores antes de tomar uma decisão sobre qual valor de K utilizar.

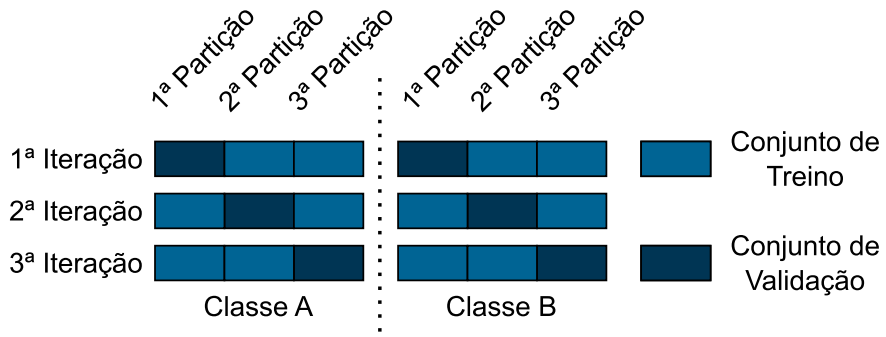
4.5.2.3 Estratificada *K-fold*

A validação cruzada Estratificada *K-fold* (*Stratified K-fold*) (KOHAVI et al., 1995) é uma variação da validação cruzada *K-fold* utilizada para problemas de classificação. A técnica funciona criando todas as partições de forma estratificada, o que significa que a proporção de diferentes classes em cada partição é similar à proporção dessas mesmas classes no conjunto de dados completo.

A Figura 26 exemplifica a estratégia Estratificada *K-fold* com $K = 3$ e duas classes A e B. Podemos exemplificar a classe A como sendo a classe dos exemplos positivos, que representam 30% de todos os dados, enquanto a classe B representa a classe dos exemplos negativos, que representam os outros 70%. Neste caso, cada partição vai apresentar a

mesma proporção quando juntamos as partições por classe em uma única partição.

Figura 26 – Exemplificação da estratégia Estratificada *K-fold* com $K = 3$ e duas classes.



Fonte: Elaborada pelo autor (2023).

Esta estratégia ajuda a garantir que o modelo seja treinado e avaliado em amostras que sejam representativas em relação aos dados reais, o que pode ser especialmente importante quando o conjunto de dados é desequilibrado.

5 REDES NEURAIS ARTIFICIAIS

As Redes Neurais Artificiais (*Artificial Neural Network* - ANN) (BISHOP et al., 1995) constituem um tipo de modelo matemático inspirado na estrutura das redes neurais do sistema nervoso humano e animal. As ANNs têm a capacidade de aproximar funções não lineares por meio de seu processo de aprendizado, sendo amplamente empregadas em diversas aplicações. A seguir, é apresentada uma breve história do início das ANNs e, posteriormente, aborda-se a evolução dessa técnica até o estágio atual deste trabalho.

McCulloch e Pitts (1943) desenvolveram um trabalho fundamental no campo da AI e no estudo de como um cérebro opera, sendo o primeiro trabalho a descrever um modelo artificial e matemático de um neurônio biológico. No referido estudo, é apresentado um modelo para a maneira pela qual o cérebro processa informações e toma decisões, sugerindo que o cérebro é composto por neurônios simples que se interconectam para formar sistemas mais complexos. Esses neurônios simples podem ser compreendidos como unidades de processamento lógico capazes de realizar operações básicas, como os operadores lógicos "E", "OU" e "NÃO".

Este modelo propunha que esses neurônios simples poderiam ser representados matematicamente como dígitos binários (0 ou 1) e que suas conexões também poderiam ser representadas assim. Dessa forma, a saída de um neurônio seria determinada pelas entradas que ele recebe de outros neurônios, juntamente com um valor limiar. Retornaríamos 1 se o valor somado recebido como entrada fosse maior que o limiar; caso contrário, retornaríamos 0. Esse modelo, apesar de simples, pode ser (e foi) estendido para redes de neurônios mais complexas, permitindo a representação de operações lógicas mais complexas e o processamento de informações mais elaboradas.

As duas seções a seguir apresentam a evolução deste modelo proposto, começando com o Perceptron e continuando com as Redes Perceptron Multicamadas.

5.1 Perceptron

Como evolução do modelo proposto anteriormente, Rosenblatt (1958) propuseram uma nova abordagem para a solução do problema de reconhecimento de padrões, denominada *Perceptron*. Assim como o modelo proposto por McCulloch e Pitts (1943), o *Perceptron* também é composto por neurônios simples que podem ser conectados entre si para formar uma rede e também podem ser vistos como unidades lógicas capazes de realizar operações básicas, como "E", "OU" e "NÃO".

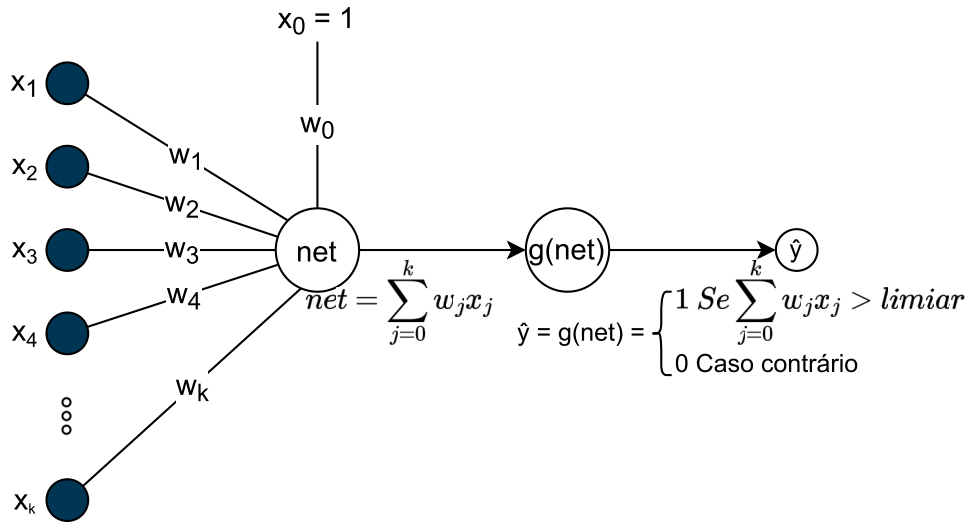
Como contribuição, o *Perceptron* introduziu o conceito de pesos (*weights*) e tendência (*bias*) nas conexões dos neurônios. Esses pesos e tendência podem ser ajustados para modificar o comportamento da rede, permitindo que ela seja capaz de realizar operações cada vez mais complexas. Desta forma, a saída de um neurônio no *Perceptron* é obtida

por meio dos dados de entrada, dos pesos e da tendência das conexões, além de um valor limiar e da função $g(net)$, definida conforme a Equação 5.1.

$$g(\sum_{j=0}^{k} w_j x_j) = g(net) = \begin{cases} 1 & \text{se } net > \text{limiar} \\ 0 & \text{caso contrário} \end{cases} \tag{5.1}$$

onde w_0 representa a tendência, w_1 até w_k representam os pesos, x_j representa o j-ésimo valor de entrada e k representa a quantidade de valores de entrada. Se a soma das entradas multiplicadas pelos pesos mais a tendência, representado por net, for maior que o valor limiar, o neurônio produzirá uma saída 1. Se a soma for menor ou igual que o valor limiar, o neurônio produzirá uma saída 0. A Figura 27 apresenta o modelo *Perceptron* descrito.

Figura 27 – Modelo *Perceptron*.



Fonte: Elaborada pelo autor (2023).

Considerando que os pesos e a tendência foram definidos previamente (geralmente inicializados aleatoriamente), para ajustar seus valores, o *Perceptron* realiza o cálculo do erro considerando o conjunto de treinamento para cada instância i. O erro quadrático é utilizado como a função de perda, pois fornece uma função convexa no processo de otimização (MELLO; PONTI, 2018). Apesar disso, outras funções de perda podem ser empregadas, se necessário. Desta forma, o *Perceptron* busca, ao adaptar os pesos e a tendência, encontrar a função de aproximação linear capaz de conduzir a função objetivo para o seu mínimo. A Equação 5.2 define a função objetivo que o *Perceptron* deseja minimizar, considerando o erro quadrático e $f(\mathbf{x})$ como o modelo.

$$E^2 = \sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - g(net_i))^2 = \sum_{i=1}^{n}(y_i - g(\sum_{j=0}^{k} w_j x_{i,j}))^2 \tag{5.2}$$

Tendo em vista que o gradiente aponta na direção da subida mais íngreme e sua magnitude é a taxa de variação da função nessa direção (BURGES et al., 2010), podemos considerar seu valor negativo como indicativo da direção da descida mais íngreme, conhecido como o método Gradiente Descendente (*Gradient Descent* - GD). Desta forma, podemos estabelecer o seguinte processo para atualização dos valores dos pesos e da tendência, com o objetivo de minimizar E^2. Este processo pode ser descrito pela Equação 5.3.

$$w_j(t+1) = w_j(t) - \eta \frac{\partial E^2}{\partial w_j} \tag{5.3}$$

onde $w_j(t)$ e $w_j(t+1)$ representam o valor de w_j no tempo t e $t+1$, respectivamente, e η denota a taxa de aprendizado.

Para facilitar o processo de diferenciação, Rosenblatt (1958) realizou uma relaxação na função $f(\mathbf{x}_i)$ para o processo de atualização, passando da Equação 5.4 para a Equação 5.5.

$$f(\mathbf{x}_i) = g(net_i) = g(\sum_{j=0}^{k} w_j x_{i,j}) \tag{5.4}$$

$$f(\mathbf{x}_i) = net_i = \sum_{j=0}^{k} w_j x_{i,j} \tag{5.5}$$

Desta forma, a derivada parcial pode ser encontrada através da regra da cadeia, conforme apresenta a Equação 5.6, e a atualização dos parâmetros do modelo pode ser feita da seguinte conforme apresenta a Equação 5.7.

$$\frac{\partial E^2}{\partial w_j} = 2\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))\frac{\partial [y_i - f(\mathbf{x}_i)]}{\partial w_j} = 2\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))(-x_{i,j}) \tag{5.6}$$

$$w_j(t+1) = w_j(t) - \eta\frac{\partial E^2}{\partial w_j} = w_j(t) - \eta 2\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))(-x_{i,j}) \tag{5.7}$$

É importante destacar que, ao atualizar o valor da tendência, temos, para todas as instâncias i, o valor de $x_{i,0} = 1$. Essas equações são responsáveis pelo processo de aprendizado do modelo *Perceptron*.

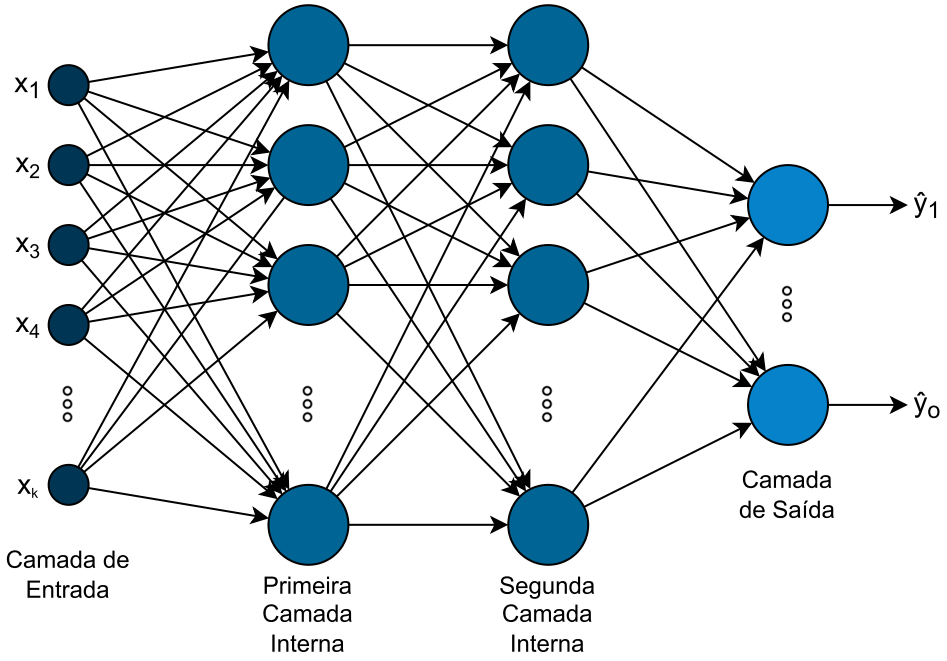
Apesar de ser um modelo poderoso para solucionar problemas que sejam linearmente separáveis, Minsky e Papert (1969) apresentaram em seu livro a capacidade limitada do algoritmo *Perceptron*. Por meio de análise matemática, eles demonstraram que o *Perceptron* não pode aprender a classificar corretamente problemas que não sejam linearmente separáveis, independentemente do número de exemplos de treinamento fornecidos. Isso levou a uma considerável perda de interesse pelo estudo de ANNs na época, mas a pesquisa

em ANNs continuou e, com o tempo, outros algoritmos foram desenvolvidos para superar essa limitação do *Perceptron*.

5.2 Redes Perceptron Multicamadas

Como evolução do *Perceptron*, surgiram as Redes Perceptron Multicamadas (*Multi-Layer Perceptron* - MLP) (HAYKIN, 1994; BISHOP et al., 1995), uma rede que consiste de pelo menos três camadas: uma camada de entrada, uma ou mais camadas internas e, por fim, uma camada de saída. A Figura 28 apresenta a arquitetura de um modelo MLP com duas camadas internas.

Figura 28 – Arquitetura de um MLP com duas camadas internas.



Fonte: Elaborada pelo autor (2023).

O MLP é um tipo de rede de Alimentação Direta (*Feedforward*), ou seja, cada neurônio de uma camada se conecta com todos os outros neurônios da próxima camada, sem existirem caminhos de volta. Desta forma, todas as conexões na rede têm a mesma direção, indo da camada de entrada até a camada de saída.

Similar ao *Perceptron*, cada neurônio presente na arquitetura recebe como entrada a saída da camada anterior multiplicada pelos pesos e tendência. Esses valores são então somados e passam por uma Função de Ativação, que exploraremos com mais detalhes na seção 5.3. O valor gerado por essa função é, então, propagado na arquitetura, servindo como entrada para a camada seguinte, até alcançar a camada de saída. Para problemas de regressão, a camada de saída representa a predição final, enquanto para problemas de

classificação, a camada de saída pode produzir uma probabilidade ou pontuação (*score*) para cada classe, sendo que a classe com o maior valor é selecionada como a previsão.

É fundamental destacar que os pesos representam a força da conexão entre os neurônios. Valores de pesos que tendem a zero indicam que a informação vinda do neurônio anterior não traz informações relevantes, em um processo de otimização, para a predição dos neurônios subsequentes.

O interesse pelo estudo de ANNs voltou principalmente devido à capacidade do MLP de ser um aproximador universal. Cybenko (1989) demonstrou que, com um número adequado de neurônios na camada interna de uma rede, é possível aproximar qualquer mapeamento contínuo em uma região compacta do espaço. As camadas intermediárias de uma ANN funcionam como extratoras de características que, por meio de seus pesos, codificam as particularidades dos padrões de entrada. Dessa forma, a rede é capaz de criar sua própria representação do problema, permitindo uma aproximação universal. Ademais, as camadas internas do MLP acabam determinando o número de hiperplanos para dividir o espaço de entrada (MELLO; PONTI, 2018).

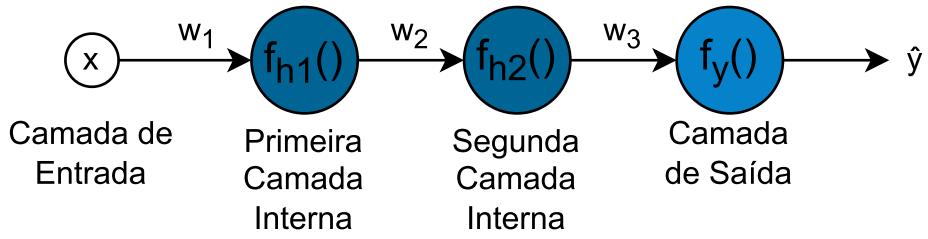
Da mesma maneira que atualizamos os pesos e as tendências no *Perceptron*, é necessário realizar esse processo para o MLP. A questão, que foi um dos principais desafios da área, consistiu em como propagar o erro da predição para os pesos e as tendências mais internos. Para abordar essa questão, Rumelhart et al. (1986) desenvolveram o algoritmo de Retropropagação (*Backpropagation*). Em linhas gerais, este algoritmo opera da seguinte forma:

1. Realizar uma inicialização (normalmente aleatória) dos pesos e tendências;
2. Realizar o processo de predição da rede com os dados de entrada para gerar as saídas previstas;
3. Calcular o erro entre a previsão da rede e os valores reais usando uma função ou métrica de perda;
4. Propagar o erro do modelo para trás através da rede, começando inicialmente pela camada de saída e indo em direção à camada de entrada. Para cada uma das camadas, usar o sinal do erro para ajustar os pesos e as tendências de forma a reduzir o erro geral.
5. Repetir as etapas de 2 até 4 para todas as instâncias de treinamento e por várias épocas (iterações de treinamento) até que a rede convirja para um critério de parada.

A etapa principal do algoritmo de Retropropagação é a 4^{a}, responsável pelo processo de propagação do erro da predição para trás, permitindo assim ajustar os pesos com base no quanto ele impacta na predição. Para demonstrar o processo de ajuste dos pesos,

adotamos uma arquitetura mais simples, presente na Figura 29. Esta arquitetura contém duas camadas internas, onde cada uma contém apenas um neurônio, e não existem os valores de tendência para ajustar. Isso pode ser generalizado considerando a tendência como um peso.

Figura 29 – Arquitetura exemplo para processo de ajuste dos pesos.



Fonte: Elaborada pelo autor (2023).

Considerando que o valor x representa a entrada, o valor h_1 denota a saída da primeira camada interna, h_2 representa a saída da segunda camada interna e, finalmente, $\hat{y}$ corresponde à saída da camada de saída. Além disso, consideramos f_{h1}, f_{h2} e f_y como funções de ativação diferenciáveis. Dessa forma, temos as Equações 5.8, 5.9 e 5.10.

$$h_1 = f_{h1}(x * w_1) \tag{5.8}$$

$$h_2 = f_{h2}(h_1 * w_2) = f_{h2}(f_{h1}(x * w_1) * w_2) \tag{5.9}$$

$$\hat{y} = f_y(h_2 * w_3) = f_y(f_{h2}(h_1 * w_2) * w_3) = f_y(f_{h2}(f_{h1}(x * w_1) * w_2) * w_3) \tag{5.10}$$

Dessa forma, de maneira similar ao *Perceptron*, é possível definir uma equação que se busca minimizar durante o processo de treinamento do MLP. Sendo então expressa a Equação 5.11 (trata-se de uma medida de perda, mas existem outras possíveis).

$$E^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i))^2 = \sum_{i=1}^{n}(y_i - f_y(f_{h2}(f_{h1}(x_i * w_1) * w_2) * w_3))^2 \tag{5.11}$$

Podemos atualizar os pesos seguindo, novamente, o mesmo processo que foi apresentado no *Perceptron*, que consiste em utilizar o GD, como mostrado na Equação 5.3.

Apesar das equações de atualização serem as mesmas para todos os pesos, independentemente da camada, as derivadas parciais são diferentes. Deste modo, para conseguirmos atualizar os valores, devemos definir qual é o resultado da derivada parcial

da equação E^2 para cada peso. Através da regra da cadeia, podemos encontrar todas as derivadas parciais de cada peso, sendo elas apresentadas nas Equações 5.12, 5.13 e 5.14.

$$\frac{\partial E^2}{\partial w_1} = \frac{\partial E^2}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1} \tag{5.12}$$

$$\frac{\partial E^2}{\partial w_2} = \frac{\partial E^2}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_2} \tag{5.13}$$

$$\frac{\partial E^2}{\partial w_3} = \frac{\partial E^2}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_3} \tag{5.14}$$

Os valores apresentados pela regra da cadeia nas três equações são expandidos nas Equações 5.15, 5.19, 5.22, 5.25, 5.28 e 5.31.

$$\frac{\partial E^2}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} \left(\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right) \tag{5.15}$$

$$= \sum_{i=1}^{n} \frac{\partial}{\partial \hat{y}} (y_i - \hat{y}_i)^2 \tag{5.16}$$

$$= \sum_{i=1}^{n} 2(y_i - \hat{y}_i)(-1) \tag{5.17}$$

$$= -2 \sum_{i=1}^{n} (y_i - \hat{y}_i) \tag{5.18}$$

$$\frac{\partial \hat{y}}{\partial w_3} = \frac{\partial f_y(h_2 * w_3)}{\partial w_3} \tag{5.19}$$

$$= \frac{\partial (h_2 * w_3)}{\partial w_3} \cdot \frac{\partial f_y(\xi)}{\partial \xi} |_{\xi = h_2 w_3} \tag{5.20}$$

$$= h_2 \cdot \frac{\partial f_y(\xi)}{\partial \xi} |_{\xi = h_2 w_3} \tag{5.21}$$

$$\frac{\partial \hat{y}}{\partial h_2} = \frac{\partial f_y(h_2 * w_3)}{\partial h_2} \tag{5.22}$$

$$= \frac{\partial (h_2 * w_3)}{\partial h_2} \cdot \frac{\partial f_y(\xi)}{\partial \xi} |_{\xi = h_2 w_3} \tag{5.23}$$

$$= w_3 \cdot \frac{\partial f_y(\xi)}{\partial \xi} |_{\xi = h_2 w_3} \tag{5.24}$$

$$\frac{\partial h_2}{\partial w_2} = \frac{\partial f_{h2}(h_1 * w_2)}{\partial w_2} \tag{5.25}$$

$$= \frac{\partial (h_1 * w_2)}{\partial w_2} \cdot \frac{\partial f_{h2}(\xi)}{\partial \xi} |_{\xi = h_1 * w_2} \tag{5.26}$$

$$= h_1 \cdot \frac{\partial f_{h2}(\xi)}{\partial \xi} |_{\xi = h_1 * w_2} \tag{5.27}$$

$$\frac{\partial h_2}{\partial h_1} = \frac{\partial f_{h2}(h_1 * w_2)}{\partial h_1} \tag{5.28}$$

$$= \frac{\partial (h_1 * w_2)}{\partial h_1} \cdot \frac{\partial f_{h2}(\xi)}{\partial \xi}|_{\xi = h_1 * w_2} \tag{5.29}$$

$$= w_2 \cdot \frac{\partial f_{h2}(\xi)}{\partial \xi}|_{\xi = h_1 * w_2} \tag{5.30}$$

$$\frac{\partial h_1}{\partial w_1} = \frac{\partial f_{h1}(x * w_1)}{\partial w_1} \tag{5.31}$$

$$= \frac{\partial (x * w_1)}{\partial w_1} \cdot \frac{\partial f_{h1}(\xi)}{\partial \xi}|_{\xi = x * w_1} \tag{5.32}$$

$$= x \cdot \frac{\partial f_{h1}(\xi)}{\partial \xi}|_{\xi = x * w_1} \tag{5.33}$$

onde $\partial f_{h1}(\xi)/\partial \xi$, $\partial f_{h2}(\xi)/\partial \xi$ e $\partial f_y(\xi)/\partial \xi$ são as derivadas parciais das funções de ativação f_{h1}, f_{h2} e f_y, respectivamente, em relação ao valor ξ de entrada. A seção 5.3 apresenta o conjunto de funções de ativação e suas derivadas em relação ao valor de entrada e aos parâmetros. Com todos os valores expandidos, chegamos nas Equações 5.34, 5.37 e 5.40.

$$\frac{\partial E^2}{\partial w_1} = \frac{\partial E^2}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1} \tag{5.34}$$

$$= -2 \sum_{i=1}^{n} (y_i - \hat{y}_i) \cdot w_3 \frac{\partial f_y(\xi)}{\partial \xi}|_{\xi = h_{2,i} w_3} \cdot w_2 \frac{\partial f_{h2}(\xi)}{\partial \xi}|_{\xi = h_{1,i} * w_2} \cdot x_i \frac{\partial f_{h1}(\xi)}{\partial \xi}|_{\xi = x_i * w_1} \tag{5.35}$$

$$= -2 \sum_{i=1}^{n} (x_i * w_2 * w_3) \cdot (y_i - \hat{y}_i) \cdot \frac{\partial f_y(\xi)}{\partial \xi}|_{\xi = h_{2,i} w_3} \cdot \frac{\partial f_{h2}(\xi)}{\partial \xi}|_{\xi = h_{1,i} * w_2} \cdot \frac{\partial f_{h1}(\xi)}{\partial \xi}|_{\xi = x_i * w_1} \tag{5.36}$$

$$\frac{\partial E^2}{\partial w_2} = \frac{\partial E^2}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_2} \tag{5.37}$$

$$= -2 \sum_{i=1}^{n} (y_i - \hat{y}_i) \cdot w_3 \frac{\partial f_y(\xi)}{\partial \xi}|_{\xi = h_{2,i} w_3} \cdot h_{1,i} \frac{\partial f_{h2}(\xi)}{\partial \xi}|_{\xi = h_{1,i} * w_2} \tag{5.38}$$

$$= -2 \sum_{i=1}^{n} (h_{1,i} * w_3) \cdot (y_i - \hat{y}_i) \cdot \frac{\partial f_y(\xi)}{\partial \xi}|_{\xi = h_{2,i} w_3} \cdot \frac{\partial f_{h2}(\xi)}{\partial \xi}|_{\xi = h_{1,i} * w_2} \tag{5.39}$$

$$\frac{\partial E^2}{\partial w_3} = \frac{\partial E^2}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_3} \tag{5.40}$$

$$= -2 \sum_{i=1}^{n} (y_i - \hat{y}_i) \cdot h_{2,i} \frac{\partial f_y(\xi)}{\partial \xi}|_{\xi = h_{2,i} w_3} \tag{5.41}$$

$$= -2 \sum_{i=1}^{n} (h_{2,i}) \cdot (y_i - \hat{y}_i) \cdot \frac{\partial f_y(\xi)}{\partial \xi}|_{\xi = h_{2,i} w_3} \tag{5.42}$$

Com estes valores calculados, conseguimos dar sequência na atualização dos pesos da rede. Isso ocorre porque a derivada parcial do erro em relação a cada um dos pesos da

rede nos indica o quanto aquele peso contribui para o erro nas previsões (RUMELHART et al., 1986). Desta maneira, o cálculo desta informação nos permite atualizar os pesos de forma a reduzir o erro.

5.3 Funções de Ativação

A Função de Ativação (*Activation Function* - AF) é um elemento fundamental das ANNs, pois ajuda no processo de aprendizagem e tomada de decisões com base nos dados de entrada. As AFs são utilizadas nos neurônios, tendo como entrada a soma ponderada de todas as entradas da camada anterior, e sua saída é enviada para a próxima camada da rede. As AFs desempenham um papel fundamental ao introduzir não linearidade no modelo, permitindo que ele seja capaz de representar espaços mais complexos.

Diferentes tipos de AFs são adequados para diferentes tipos de problemas, e a escolha da função correta pode afetar significativamente o desempenho do modelo (GLOROT; BENGIO, 2010). A seguir, listamos todas as AFs estudadas neste trabalho, incluindo funções de ativação comumente utilizadas na literatura e algumas apresentadas na revisão bibliográfica.

5.3.1 BLU

Nesta subseção, definimos a AF BLU, que recebe como entrada o valor x e os parâmetros α e β. Esses parâmetros possuem um valor padrão, normalmente utilizados na literatura, que é utilizado pela AF, sendo esses valores 0.5 e 0.5, respectivamente. A AF BLU é definida na Equação 5.43.

$$BLU(x, \alpha, \beta) = -\beta \left(\alpha - \sqrt{\alpha^2 + x^2}\right) + x \tag{5.43}$$

A Figura 30 apresenta a AF BLU com seus valores padrões à esquerda e, à direita, outra forma para diferentes valores de parâmetros.

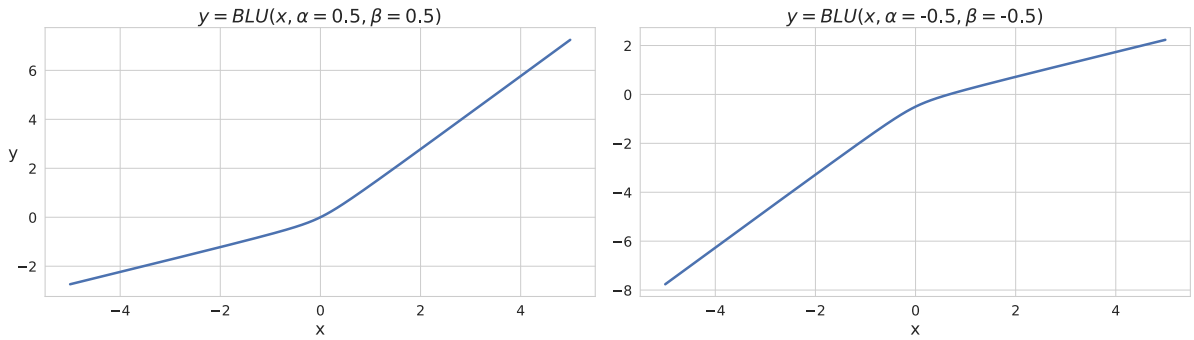
Além disso, temos as derivadas parciais da função apresentadas pelas Equações 5.44, 5.45 e 5.46.

$$\frac{\partial BLU(x, \alpha, \beta)}{\partial x} = \frac{\beta x}{\sqrt{\alpha^2 + x^2}} + 1 \tag{5.44}$$

$$\frac{\partial BLU(x, \alpha, \beta)}{\partial \alpha} = \frac{\alpha\beta}{\sqrt{\alpha^2 + x^2}} - \beta \tag{5.45}$$

$$\frac{\partial BLU(x, \alpha, \beta)}{\partial \beta} = -\alpha + \sqrt{\alpha^2 + x^2} \tag{5.46}$$

Figura 30 – Função de Ativação BLU com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

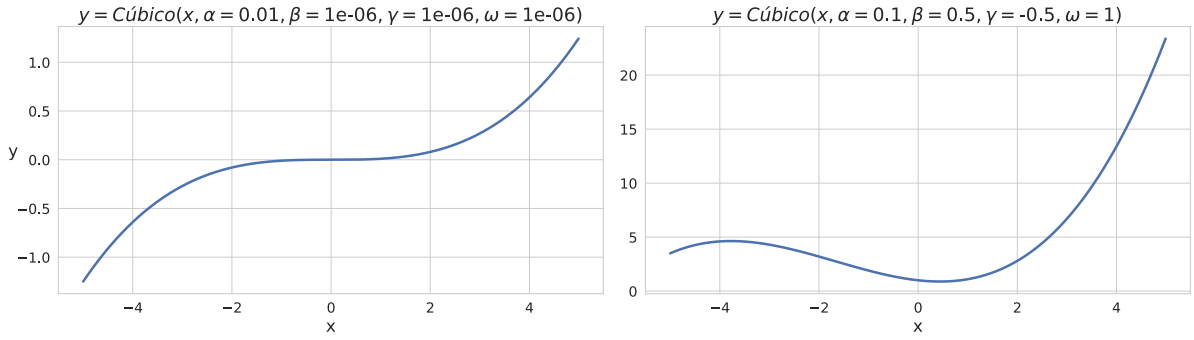
5.3.2 Cúbico

Nesta subseção, definimos a AF Cúbica, que recebe como entrada o valor x e os parâmetros α, β, γ e ω. Esses parâmetros possuem um valor padrão, normalmente utilizados na literatura, que é utilizado pela AF, sendo esses valores 0.01, 1×10^{-6}, 1×10^{-6} e 1×10^{-6}, respectivamente. A AF Cúbica é definida na Equação 5.47.

$$Cúbico(x, \alpha, \beta, \gamma, \omega) = \alpha x^3 + \beta x^2 + \gamma x + \omega \tag{5.47}$$

A Figura 31 apresenta a AF Cúbico com seus valores padrões à esquerda e, à direita, outra forma para diferentes valores de parâmetros.

Figura 31 – Função de Ativação Cúbico com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

Além disso, temos as derivadas parciais da função apresentadas pelas Equações

5.48, 5.49, 5.50, 5.51 e 5.52.

$$\frac{\partial Cúbico(x, \alpha, \beta, \gamma, \omega)}{\partial x} = 3\alpha x^2 + 2\beta x + \gamma \tag{5.48}$$

$$\frac{\partial Cúbico(x, \alpha, \beta, \gamma, \omega)}{\partial \alpha} = x^3 \tag{5.49}$$

$$\frac{\partial Cúbico(x, \alpha, \beta, \gamma, \omega)}{\partial \beta} = x^2 \tag{5.50}$$

$$\frac{\partial Cúbico(x, \alpha, \beta, \gamma, \omega)}{\partial \gamma} = x \tag{5.51}$$

$$\frac{\partial Cúbico(x, \alpha, \beta, \gamma, \omega)}{\partial \omega} = 1 \tag{5.52}$$

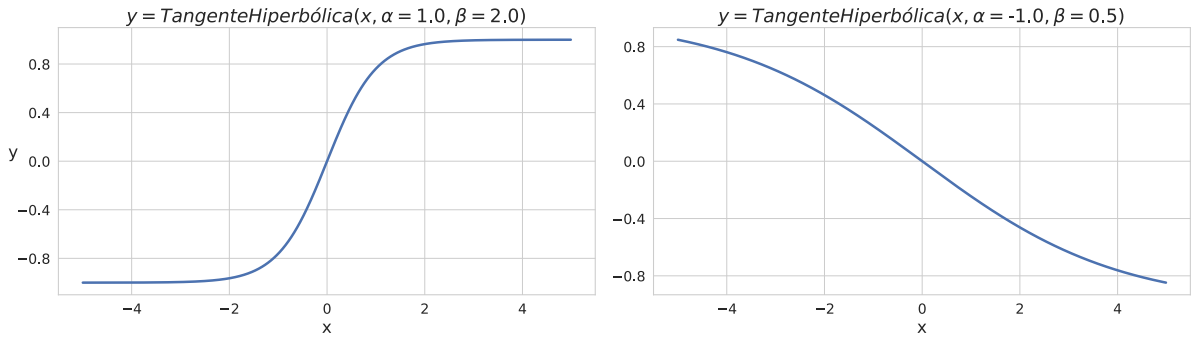
5.3.3 Tangente Hiperbólica

Nesta subseção, definimos a AF Tangente Hiperbólica, que recebe como entrada o valor x e os parâmetros α e β. Esses parâmetros possuem um valor padrão, normalmente utilizados na literatura, que é utilizado pela AF, sendo esses valores 1.0 e 2.0, respectivamente. A AF Tangente Hiperbólica é definida na Equação 5.53.

$$TangenteHiperbólica(x, \alpha, \beta) = \alpha \tanh\left(\frac{\beta x}{2}\right) \tag{5.53}$$

A Figura 32 apresenta a AF Tangente Hiperbólica com seus valores padrões à esquerda e, à direita, outra forma para diferentes valores de parâmetros.

Figura 32 – Função de Ativação Tangente Hiperbólica com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

Além disso, temos as derivadas parciais da função apresentadas pelas Equações 5.54, 5.55 e 5.56.

$$\frac{\partial TangenteHiperbólica(x,\alpha,\beta)}{\partial x} = \frac{\alpha\beta}{2\cosh^2\left(\frac{\beta x}{2}\right)} \tag{5.54}$$

$$\frac{\partial TangenteHiperbólica(x,\alpha,\beta)}{\partial \alpha} = \tanh\left(\frac{\beta x}{2}\right) \tag{5.55}$$

$$\frac{\partial TangenteHiperbólica(x,\alpha,\beta)}{\partial \beta} = \frac{\alpha x}{2\cosh^2\left(\frac{\beta x}{2}\right)} \tag{5.56}$$

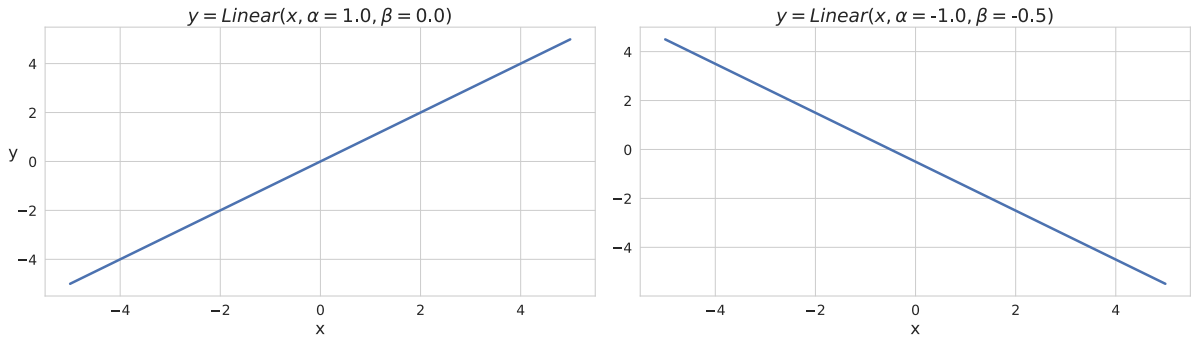
5.3.4 Linear

Nesta subseção, definimos a AF Linear, que recebe como entrada o valor x e os parâmetros α e β. Esses parâmetros possuem um valor padrão, normalmente utilizados na literatura, que é utilizado pela AF, sendo esses valores 1.0 e 0.0, respectivamente. A AF Linear é definida na Equação 5.57.

$$Linear(x,\alpha,\beta) = \alpha x + \beta \tag{5.57}$$

A Figura 33 apresenta a AF Linear com seus valores padrões à esquerda e, à direita, outra forma para diferentes valores de parâmetros.

Figura 33 – Função de Ativação Linear com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

Além disso, temos as derivadas parciais da função apresentadas pelas Equações 5.58, 5.59 e 5.60.

$$\frac{\partial Linear(x,\alpha,\beta)}{\partial x} = \alpha \tag{5.58}$$

$$\frac{\partial Linear(x,\alpha,\beta)}{\partial \alpha} = x \tag{5.59}$$

$$\frac{\partial Linear(x, \alpha, \beta)}{\partial \beta} = 1 \tag{5.60}$$

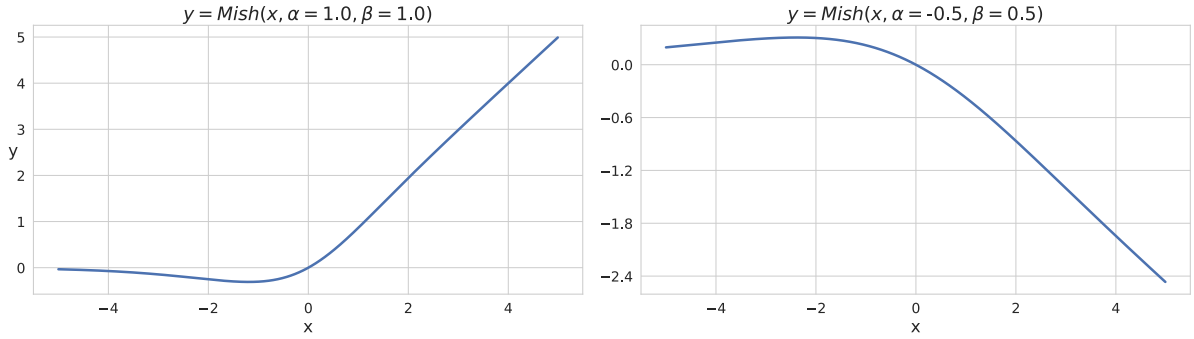
5.3.5 Mish

Nesta subseção, definimos a AF Mish, que recebe como entrada o valor x e os parâmetros α e β. Esses parâmetros possuem um valor padrão, normalmente utilizados na literatura, que é utilizado pela AF, sendo esses valores 1.0 e 1.0, respectivamente. A AF Mish é definida na Equação 5.61.

$$Mish(x, \alpha, \beta) = \alpha x \tanh\left(\log\left(e^{\beta x} + 1\right)\right) \tag{5.61}$$

A Figura 34 apresenta a AF Mish com seus valores padrões à esquerda e, à direita, outra forma para diferentes valores de parâmetros.

Figura 34 – Função de Ativação Mish com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

Além disso, temos as derivadas parciais da função apresentadas pelas Equações 5.62, 5.63 e 5.64.

$$\frac{\partial Mish(x, \alpha, \beta)}{\partial x} = \frac{\alpha\left(\frac{\beta x e^{\beta x}}{\cosh^2\left(\log\left(e^{\beta x}+1\right)\right)} + \left(e^{\beta x}+1\right)\tanh\left(\log\left(e^{\beta x}+1\right)\right)\right)}{e^{\beta x}+1} \tag{5.62}$$

$$\frac{\partial Mish(x, \alpha, \beta)}{\partial \alpha} = x \tanh\left(\log\left(e^{\beta x}+1\right)\right) \tag{5.63}$$

$$\frac{\partial Mish(x, \alpha, \beta)}{\partial \beta} = \frac{\alpha x^2 e^{\beta x}}{\left(e^{\beta x}+1\right)\cosh^2\left(\log\left(e^{\beta x}+1\right)\right)} \tag{5.64}$$

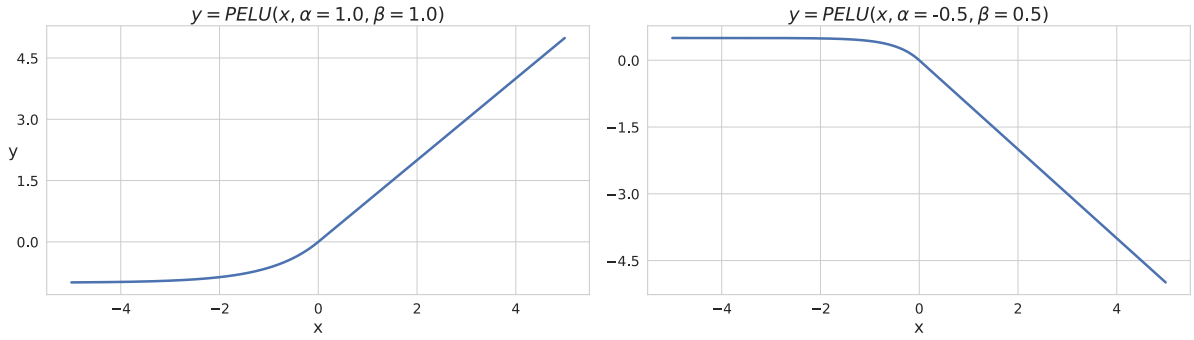
5.3.6 PELU

Nesta subseção, definimos a AF PELU, que recebe como entrada o valor x e os parâmetros α e β. Esses parâmetros possuem um valor padrão, normalmente utilizados na literatura, que é utilizado pela AF, sendo esses valores 1.0 e 1.0, respectivamente. A AF PELU é definida na Equação 5.65.

$$PELU(x, \alpha, \beta) = \begin{cases} \frac{\alpha x}{\beta} & \text{se } x \geq 0 \\ \alpha \left(e^{\frac{x}{\beta}} - 1 \right) & \text{caso contrário} \end{cases} \tag{5.65}$$

A Figura 35 apresenta a AF PELU com seus valores padrões à esquerda e, à direita, outra forma para diferentes valores de parâmetros.

Figura 35 – Função de Ativação PELU com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

Além disso, temos as derivadas parciais da função apresentadas pelas Equações 5.66, 5.67 e 5.68.

$$\frac{\partial PELU(x, \alpha, \beta)}{\partial x} = \begin{cases} \frac{\alpha}{\beta} & \text{se } x \geq 0 \\ \frac{\alpha e^{\frac{x}{\beta}}}{\beta} & \text{caso contrário} \end{cases} \tag{5.66}$$

$$\frac{\partial PELU(x, \alpha, \beta)}{\partial \alpha} = \begin{cases} \frac{x}{\beta} & \text{se } x \geq 0 \\ e^{\frac{x}{\beta}} - 1 & \text{caso contrário} \end{cases} \tag{5.67}$$

$$\frac{\partial PELU(x, \alpha, \beta)}{\partial \beta} = \begin{cases} -\frac{\alpha x}{\beta^2} & \text{se } x \geq 0 \\ -\frac{\alpha x e^{\frac{x}{\beta}}}{\beta^2} & \text{caso contrário} \end{cases} \tag{5.68}$$

5.3.7 PReLU

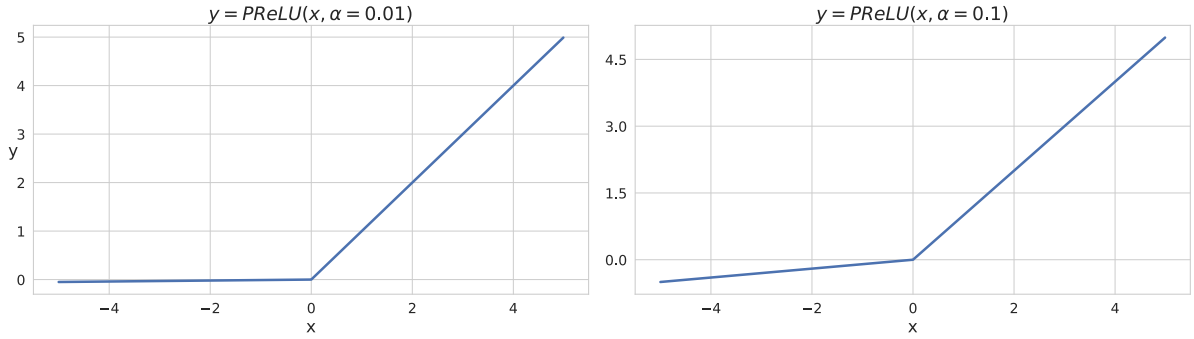
Nesta subseção, definimos a AF PReLU, que recebe como entrada o valor x e o parâmetro α. Esse parâmetro possui um valor padrão, normalmente utilizado na literatura,

que é utilizado pela AF, sendo esse valor 0.01. A AF PReLU é definida na Equação 5.69.

$$PReLU(x, \alpha) = \begin{cases} x & \text{se } x \geq 0 \\ \alpha x & \text{caso contrário} \end{cases} \tag{5.69}$$

A Figura 36 apresenta a AF PReLU com seu valor padrão à esquerda e, à direita, outra forma para diferente valor de parâmetro.

Figura 36 – Função de Ativação PReLU com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

Além disso, temos as derivadas parciais da função apresentadas pelas Equações 5.70 e 5.71.

$$\frac{\partial PReLU(x, \alpha)}{\partial x} = \begin{cases} 1 & \text{se } x \geq 0 \\ \alpha & \text{caso contrário} \end{cases} \tag{5.70}$$

$$\frac{\partial PReLU(x, \alpha)}{\partial \alpha} = \begin{cases} 0 & \text{se } x \geq 0 \\ x & \text{caso contrário} \end{cases} \tag{5.71}$$

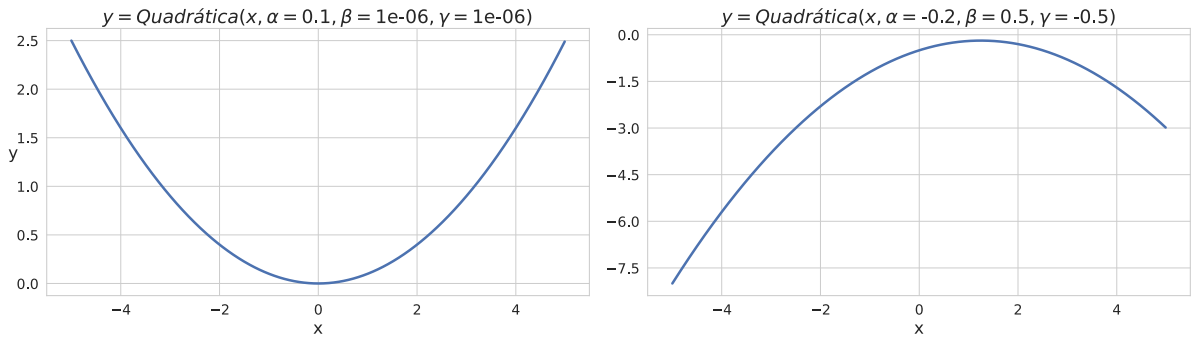
5.3.8 Quadrática

Nesta subseção, definimos a AF Quadrática, que recebe como entrada o valor x e os parâmetros α, β e γ. Esses parâmetros possuem um valor padrão, normalmente utilizados na literatura, que é utilizado pela AF, sendo esses valores 0.1, 1×10^{-6} e 1×10^{-6}, respectivamente. A AF Quadrática é definida na Equação 5.72.

$$Quadrática(x, \alpha, \beta, \gamma) = \alpha x^2 + \beta x + \gamma \tag{5.72}$$

A Figura 37 apresenta a AF Quadrática com seus valores padrões à esquerda e, à direita, outra forma para diferentes valores de parâmetros.

Figura 37 – Função de Ativação Quadrática com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

Além disso, temos as derivadas parciais da função apresentadas pelas Equações 5.73, 5.74, 5.75 e 5.76.

$$\frac{\partial Quadrática(x, \alpha, \beta, \gamma)}{\partial x} = 2\alpha x + \beta \tag{5.73}$$

$$\frac{\partial Quadrática(x, \alpha, \beta, \gamma)}{\partial \alpha} = x^2 \tag{5.74}$$

$$\frac{\partial Quadrática(x, \alpha, \beta, \gamma)}{\partial \beta} = x \tag{5.75}$$

$$\frac{\partial Quadrática(x, \alpha, \beta, \gamma)}{\partial \gamma} = 1 \tag{5.76}$$

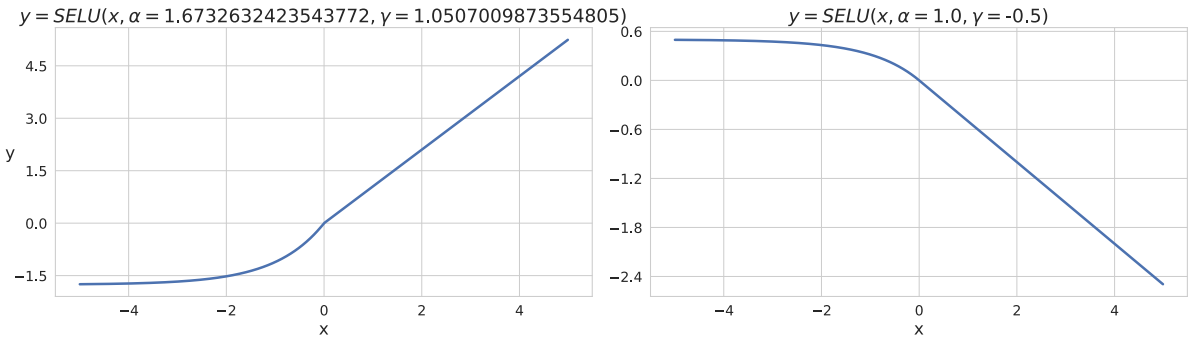
5.3.9 SELU

Nesta subseção, definimos a AF SELU, que recebe como entrada o valor x e os parâmetros α e γ. Esses parâmetros possuem um valor padrão, normalmente utilizados na literatura, que é utilizado pela AF, sendo esses valores 1.6732632423543772 e 1.0507009873554805, respectivamente. A AF SELU é definida na Equação 5.77.

$$SELU(x, \alpha, \gamma) = \begin{cases} \gamma x & \text{se } x \geq 0 \\ \alpha\gamma\left(e^x - 1\right) & \text{caso contrário} \end{cases} \tag{5.77}$$

A Figura 38 apresenta a AF SELU com seus valores padrões à esquerda e, à direita, outra forma para diferentes valores de parâmetros.

Figura 38 – Função de Ativação SELU com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

Além disso, temos as derivadas parciais da função apresentadas pelas Equações 5.78, 5.79 e 5.80.

$$\frac{\partial SELU(x,\alpha,\gamma)}{\partial x} = \begin{cases} \gamma & \text{se } x \geq 0 \\ \alpha\gamma e^{x} & \text{caso contrário} \end{cases} \tag{5.78}$$

$$\frac{\partial SELU(x,\alpha,\gamma)}{\partial \alpha} = \begin{cases} 0 & \text{se } x \geq 0 \\ \gamma\left(e^{x}-1\right) & \text{caso contrário} \end{cases} \tag{5.79}$$

$$\frac{\partial SELU(x,\alpha,\gamma)}{\partial \gamma} = \begin{cases} x & \text{se } x \geq 0 \\ \alpha\left(e^{x}-1\right) & \text{caso contrário} \end{cases} \tag{5.80}$$

5.3.10 Sigmóide

Nesta subseção, definimos a AF Sigmóide, que recebe como entrada o valor x e os parâmetros α e β. Esses parâmetros possuem um valor padrão, normalmente utilizados na literatura, que é utilizado pela AF, sendo esses valores 1.0 e 1.0, respectivamente. A AF Sigmóide é definida na Equação 5.81.

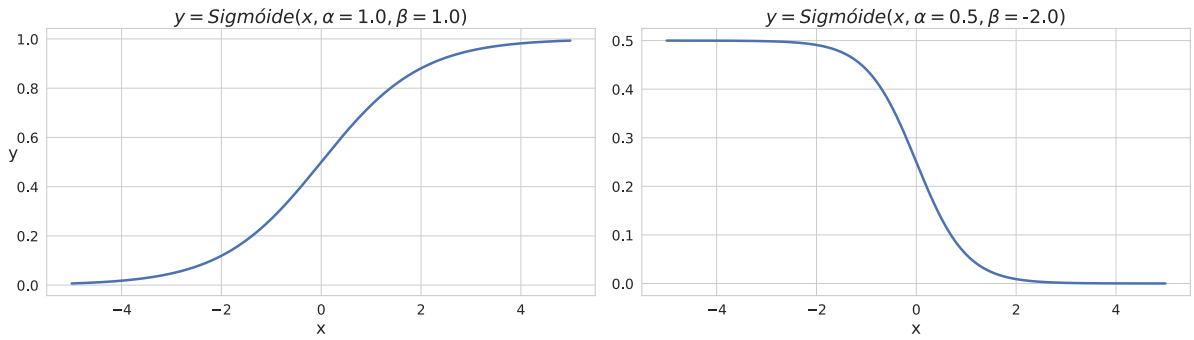
$$Sigmóide(x,\alpha,\beta) = \frac{\alpha e^{\beta x}}{e^{\beta x}+1} \tag{5.81}$$

A Figura 39 apresenta a AF Sigmóide com seus valores padrões à esquerda e, à direita, outra forma para diferentes valores de parâmetros.

Além disso, temos as derivadas parciais da função apresentadas pelas Equações 5.82, 5.83 e 5.84.

$$\frac{\partial Sigmóide(x,\alpha,\beta)}{\partial x} = \frac{\alpha\beta}{4\cosh^{2}\left(\frac{\beta x}{2}\right)} \tag{5.82}$$

Figura 39 – Função de Ativação Sigmóide com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

$$\frac{\partial Sigmóide(x, \alpha, \beta)}{\partial \alpha} = \frac{e^{\beta x}}{e^{\beta x} + 1} \tag{5.83}$$

$$\frac{\partial Sigmóide(x, \alpha, \beta)}{\partial \beta} = \frac{\alpha x}{4 \cosh^2\left(\frac{\beta x}{2}\right)} \tag{5.84}$$

5.3.11 SoftExponential

Nesta subseção, definimos a AF SoftExponential, que recebe como entrada o valor x e o parâmetro α. Esse parâmetro possui um valor padrão que é utilizado pela AF, sendo esse valor 0.1. A AF SoftExponential é definida na Equação 5.85.

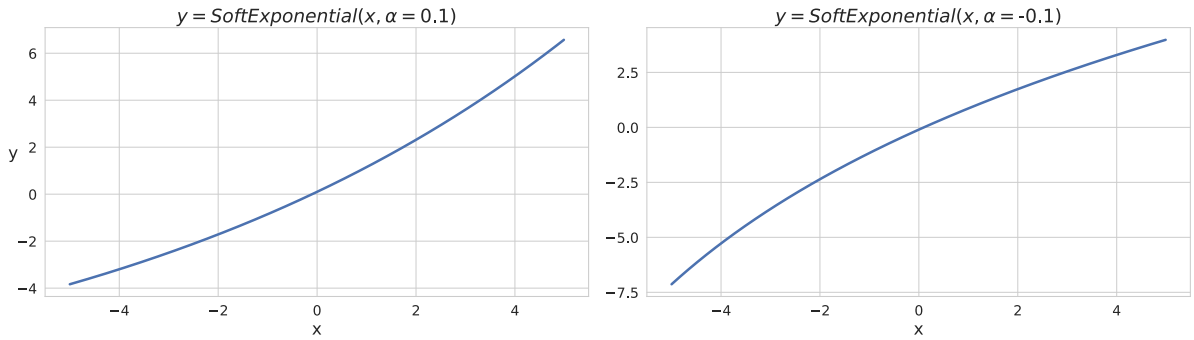
$$SoftExponential(x, \alpha) = \begin{cases} x & \text{se } \alpha = 0 \\ \frac{\alpha^2 + e^{\alpha x} - 1}{\alpha} & \text{se } \alpha > 0 \\ -\frac{\log(-\alpha(\alpha + x) + 1)}{\alpha} & \text{caso contrário} \end{cases} \tag{5.85}$$

A Figura 40 apresenta a AF SoftExponential com seus valor padrão à esquerda e, à direita, outra forma para diferente valor de parâmetro.

Além disso, temos as derivadas parciais da função apresentadas pelas Equações 5.86 e 5.87.

$$\frac{\partial SoftExponential(x, \alpha)}{\partial x} = \begin{cases} 1 & \text{se } \alpha = 0 \\ e^{\alpha x} & \text{se } \alpha > 0 \\ -\frac{1}{\alpha(\alpha + x) - 1} & \text{caso contrário} \end{cases} \tag{5.86}$$

Figura 40 – Função de Ativação SoftExponential com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

$$\frac{\partial SoftExponential(x,\alpha)}{\partial \alpha} = \begin{cases} 0 & \text{se } \alpha = 0 \\ \frac{\alpha^2+\alpha x e^{\alpha x}-e^{\alpha x}+1}{\alpha^2} & \text{se } \alpha > 0 \\ \frac{-\alpha(2\alpha+x)+(\alpha(\alpha+x)-1)\log(-\alpha(\alpha+x)+1)}{\alpha^2(\alpha(\alpha+x)-1)} & \text{caso contrário} \end{cases} \tag{5.87}$$

5.3.12 SoftPlus

Nesta subseção, definimos a AF SoftPlus, que recebe como entrada o valor x e os parâmetros α e β. Esses parâmetros possuem um valor padrão, normalmente utilizados na literatura, que é utilizado pela AF, sendo esses valores 1.0 e 1.0, respectivamente. A AF SoftPlus é definida na Equação 5.88.

$$SoftPlus(x,\alpha,\beta) = \alpha \log\left(e^{\beta x}+1\right) \tag{5.88}$$

A Figura 41 apresenta a AF SoftPlus com seus valores padrões à esquerda e, à direita, outra forma para diferentes valores de parâmetros.

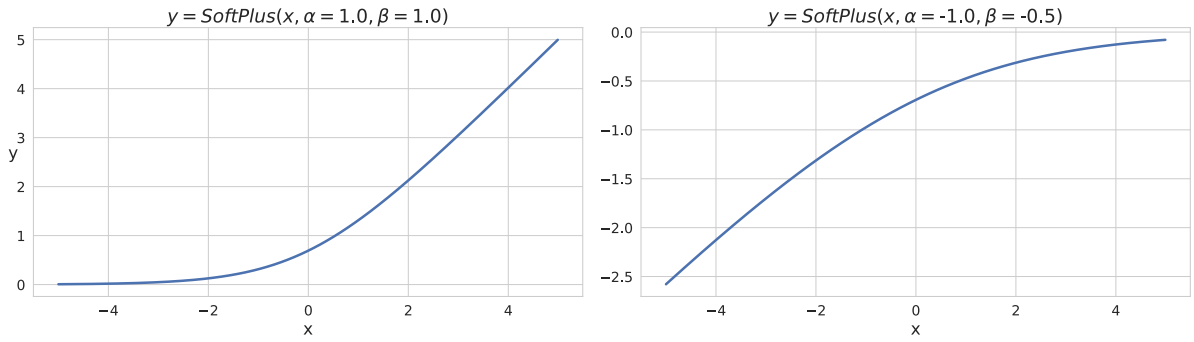
Além disso, temos as derivadas parciais da função apresentadas pelas Equações 5.89, 5.90 e 5.91.

$$\frac{\partial SoftPlus(x,\alpha,\beta)}{\partial x} = \frac{\alpha\beta e^{\beta x}}{e^{\beta x}+1} \tag{5.89}$$

$$\frac{\partial SoftPlus(x,\alpha,\beta)}{\partial \alpha} = \log\left(e^{\beta x}+1\right) \tag{5.90}$$

$$\frac{\partial SoftPlus(x,\alpha,\beta)}{\partial \beta} = \frac{\alpha x e^{\beta x}}{e^{\beta x}+1} \tag{5.91}$$

Figura 41 – Função de Ativação SoftPlus com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

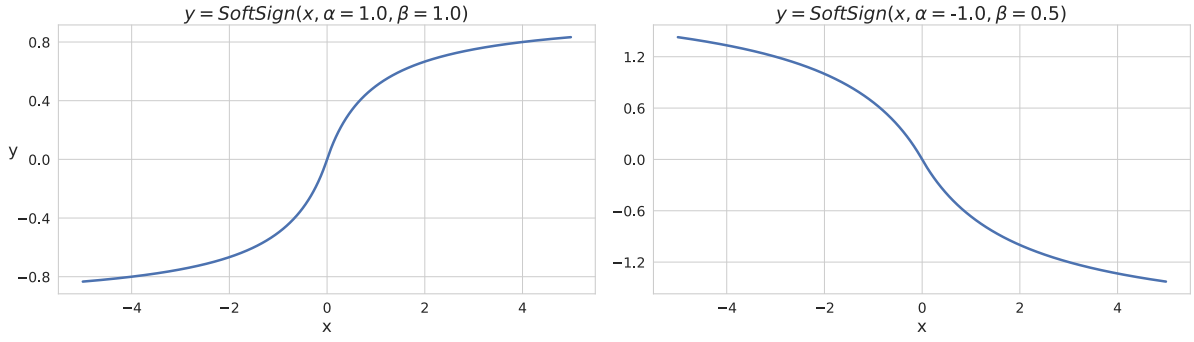
5.3.13 SoftSign

Nesta subseção, definimos a AF SoftSign, que recebe como entrada o valor x e os parâmetros α e β. Esses parâmetros possuem um valor padrão, normalmente utilizados na literatura, que é utilizado pela AF, sendo esses valores 1.0 e 1.0, respectivamente. A AF SoftSign é definida na Equação 5.92.

$$SoftSign(x, \alpha, \beta) = \frac{\alpha x}{\beta |x| + 1} = \begin{cases} \frac{\alpha x}{\beta x + 1} & \text{se } x \geq 0 \\ -\frac{\alpha x}{\beta x - 1} & \text{caso contrário} \end{cases} \tag{5.92}$$

A Figura 42 apresenta a AF SoftSign com seus valores padrões à esquerda e, à direita, outra forma para diferentes valores de parâmetros.

Figura 42 – Função de Ativação SoftSign com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

Além disso, temos as derivadas parciais da função apresentadas pelas Equações

5.93, 5.94 e 5.95.

$$\frac{\partial SoftSign(x,\alpha,\beta)}{\partial x} = \begin{cases} \frac{\alpha}{(\beta x+1)^2} & \text{se } x \geq 0 \\ \frac{\alpha}{(\beta x-1)^2} & \text{caso contrário} \end{cases} \tag{5.93}$$

$$\frac{\partial SoftSign(x,\alpha,\beta)}{\partial \alpha} = \begin{cases} \frac{x}{\beta x+1} & \text{se } x \geq 0 \\ -\frac{x}{\beta x-1} & \text{caso contrário} \end{cases} \tag{5.94}$$

$$\frac{\partial SoftSign(x,\alpha,\beta)}{\partial \beta} = \begin{cases} -\frac{\alpha x^2}{(\beta x+1)^2} & \text{se } x \geq 0 \\ \frac{\alpha x^2}{(\beta x-1)^2} & \text{caso contrário} \end{cases} \tag{5.95}$$

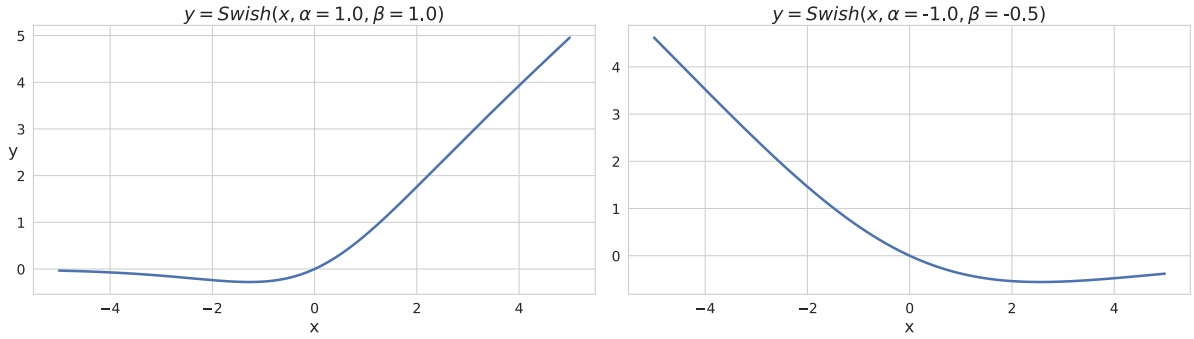
5.3.14 Swish

Nesta subseção, definimos a AF Swish, que recebe como entrada o valor x e os parâmetros α e β. Esses parâmetros possuem um valor padrão, normalmente utilizados na literatura, que é utilizado pela AF, sendo esses valores 1.0 e 1.0, respectivamente. A AF Swish é definida na Equação 5.96.

$$Swish(x,\alpha,\beta) = \frac{\alpha x e^{\beta x}}{e^{\beta x}+1} \tag{5.96}$$

A Figura 43 apresenta a AF Swish com seus valores padrões à esquerda e, à direita, outra forma para diferentes valores de parâmetros.

Figura 43 – Função de Ativação Swish com diferentes valores de parâmetros.



Fonte: Elaborada pelo autor (2023).

Além disso, temos as derivadas parciais da função apresentadas pelas Equações 5.97, 5.98 e 5.99.

$$\frac{\partial Swish(x,\alpha,\beta)}{\partial x} = \frac{\alpha\left(\beta x + e^{\beta x} + 1\right)}{4\cosh^2\left(\frac{\beta x}{2}\right)} \tag{5.97}$$

$$\frac{\partial Swish(x,\alpha,\beta)}{\partial \alpha} = \frac{xe^{\beta x}}{e^{\beta x}+1} \tag{5.98}$$

$$\frac{\partial Swish(x,\alpha,\beta)}{\partial \beta} = \frac{\alpha x^2}{4\cosh^2\left(\frac{\beta x}{2}\right)} \tag{5.99}$$

5.3.15 Fourier

Nesta subseção, definimos a AF Fourier, inspirada na própria Série de Fourier (STEIN; SHAKARCHI, 2011), que recebe como entrada o valor x e os parâmetros α_0, α_1, ..., α_n, β_1, ..., β_n e L_1, ..., L_n. Esses parâmetros possuem um valor padrão, normalmente utilizados na literatura, que é utilizado pela AF neste trabalho, sendo esses valores:

$$\alpha_0 = 1.0$$

$$\alpha_i = (0.5/i)\ \forall i \in 1, 2, ..., n$$

$$\beta_i = (0.5/i) * (-1)^i\ \forall i \in 1, 2, ..., n$$

$$L_i = 5.0\ \forall i \in 1, 2, ..., n$$

O valor n é um hiperparâmetro definido pelo usuário antes da fase de treinamento. À medida que n tende ao infinito, a série de Fourier se torna uma representação mais precisa da função original que se deseja aproximar (STEIN; SHAKARCHI, 2011). A equação da AF Fourier é definida como: A AF Fourier é definida na Equação 5.100.

$$Fourier(x, \alpha_0, \alpha_1, ..., L_n) = \frac{\alpha_0}{2} + \sum_{i=1}^{n\to\infty} \alpha_i \cos(\frac{i\pi x}{L_i}) + \beta_i \sin(\frac{i\pi x}{L_i}) \tag{5.100}$$

A Figura 44 apresenta a AF Fourier com seus valores padrões à esquerda e, à direita, outra forma para diferentes valores de parâmetros.

Além disso, temos as derivadas parciais da função apresentadas pelas Equações 5.101, 5.102, 5.103, 5.104 e 5.105.

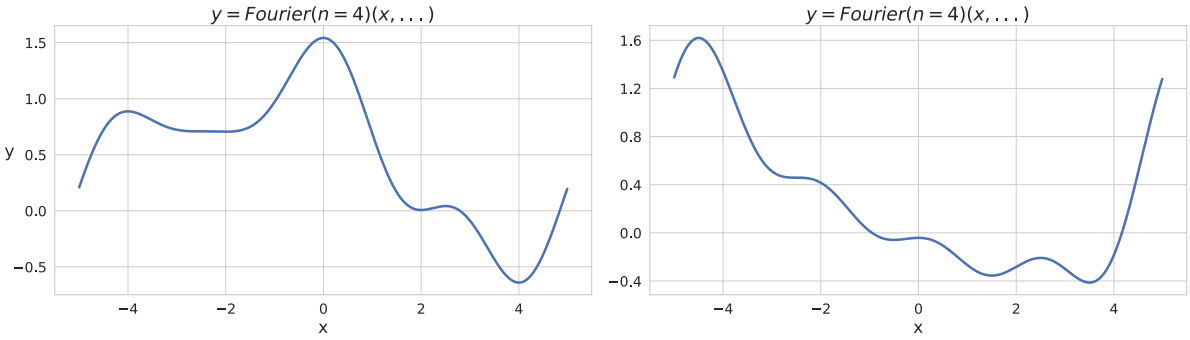
$$\frac{\partial Fourier(x, \alpha_0, \alpha_1, ..., L_n)}{\partial x} = \sum_{i=1}^{n\to\infty} \left(-\frac{i\pi\alpha_i}{L_i} \sin\left(\frac{i\pi x}{L_i}\right) + \frac{i\pi\beta_i}{L_i} \cos\left(\frac{i\pi x}{L_i}\right) \right) \tag{5.101}$$

$$\frac{\partial Fourier(x, \alpha_0, \alpha_1, ..., L_n)}{\partial \alpha_0} = \frac{1}{2} \tag{5.102}$$

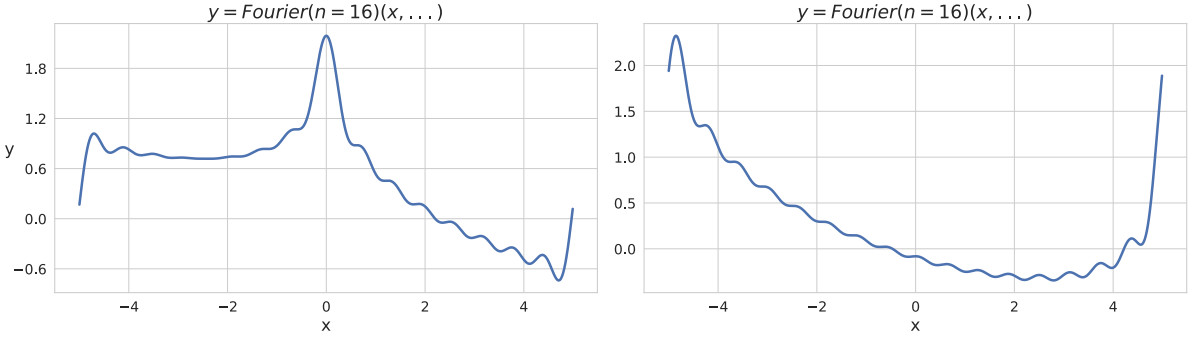
$$\frac{\partial Fourier(x, \alpha_0, \alpha_1, ..., L_n)}{\partial \alpha_i} = \cos(\frac{i\pi x}{L_i})\ \forall i \in 1, 2, ..., n \tag{5.103}$$

Figura 44 – Função de Ativação Série de Fourier com diferentes valores de parâmetros.

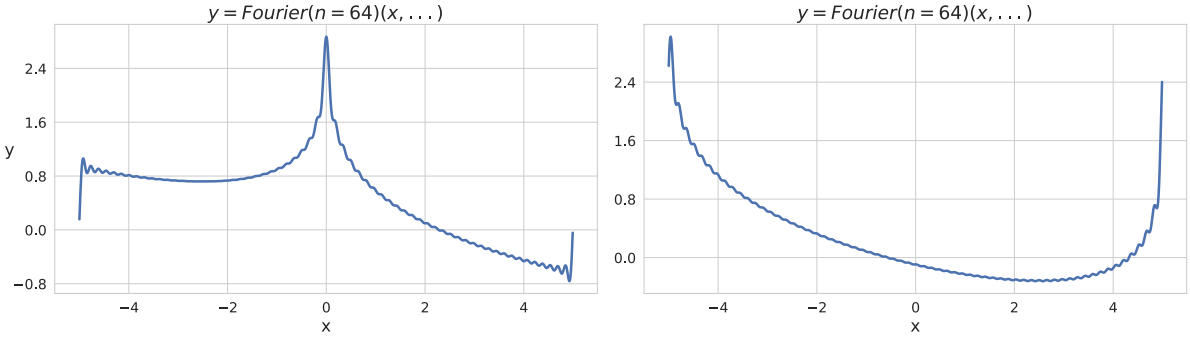
(a) $n = 4$.



(b) $n = 16$.



(c) $n = 64$.



Fonte: Elaborada pelo autor (2023).

$$\frac{\partial Fourier(x, \alpha_0, \alpha_1, ..., L_n)}{\partial \beta_i} = \sin(\frac{i\pi x}{L_i}) \ \forall i \in 1, 2, ..., n \tag{5.104}$$

$$\frac{\partial Fourier(x, \alpha_0, \alpha_1, ..., L_n)}{\partial L_i} = \frac{i\pi\alpha_i x \sin\left(\frac{i\pi x}{L_i}\right)}{L_i^2} - \frac{i\pi\beta_i x \cos\left(\frac{i\pi x}{L_i}\right)}{L_i^2} \ \forall i \in 1, 2, ..., n \tag{5.105}$$

É importante destacar que quanto maior o valor de n, maior a quantidade de parâmetros. Com $n = 0$, temos apenas um parâmetro, o α_0. Com $n = 1$, temos 4 parâmetros: α_0, α_1, β_1 e L_1. Desta forma, sabemos que teremos $3n + 1$ parâmetros dentro da função.

5.4 Funções De Ativação Adaptativas

A Função de Ativação Adaptativa (*Adaptive Activation Function* - AAF) difere-se da AF apenas pelo fato de que, durante o processo de treinamento da ANN, os parâmetros são adaptados da mesma forma que os pesos e tendências. Desta forma, inicializamos as ANN com as funções em seus valores padrões e, durante o treinamento, esses parâmetros são ajustados. Isso torna possível encontrar formas diferentes das tradicionais, conforme apresentado na Seção 5.3.

Considerando o exemplo da Figura 29, presente na Seção 5.2 sobre Retropropagação, e a função f_y sendo uma AAF com dois parâmetros ajustáveis, α e β, temos que o processo de atualização seria conforme as Equações 5.106 e 5.107, respectivamente.

$$\alpha(t+1) = \alpha(t) - \eta \frac{\partial E^2}{\partial \alpha} \tag{5.106}$$

$$\beta(t+1) = \beta(t) - \eta \frac{\partial E^2}{\partial \beta} \tag{5.107}$$

onde $\alpha(t)$ e $\alpha(t+1)$ representam os valores de α nos tempos t e $t+1$, respectivamente, $\beta(t)$ e $\beta(t+1)$ representam os valores de β nos tempos t e $t+1$, respectivamente, e η é uma taxa de aprendizado do algoritmo de treinamento.

Utilizando a regra da cadeia, obtemos as Equações 5.108 e 5.109.

$$\frac{\partial E^2}{\partial \alpha} = \frac{\partial E^2}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \alpha} = \frac{\partial E^2}{\partial \hat{y}} \cdot \frac{\partial f_y(h_2 * w_3)}{\partial \alpha} = -2 \sum_{i=1}^{n} (y_i - \hat{y}_i) \cdot \frac{\partial f_y(\xi)}{\partial \alpha}|_{\xi = h_2 * w_3} \tag{5.108}$$

$$\frac{\partial E^2}{\partial \beta} = \frac{\partial E^2}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \beta} = \frac{\partial E^2}{\partial \hat{y}} \cdot \frac{\partial f_y(h_2 * w_3)}{\partial \beta} = -2 \sum_{i=1}^{n} (y_i - \hat{y}_i) \cdot \frac{\partial f_y(\xi)}{\partial \beta}|_{\xi = h_2 * w_3} \tag{5.109}$$

onde $\partial f_y(\xi)/\partial \alpha$ e $\partial f_y(\xi)/\partial \beta$ representam as derivadas parciais da função de ativação f_y em relação a α e β, respectivamente, no valor ξ de entrada.

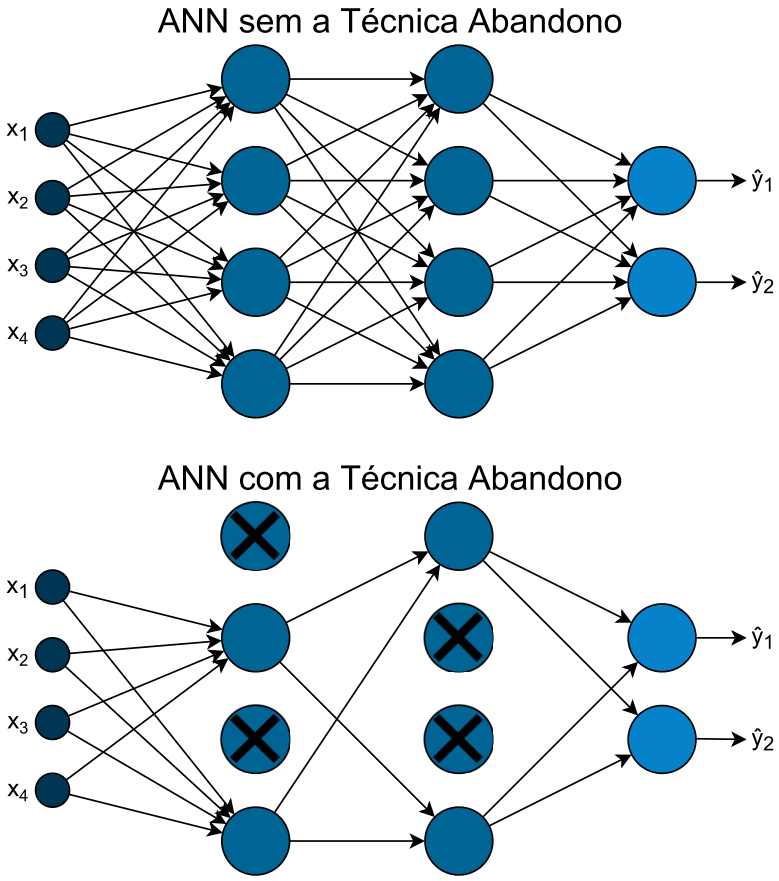
Desta forma, por meio do processo de adaptação das AF, estamos buscando pelas curvas/superfícies ótimas que uma AF poderia assumir para minimizar uma determinada métrica. Esse processo torna as ANN com AAF mais adaptáveis/ajustáveis a diferentes problemas do que as ANN com AF. Apesar disso, esse processo de adaptação pode trazer problemas previamente discutidos, como o sobreajuste.

5.5 Abandono

O Abandono (*Dropout*) (SRIVASTAVA et al., 2014) é uma técnica de regularização desenvolvida para prevenir o sobreajuste em uma ANN. Para lidar com esse problema, a técnica propõe realizar, de forma aleatória, o desligamento temporário de alguns neurônios

durante cada iteração da etapa de treinamento. É importante destacar que a cada iteração da etapa de treinamento, diferentes neurônios são selecionados para serem desligados. Dessa maneira, com o desligamento de alguns neurônios, o modelo é forçado a transmitir as informações importantes por outros caminhos, promovendo o compartilhamento de informação entre os neurônios. A Figura 45 ilustra o processo de desligamento dos neurônios, em que as conexões são eliminadas devido a esse desligamento.

Figura 45 – Exemplo de uma ANN com e sem a técnica Abandono.



Fonte: Elaborada pelo autor (2023).

A técnica é tipicamente aplicada nas camadas internas de uma ANN, podendo também ser aplicada na camada de entrada. Ela opera com uma determinada probabilidade de desligamento, normalmente entre 20% e 50%, mas essa é um hiperparâmetro que deve ser ajustado para cada aplicação. Quanto maior for o valor dessa probabilidade, mais desafiador será para a ANN aprender, podendo, em alguns casos, levar ao subajuste.

5.6 Normalização em Lote

A Normalização em Lote (*Batch Normalization* - BN) (IOFFE; SZEGEDY, 2015) é uma técnica de normalização das entradas das camadas desenvolvida para ANNs profundas. Essa técnica surgiu para tratar o problema conhecido como Mudança de Covariável Interna (*Internal Covariate Shift*), em que a distribuição das entradas de cada camada muda durante o treinamento à medida que os parâmetros das camadas anteriores também mudam. Com isso, o processo de treinamento é desacelerado, pois requer taxas de aprendizado mais baixas e inicialização cuidadosa dos parâmetros, além de tornar extremamente difícil treinar modelos com não linearidades saturadas. Não linearidades saturadas são características de certas AF em ANN, como a AF Sigmóide e a AF Tangente Hiperbólica, onde a saída da função tende a se aproximar de um valor máximo ou mínimo à medida que a entrada se torna muito grande ou muito pequena.

A ideia da técnica é realizar a normalização das ativações de cada camada da ANN, de modo que elas produzam valores com média zero e desvio padrão igual a um. Esse processo é executado com mini-lotes de dados, uma vez que calcular a média e o desvio padrão de toda a base pode ser computacionalmente custoso e retardar o processo de treinamento. Além disso, ao trabalhar com mini-lotes de dados, o processo de normalização se torna mais robusto aos ruídos dos dados, visto que os valores estão sujeitos a menos variação do que se fossem calculados sobre todo o conjunto de dados.

O processo de normalização é realizado da seguinte forma:

1. Cálculo da média do mini-lote L composto por m valores (Equação 5.110).

$$\mu_L = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{5.110}$$

2. Cálculo da variância do mini-lote L composto por m valores (Equação 5.111).

$$\sigma_L^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_L)^2 \tag{5.111}$$

3. Normalize cada elemento i no mini-lote L, onde o valor ϵ é utilizado para evitar divisão por zero (Equação 5.112).

$$\hat{x}_i = \frac{x_i - \mu_L}{\sqrt{\sigma_L^2 + \epsilon}} \tag{5.112}$$

4. Aplique escala e deslocamento ao valor normalizado usando parâmetros aprendidos γ e β (Equação 5.113).

$$y_i = \gamma\hat{x}_i + \beta \tag{5.113}$$

Considerando uma métrica de perda ℓ e os parâmetros γ e β, podemos realizar a atualização seguindo o mesmo processo utilizado para atualizar os pesos e tendências. Primeiramente, aplicamos a regra da cadeia para avaliar o impacto dos parâmetros em relação à métrica de perda ℓ, conforme as Equações 5.114 e 5.115.

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \frac{\partial y_i}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i \tag{5.114}$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \frac{\partial y_i}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \tag{5.115}$$

e esses valores são empregados para atualizar os parâmetros, conforme as Equações 5.116 e 5.117.

$$\gamma(t+1) = \gamma(t) - \eta \frac{\partial \ell}{\partial \gamma} \tag{5.116}$$

$$\beta(t+1) = \beta(t) - \eta \frac{\partial \ell}{\partial \beta} \tag{5.117}$$

onde $\gamma(t)$ e $\gamma(t+1)$ representam o valor de γ no tempo t e $t+1$, respectivamente. Da mesma forma, $\beta(t)$ e $\beta(t+1)$ representam o valor de β no tempo t e $t+1$, respectivamente, e η é a taxa de aprendizado do algoritmo de treinamento.

O processo de normalização descrito anteriormente depende do mini-lote, permitindo eficiência no treinamento. Apesar disso, esse processo não é necessário nem desejável durante a inferência, pois desejamos que a saída dependa apenas do valor de entrada, de maneira determinística. Para atingir esse objetivo, utilizamos a normalização dada pela Equação 5.118.

$$\hat{x} = \frac{x - E[x]}{\sqrt{Var[X] + \epsilon}} \tag{5.118}$$

onde $E[x] = E_L[\mu_L]$ e $Var[X] = \frac{m}{m+1} \cdot E_L[\sigma_L^2]$ representam expectativas relacionadas ao treinamento dos mini-lotes de tamanho m. Aqui, μ_L denota as médias amostrais e σ_L^2 representa as variações amostrais. Em vez disso, através do uso de médias móveis, é possível acompanhar a precisão de um modelo durante o treinamento.

Durante os experimentos com BN, os autores observaram que utilizar a técnica de Abandono em conjunto com o BN não faz tanto sentido, podendo, assim, ser removida ou perder a sua eficácia. Para explorar essa questão, foi descrita, na próxima seção, a combinação de ambas as técnicas.

5.7 Ordem do Abandono e Normalização em Lote

Li et al. (2019) desenvolveram um estudo para responder ao motivo pelo qual a aplicação das duas técnicas, Abandono e BN, frequentemente leva a um desempenho pior

quando combinadas. Realizaram uma investigação teórica e prática sobre o motivo disso ocorrer e descobriram que a técnica de Abandono altera a variância de um neurônio quando passamos do treino para o teste, visto que os desligamentos aleatórios não aconteceriam mais. Entretanto, a técnica BN manteria a sua variância estatística, obtida de forma cumulativa durante a etapa de treinamento, na fase de teste. Essa inconsistência nos valores de variância é o que causa o comportamento numérico instável na inferência e, consequentemente, previsões mais incertas quando a técnica de Abandono é aplicada antes da técnica de BN.

Os autores propõem duas soluções para resolver esse problema, alterando a técnica do Abandono. A primeira é simplesmente aplicar a técnica do Abandono somente após o último bloco que utilizou o BN, onde eles observaram uma melhoria nos resultados numéricos. A segunda é fazer uma modificação na fórmula da técnica do Abandono para torná-la menos sensível à variância, o que diminui o problema de deslocamento e estabiliza os comportamentos numéricos. Dessa maneira, descobriram que, em geral, a técnica "Incerteza Fora" (*Uncertainty Out* - Uout) (YI et al., 2020), uma variação da técnica do Abandono que desativa neurônios durante a inferência, poderia ser usada em conjunto com BN e também traria melhores resultados numéricos.

Apesar disso, Chen et al. (2019) apresentaram um estudo em que propõem uma camada denominada "Componente Independente" (*Independent Component* - IC), que tem como propósito realizar uma "normalização de dados" [1], processo que envolve a remoção de qualquer relação linear entre as características dos dados, transformando-as em independentes e identicamente distribuídas. Essa camada IC é composta pelas técnicas BN e Abandono em sequência, sendo aplicada entre uma AF e os próximos pesos.

A motivação do trabalho foi buscar uma nova abordagem, que fosse computacionalmente eficiente, para realizar esse processo de "normalização de dados" entre as camadas de uma ANN. De acordo com as análises teóricas apresentadas no trabalho, a camada IC tem o potencial de reduzir a informação mútua entre os neurônios por um fator de quatro em relação ao parâmetro p da camada de Abandono. Além disso, conseguiram produzir um processo de treinamento mais estável, com maior velocidade de convergência e melhor desempenho de generalização ao reformularem a arquitetura ResNet (HE et al., 2016) com camadas IC.

Dessa forma, podemos observar que a utilização de ambas as técnicas, BN e Abandono, sem um uso adequado, pode trazer malefícios para o desempenho da ANN. Por outro lado, a utilização de ambas as técnicas em locais específicos da ANN pode trazer grandes benefícios.

[1] O termo é conhecido como "branqueamento" ("*whitening*"), mas pode ser considerado um termo ofensivo, e por esse motivo não será utilizado neste trabalho.

5.8 Normalização de Peso

A Normalização de Peso (*Weight Normalization*) (SALIMANS; KINGMA, 2016) é uma técnica desenvolvida com o intuito de realizar uma reparametrização dos pesos da ANN, desacoplando a escala dos pesos de sua direção. Isso é feito ao expressar os vetores de peso em termos de novos parâmetros, como mostra a Equação 5.119.

$$\mathbf{w} = \frac{g}{||\mathbf{v}||}\mathbf{v} \tag{5.119}$$

onde $\mathbf{w}$ e $\mathbf{v}$ são vetores k-dimensionais, g é um escalar, e $||\mathbf{v}||$ é a norma euclidiana de $\mathbf{v}$. Ao realizar essa reparametrização dos pesos, temos que $||\mathbf{w}|| = g$, independentemente dos parâmetros de $\mathbf{v}$.

Ao realizar esse desacoplamento, é possível realizar a busca individual dos parâmetros, cada um com uma responsabilidade diferente, onde g é responsável pela escala e $\mathbf{v}/||\mathbf{v}||$ pela direção. Essa reparametrização permite um treinamento mais rápido e uma melhor generalização, pois ajuda a evitar que os pesos se tornem muito grandes ou muito pequenos, o que pode causar problemas como o desaparecimento ou a explosão de gradientes.

Além disso, os autores demonstram por meio de experimentos que a técnica pode ser combinada com a técnica de Abandono e BN para obter desempenhos melhores. A combinação dessas técnicas é capaz de alcançar várias formas de regularização, o que pode ajudar a prevenir a superajustagem e aprimorar a generalização (ZHANG et al., 2018).

6 ABORDAGEM EVOLUTIVA DE REDES NEURAIS ARTIFICIAIS

O estudo desenvolvido nesta tese visa investigar uma nova abordagem evolutiva para encontrar a melhor arquitetura para uma ANN. A ideia principal do trabalho é a utilização de um processo de busca multiobjetivo, em combinação com AAF, com o intuito de encontrar arquiteturas mais simples e compactas, sem comprometer a capacidade de predição.

A hipótese adotada para este estudo é que é possível descobrir neurônios mais avançados com AAF, que se adaptariam melhor aos problemas, diminuindo a necessidade de outros neurônios e, por consequência, simplificando a arquitetura da ANN. Essas pesquisas são relevantes devido à necessidade de encontrar soluções mais eficientes e sustentáveis.

6.1 Definição da Solução

Para criar e treinar um modelo de ANN, é necessário, em primeiro lugar, estabelecer a relação entre a possível solução e os parâmetros utilizados para construir e treinar a arquitetura do modelo. Esse processo envolve definir o significado de cada posição da solução, estabelecer limites inferiores e superiores para cada um deles e determinar o tipo de variável a ser utilizada. Segue a lista de cada variável da solução:

- Taxa de Aprendizado. A variável assume valores reais. Os limites inferior e superior são hiperparâmetros definidos pelo usuário (valor muito dependente do problema abordado).
- Número de Épocas. A variável assume valores inteiros. Os limites inferior e superior são hiperparâmetros definidos pelo usuário (valor muito dependente do problema abordado).
- Tamanho dos Mini-Lotes. A variável assume valores inteiros. Os limites inferior e superior são hiperparâmetros definidos pelo usuário.
- Quantidade de Camadas Internas. A variável assume valores inteiros. O limite inferior é igual a 1 e o superior é um hiperparâmetro definido pelo usuário.
- Para cada camada interna:
  - Utilizar Normalização em Lote. A variável assume valores inteiros (binários), representando a ativação (1) ou não (0) da normalização em lote. O limite inferior é igual a 0 e o superior é igual a 1.

  - Utilizar Normalização de Peso. A variável assume valores inteiros (binários), representando a ativação (1) ou não (0) da normalização de peso. O limite inferior é igual a 0 e o superior é igual a 1.
  - Probabilidade de Abandono. A variável assume valores reais. O limite inferior é igual a 0 e o superior é igual a 1.
  - Quantidade de Neurônios. A variável assume valores inteiros. O limite inferior é igual a 1 e o superior é um hiperparâmetro definido pelo usuário (valor muito dependente do problema abordado).
  - Identificador da Função de Ativação. A variável assume valores inteiros. O limite inferior é igual a 1 e o superior é igual ao tamanho da lista de possíveis Funções de Ativação, definida como hiperparâmetro pelo usuário.

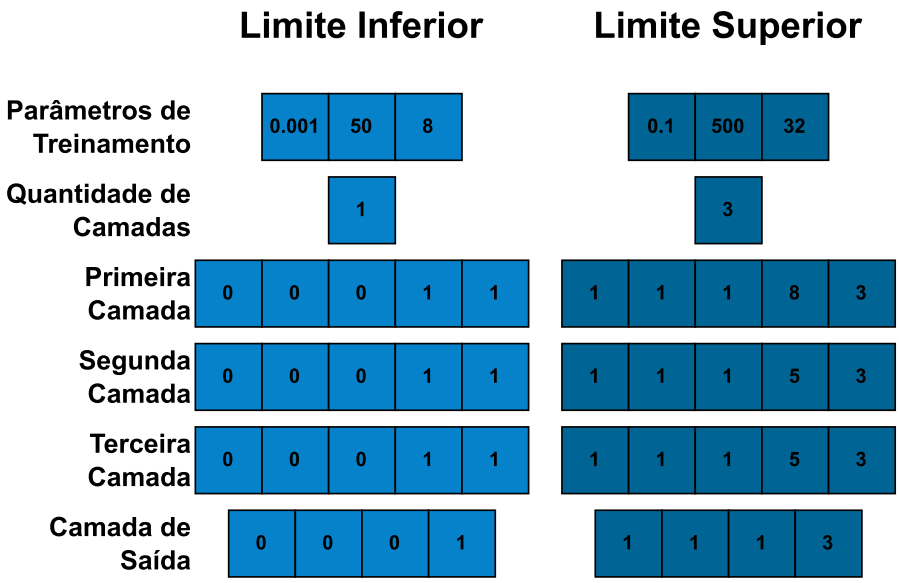
- Para a camada de saída:
  - Utilizar Normalização em Lote. A variável assume valores inteiros (binários), representando a ativação (1) ou não (0) da normalização em lote. O limite inferior é igual a 0 e o superior é igual a 1.
  - Utilizar Normalização de Peso. A variável assume valores inteiros (binários), representando a ativação (1) ou não (0) da normalização de peso. O limite inferior é igual a 0 e o superior é igual a 1.
  - Probabilidade de Abandono. A variável assume valores reais. O limite inferior é igual a 0 e o superior é igual a 1.
  - Identificador da Função de Ativação. A variável assume valores inteiros. O limite inferior é igual a 1 e o superior é igual ao tamanho da lista de possíveis Funções de Ativação, definida como hiperparâmetro pelo usuário.

Os três primeiros valores da solução são utilizados durante o processo de treinamento da ANN, enquanto os demais são utilizados para definir/construir a sua arquitetura.

Como exemplo, considere as tuplas (0.001, 0.1), (50, 500) e (8, 32) como os limites inferior e superior para a taxa de aprendizado, número de épocas e tamanho dos mini-lotes, respectivamente. Além disso, a tupla (8, 5, 5) representa, para cada posição, o valor máximo de neurônios por camada, e o seu comprimento indica a quantidade máxima de camadas. Por fim, as funções Linear, SELU e Sigmóide compõem a lista de funções de ativação possíveis. A Figura 46 representa, para o exemplo descrito anteriormente, os limites inferior e superior das possíveis soluções do processo de busca.

Supondo ainda que se trata de um problema de regressão, onde desejamos prever dois valores através de quatro variáveis explicativas, temos que a Figura 47 representa uma possível solução aleatória deste espaço. Para esta solução, a taxa de aprendizado é 0.05, a quantidade de épocas de treinamento é 256 e o tamanho do mini-lote é 16. Além

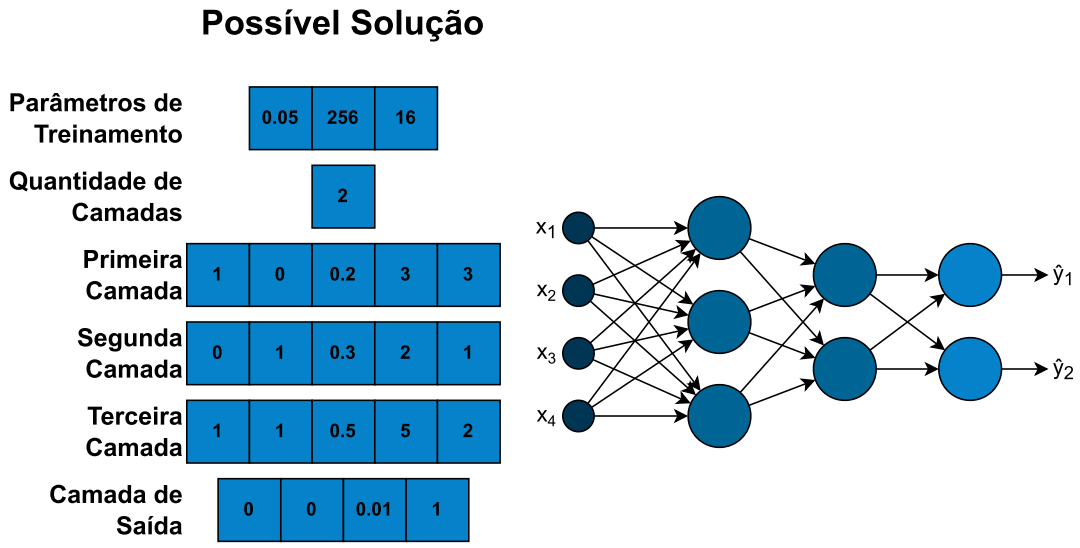
Figura 46 – Limite inferior e superior das possíveis soluções do problema exemplo.



Fonte: Elaborada pelo autor (2023).

disso, embora seja possível ter três camadas internas, a solução indica a utilização de duas, sendo assim necessário descartar qualquer informação presente sobre a terceira camada. A primeira camada possui normalização em lote, mas não tem normalização de peso. Tem uma taxa de abandono igual a 0.2, possui três neurônios e utiliza a função de ativação "Sigmóide". A segunda camada não possui normalização em lote, mas tem normalização de peso. Tem uma taxa de abandono igual a 0.3, possui dois neurônios e utiliza a função de ativação "Linear". A terceira camada, apesar de possuir valores, eles são descartados, conforme explicado anteriormente. A camada de saída não possui normalização em lote e nem de peso. Tem uma taxa de abandono igual a 0.01 e utiliza a função de ativação "Linear".

Figura 47 – Exemplo de possível solução aleatória dentro dos limites inferiores e superiores do problema.



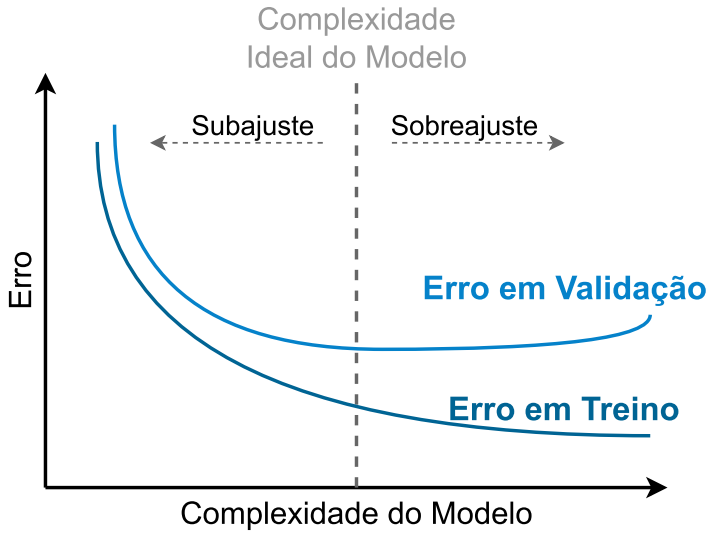
Fonte: Elaborada pelo autor (2023).

6.2 Definição das Funções Objetivo

Com o processo anterior, conseguimos, para qualquer vetor solução, construir a ANN e definir alguns dos seus parâmetros de treinamento. Através disso, com o propósito do trabalho, definimos funções objetivo que reflitam a qualidade da solução avaliada. Essas funções objetivo visam modelos de melhor desempenho no processo de predição e, ao mesmo tempo, que sejam simples/compactos.

Iniciando com a primeira função objetivo, relacionada à performance do modelo, buscamos as ANN que apresentam os melhores valores de métricas, conforme as apresentadas na subseção 4.5.1. Arquiteturas muito simples vão apresentar comportamento de subajuste, exibindo valores de métricas não desejáveis. Enquanto arquiteturas muito complexas terão o comportamento oposto, o sobreajuste, demonstrando valores de métricas boas, mas que também não são desejáveis devido às características dos modelos superajustados. O subajuste é resolvido com o processo de busca, pois essas arquiteturas gradualmente cedem lugar para aquelas que melhoram o valor das métricas. Por outro lado, é necessário adotar uma abordagem para tratar a questão do sobreajuste, visto que esses modelos tendem a aprender os ruídos dos dados, resultando na minimização das métricas escolhidas nos dados de treinamento, mas apresentando resultados insatisfatórios para dados nunca antes observados. A Figura 48, extraída e adaptada de Yu et al. (2007), apresenta o erro das métricas de desempenho em treinamento e validação em relação à complexidade do modelo.

Figura 48 – Representação do erro de treino e validação em comparação a complexidade do modelo.



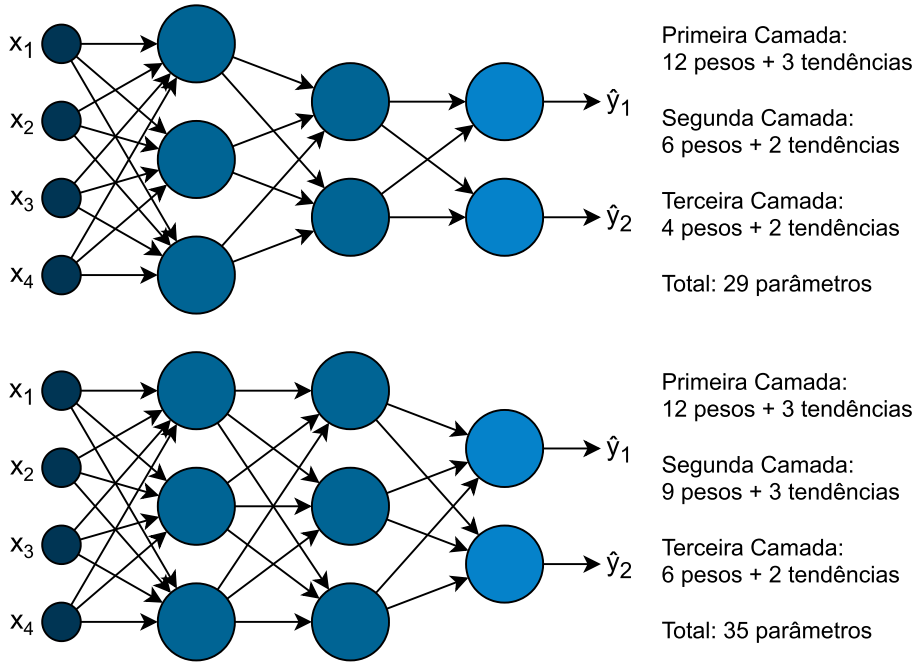
Fonte: Extraída e adaptada de Yu et al. (2007).

Para evitar que arquiteturas que levam ao sobreajuste sejam consideradas como boas soluções, optamos por utilizar a validação cruzada no processo de avaliação, sendo a técnica de validação cruzada e seus parâmetros definidos anteriormente a otimização da arquitetura. Dessa forma, os dados iniciais que seriam destinados apenas ao treinamento são divididos entre conjuntos de treinamento e validação. O modelo é então treinado no novo conjunto de treinamento e sua métrica de desempenho é avaliada no conjunto de validação. Desse modo, se o modelo estiver sofrendo de sobreajuste nos dados de treinamento, seu desempenho nos dados de validação será comprometido. Isso faz com que, durante o processo de busca, tais soluções percam competitividade. Na validação cruzada, esse processo de divisão dos dados é repetido várias vezes, e o valor considerado como a função objetivo é a média da métrica de desempenho do modelo nos conjuntos de validação.

A segunda função objetivo está relacionada ao Principio da Parcimônia, ou seja, encontrar modelos mais simples. Procuramos encontrar ANNs com o menor número de parâmetros ajustáveis. Como mencionado anteriormente, arquiteturas muito simples, com um menor número de parâmetros, tendem a apresentar comportamento de subajuste, enquanto arquiteturas muito complexas, com um maior número de parâmetros, tendem a apresentar sobreajuste. Entretanto, é importante observar que existem ANNs que não manifestam esses comportamentos de subajuste ou sobreajuste, mas ainda possuem um excesso de parâmetros ajustáveis. Assim, ao definir essa função objetivo, nosso objetivo é encontrar ANNs que possuam ótimo poder preditivo e capacidade de generalização, sem sofrerem de subajuste ou sobreajuste, ao mesmo tempo que sejam simples, com poucos

parâmetros para ajustar. A Figura 49 ilustra duas arquiteturas bastante similares, diferindo por apenas um neurônio, juntamente com suas quantidades de parâmetros ajustáveis.

Figura 49 – Exemplo de arquiteturas e suas quantidades de parâmetros ajustáveis.



Fonte: Elaborada pelo autor (2023).

Ao aderir ao Princípio da Parcimônia e alcançarmos arquiteturas mais simples, usufruímos da vantagem de que elas se tornam mais ágeis tanto no processo de treinamento quanto em sua inferência. Além disso, isso nos possibilita integrá-las em dispositivos com recursos computacionais limitados para armazenamento em memória. Também se observa uma redução nos impactos ambientais, uma vez que o tempo de utilização dos recursos computacionais é reduzido.

6.3 Conjunto de Funções de Ativação

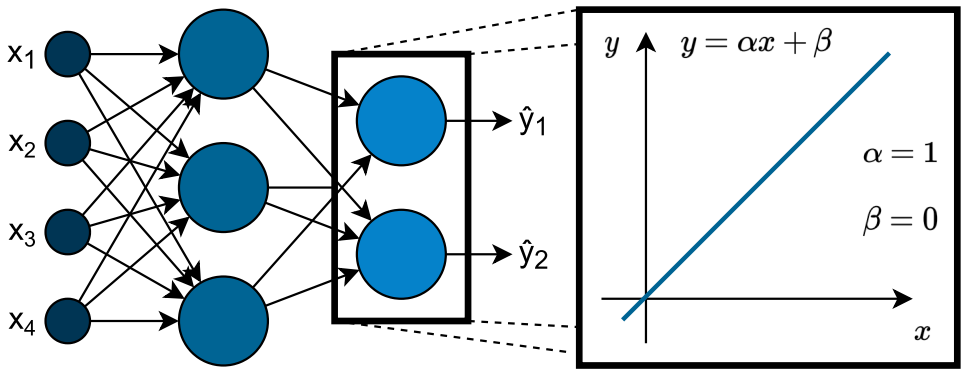
Conforme descrito na definição da solução, algumas das variáveis da solução estão relacionadas ao identificador da AF. Para que esse processo de busca ocorra, o usuário precisa definir uma lista com as possíveis AFs que deseja avaliar. Como exemplo, em um problema, podemos definir a busca entre as funções Sigmoid, Linear e Swish, enquanto em outro problema, podemos definir a busca entre todas as funções descritas na seção 5.4. Esse procedimento nos permite realizar a avaliação de diferentes funções, desde as AFs tradicionais até as AAFs, identificando quais apresentam, ao final da busca, o melhor desempenho preditivo e a menor complexidade de arquitetura.

Definimos quatro possíveis listas de funções neste trabalho, denominadas como tipo adaptativo. A lista a seguir detalha as diferenças entre os tipos adaptativos:

- Normal: Trata-se de uma lista de AF tradicionais, que não são adaptadas durante a etapa de treinamento de um modelo. Essa abordagem é a tradicional na maioria dos estudos sobre ANN.
- Camada: Trata-se de uma lista de AAFs, que são adaptadas durante a etapa de treinamento de um modelo, similar a adaptação dos pesos e tendências. Nessa abordagem adaptativa, as AAFs são adaptadas por camada, onde seus parâmetros adaptativos são ajustados para todos os neurônios de forma única e compartilhada.
- Neurônio: Trata-se de uma lista de AAFs, que são adaptadas durante a etapa de treinamento de um modelo, similar a adaptação dos pesos e tendências. Nessa abordagem adaptativa, as AAFs são adaptadas por neurônio, onde seus parâmetros adaptativos são ajustados individualmente para cada um dos neurônios da rede.
- Mistura: Trata-se de uma lista que combina as diferentes abordagens descritas anteriormente. Essa abordagem, apesar de possível, não foi explorada neste trabalho.

As Figuras 50, 51 e 52 ilustram os três primeiros tipos adaptativos utilizados neste estudo. A Figura 50 apresenta uma função linear na camada de saída, onde os valores de α e β permanecem inalterados. A Figura 51 apresenta uma função linear adaptativa na camada de saída, onde os valores de α e β se alteraram durante o processo de treinamento para toda a camada. Por fim, a Figura 52 mostra duas funções lineares adaptativas na camada de saída, uma para cada neurônio, onde os valores de α e β se alteraram durante o processo de treinamento.

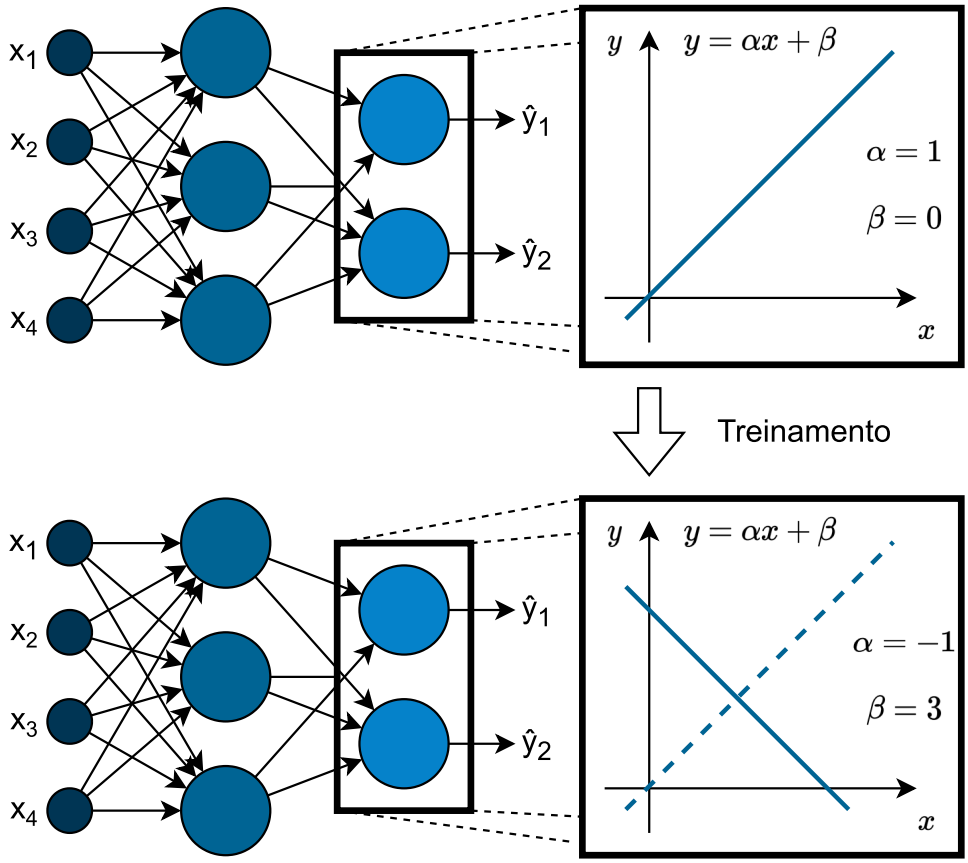
Figura 50 – Exemplo do tipo adaptativo Normal.



Fonte: Elaborada pelo autor (2023).

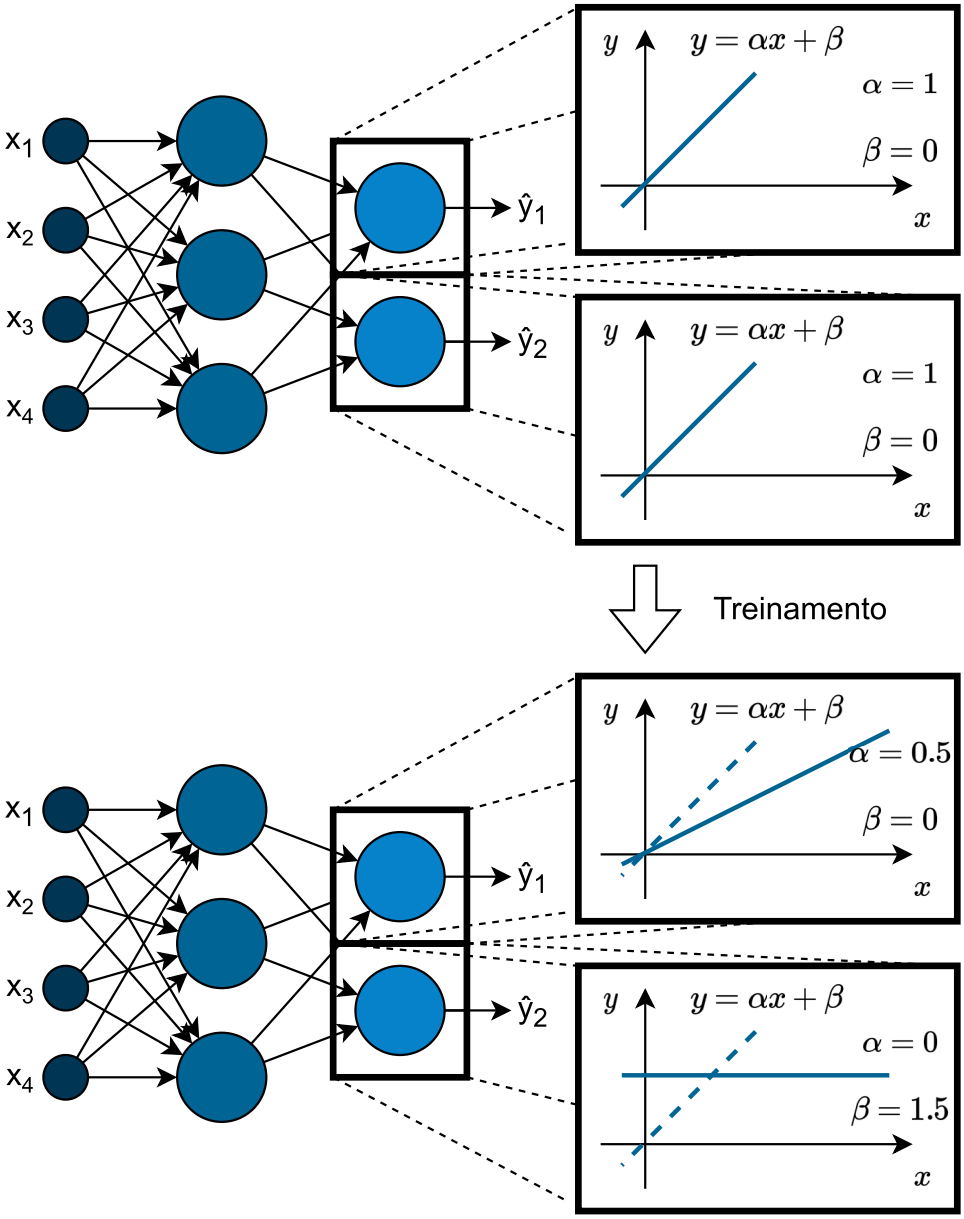
Com todas as etapas da abordagem proposta definidas, conseguimos, por meio de um conjunto de experimentos, validar a hipótese de que é possível descobrir neurônios mais avançados com AAF, os quais se adaptariam melhor aos problemas, reduzindo a necessidade de outros neurônios e, por consequência, simplificando a arquitetura da ANN.

Figura 51 – Exemplo do tipo adaptativo Camada.



Fonte: Elaborada pelo autor (2023).

Figura 52 – Exemplo do tipo adaptativo Neurônio.



Fonte: Elaborada pelo autor (2023).

6.4 Pegadas de Carbono

É fundamental, com a evolução dos algoritmos/modelos, medir e reportar de maneira transparente a pegada de carbono das soluções desenvolvidas, de modo que os cientistas possam desempenhar um papel ativo na sustentabilidade ambiental (LOTTICK et al., 2019).

As emissões de dióxido de carbono (CO_2) podem ser calculadas como sendo a intensidade de carbono da eletricidade consumida para computação (quantificada de gramas de CO_2 emitidos por quilowatt-hora de eletricidade) multiplicado pela potência consumida pela infraestrutura de computação (quantificada em quilowatt-hora) (SCHMIDT et al., 2020).

A intensidade de carbono da eletricidade consumida é calculada como uma média ponderada das emissões das diferentes fontes de energia que são usadas para gerar eletricidade, incluindo combustíveis fósseis e energias renováveis. Os combustíveis fósseis - carvão, petróleo e gás natural - estão ligados a uma quantidade específica de emissão de dióxido de carbono para cada quilowatt-hora de eletricidade gerada. Além disso, fontes renováveis ou de baixo carbono incluem energia solar, hidroeletricidade, biomassa, geotérmica e outras.

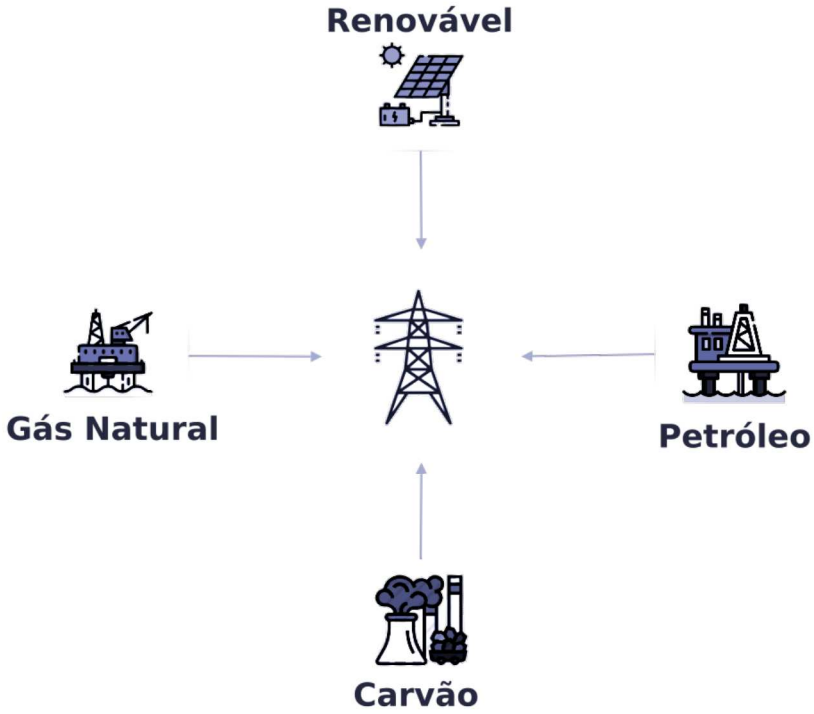
A rede de energia mais próxima possui uma mistura de combustíveis fósseis e fontes de energia de baixo carbono, denominada Mistura de Energia. A Figura 53, extraída e adaptada de Schmidt et al. (2020), apresenta uma esquematização da rede de energia composta por diferentes fontes de energia.

Com base na combinação de fontes de energia na rede local, é possível estimar a quantidade de poluição presente na eletricidade que utilizamos. À medida que a utilização de aparelhos eletrônicos aumenta, a poluição também aumenta. No entanto, é importante buscar soluções eficientes e com melhor desempenho, pois isso não apenas reduz o consumo de energia, mas também contribui para a diminuição da poluição.

No contexto deste trabalho, foi adotada a quantidade de parâmetros do modelo como uma medida substituta (*proxy*) para tentar reduzir o consumo de energia ao optar por modelos mais simples. Entretanto, utilizamos o pacote CodeCarbon (SCHMIDT et al., 2020), que utiliza a mesma formulação de cálculo descrito anteriormente, para avaliar o impacto de carbono nos experimentos numéricos que serão discutidos no Capítulo 7.

Dessa forma, este trabalho busca alcançar um equilíbrio entre a eficiência energética e o desempenho dos modelos, valorizando a consciência ambiental.

Figura 53 – Mistura de Energia: Combustíveis Fósseis e Fontes de Baixo Carbono na Rede de Energia.



Fonte: Extraída e adaptada de Schmidt et al. (2020).

7 EXPERIMENTOS COMPUTACIONAIS

Este capítulo é dedicado a apresentar os experimentos computacionais realizados e os resultados, discussões e conclusões obtidas. As seções 7.1 e 7.2 apresentam os experimentos que compararam a eficiência das AAFs em relação às AFs em problemas de classificação e regressão. A seção 7.3 apresenta o experimento que combinou abordagens evolutivas com AAFs para encontrar o melhor modelo em termos de métrica e complexidade em problemas de Autoencoder. Por fim, no experimento da seção 7.4, estendeu-se a aplicação das abordagens evolutivas combinadas com AAFs, desta vez voltadas para otimizar o modelo em termos de métrica e complexidade em um contexto real de detecção de tumores cerebrais.

Todas as tabelas e figuras apresentados aqui utilizam um mapa de cores com brilho linearmente decrescente (ou crescente). Isso significa que as informações serão preservadas se impressas em preto e branco ou visualizadas por pessoas com deficiência de daltonismo.

Além disso, os valores apresentados com a cor **"Azul Escuro"** representam os melhores valores encontrados, enquanto os valores apresentados com a cor **"Azul Claro"** representam os piores valores encontrados.

7.1 EXPERIMENTO I - AVALIAÇÃO DAS FUNÇÕES ADAPTATIVAS PARA PROBLEMAS DE CLASSIFICAÇÃO

Este experimento numérico visa realizar a avaliação das funções de ativação adaptativas para um conjunto de dez problemas de classificação. O conjunto de dados foi obtido por meio do "Bancos de Teste de Aprendizado de Máquina da Universidade da Pensilvânia" (*Penn Machine Learning Benchmarks* - PMLB), uma extensa coleção de *benchmark* para avaliação e comparação de técnicas de aprendizado de máquina. Para o processo de avaliação das técnicas, realizamos a separação aleatória de 20% do conjunto de dados de forma estratificada para teste e os 80% restantes para etapa de treino. A Tabela 4 apresenta um resumo das bases de dados utilizadas no primeiro experimento numérico de classificação, contendo seu nome, quantidade utilizada no treinamento e teste, o número de características e a quantidade de classes.

A comparação das habilidades de aprendizagem das AFs e das AAFs foi realizada usando a mesma estrutura de arquitetura (número de neurônios por camada) para cada base de dados estudada. Essa escolha garante que, ao comparar as estratégias, as arquiteturas subjacentes sejam idênticas em termos de número de camadas e neurônios, sendo a presença de parâmetros adaptativos dentro das AFs o único fator diferenciador. Além disso, criamos um modelo de ANN baseado na Máquina de Aprendizado Extremo (*Extreme Learning Machine* - ELM) (HUANG et al., 2006), onde os pesos das camadas internas são congelados, com apenas uma camada oculta contendo o número de neurônios igual ao dobro do número

Tabela 4 – Sumarização das bases de dados de classificação utilizadas no primeiro experimento numérico.

Base de Dados	Problema	Instâncias de Treinamento	Instâncias de Teste	Qtd. de Características	Qtd. de Classes
1	analcatdata_lawsuit	211	53	4	2
2	australian	552	138	14	2
3	breast_w	559	140	9	2
4	buggyCrx	552	138	15	2
5	dna	2548	638	180	3
6	hypothyroid	2530	633	25	2
7	monk1	444	112	6	2
8	penguins	266	67	7	3
9	sonar	166	42	60	2
10	threeOf9	409	103	9	2

Fonte: Elaborada pelo autor (2023).

de características da base de dados modelada. A escolha deste modelo como base se deu pelo fato de ser um modelo simples e de fácil treinamento, atingindo resultados bastante relevantes na literatura. Além disso, a escolha da arquitetura foi feita de forma empírica pensando em modelos compactos, sem a necessidade de muitos neurônios ou camadas.

Para todos os experimentos realizados, variando a base de dados, a AF e o tipo adaptativo (Normal, Camada e Neurônio), nós treinamos uma ANN usando o otimizador Adam (KINGMA; BA, 2014) durante 500 épocas com um mini-lote de 32 e uma taxa de aprendizado de 10^{-4}, que foi ajustada a cada dez épocas usando um fator multiplicativo de 0.99. Todos estes valores também foram definidos de forma empírica. Realizar a combinação do otimizador Adam com a taxa de aprendizado ajustada a cada dez épocas permite a adaptabilidade das taxas de aprendizado individuais do Adam, ao mesmo tempo em que fornece um ajuste global e gradual, resultando em melhor estabilidade, capacidade de ajuste fino e prevenção de superajuste durante o processo de treinamento (LOSHCHILOV; HUTTER, 2018). Realizamos 30 execuções independentes de treinamento e teste, calculando a métrica F1 e a *Log Loss* para cada uma delas.

As Tabelas 18, 19, 20, 21, 22, 23, 24, 25, 26 e 27, presentes no Anexo A, apresentam os resultados sumarizados para cada função de ativação e tipo adaptativo da ANN, exibindo para as métricas F1 e *Log Loss* o valor mínimo, médio, mediano e máximo, além do desvio padrão. É possível observar que, em geral, as AAFs possuem melhores valores em comparação com as AFs, indicando que podem maximizar melhor a métrica F1 e minimizar melhor o *Log Loss*. Por outro lado, é possível observar que os resultados das estratégias adaptativas por Camada e Neurônio se alternam em relação a qual é melhor, dependendo da base de dados, da AF e da métrica de avaliação, não sendo possível definir qual estratégia é melhor de forma geral.

Para complementar essas informações, as Figuras 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87 e 88, também presentes no Anexo A, representam os *Violin plots* das métricas F1 e *Log Loss* para os tipos adaptativos das funções de ativação, considerando as 30 execuções independentes. Optamos por usar os *Violin plots* porque essa visualização combina as informações presentes em um *boxplot* juntamente com um *plot* de densidade.

Com base nas informações fornecidas no Anexo A até o momento, a Tabela 5 apresenta a frequência, em percentagem, com que cada tipo de adaptação obteve resultados classificados como "Pior", "Intermediário" e "Melhor" para cada métrica no Experimento 1. Os resultados demonstram que, em geral, o tipo adaptativo Neurônio apresentou a frequência mais alta na categoria "Melhor" para ambas as métricas, indicando sua eficácia em comparação com os outros tipos adaptativos. Em contrapartida, o tipo adaptativo Normal teve uma alta frequência na categoria "Pior" para ambas as métricas, sugerindo que essa abordagem pode ser menos eficaz nesse aspecto. Ao analisarmos o tipo adaptativo Camada, é evidente que ele proporciona desempenho superior em comparação com o tipo adaptativo Normal em todas as situações, porém, apresenta desempenho inferior em relação ao tipo adaptativo Neurônio, também em todos os cenários.

Tabela 5 – Comparação Percentual de Desempenho por Tipo Adaptativo para o Experimento 1.

Tipo	F1			Log Loss		
Adaptativo	Pior	Intermediário	Melhor	Pior	Intermediário	Melhor
Normal	**17.38%**	10.48%	5.48%	**30.71%**	0.0	2.62%
Camada	11.67%	7.62%	14.05%	2.86%	13.57%	16.90%
Neurônio	7.14%	4.52%	**21.67%**	0.0%	12.86%	**20.48%**

Fonte: Elaborada pelo autor (2023).

Para auxiliar na visualização e compreensão dos resultados obtidos, utilizamos os Perfis de Desempenho, descrito na Subseção 3.3.5.4. O perfil de desempenho é uma distribuição cumulativa de uma métrica de desempenho usada para avaliar várias soluções para o mesmo conjunto de problemas e identificar, entre todas as soluções avaliadas, aquela que produz os melhores resultados. Em termos gerais, a estratégia que apresenta a maior área sob a curva é considerada a melhor entre as estratégias analisadas, levando em consideração os problemas em que foram aplicadas para resolução.

A Tabela 6 e as Figuras 54 e 55 exibem os resultados dos Perfis de Desempenho para diferentes funções de ativação, tipos adaptativos e as dez bases de dados de classificação estudadas, considerando a métrica F1. A Figura 54 apresenta os Perfis de Desempenho construídos usando o valor médio obtido nas 30 execuções independentes, enquanto a Figura 55 apresenta os Perfis de Desempenho construídos usando o melhor valor obtido (o valor máximo). É possível observar que a estratégia que utiliza AAFs (Camada e Neurônio) apresenta melhores resultados nas bases de dados estudadas em comparação

com a estratégia que usa apenas AF tradicional (Normal). Além disso, a estratégia por Neurônio apresenta melhores resultados quando usamos a média como medida, enquanto a estratégia por Camada apresenta melhores resultados quando usamos o melhor valor como medida.

Tabela 6 – Área normalizada sob as curvas dos perfis de desempenho do primeiro experimento para métrica F1.

Função de Ativação	Média			Máximo		
	Normal	Camada	Neurônio	Normal	Camada	Neurônio
BLU	93.70%	**86.14%**	**98.94%**	88.43%	**92.47%**	**85.57%**
Cúbico	**74.26%**	**98.07%**	93.92%	**76.66%**	**100.00%**	92.02%
Linear	**87.03%**	95.17%	**98.48%**	**87.83%**	**91.93%**	90.43%
Mish	85.99%	**85.14%**	**95.87%**	**87.06%**	**99.45%**	90.15%
PELU	87.96%	**84.89%**	**97.16%**	**86.23%**	85.42%	**80.76%**
PReLU	**77.96%**	88.62%	**94.97%**	**82.91%**	**93.25%**	92.89%
Quadrática	**70.70%**	**93.66%**	88.98%	**87.18%**	**97.83%**	92.93%
SELU	92.47%	**86.61%**	**98.77%**	**80.22%**	90.14%	**95.69%**
Sigmóide	**88.78%**	98.07%	**98.43%**	**89.48%**	**99.80%**	96.44%
SoftExponential	**88.21%**	95.86%	**99.12%**	92.51%	**94.30%**	**87.44%**
SoftPlus	**84.80%**	93.84%	**94.69%**	**86.11%**	**95.32%**	88.79%
SoftSign	88.63%	**83.99%**	**95.14%**	**89.76%**	**93.59%**	91.94%
Swish	83.14%	**82.55%**	**96.04%**	**90.28%**	**97.02%**	95.88%
Tangente Hiperbólica	89.48%	**84.29%**	**98.07%**	93.98%	**91.87%**	**95.07%**

Fonte: Elaborada pelo autor (2023).

A Tabela 7 e as Figuras 56 e 57 apresentam os resultados dos Perfis de Desempenho para diferentes funções de ativação, tipos adaptativos e as dez bases de dados de classificação estudadas, considerando a métrica *Log Loss*. A Figura 56 exibe os Perfis de Desempenho construídos usando o valor médio obtido nas 30 execuções independentes, enquanto a Figura 57 exibe os Perfis de Desempenho construídos usando o melhor valor obtido (o valor mínimo). É possível observar que a estratégia que utiliza AAFs (Camada e Neurônio) apresenta melhores resultados nas bases de dados estudadas em comparação com a estratégia que usa apenas AF tradicional (Normal). Além disso, a estratégia por Camada demonstrou ser predominantemente melhor do que a estratégia por Neurônio, independentemente de ser calculado usando a média ou o melhor valor.

Tabela 7 – Área normalizada sob as curvas dos perfis de desempenho do primeiro experimento para métrica *Log Loss*.

Função de Ativação	Média			Mínimo		
	Normal	Camada	Neurônio	Normal	Camada	Neurônio
BLU	86.14%	**97.36%**	94.77%	86.79%	**98.97%**	92.38%
Cúbico	85.96%	**99.32%**	95.65%	85.56%	**99.56%**	94.88%
Linear	87.18%	**98.88%**	94.96%	85.63%	**99.54%**	94.30%
Mish	88.50%	**98.93%**	97.10%	87.18%	**99.26%**	95.53%
PELU	88.33%	**98.97%**	96.84%	86.43%	**99.92%**	94.39%
PReLU	89.00%	**98.92%**	97.89%	88.63%	**99.03%**	96.96%
Quadrática	86.81%	96.79%	**97.63%**	78.66%	**98.69%**	88.75%
SELU	86.55%	**98.58%**	96.34%	85.94%	**99.72%**	92.93%
Sigmóide	84.29%	**99.14%**	93.36%	87.94%	**99.71%**	97.05%
SoftExponential	85.55%	**97.38%**	94.25%	85.19%	**98.72%**	93.26%
SoftPlus	90.62%	**99.39%**	99.07%	89.75%	**100.00%**	97.32%
SoftSign	88.90%	**98.94%**	97.24%	87.23%	**99.43%**	95.74%
Swish	88.06%	**99.16%**	96.94%	87.65%	**99.24%**	95.73%
Tangente Hiperbólica	87.76%	**98.93%**	96.76%	85.92%	**99.43%**	93.40%

Fonte: Elaborada pelo autor (2023).

Por fim, considerando o Teorema Central do Limite (PÓLYA, 1920; FISCHER, 2011), onde, com uma amostra maior, a distribuição amostral da média se aproxima da normalidade, e essa condição geralmente é atendida se o tamanho da amostra for maior do que 25/30 (HOGG et al., 2010), podemos então assumir que a distribuição dos resultados se aproxima de uma distribuição normal. Isso nos permite utilizar alguns testes estatísticos que dependem de a distribuição ser normal, a fim de ampliar a análise comparativa dos resultados.

Desta forma, realizamos uma Análise de Variância simples (*One-way Analysis of Variance* - ANOVA) (FISHER, 1919) com um intervalo de confiança de 95%. O ANOVA tem como objetivo identificar, estatisticamente, a hipótese nula de que não há diferença significativa entre as médias de cada grupo (tipo adaptativo). Os resultados obtidos indicaram que, em alguns casos, o teste rejeitou a hipótese nula e que existem grupos adaptativos que apresentam desempenho diferente. Apesar disso, o ANOVA não é capaz de identificar quais são os grupos que possuem diferença nas médias.

Para isso, utilizamos o teste de Tukey (TUKEY, 1977), que realiza a comparação de todos os possíveis pares de médias, com um intervalo de confiança de 95%, para verificar onde a diferença entre as médias é significativa. A Figura 58 mostra uma contagem,

com valor máximo igual ao número de bases de dados, das situações em que os tipos de funções adaptativas apresentaram diferença média significativa de acordo com o teste de Tukey, possibilitando identificar em quantos problemas os tipos adaptativos apresentaram resultados diferentes.

É possível observar que as funções Cúbicas e Quadráticas apresentaram uma maior contagem de problemas que demonstraram uma diferença média significativa de acordo com o teste de Tukey, o que indica que, para as bases analisadas, o uso de uma AAF é relevante. Por outro lado, as funções Sigmoides e SoftPlus apresentaram a menor contagem, o que torna o uso de uma AAF menos relevante. As outras funções apresentaram um desempenho intermediário, o que sugere que as AAFs podem trazer benefícios para alguns problemas.

Ao compilar todas as análises realizadas, chegamos à conclusão de que, para as bases analisadas, a estratégia com AAF (Camada e Neurônio) é superior à estratégia AF tradicional (Normal). Além disso, a estratégia adaptativa por Camada foi superior à estratégia adaptativa por Neurônio. Apesar de a estratégia adaptativa por Neurônio apresentar mais parâmetros adaptativos, permitindo-lhe um ajuste maior do que a estratégia adaptativa por Camada, ela demonstrou desempenho inferior. Isso pode ser atribuído ao fato de que essa estratégia está mais suscetível ao sobreajuste nos dados.

7.1.1 Pegadas de Carbono

Conforme descrito na seção 6.4, utilizamos o pacote CodeCarbon (SCHMIDT et al., 2020) para avaliar o impacto de carbono nos experimentos numéricos. A análise da pegada de carbono para este experimento foi realizada na cidade de São Paulo, localizada no estado de São Paulo, Brasil. A máquina utilizada para a análise estava equipada com as seguintes configurações: um processador Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, com 12 núcleos de CPU, e 16,0 GiB de memória RAM. A parte gráfica contava com uma placa de vídeo NVIDIA GeForce RTX 2060, apesar de não ter sido utilizada. O sistema operacional empregado foi o Ubuntu 22.04.2 LTS. Além disso, o experimento foi executado utilizando o Python na versão 3.7.13. Ao se tratar de Pegadas de Carbono, todos os experimentos posteriores utilizaram esta mesma configuração para gerar os estudos e, por este motivo, não serão apresentados novamente.

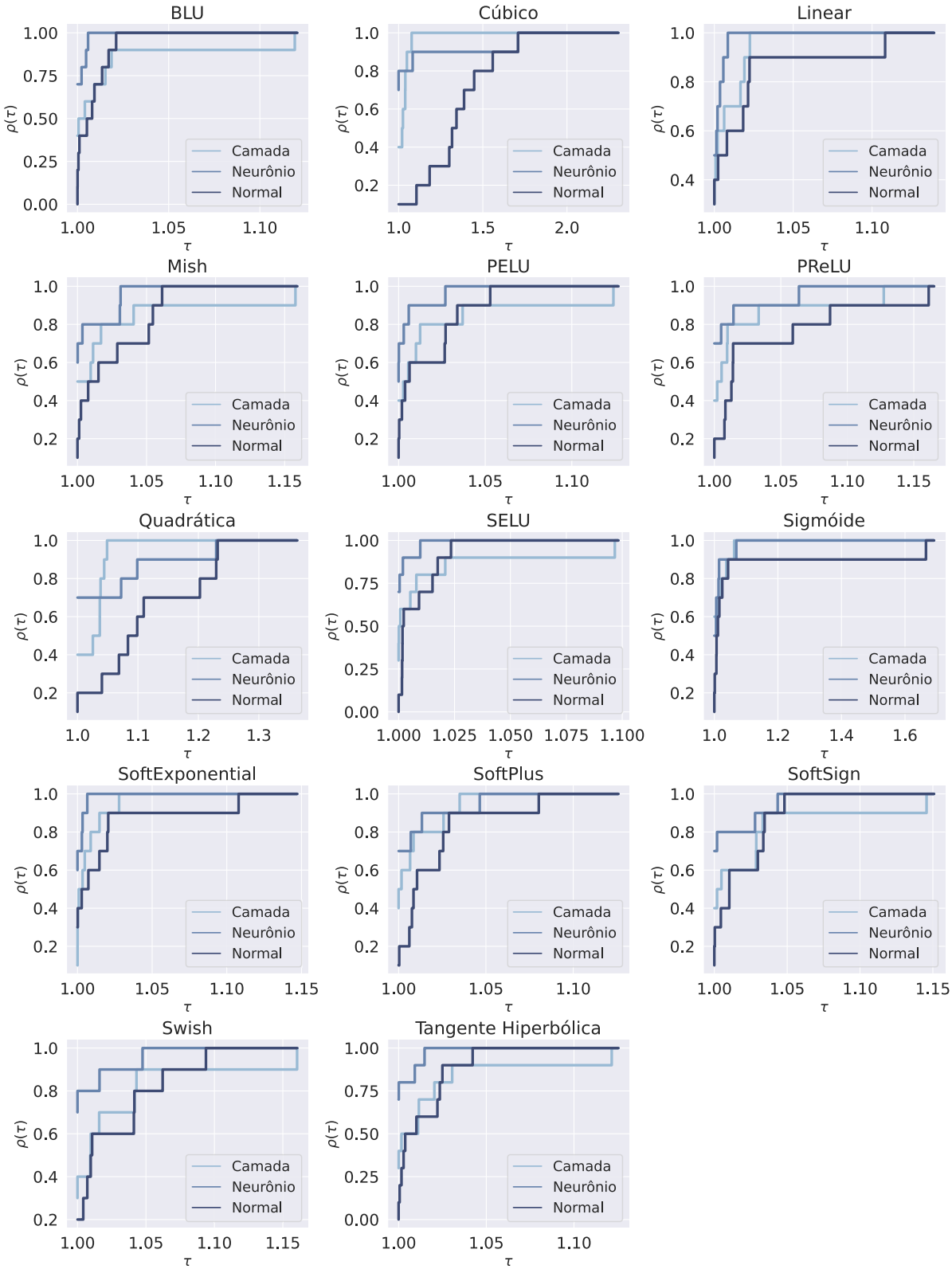
A Tabela 28, presente no Anexo A, exibe as estimativas médias de emissão de CO_2, em quilogramas, associadas ao treinamento de cada modelo no primeiro experimento numérico. Essa média é proveniente de cinco execuções focadas na estimativa da emissão de CO_2. É possível observar que, para este experimento, onde a arquitetura é fixa e avaliamos apenas o impacto da função adaptativa, a estratégia Normal apresentou o menor valor de emissão de CO_2 na maioria dos casos, enquanto a estratégia Neurônio apresentou o maior valor. Apesar disso, a diferença entre eles é muito pequena, sendo menor do que

1×10^{-4}.

Com o intuito de fazer uma comparação, uma carga de telefone celular é estimada em emitir 8.22 g, uma milha percorrida de carro é estimada em emitir 0.398 kg, um galão de gasolina consumido é estimado em emitir 8.887 kg, um barril de petróleo bruto consumido é estimado em emitir 430 kg, o uso médio de energia de uma residência é estimado em emitir 8300 kg e um vagão de trem de carvão é estimado em emitir 181290 kg, tudo baseado em consumos nos Estados Unidos (DODGE et al., 2022).

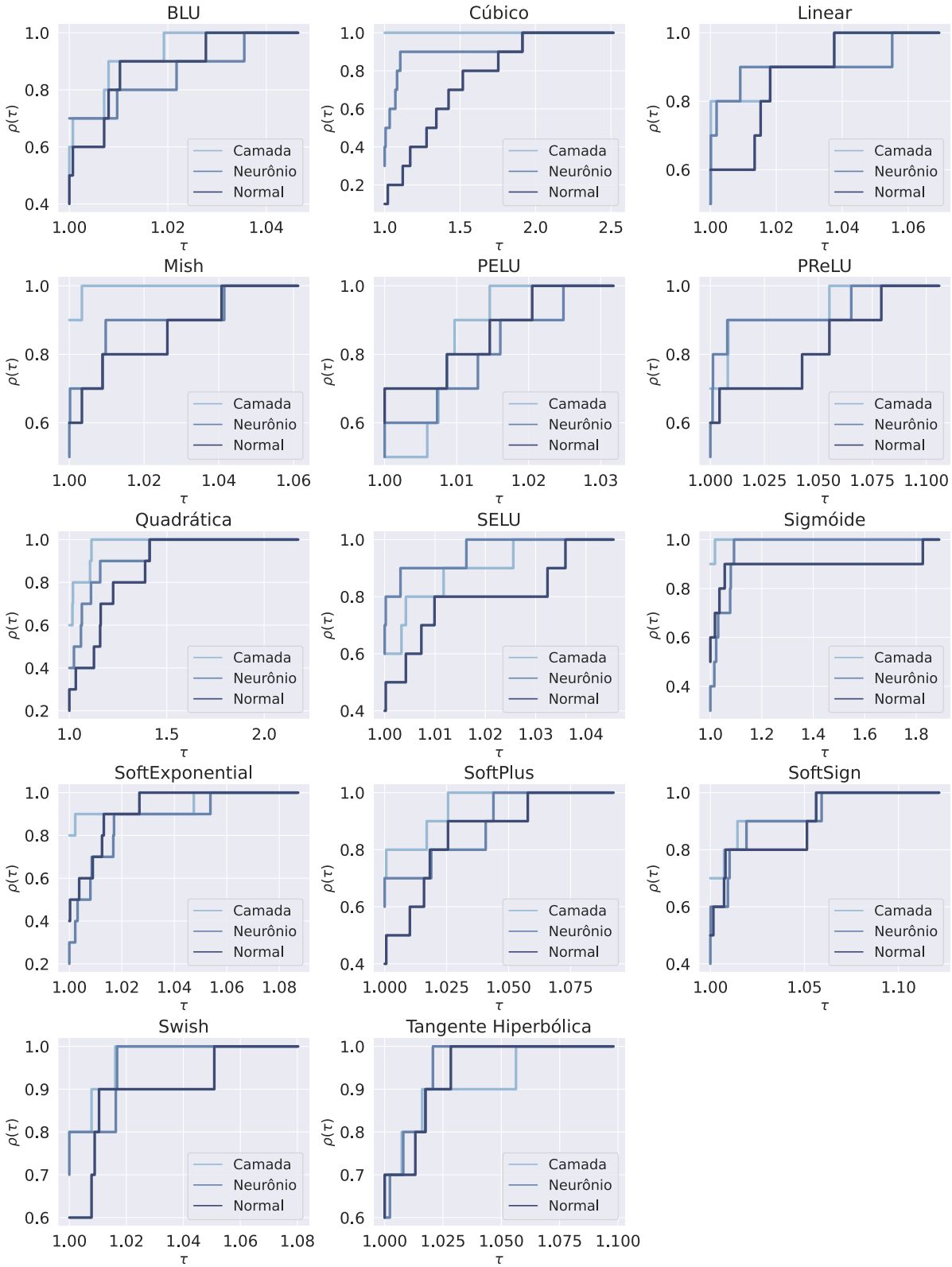
Apesar de as estratégias adaptativas apresentarem maior emissão de CO_2, o valor máximo registrado foi de 6.42×10^{-5} kg, representando menos de 1% do que é gasto com o carregamento de um telefone.

Figura 54 – Curvas dos perfis de desempenho do primeiro experimento para métrica F1 utilizando o valor médio.



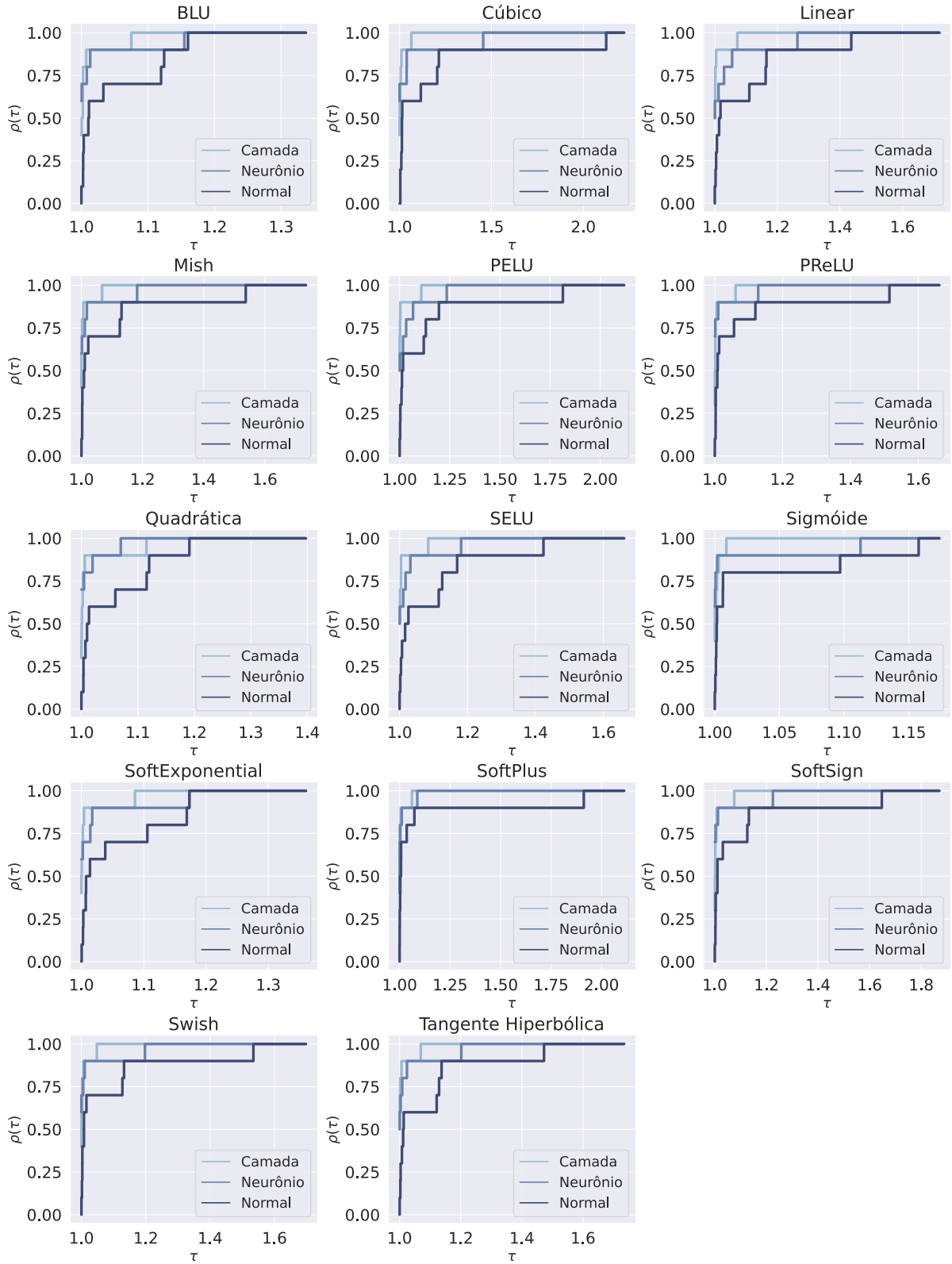
Fonte: Elaborada pelo autor (2023).

Figura 55 – Curvas dos perfis de desempenho do primeiro experimento para métrica F1 utilizando o valor máximo.



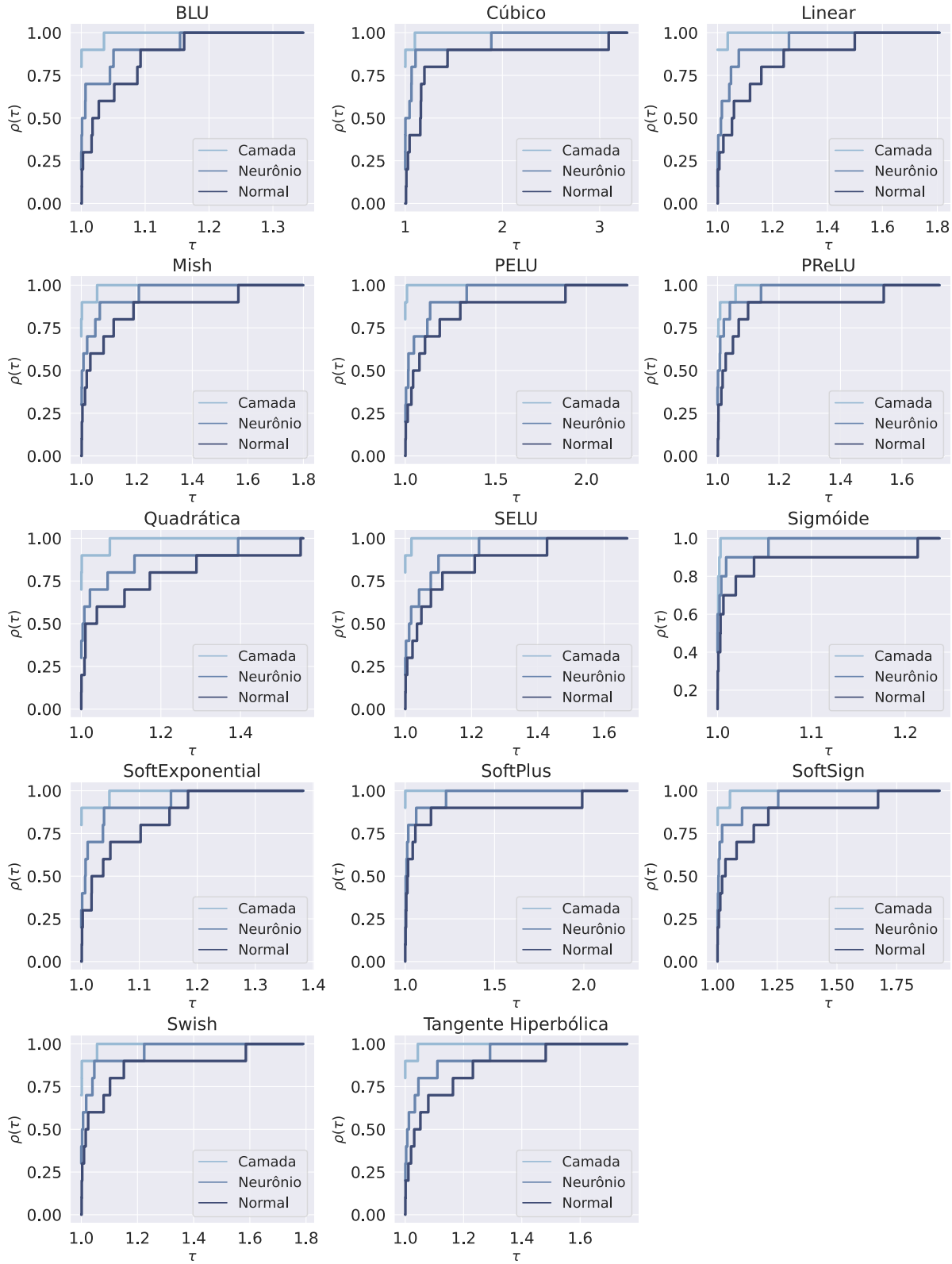
Fonte: Elaborada pelo autor (2023).

Figura 56 – Curvas dos perfis de desempenho do primeiro experimento para métrica *Log Loss* utilizando o valor médio.



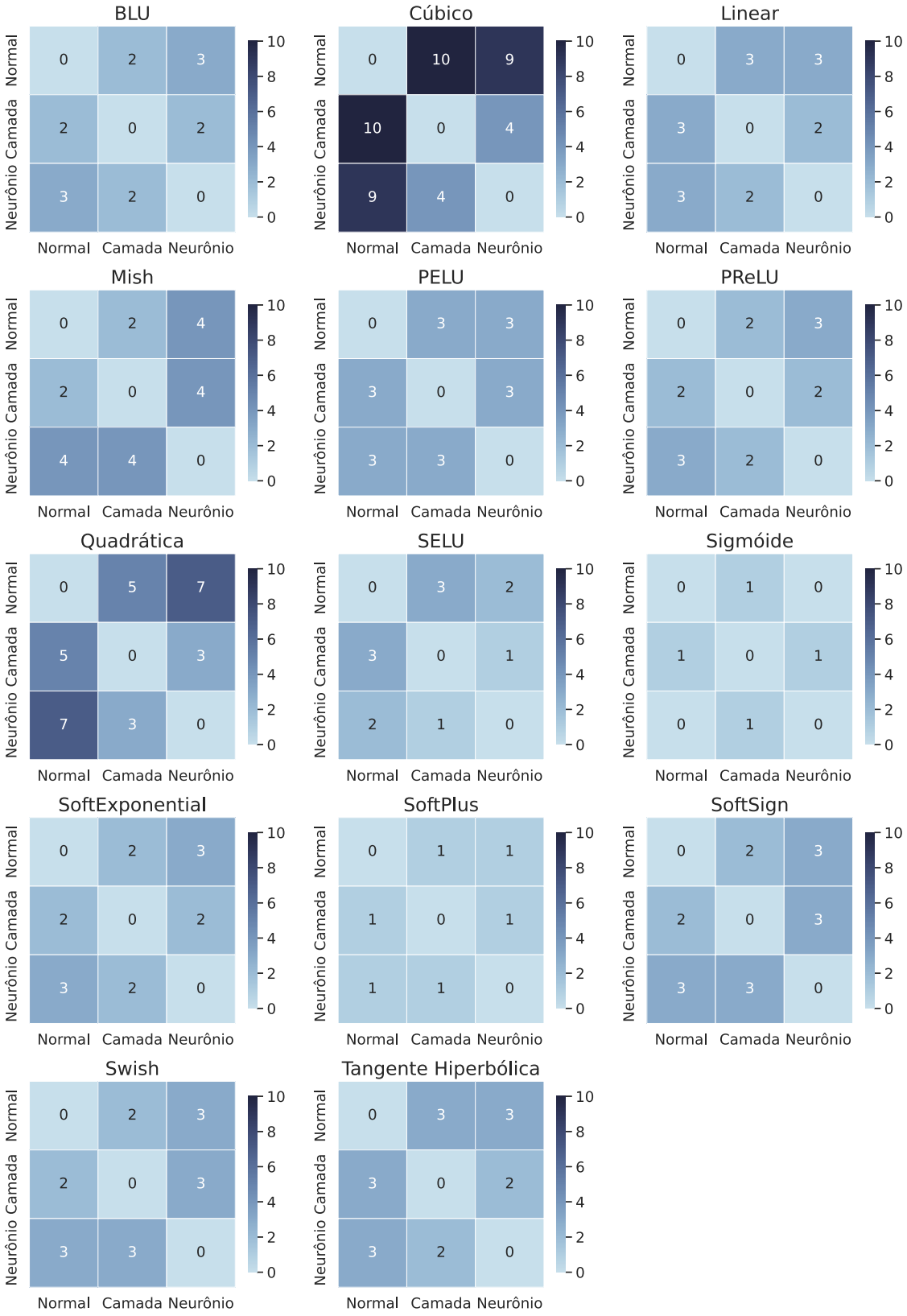
Fonte: Elaborada pelo autor (2023).

Figura 57 – Curvas dos perfis de desempenho do primeiro experimento para métrica *Log Loss* utilizando o valor mínimo.



Fonte: Elaborada pelo autor (2023).

Figura 58 – Contagem dos tipos de funções adaptativas que apresentaram diferença média significativa de acordo com o teste de Tukey para o primeiro experimento.



Fonte: Elaborada pelo autor (2023).

7.2 EXPERIMENTO II - AVALIAÇÃO DAS FUNÇÕES ADAPTATIVAS PARA PROBLEMAS DE REGRESSÃO

Este experimento numérico tem como objetivo realizar a avaliação das funções de ativação adaptativas para um conjunto de dez problemas de regressão. Assim como foi feito no problema de classificação, obtemos o conjunto de dados por meio do PMLB (OLSON et al., 2017). No processo de avaliação das técnicas, realizamos a separação aleatória de 20% do conjunto de dados para teste e os 80% restantes para etapa de treino. A Tabela 8 apresenta um resumo das bases de dados utilizadas no segundo experimento numérico de regressão, contendo seus nomes, quantidades utilizadas no treinamento e teste, e o número de características.

Tabela 8 – Sumarização das bases de dados de regressão utilizadas no segundo experimento numérico.

Base de Dados	Problema	Instancias de Treinamento	Instancias de Teste	Qtd. de Características
1	503_wind	5259	1315	14
2	1027_ESL	390	98	4
3	1028_SWD	800	200	10
4	1029_LEV	800	200	4
5	1030_ERA	800	200	4
6	1089_USCrime	37	10	13
7	1096_FacultySalaries	40	10	4
8	4544_GeographicalOriginalofMusic	847	212	117
9	294_satellite_image	5148	1287	36
10	573_cpu_act	6553	1639	21

Fonte: Elaborada pelo autor (2023).

Adotando o mesmo procedimento do primeiro experimento, a comparação das habilidades de aprendizagem das AFs e das AAFs foi realizada utilizando a mesma estrutura de arquitetura (número de neurônios por camada) para cada base de dados estudada. Essa escolha assegura que, ao comparar as estratégias, as arquiteturas subjacentes sejam idênticas em termos de número de camadas e neurônios, sendo a presença de parâmetros adaptativos dentro das AFs o único fator diferenciador. Adicionalmente, foi desenvolvido um modelo de ANN baseado no ELM (onde os pesos das camadas internas são congelados), com apenas uma camada oculta contendo o número de neurônios igual ao dobro do número de características da base de dados modelada. A escolha deste modelo como base se deu pelo fato de ser um modelo simples e de fácil treinamento, atingindo resultados bastante relevantes na literatura. Além disso, a escolha da arquitetura foi feita de forma empírica pensando em modelos compactos, sem a necessidade de muitos neurônios ou camadas.

Para todos os experimentos realizados, nos quais variamos a base de dados, a AF e o tipo adaptativo (Normal, Camada e Neurônio), nós treinamos uma ANN usando o

otimizador Adam (KINGMA; BA, 2014) ao longo de 500 épocas, com um mini-lote de 32 e uma taxa de aprendizado de 10^{-4}, a qual foi ajustada a cada dez épocas por meio de um fator multiplicativo de 0.99. Todos estes valores também foram definidos de forma empírica. Realizamos 30 execuções independentes de treinamento e teste, calculando as métricas R^2 e RMSE para cada uma dessas execuções.

As Tabelas 29, 30, 31, 32, 33, 34, 35, 36, 37 e 38, presentes no Anexo B, apresentam os resultados sumarizados para cada função de ativação e tipo adaptativo da ANN, exibindo, para as métricas R^2 e RMSE, o valor mínimo, médio, mediano e máximo, além do desvio padrão. É possível observar que, em geral, as AAFs possuem melhores valores em comparação com as AFs, indicando que podem maximizar melhor a métrica R^2 e minimizar melhor o RMSE. Por outro lado, é possível observar que os resultados da estratégia adaptativa por Camada e Neurônio se igualaram em muitas comparações e se intercalam em qual era melhor, dependendo da base de dados, da AF e da métrica de avaliação. Novamente, assim como já foi apresentado no primeiro experimento, não foi possível definir qual estratégia é melhor em geral.

Para complementar estas informações, as Figuras 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101 e 102, também presentes no Anexo B, representam os *Violin plots* das métricas R^2 e RMSE para os tipos adaptativos das funções de ativação, considerando as 30 execuções independentes.

Com base nas informações fornecidas no Anexo B até o momento, a Tabela 9 apresenta a frequência, em percentagem, com que cada tipo de adaptação obteve resultados classificados como "Pior", "Intermediário" e "Melhor" para cada métrica no Experimento 2. Observa-se que o tipo adaptativo Normal apresentou uma frequência significativamente alta na categoria "Pior" para ambas as métricas, R^2 e RMSE. Em contraste, o tipo adaptativo Neurônio demonstrou um desempenho superior, sendo o mais frequente na categoria "Melhor" para ambas as métricas. O tipo adaptativo Camada também obteve bons resultados, ficando entre o Normal e o Neurônio em termos de desempenho, indicando sua eficácia nesta configuração experimental.

Tabela 9 – Comparação Percentual de Desempenho por Tipo Adaptativo para o Experimento 2.

Tipo	R^2			RMSE		
Adaptativo	Pior	Intermediário	Melhor	Pior	Intermediário	Melhor
Normal	**32.14%**	0.0%	1.19%	**32.14%**	0.0%	1.19%
Camada	0.48%	13.33%	19.52%	0.24%	13.33%	19.76%
Neurônio	0.48%	3.10%	**29.76%**	0.48%	4.05%	**28.81%**

Fonte: Elaborada pelo autor (2023).

Para fomentar as análises, decidimos mais uma vez prosseguir com a utilização dos Perfis de Desempenho (Subseção 3.3.5.4). A Tabela 10 e as Figuras 59 e 60 mostram

os resultados dos Perfis de Desempenho para as diferentes funções de ativação, tipos adaptativos e as dez bases de dados de regressão estudadas, considerando a métrica R^2. A Figura 59 apresenta os Perfis de Desempenho construídos utilizando o valor médio obtido nas 30 execuções independentes, enquanto a Figura 60 mostra os Perfis de Desempenho construídos utilizando o melhor valor obtido (o valor máximo). É possível observar que a estratégia que utiliza AAF (Camada e Neurônio) apresenta, na maioria das vezes, resultados superiores nas bases de dados estudadas quando comparada com a estratégia que utiliza apenas a AF tradicional (Normal). Além disso, a estratégia por Neurônio revela melhores resultados quando consideramos o melhor valor como critério de avaliação. Por outro lado, ao utilizar o valor médio, ambas as estratégias adaptativas apresentaram os melhores resultados, oscilando entre si.

Tabela 10 – Área normalizada sob as curvas dos perfis de desempenho do segundo experimento para Métrica R^2.

Função de Ativação	Média			Máximo		
	Normal	Camada	Neurônio	Normal	Camada	Neurônio
BLU	**88.20%**	89.02%	**89.99%**	**61.13%**	90.14%	**99.21%**
Cúbico	**92.78%**	98.69%	**98.91%**	**98.29%**	99.86%	**100.00%**
Linear	**85.96%**	85.08%	**83.99%**	**86.02%**	97.26%	**98.68%**
Mish	**85.60%**	**89.46%**	88.94%	**72.40%**	98.92%	**99.33%**
PELU	86.35%	**86.39%**	**85.64%**	**86.21%**	98.72%	**99.80%**
PReLU	**81.03%**	93.74%	**93.77%**	**80.11%**	97.67%	**99.77%**
Quadrática	**90.05%**	95.84%	**96.06%**	**84.48%**	99.94%	**100.00%**
SELU	**86.88%**	87.18%	**87.47%**	**75.53%**	94.61%	**98.52%**
Sigmóide	**6.48%**	25.30%	**25.31%**	**53.81%**	**99.63%**	99.57%
SoftExponential	85.45%	**85.54%**	**85.24%**	**69.86%**	84.10%	**95.73%**
SoftPlus	**79.50%**	**83.24%**	83.07%	**91.17%**	98.82%	**99.56%**
SoftSign	**86.54%**	**88.72%**	88.12%	**63.60%**	**99.63%**	98.65%
Swish	**85.26%**	**90.03%**	89.78%	**67.88%**	98.83%	**99.35%**
Tangente Hiperbólica	**85.08%**	**87.03%**	85.86%	**76.36%**	**99.60%**	99.46%

Fonte: Elaborada pelo autor (2023).

A Tabela 11 e as Figuras 61 e 62 mostram os resultados dos Perfis de Desempenho para as diferentes funções de ativação, diferentes tipos adaptativos e as dez bases de dados de classificação estudadas para a métrica RMSE. A Figura 61 apresenta os Perfis de Desempenho construídos utilizando o valor médio obtido nas 30 execuções independentes, enquanto a Figura 62 apresenta os Perfis de Desempenho construídos utilizando o melhor valor obtido (o valor mínimo). É possível observar que a estratégia que utiliza AAF

(Camada e Neurônio) apresenta melhores resultados nas bases de dados estudadas em comparação com a estratégia que usa apenas AF tradicional (Normal). Além disso, a estratégia por Neurônio foi predominantemente melhor do que a estratégia por Camada, independentemente se o cálculo foi realizado utilizando a média ou o melhor valor.

Tabela 11 – Área normalizada sob as curvas dos perfis de desempenho do segundo experimento para Métrica RMSE.

Função de Ativação	Média			Mínimo		
	Normal	Camada	Neurônio	Normal	Camada	Neurônio
BLU	**76.15%**	90.35%	**99.98%**	**71.78%**	92.85%	**99.66%**
Cúbico	**81.44%**	99.03%	**99.94%**	**63.28%**	98.82%	**100.00%**
Linear	**88.15%**	99.80%	**99.84%**	**75.94%**	97.96%	**98.43%**
Mish	**61.64%**	88.74%	**99.25%**	**60.39%**	93.83%	**99.39%**
PELU	**77.72%**	95.68%	**99.93%**	**82.52%**	97.04%	**99.88%**
PReLU	**65.94%**	92.98%	**98.95%**	**71.06%**	96.15%	**98.70%**
Quadrática	**76.13%**	98.28%	**100.00%**	**73.14%**	98.85%	**99.96%**
SELU	**76.90%**	92.62%	**99.89%**	**64.12%**	88.55%	**99.40%**
Sigmóide	**70.63%**	99.04%	**99.09%**	**84.11%**	99.64%	**99.67%**
SoftExponential	**74.82%**	94.07%	**100.00%**	**80.21%**	93.78%	**99.56%**
SoftPlus	**60.86%**	96.74%	**99.12%**	**60.57%**	93.37%	**98.16%**
SoftSign	**72.11%**	97.06%	**99.24%**	**78.09%**	**99.51%**	99.03%
Swish	**62.91%**	91.45%	**99.23%**	**62.32%**	94.45%	**99.78%**
Tangente Hiperbólica	**75.43%**	98.20%	**99.81%**	**67.94%**	99.43%	**99.55%**

Fonte: Elaborada pelo autor (2023).

Novamente, para ampliar a análise comparativa dos resultados, utilizamos o ANOVA com um intervalo de confiança de 95% para verificar a hipótese nula de que não há diferença significativa entre as médias de cada grupo (tipo adaptativo). Os resultados obtidos indicaram que, em alguns casos, o teste rejeitou a hipótese nula e que existem grupos adaptativos que apresentam desempenho diferente. Apesar disso, o ANOVA não é capaz de identificar quais grupos possuem diferença nas médias.

Dessa forma, empregamos o teste de Tukey, com um intervalo de confiança de 95%, para verificar onde a diferença entre as médias é significativa. A Figura 63 apresenta uma contagem, com valor máximo igual ao número de bases de dados, das situações em que os tipos de funções adaptativas demonstraram diferença média significativa, de acordo com o teste de Tukey, possibilitando identificar em quantos problemas os tipos adaptativos apresentaram resultados diferentes.

Novamente, assim como no primeiro experimento, podemos observar que as funções Cúbicas e Quadráticas apresentaram uma contagem maior de casos que exibiram diferença média significativa, de acordo com o teste de Tukey. Isso indica que, para as bases analisadas, o uso de uma AAF é relevante. Por outro lado, um grande número de funções, como a BLU, Linear, Mish, PELU, SELU, SoftExponential e Tangente Hiperbólica, apresentaram a contagem menor, diminuindo a relevância do uso de uma AAF. As funções restantes mostraram um desempenho ligeiramente melhor, mas ainda indicaram que as AAF não foram tão eficazes quanto esperado.

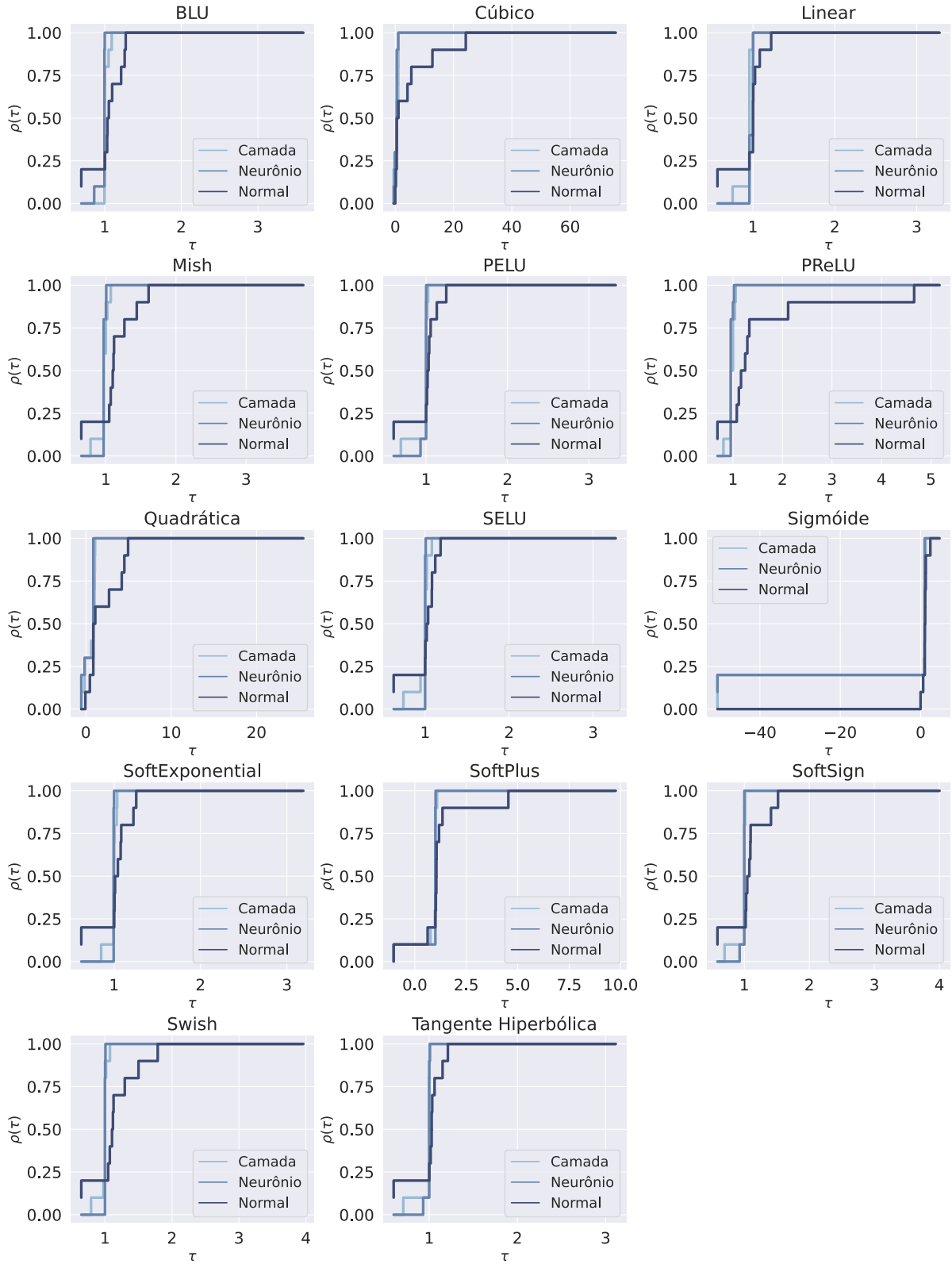
Ao compilar todas as análises realizadas, chegamos à conclusão de que, para as bases analisadas, a estratégia com AAF (Camada e Neurônio) é superior à estratégia de AF tradicional (Normal). Apesar disso, nos testes estatísticos, a diferença não é significativa. Além disso, a estratégia adaptativa por Neurônio foi superior à estratégia adaptativa por Camada. Para esses problemas, a quantidade adicional de parâmetros na estratégia adaptativa por Neurônio permitiu que ela alcançasse resultados melhores, sem sofrer com sobreajuste.

7.2.1 Pegadas de Carbono

A Tabela 39, presente no Anexo B, exibe as estimativas médias de emissão de CO_2, em quilogramas, associadas ao treinamento de cada modelo no segundo experimento numérico. Essa média é proveniente de cinco execuções focadas na estimativa da emissão de CO_2. É possível observar que, para este experimento, em que a arquitetura é fixa e apenas avaliamos o impacto da função adaptativa, a estratégia Normal apresentou, na maioria dos casos, o menor valor de emissão de CO_2, enquanto a estratégia Neurônio mostrou o maior valor. Apesar disso, a diferença entre eles é muito pequena, sendo menor do que 1×10^{-4}.

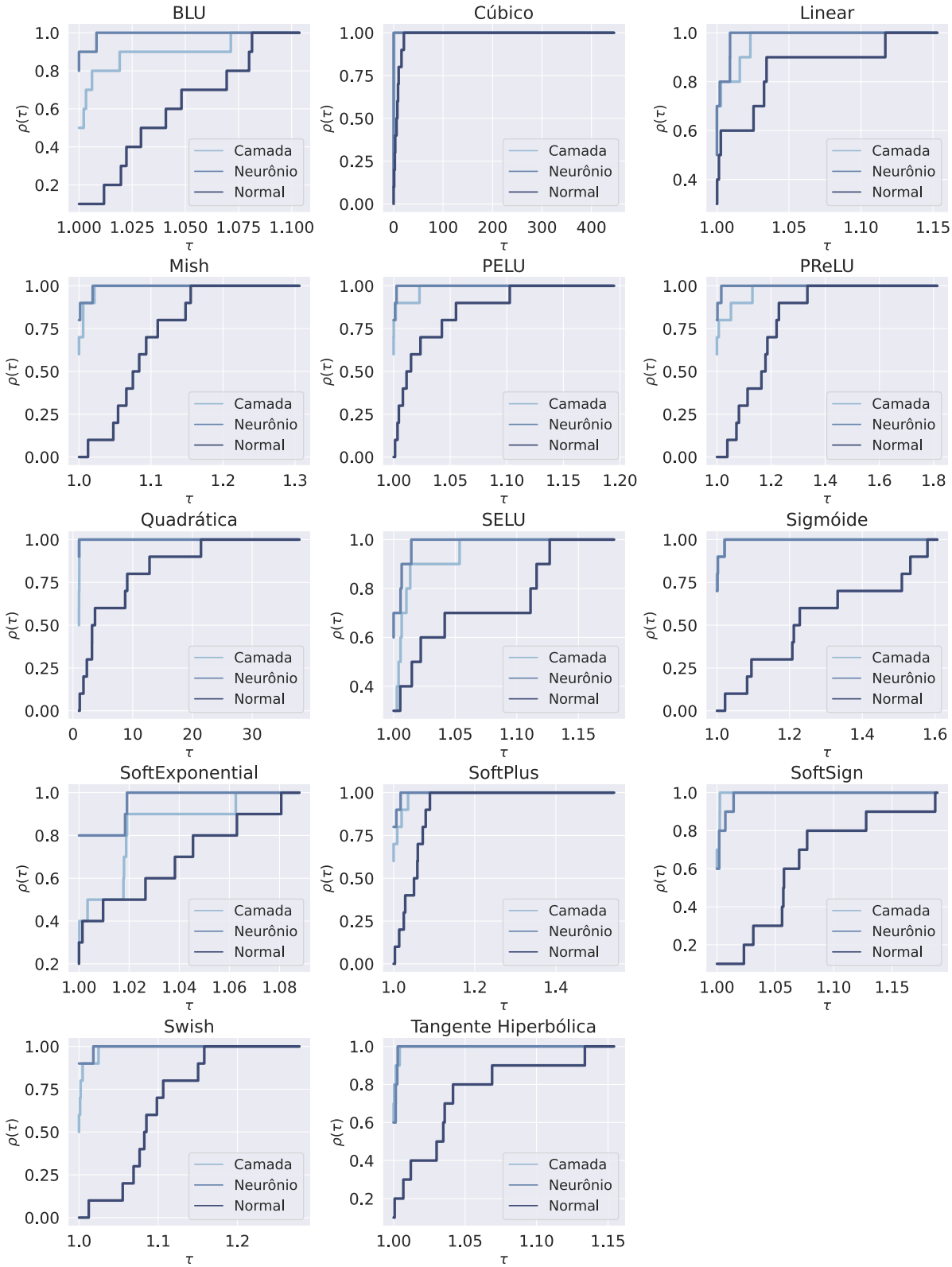
Embora as estratégias adaptativas tenham apresentado uma emissão de CO_2 maior, o valor máximo obtido foi de 1.24×10^{-4} kg, o que equivale a menos de 2% do gasto energético de um carregador de telefone.

Figura 59 – Curvas dos perfis de desempenho do segundo experimento para métrica R^2 utilizando o valor médio.



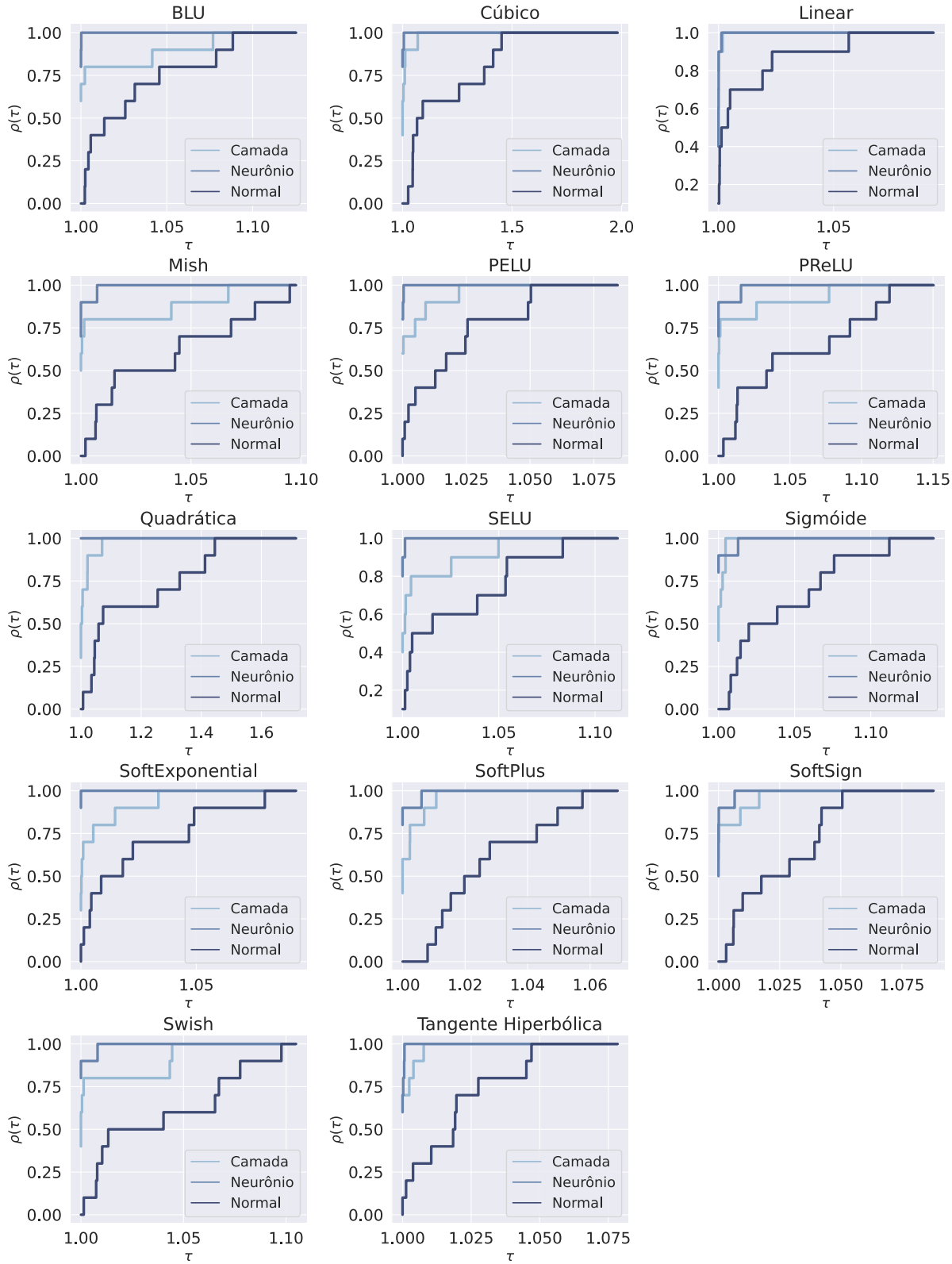
Fonte: Elaborada pelo autor (2023).

Figura 60 – Curvas dos perfis de desempenho do segundo experimento para métrica R^2 utilizando o valor máximo.



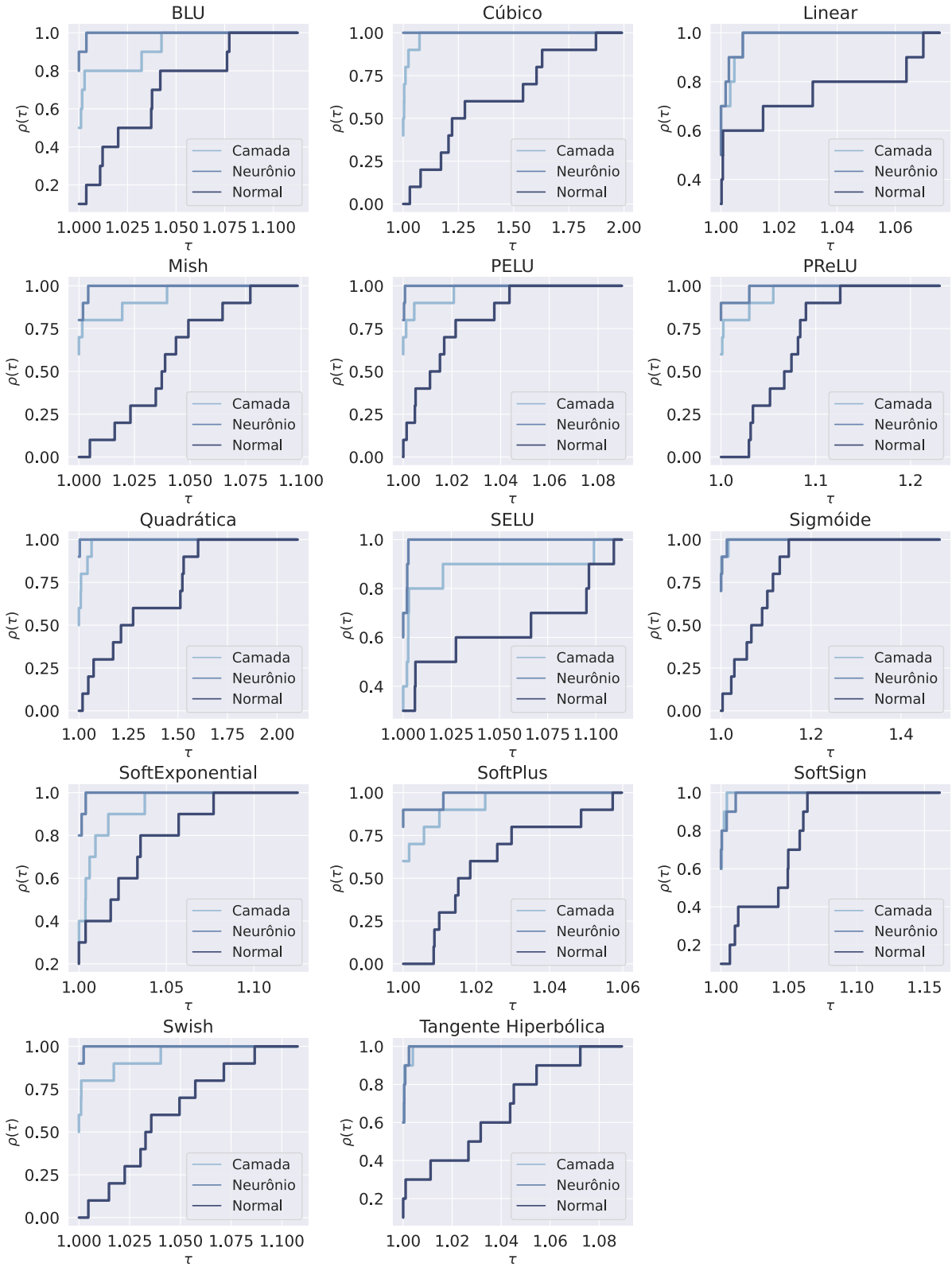
Fonte: Elaborada pelo autor (2023).

Figura 61 – Curvas dos perfis de desempenho do segundo experimento para métrica RMSE utilizando o valor médio.



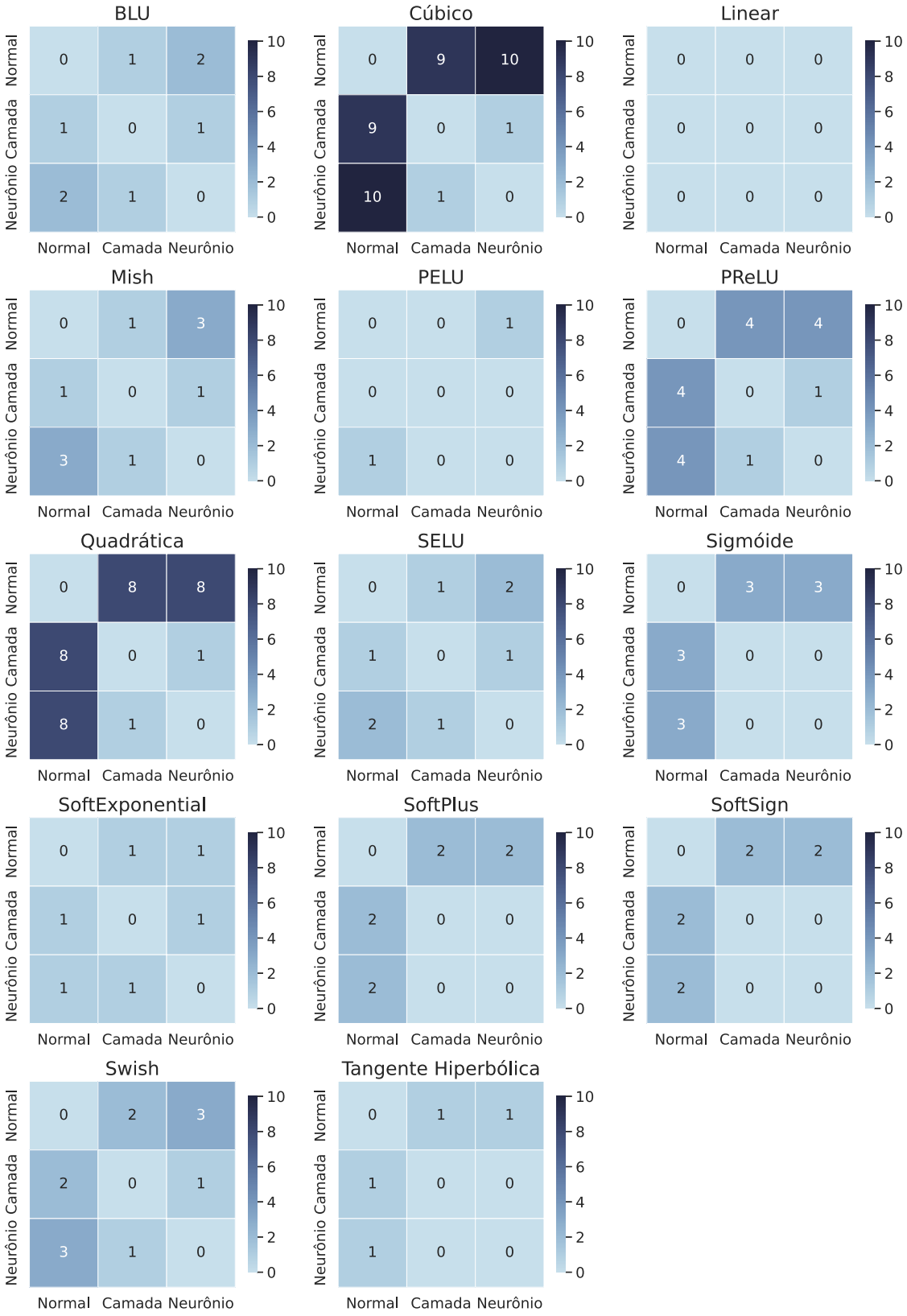
Fonte: Elaborada pelo autor (2023).

Figura 62 – Curvas dos perfis de desempenho do segundo experimento para métrica RMSE utilizando o valor mínimo.



Fonte: Elaborada pelo autor (2023).

Figura 63 – Contagem dos tipos de funções adaptativas que apresentaram diferença média significativa de acordo com o teste de Tukey para o segundo experimento.



Fonte: Elaborada pelo autor (2023).

7.3 EXPERIMENTO III - AVALIAÇÃO DA BUSCA DE ARQUITETURA COM OS TIPOS ADAPTATIVOS PARA PROBLEMAS DE AUTOENCODER

Este experimento numérico tem como objetivo realizar uma busca multiobjetivo, maximizando a métrica e minimizando a quantidade de parâmetros, de uma ANN para cinco problemas de regressão no contexto de *Autoencoder*. Assim como foi feito nos dois experimentos anteriores, os conjuntos de dados foram obtidos por meio do PMLB (OLSON et al., 2017). Para a avaliação das técnicas, procedemos com a separação aleatória de 20% dos conjuntos de dados para o teste. A Tabela 12 apresenta um resumo das bases de dados utilizadas no terceiro experimento numérico, contendo seus nomes, quantidades utilizadas tanto no treinamento quanto no teste, além do número de características.

Tabela 12 – Sumarização das bases de dados utilizadas no terceiro experimento numérico.

Base de Dados	Problema	Instancias de Treinamento	Instancias de Teste	Qtd. de Características
1	4544_GeographicalOriginalofMusic	847	212	117
2	505_tecator	192	48	124
3	clean1	380	96	168
4	dna	2548	638	180
5	mfeat_pixel	1600	400	240

Fonte: Elaborada pelo autor (2023).

Adotamos, para todas as bases de dados estudadas, o mesmo procedimento de busca com o NSGA-II e o NSGA-III. Além disso, considerando a proposta deste trabalho que foi descrita no capítulo 6, algumas das variáveis de busca presentes no problema de otimização referem-se à lista de AF, podendo essas listas ser dos tipos adaptativo Normal, Camada ou Neurônio. Desta forma, podemos avaliar se as estratégias adaptativas são capazes de auxiliar no processo de busca por arquiteturas que maximizem a métrica e minimizem a quantidade de parâmetros, em comparação com as AF tradicionais. Utilizamos o NSGA-III, mesmo com apenas dois objetivos, pois o algoritmo apresenta ser mais eficiente em distribuição e diversidade, sendo útil para explorar melhor o espaço de soluções.

Utilizamos de forma empírica, para cada problema de otimização, uma população de tamanho 16 que foi evoluída por 32 gerações, totalizando 512 avaliações. Quanto maior número de avaliações melhor para a exploração do espaço do problema, mas consequentemente aumenta o tempo do experimento realizado. Sendo assim, buscamos um equilíbrio entre tempo do experimento e exploração. É importante destacar, neste contexto, que avaliamos 512 arquiteturas e configurações de ANN para identificar a frente de Pareto.

Cada indivíduo avaliado representa uma configuração de ANN e alguns de seus parâmetros de treinamento. Também de forma empírica, realizamos a busca pela quantidade de camadas internas, variando de um mínimo de uma camada até um máximo de três

camadas. Também variamos o número de neurônios por camada, de um mínimo de um até um máximo igual ao número de características da base de dados modelada. Selecionamos a função de ativação por camada de entre as mencionadas na seção 5.3 (exceto a função Fourier, descrita na subseção 5.3.15). Avaliamos se a normalização em lote, a normalização de pesos e a taxa de abandono por camada são aplicadas. Além disso, para os parâmetros de treinamento, exploramos a quantidade de épocas para o treinamento da ANN, variando de um mínimo de 50 até um máximo de 300. Também investigamos a taxa de aprendizado, variando de um mínimo de 10^{-5} até um máximo de 10^{-2}. Quanto ao tamanho do mini-lote, consideramos um mínimo de 1 e um máximo de 64 para as bases de dados 2 e 3, 256 para a base de dados 1 e 512 para as bases de dados 4 e 5, onde a diferença é decorrente do número de amostras da base de dados original.

É importante observar que entre as variáveis de busca, temos números inteiros e reais, sendo necessário, portanto, tratar o processo de recombinação e mutação de acordo. Novamente, de forma empírica, definimos alguns hiperparametros do processo de busca. Os números inteiros, neste caso, são representados com a forma binária. Desta forma, utilizamos o operador SBX, com uma probabilidade de troca de 0.5 e $\eta = 15$. Além disso, empregamos a Recombinação de n Pontos, com probabilidade de troca de 0.5 para todos os bits, como operadores de recombinação para os valores reais e inteiros, respectivamente. Adicionalmente, empregamos a mutação Polinomial, com probabilidade de troca de 1.0, e a Mutação de Inversão de Bits, com probabilidade proporcional à quantidade de bits na solução, como operadores de mutação para os valores reais e inteiros, respectivamente.

Durante a etapa de avaliação do indivíduo, realizamos uma validação cruzada com K-*fold* ($K = 3$), com o intuito de calcular um valor médio da métrica RMSE nos dados de validação e utilizar esse mesmo valor como resultado de uma das funções objetivo. É importante observar que a ANN será treinada K vezes para obter uma avaliação mais consistente da sua qualidade de predição. A outra função objetivo, referente à quantidade de parâmetros, é obtida na construção da ANN, obtendo todos os parâmetros ajustáveis da mesma.

Durante o processo de treinamento da ANN, foi utilizado o otimizador Adam (KINGMA; BA, 2014) e 20% dos dados, que foram destinados ao treinamento da ANN, após a aplicação do K-*fold*, foram utilizados para realizar um processo de parada no treinamento, caso a ANN apresentasse piora nas últimas 10 épocas.

Realizamos 30 execuções independentes no processo de busca e calculamos as métricas de Hipervolume, Espaçamento e Cobertura. Em algumas análises, retiramos três soluções da frente de Pareto encontrada, sendo elas: a que apresentou o melhor resultado em métrica, a que apresentou o menor número de parâmetros e, por fim, a solução intermediária que está mais próxima da origem, de acordo com a distância euclidiana. Para esta segunda análise, calculamos as métricas RMSE e Quantidade de Parâmetros.

Essas análises são detalhadas nas subseções a seguir.

7.3.1 Análise da Frente de Pareto

As Tabelas 40 e 41, presentes no Anexo C.1, apresentam os resultados sumarizados do terceiro experimento numérico para os otimizadores NSGA-II e NSGA-III, bem como para o tipo adaptativo da ANN. Elas exibem os valores mínimo, médio, mediano e máximo, além do desvio padrão, para as métricas Hipervolume e Espaçamento. Adicionalmente, as Figuras 103 e 104, também encontradas no Anexo C.1, ilustram os *Violin plots* das métricas Hipervolume e Espaçamento, respectivamente. Esses gráficos proporcionam uma visualização da distribuição das soluções encontradas.

É possível observar que, em geral, para a métrica de Hipervolume, as estratégias adaptativas possuem valores melhores em comparação com a estratégia Normal. Já em relação à métrica de Espaçamento, é difícil afirmar qual estratégia é superior, visto que as três apresentam resultados bons e ruins em diferentes cenários. Por outro lado, é notável mais uma vez que os resultados das estratégias adaptativas por Camada e Neurônio se igualaram em muitas comparações e alternaram em relação a qual era superior, dependendo da base de dados e da métrica de avaliação. Como nos experimentos anteriores, não foi possível determinar qual estratégia é melhor de forma geral. Quando analisamos os algoritmos de otimização, ambos apresentam resultados comparáveis, tornando ideal a realização de outras análises qualitativas dos resultados para indicar qual algoritmo apresenta melhor desempenho.

Para fomentar as análises, decidimos mais uma vez seguir com a utilização dos Perfis de Desempenho. A Tabela 13 e as Figuras 64 e 65 mostram os resultados dos Perfis de Desempenho para os diferentes algoritmos de otimização, tipos adaptativos e para as métricas Hipervolume e Espaçamento. A Figura 64 apresenta os Perfis de Desempenho construídos utilizando o valor médio obtido em 30 execuções independentes, enquanto a Figura 65 apresenta os Perfis de Desempenho construídos utilizando o melhor valor obtido (o valor máximo para Hipervolume e o valor mínimo para Espaçamento). É possível observar que as estratégias que utilizam AAF (Camada e Neurônio) apresentam melhores resultados na métrica Hipervolume em comparação com a estratégia que utiliza apenas AF tradicional (Normal) nas bases de dados estudadas. Quanto à métrica Espaçamento, percebemos que para a média, a estratégia que utiliza AAF (Camada e Neurônio) apresenta melhores resultados, enquanto que para o melhor valor, a estratégia que usa apenas AF tradicional (Normal) é mais eficaz.

Além disso, percebe-se que em ambos os casos, para a média e o melhor valor da métrica Hipervolume, o algoritmo NSGA-III apresentou os melhores resultados em comparação com o NSGA-II. Por outro lado, em relação à métrica Espaçamento, observa-se que o algoritmo NSGA-II obteve os melhores resultados quando comparado ao NSGA-III.

Tabela 13 – Área normalizada sob as curvas dos perfis de desempenho para Métricas Hipervolume e Espaçamento.

Análise	Métrica	NSGA-II			NSGA-III		
		Normal	Camada	Neurônio	Normal	Camada	Neurônio
Média	Hipervolume	**59.29%**	88.75%	89.97%	68.90%	**96.91%**	93.37%
	Espaçamento	72.24%	85.54%	**90.61%**	**29.52%**	32.39%	43.84%
Melhor	Hipervolume	**57.02%**	90.29%	90.75%	73.13%	87.94%	**98.09%**
	Espaçamento	**93.96%**	70.77%	92.36%	63.58%	63.33%	**54.79%**

Fonte: Elaborada pelo autor (2023).

Novamente, para ampliar a análise comparativa dos resultados, utilizamos o ANOVA com um intervalo de confiança de 95% para verificar a hipótese nula de que não há diferença significativa entre as médias de cada grupo (otimizador e tipo adaptativo). Os resultados obtidos indicaram que, em todos os casos, o teste rejeitou a hipótese nula e que existem grupos compostos por otimizadores e tipos adaptativos que apresentam desempenho diferente. No entanto, o ANOVA não é capaz de identificar quais grupos apresentam diferenças nas médias.

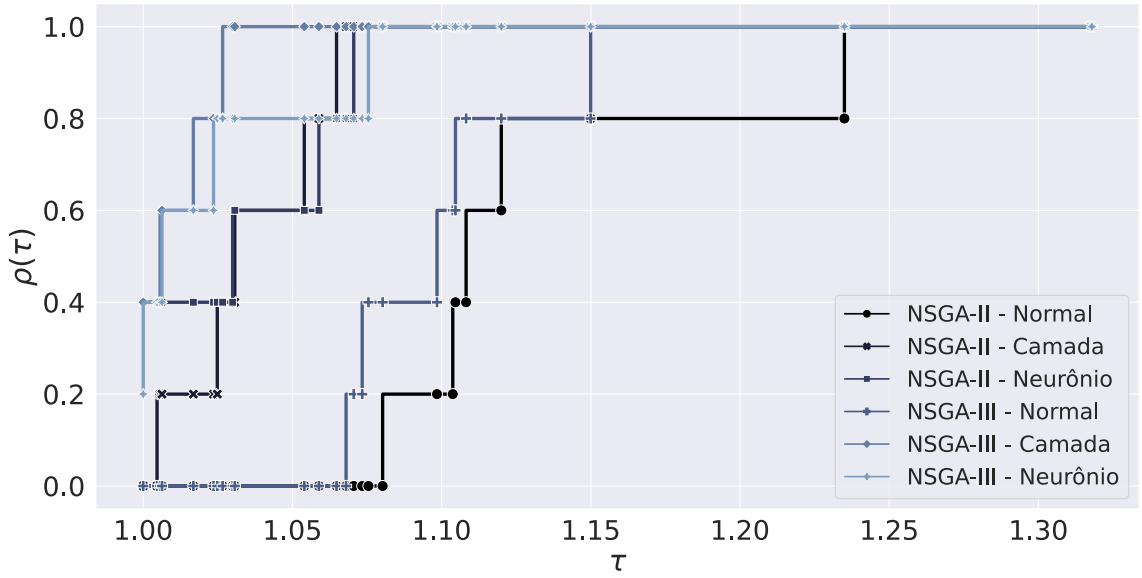
Utilizamos, portanto, o teste de Tukey com um intervalo de confiança de 95% para verificar onde a diferença entre as médias é significativa. A Figura 66 apresenta uma contagem, com o valor máximo igual ao número de bases de dados, das situações em que os otimizadores e tipos de funções adaptativas demonstraram diferença média significativa, de acordo com o teste de Tukey, para as duas métricas em estudo.

Ao analisar a Figura 66a, que apresenta a métrica Hipervolume, é possível observar que, ao fixar o otimizador e comparar os tipos adaptativos, percebe-se uma diferença significativa entre a estratégia Normal e as estratégias Camada e Neurônio em todas as bases de dados. Agora, na comparação entre as estratégias Camada e Neurônio, nota-se que, em alguns casos, existe uma diferença, mas ela não é tão relevante. Ao analisar os algoritmos de otimização, observa-se que a estratégia Normal em conjunto com o NSGA-II também apresenta diferenças em relação aos adaptativos do NSGA-III, abrangendo todos os problemas. Por outro lado, quando comparada a estratégia Normal com o NSGA-III em relação aos adaptativos do NSGA-II, percebe-se uma diminuição no número de problemas que apresentam diferenças.

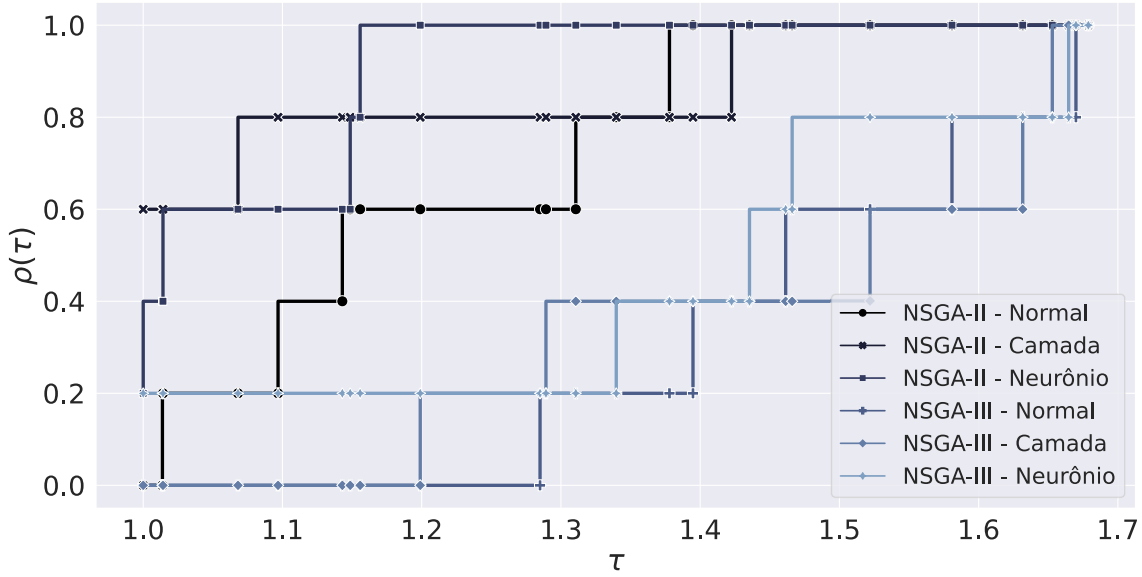
Agora, analisando a Figura 66b, que apresenta a métrica Espaçamento, podemos observar que, ao fixarmos o otimizador e compararmos os tipos adaptativos, não observamos, na maioria das vezes, diferença entre os grupos. Ao compararmos os otimizadores NSGA-II e NSGA-III, observamos uma grande quantidade de problemas que apresentaram diferenças entre os grupos. Ao combinar todos os resultados obtidos até este momento, o NSGA-II apresentou melhores resultados em comparação com o NSGA-III em relação à métrica

Figura 64 – Curvas dos perfis de desempenho do terceiro experimento utilizando o valor médio.

(a) Hipervolume



(b) Espaçamento



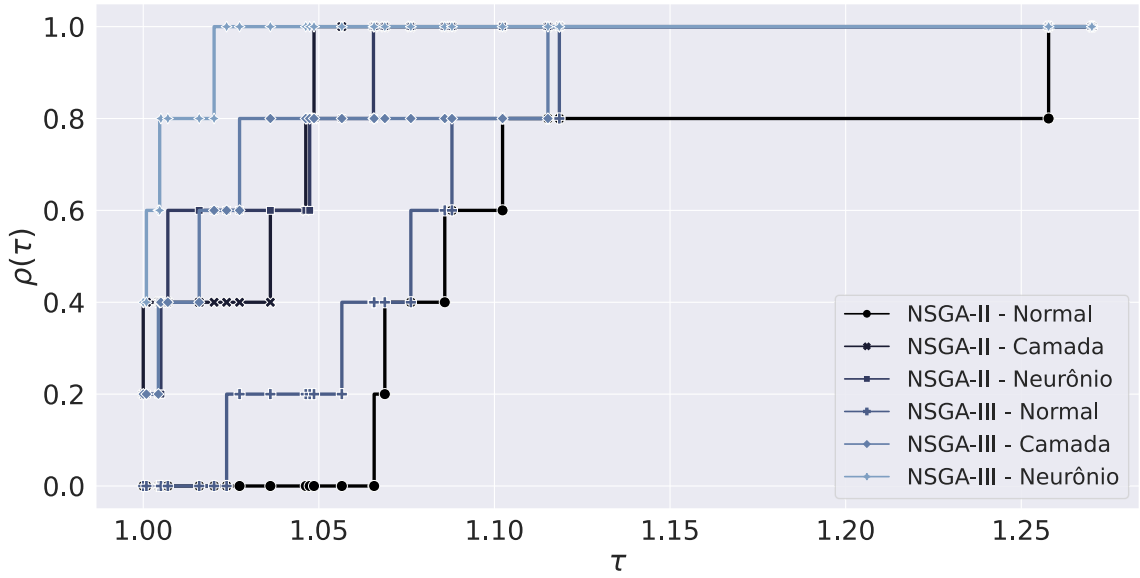
Fonte: Elaborada pelo autor (2023).

Espaçamento.

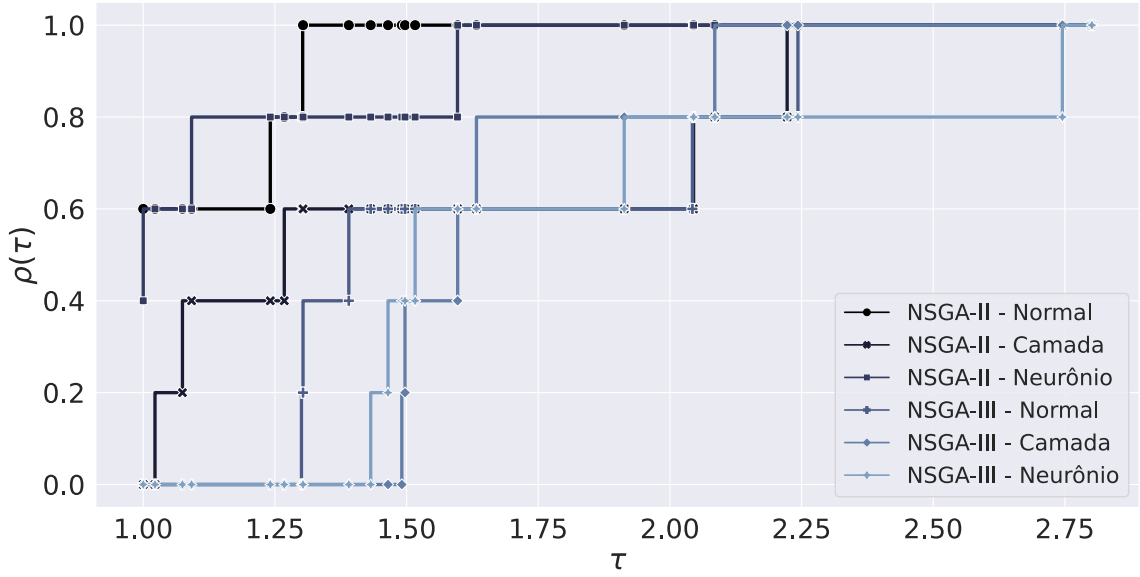
Realizamos a agregação das 30 execuções independentes para a formação de uma única frente de Pareto para cada base de dados, algoritmos de otimização e tipos adaptativos. Essas frentes de Pareto podem ser observadas na Figura 105, presente no Anexo C.1. Através dessas frentes de Pareto, podemos calcular a métrica de Cobertura, conforme apresentado na Tabela 14.

Figura 65 – Curvas dos perfis de desempenho do terceiro experimento utilizando o melhor valor.

(a) Hipervolume



(b) Espaçamento

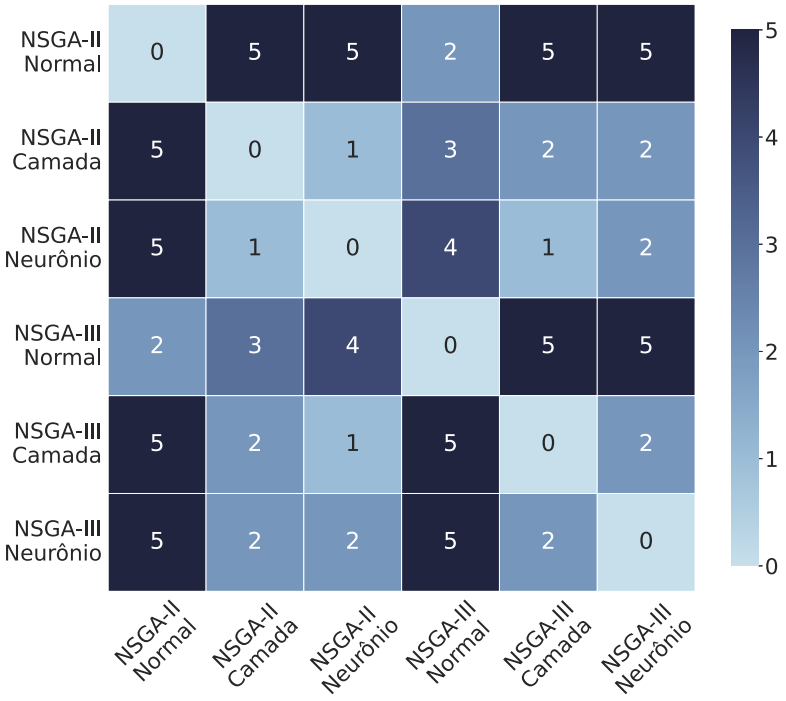


Fonte: Elaborada pelo autor (2023).

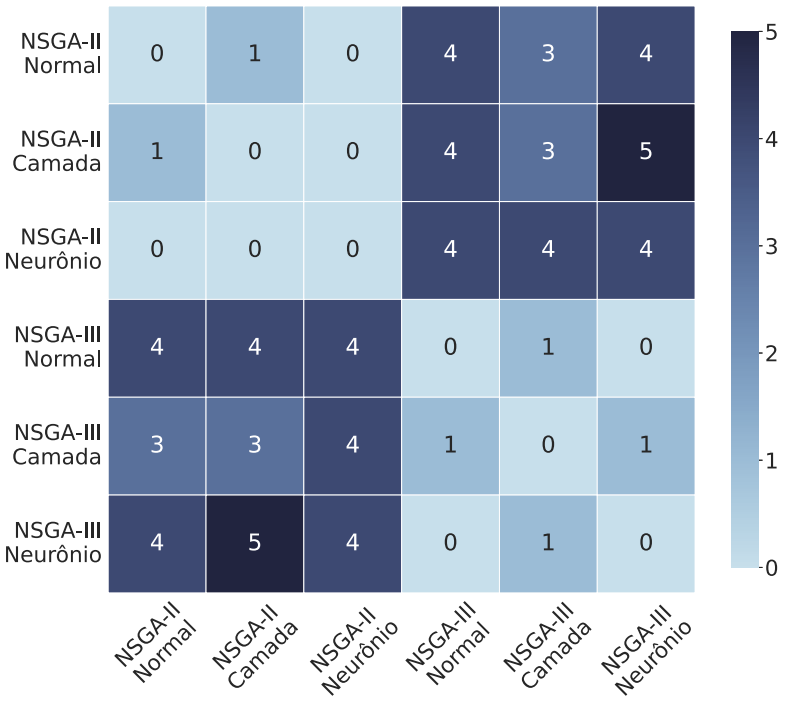
Observamos que, independentemente do algoritmo de otimização, as estratégias Camada e Neurônio possuem, em geral, um valor maior de cobertura em relação à estratégia Normal. Ao compararmos os algoritmos de otimização para a estratégia Normal, observamos que eles apresentam um comportamento similar de cobertura entre si. Por outro lado, é possível observar que, independentemente do algoritmo de otimização, a estratégia Camada acaba sendo menos coberta pelas outras estratégias, apresentando assim uma frente de Pareto menos dominante.

Figura 66 – Contagem dos otimizadores e tipos adaptativos que apresentaram diferença média significativa de acordo com o teste de Tukey para o terceiro experimento.

(a) Hipervolume



(b) Espaçamento



Fonte: Elaborada pelo autor (2023).

Compilando todas as análises realizadas nesta subseção, podemos observar que as estratégias adaptativas, tanto por Camada quanto por Neurônio, apresentam resultados

Tabela 14 – Comparação da métrica de Cobertura.

Base de Dados	Conjunto A		Conjunto B					
			NSGA-II			NSGA-III		
			Normal	Camada	Neurônio	Normal	Camada	Neurônio
1	NSGA-II	Normal	0.000	8.600	35.560	45.310	11.390	32.880
		Camada	**73.680**	0.000	**73.330**	**81.250**	48.100	**83.560**
		Neurônio	49.120	13.980	0.000	51.560	22.780	46.580
	NSGA-III	Normal	14.040	4.300	21.110	0.000	2.530	31.510
		Camada	64.910	8.600	53.330	73.440	0.000	65.750
		Neurônio	45.610	10.750	14.440	54.690	11.390	0.000
2	NSGA-II	Normal	0.000	4.000	8.570	42.860	4.350	8.700
		Camada	**77.780**	0.000	**77.140**	**71.430**	**52.170**	30.430
		Neurônio	**72.220**	12.000	0.000	64.290	13.040	0.000
	NSGA-III	Normal	16.670	4.000	20.000	0.000	8.700	13.040
		Camada	**72.220**	16.000	62.860	**71.430**	0.000	17.390
		Neurônio	**77.780**	36.000	65.710	64.290	43.480	0.000
3	NSGA-II	Normal	0.000	3.330	23.420	20.000	2.860	18.670
		Camada	**75.510**	0.000	41.440	**76.000**	21.430	48.000
		Neurônio	**63.270**	45.560	0.000	66.000	48.570	36.000
	NSGA-III	Normal	40.820	3.330	20.720	0.000	1.430	20.000
		Camada	**73.470**	34.440	40.540	**82.000**	0.000	46.670
		Neurônio	59.180	32.220	13.510	60.000	27.140	0.000
4	NSGA-II	Normal	0.000	10.130	36.560	52.630	17.760	48.180
		Camada	**71.630**	0.000	**77.960**	**81.050**	44.860	**80.000**
		Neurônio	49.650	5.060	0.000	68.420	11.210	55.450
	NSGA-III	Normal	16.310	2.530	15.050	0.000	4.670	30.000
		Camada	**56.740**	20.250	51.080	**71.580**	0.000	**61.820**
		Neurônio	34.040	12.030	25.810	49.470	14.950	0.000
5	NSGA-II	Normal	0.000	6.400	19.350	25.580	9.180	13.790
		Camada	**80.910**	0.000	25.810	**80.230**	24.490	22.990
		Neurônio	**82.730**	**68.800**	0.000	81.400	**79.590**	27.590
	NSGA-III	Normal	33.640	6.400	19.350	0.000	8.160	13.790
		Camada	**80.910**	26.400	20.160	**80.230**	0.000	20.690
		Neurônio	**80.910**	**68.800**	25.810	**82.560**	**79.590**	0.000

Fonte: Elaborada pelo autor (2023).

melhores do que a estratégia Normal. Ao analisarmos puramente a métrica Hipervolume, essa conclusão fica mais evidente, uma vez que, para a métrica Espaçamento, os tipos adaptativos não apresentaram diferença significativa entre os grupos.

Ao compararmos a estratégia por Camada em relação à estratégia por Neurônio, observamos que ambas apresentam um desempenho competitivo entre si, apesar de a estratégia por Camada apresentar melhores valores para a métrica Cobertura. Isso nos leva a considerar a estratégia por Camada como mais vantajosa em relação às outras.

7.3.2 Análise de Soluções da Frente de Pareto

As Tabelas 42 e 43, presentes no Anexo C.2, apresentam os resultados sumarizados do terceiro experimento numérico para os otimizadores NSGA-II e NSGA-III, as soluções extraídas da frente de Pareto e o tipo adaptativo da ANN, exibindo os valores mínimo, médio, mediano e máximo para as métricas RMSE e Quantidade de Parâmetros, além do desvio padrão para as 30 execuções independentes. Complementando, as Figuras 106 e 107, também presentes no Anexo C.2, representam os *Violin plots* das métricas RMSE e Quantidade de Parâmetros, respectivamente, proporcionando uma visualização da distribuição das soluções encontradas.

Apesar de ser esperado, é importante destacar que as soluções extraídas de execuções independentes faziam parte de uma frente de Pareto, ou seja, nenhuma destas soluções dominava a outra. Desta forma, nos resultados, podemos observar que as soluções que foram extraídas por apresentarem os melhores resultados em relação à métrica RMSE serão aquelas que também apresentam os melhores resultados para a mesma métrica. Isso se mantém também para a Quantidade de Parâmetros.

Em geral, independentemente da solução extraída da frente de Pareto, para a métrica RMSE, as estratégias adaptativas possuem valores melhores em comparação com a estratégia Normal. Quanto à métrica de Quantidade de Parâmetros, é difícil observar qual estratégia é melhor, visto que as três apresentam resultados bons e ruins em diferentes cenários. Por outro lado, é possível observar novamente que os resultados das estratégias adaptativas por Camada e por Neurônio se igualam em muitas comparações e se alternam em relação a qual era a melhor, dependendo da base de dados e da métrica de avaliação. Assim como já foi apresentado nos primeiros e segundos experimentos, novamente não foi possível definir qual estratégia é melhor de maneira geral. Ao analisarmos os algoritmos de otimização, ambos apresentam resultados comparáveis, sendo necessárias outras análises qualitativas dos resultados para indicar qual algoritmo apresenta melhor desempenho.

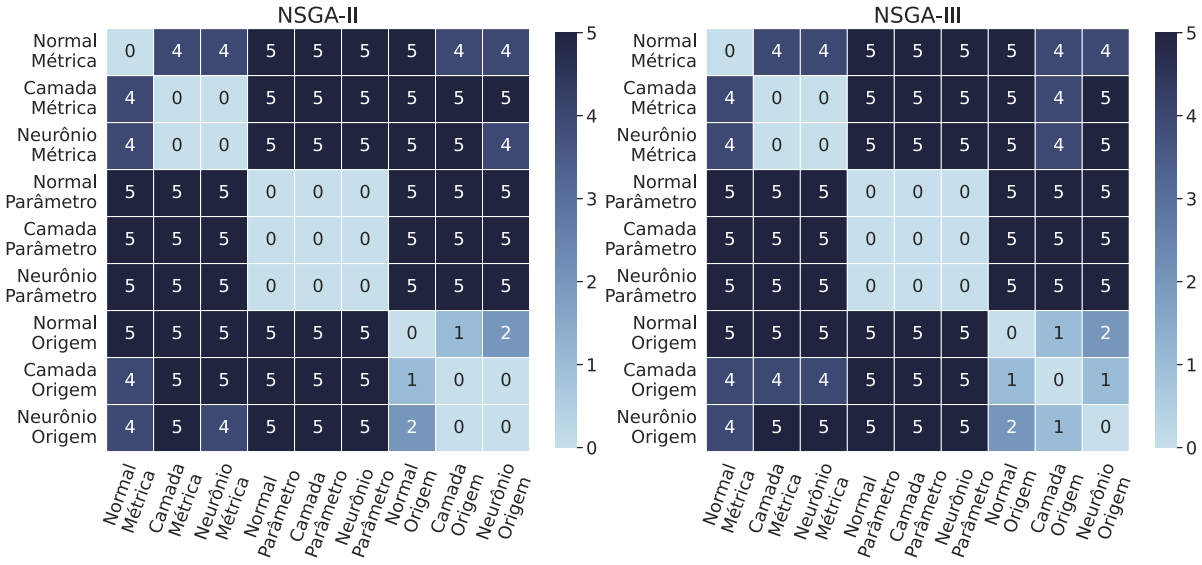
Como a análise da Frente de Pareto, apresentada na subseção anterior, já apresentava os resultados esperados, continuamos nesta subseção apenas com os testes estatísticos para verificar diferenças entre os grupos. Desta forma, utilizamos a ANOVA com um intervalo de confiança de 95% para testar a hipótese nula de que não há diferença significativa entre as médias de cada grupo (otimizador, solução extraída e tipo adaptativo). Os resultados obtidos indicaram que, em todos os casos, o teste rejeitou a hipótese nula e que existem grupos formados pelo otimizador, solução extraída e tipos adaptativos que têm desempenhos diferentes. Apesar disso, a ANOVA não é capaz de identificar quais grupos possuem diferença nas médias.

Utilizamos o teste de Tukey, com um intervalo de confiança de 95%, para identificar onde a diferença entre as médias é significativa. A Figura 67 exibe uma contagem, com valor máximo igual ao número de bases de dados, das vezes em que os otimizadores,

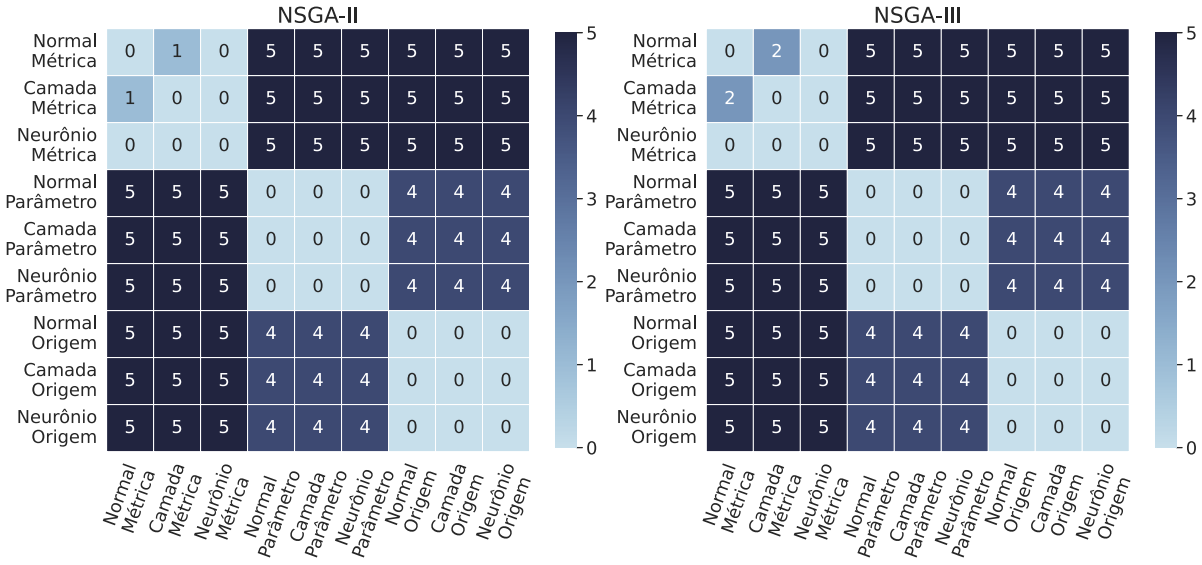
soluções extraídas e tipos de funções adaptativas apresentaram diferença média significativa, conforme indicado pelo teste de Tukey, para as duas métricas estudadas.

Figura 67 – Contagem dos otimizadores, soluções extraídas e tipos adaptativos que apresentaram diferença média significativa de acordo com o teste de Tukey para o terceiro experimento.

(a) RMSE



(b) Quantidade de Parâmetros



Fonte: Elaborada pelo autor (2023).

Analisando a Figura 67a, que apresenta a métrica RMSE, podemos observar que, ao analisarmos soluções extraídas que demonstram uma melhor Quantidade de Parâmetros, independentemente do tipo adaptativo, elas não apresentam diferença significativa. Ao analisarmos as soluções extraídas que demonstram um melhor valor de RMSE, percebemos que na maioria dos problemas, as estratégias adaptativas apresentaram diferença significa-

tiva em relação à estratégia Normal. Para os demais casos, quando comparamos diferentes tipos de soluções extraídas, era de se esperar o comportamento observado, no qual existe, na maioria dos problemas, diferença significativa entre eles por representarem diferentes regiões da frente de Pareto.

Ao analisarmos a Figura 67b, notamos que essencialmente não há diferença significativa entre os tipos adaptativos. E mais uma vez, para os outros casos, ao compararmos diferentes tipos de soluções extraídas, era de se esperar o comportamento observado, no qual existe, na maioria dos problemas, diferença significativa entre eles por representarem diferentes regiões da frente de Pareto.

Por fim, com o objetivo de verificar a relevância das diferentes AFs, as Figuras 68 e 69 apresentam, para os otimizadores NSGA-II e NSGA-III, respectivamente, a frequência de seleção das AFs tanto para os tipos adaptativos quanto para as soluções extraídas da frente de Pareto.

É possível observar que, independentemente do otimizador e do tipo de solução analisada, a estratégia Normal apresentou, em ordem crescente, as cinco funções mais escolhidas: BLU, PELU, Soft Exponential, Linear e SELU. Nos demais casos, é possível observar diferentes comportamentos nas funções mais utilizadas, embora a função Linear esteja sempre entre as mais utilizadas. Nesse contexto, a função Linear pode desempenhar um papel muito importante na última camada da *Autoencoder*, delegando a responsabilidade do ajuste aos últimos pesos da rede.

Compilando também as análises realizadas nesta subseção, é possível observar que as estratégias adaptativas, tanto por Camada quanto por Neurônio, apresentam melhores resultados do que a estratégia Normal ao analisarmos puramente a métrica RMSE para as soluções extraídas da frente de Pareto que possuam, para essa mesma métrica, o melhor valor. Enquanto a Quantidade de Parâmetros não demonstra diferença significativa entre os grupos adaptativos. Esses resultados sugerem que é possível encontrar arquiteturas de ANN com a mesma Quantidade de Parâmetros, mas com melhores resultados de RMSE ao utilizar os tipos adaptativos, tanto por Camada quanto por Neurônio.

7.3.3 Pegadas de Carbono

A Tabela 44, presente no Anexo C, exibe as estimativas médias de emissão de CO_2 em quilogramas, associadas ao treinamento das três soluções retiradas da frente de Pareto encontrada, agregando todas as 30 execuções independentes do terceiro experimento. Essas soluções são: aquela que apresentou o melhor resultado em métrica, aquela que apresentou o menor número de parâmetros e, por fim, a solução intermediária que está mais próxima da origem, de acordo com a distância euclidiana.

Ao concentrar a análise nas colunas relacionadas às soluções com o menor número de parâmetros, destaca-se que em várias bases de dados, a abordagem do otimizador

NSGA-II demonstra uma tendência de resultar em emissões de CO_2 mais baixas em comparação com a abordagem do NSGA-III. Esses resultados sugerem que, de forma geral, o NSGA-II se mostra mais eficaz em encontrar soluções com menor impacto de CO_2. Além disso, explorando os tipos adaptativos utilizados, observa-se uma falta de padrão único. Em algumas situações, como no caso da primeira base de dados, a estratégia Camada demonstra desempenho superior, enquanto em outras, como nas bases de dados segunda e quinta, a abordagem Normal supera a abordagem Camada. O mesmo cenário ocorre com os tipos Normal e Neurônio, destacando que a influência desses tipos adaptativos na qualidade das soluções é sensível às particularidades de cada conjunto de dados. Portanto, apesar da tendência favorável ao NSGA-II sobre o NSGA-III, a seleção dos tipos adaptativos ideais para cada cenário ainda se apresenta como um fator crucial na busca por soluções de baixas emissões de CO_2.

Ao focar nas soluções intermediárias mais próximas da origem com base na distância euclidiana, percebe-se que, em grande parte dos casos, as soluções derivadas do NSGA-III exibem níveis inferiores de emissões de CO_2 em comparação com as provenientes do NSGA-II. Isso sugere que, de maneira geral, o NSGA-III tem um desempenho mais eficiente na busca por soluções com menor impacto de CO_2. Quando examinamos os diferentes tipos adaptativos, emerge uma tendência: o tipo adaptativo Camada frequentemente apresenta as emissões de CO_2 mais baixas, seguido pelo tipo Normal, e finalmente pelo Neurônio. Essa observação sugere que o tipo adaptativo Camada exerce uma influência mais positiva na mitigação das emissões de CO_2. Com base nos resultados apresentados, o NSGA-III surge como a escolha mais promissora para alcançar soluções otimizadas em relação às emissões de CO_2, e o tipo adaptativo Camada se destaca como a opção mais eficaz nesse processo de otimização.

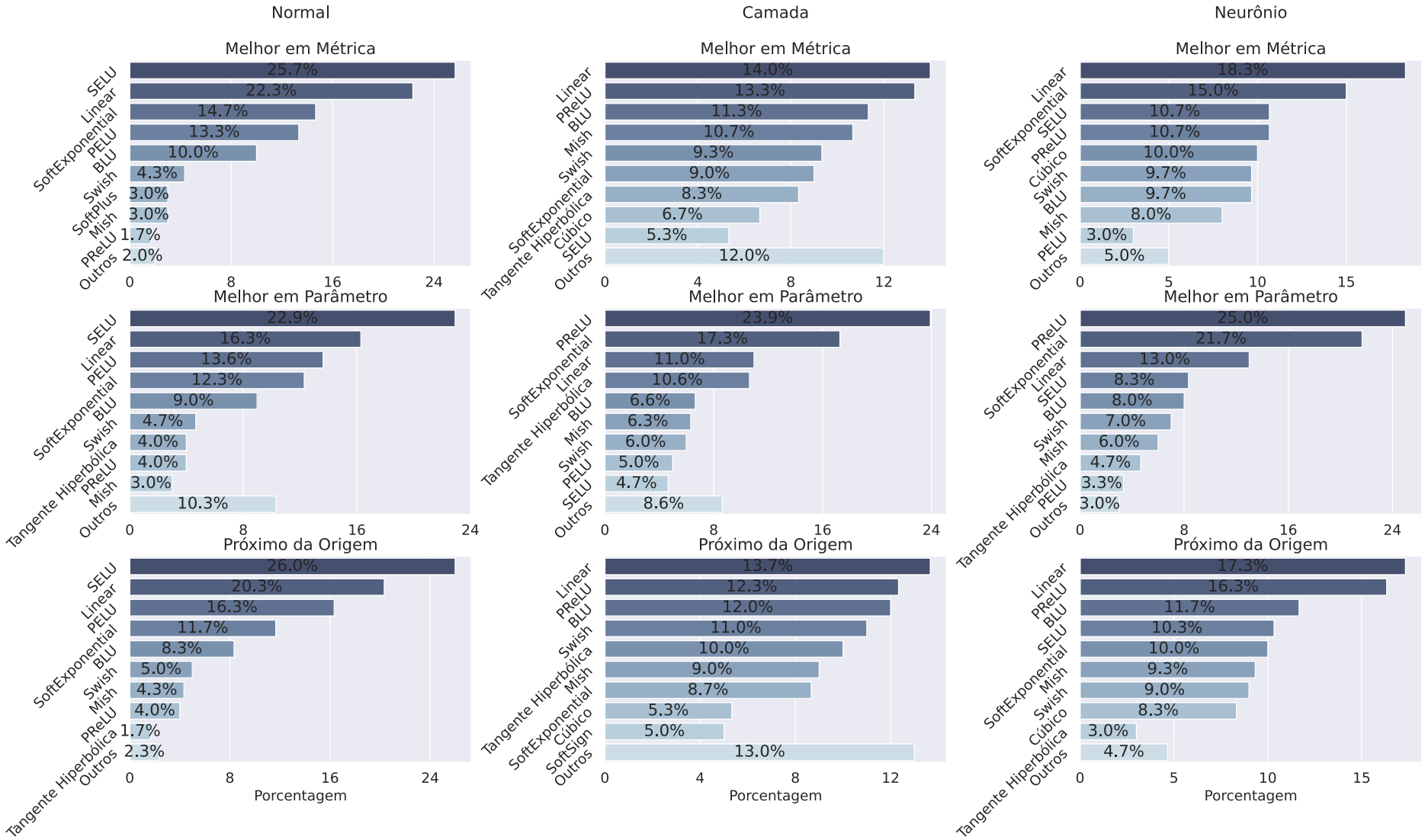
Por fim, ao enfocar nas soluções que alcançam os melhores resultados em métricas, torna-se evidente que as abordagens baseadas no otimizador NSGA-III consistentemente resultam em valores menores de emissões de CO_2 em comparação com aquelas do NSGA-II. Isso insinua que, no contexto das emissões de CO_2, o NSGA-III exibe maior eficácia na obtenção de soluções mais otimizadas. Quando examinamos os diferentes tipos adaptativos, emerge uma tendência: o tipo Normal frequentemente corresponde a valores inferiores de emissões de CO_2, seguido pelo tipo Camada, enquanto o tipo Neurônio tende a mostrar valores mais elevados. Isso sugere que o ajuste do tipo Normal exerce maior influência na redução das emissões de CO_2 em comparação com as outras métricas. Com base nos resultados apresentados, o NSGA-III emerge como a escolha preferencial para alcançar soluções com baixas emissões de CO_2. Adicionalmente, o tipo Normal demonstra ser a opção mais eficaz na otimização das soluções em termos de emissões de CO_2. No entanto, é importante notar que a escolha do otimizador e do tipo adaptativo ideal também pode depender das peculiaridades do problema e das prioridades do usuário.

A análise completa revela que o otimizador NSGA-II tende a ser mais eficaz na

busca por soluções com menor número de parâmetros, enquanto o NSGA-III se destaca em soluções intermediárias próximas à origem, exibindo consistentemente emissões de CO_2 mais baixas. A escolha entre os tipos adaptativos Normal, Camada e Neurônio desempenha um papel crucial, sendo a abordagem por Camada frequentemente associada a menores emissões de CO_2. Em suma, a seleção do otimizador e tipo adaptativo deve considerar as particularidades do problema e objetivos, oferecendo uma abordagem personalizada para otimizar eficazmente as soluções em termos de emissões de CO_2.

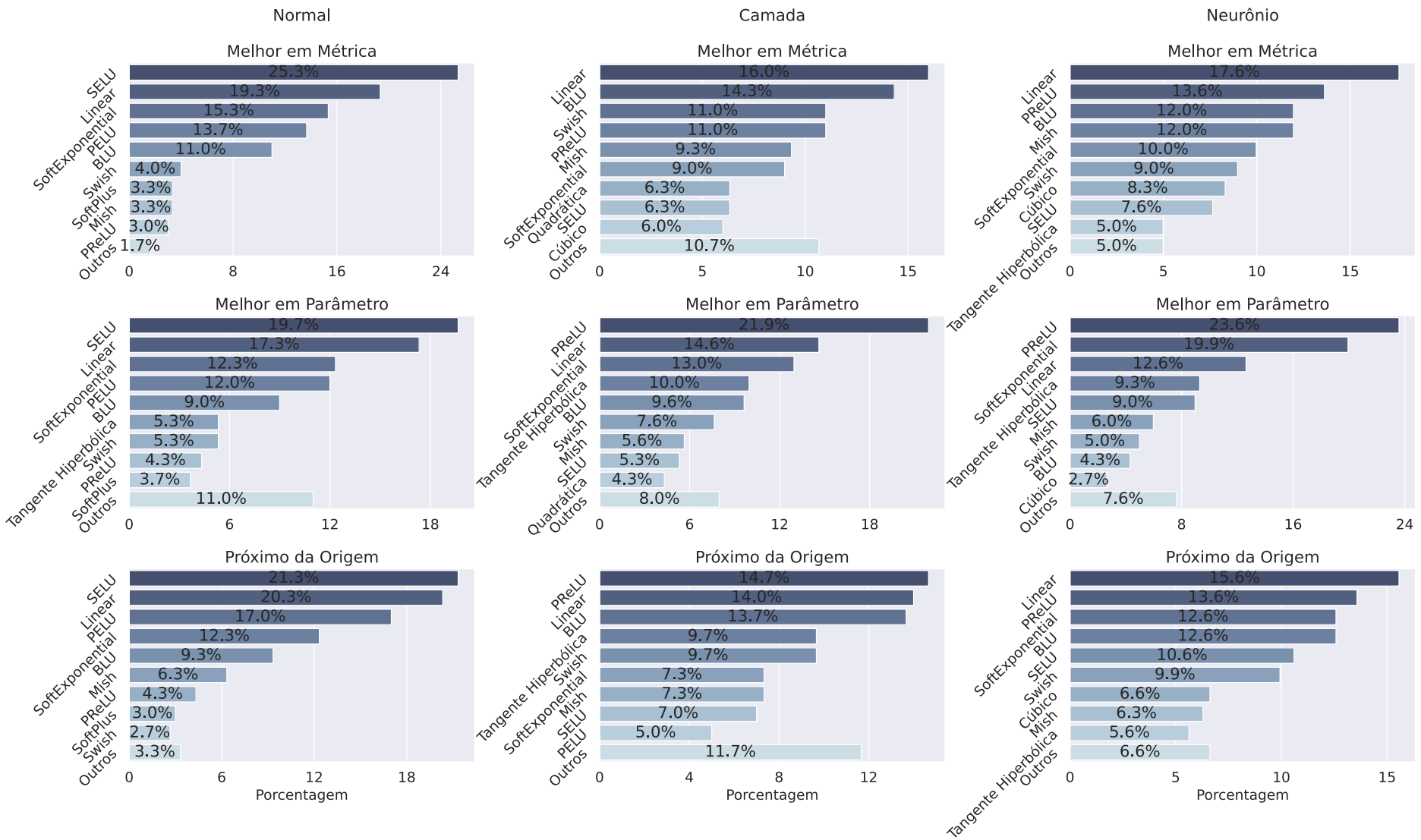
Apesar desta análise completa, o maior valor apresentado foi de $6.10e^{-6}$ kg, menos de 1% do que é gasto com a carga de um telefone.

Figura 68 – Frequência de seleção das AFs para o NSGA-II.



Fonte: Elaborada pelo autor (2023).

Figura 69 – Frequência de seleção das AFs para o NSGA-III.



Fonte: Elaborada pelo autor (2023).

7.4 EXPERIMENTO IV - AVALIAÇÃO DA BUSCA DE ARQUITETURA COM OS TIPOS ADAPTATIVOS PARA PROBLEMA DE CLASSIFICAÇÃO EM IMAGENS DE TUMOR CEREBRAL

Este experimento numérico visa realizar uma busca multiobjetivo, maximizando a métrica e minimizando a quantidade de parâmetros, de uma ANN para um problema de classificação de tumor cerebral utilizando imagens de ressonância magnética. Obtemos o conjunto de dados através da plataforma Kaggle (FELTRIN, 2023), disponibilizada por Fernando Feltrin, um profissional qualificado em engenharia de computação e técnico em radiologia, que trabalha na empresa GRS Imagem, localizada na cidade de Cruz Alta, no estado do Rio Grande do Sul, Brasil. As imagens não possuem qualquer tipo de marcação ou identificação do paciente, sendo interpretadas por radiologistas e disponibilizadas para fins de estudo.

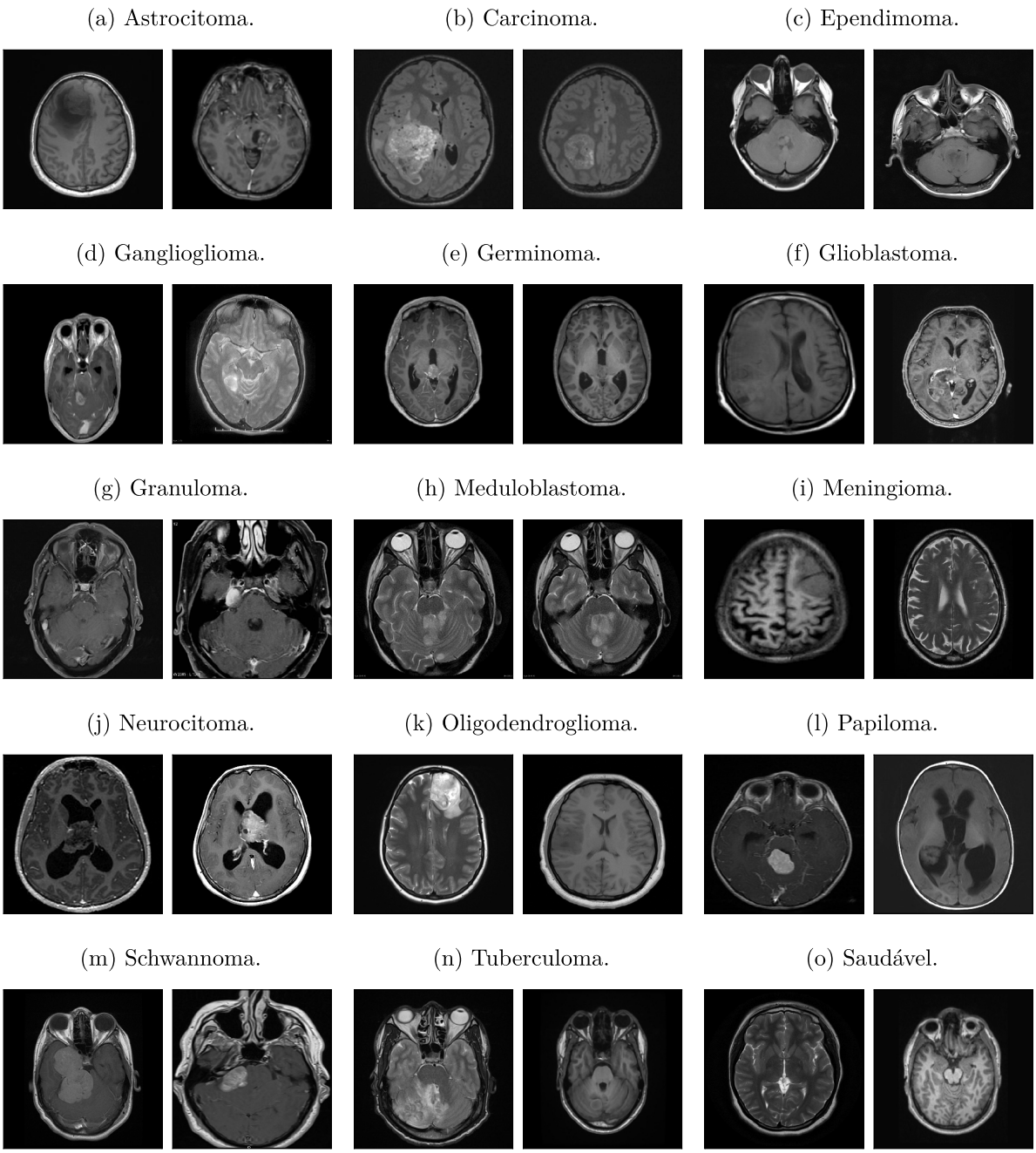
Este conjunto de dados é composto por 4479 imagens de ressonância magnética dos tipos T1, T1 com contraste e T2. Os termos "T1" e "T2" referem-se a sequências de pulso utilizadas durante o exame de ressonância magnética, que fornecem informações diferentes sobre os tecidos do corpo (HAACKE et al., 1999). Além disso, essas imagens estão divididas em 15 classes, das quais 14 representam tipos diferentes de tumores cerebrais, enquanto a classe restante representa a saúde cerebral. A Tabela 15 apresenta a distribuição de classes e a volumetria dos tumores cerebrais, e a Figura 70 mostra uma amostra das imagens para cada classe.

Tabela 15 – Distribuição de Classes e Volumetria dos Tumores Cerebrais.

Rótulo	Classe	Volumetria
0	Astrocitoma	580
1	Carcinoma	251
2	Ependimoma	150
3	Ganglioglioma	61
4	Germinoma	100
5	Glioblastoma	204
6	Granuloma	78
7	Meduloblastoma	131
8	Meningioma	874
9	Neurocitoma	457
10	Oligodendroglioma	224
11	Papiloma	237
12	Schwannoma	465
13	Tuberculoma	145
14	Saudável	522

Fonte: Elaborada pelo autor (2023).

Figura 70 – Amostra dos Tumores Cerebrais.



Fonte: Elaborada pelo autor (2023).

O objetivo principal desta tese e experimento é aprofundar-se na exploração e aprimoramento das ANNs adaptativas. Com o intuito de manter o estudo estritamente direcionado a este tema, foi adotada a rede MobileNet (HOWARD et al., 2017), uma rede convolucional eficiente e compacta, para realizar a transformação das imagens em características relevantes para o processo de classificação. Essa opção foi deliberada para ressaltar a aplicação específica das ANNs adaptativas no contexto do problema em análise, excluindo-se, portanto, o estudo de redes convolucionais a fim de evitar qualquer dispersão

de escopo.

Para isso, utilizamos uma técnica de Transferência de Conhecimento (*Transfer Learning* - TL) (PAN; YANG, 2009), onde um modelo pré-treinado, geralmente em uma tarefa ampla, como classificação de imagens, é utilizado como ponto de partida para gerar características relevantes em uma tarefa diferente e mais específica. Isso acelera o processo de treinamento e melhora o desempenho do modelo em problemas com conjuntos de dados menores ou mais complexos. Desta maneira, a rede MobileNet utilizada foi treinada no contexto de classificação de imagens em um grande conjunto de dados chamado ImageNet (DENG et al., 2009), uma base com milhões de imagens categorizadas em várias classes. Essa base é amplamente utilizada para treinar e avaliar modelos de aprendizado profundo para tarefas de classificação de imagens.

Aplicamos a rede MobileNet em imagens pré-processadas com tamanho padrão de 299 de altura por 299 de largura, obtidas após um processo de redimensionamento. Realizamos a extração da saída da última camada convolucional e aplicamos nela uma Média Global de Agrupamento (*Global Average Pooling* - GAP) (LIN et al., 2013), uma técnica que reduz a dimensão espacial de um tensor ao calcular a média da dimensão analisada. Isso nos permitiu obter, para cada imagem equivalente, um espaço de características relevantes de tamanho 1024. As características geradas não capturam todas as informações da imagem, uma vez que são um mapeamento em um espaço reduzido, onde características mais relevantes para tarefas específicas são enfatizadas em detrimento de detalhes menos significativos. Em um contexto de TL, as características mais relevantes derivam do processo de classificação da base ImageNet, sendo possível que elas sejam suficientes para a detecção dos tumores cerebrais no estudo.

Para o processo de avaliação das técnicas, realizamos a separação aleatória de 20% do conjunto de dados de forma estratificada para o teste. A volumetria total pode ser observada, mais uma vez, na Tabela 15.

Adotamos o mesmo procedimento de busca tanto com o NSGA-II quanto com o NSGA-III. Além disso, considerando a proposta deste trabalho, detalhada no capítulo 6, algumas das variáveis de busca presentes no problema de otimização referem-se à lista de AF. Essas listas podem ser dos tipos adaptativo Normal, Camada ou Neurônio. Dessa forma, conseguimos avaliar se as estratégias adaptativas são capazes de auxiliar no processo de busca por arquiteturas que maximizem a métrica enquanto minimizam a quantidade de parâmetros, em comparação com as AF tradicionais. Utilizamos o NSGA-III, mesmo com apenas dois objetivos, pois o algoritmo apresenta ser mais eficiente em distribuição e diversidade, sendo útil para explorar melhor o espaço de soluções.

Utilizamos uma população de tamanho 16 que foi evoluída ao longo de 32 gerações, totalizando 512 avaliações. Quanto maior número de avaliações melhor para a exploração do espaço do problema, mas consequentemente aumenta o tempo do experimento realizado.

Sendo assim, buscamos um equilíbrio entre tempo do experimento e exploração. É importante ressaltar que, neste contexto, conduzimos a avaliação de 512 arquiteturas e configurações de ANN para identificar a frente de Pareto.

Cada indivíduo avaliado representa uma configuração de ANN e alguns de seus parâmetros de treinamento. De forma empírica, definimos as arquiteturas para possuírem somente uma camada interna e utilizarem apenas a AF Fourier (Subseção 5.3.15), com $n = 4$, $n = 8$, $n = 16$, $n = 32$ e $n = 64$. Conduzimos a busca pelo número de neurônios por camada, variando de um mínimo de um até um máximo equivalente ao número de características da base de dados modelada. Também consideramos a inclusão de normalização em lote, normalização de pesos e a taxa de abandono por camada. Adicionalmente, em relação aos parâmetros de treinamento, exploramos a quantidade de épocas nas quais a ANN será treinada, com um mínimo de 256 e um máximo de 1024. Definimos a taxa de aprendizado variando entre um mínimo de 10^{-3} e um máximo de 10^{-2}, além do tamanho do lote de treinamento, que variou entre um mínimo de 1 e um máximo de 895.

É relevante notar que entre as variáveis de busca existem números inteiros e reais, o que demanda o tratamento apropriado nos processos de recombinação e mutação. Os números inteiros são representados por meio da representação binária. Para tal, utilizamos o operador SBX com probabilidade de troca de 0.5 e parâmetro $\eta = 15$, bem como a Recombinação de n Pontos com probabilidade de troca de 0.5 para todos os bits. Esses são os operadores empregados para recombinação de valores reais e inteiros, respectivamente. Adicionalmente, para os processos de mutação, empregamos a Mutação Polinomial com probabilidade de troca de 1.0, e a Mutação de Inversão de Bits, com probabilidade proporcional à quantidade de bits na solução, para valores reais e inteiros, respectivamente.

Durante a etapa de avaliação do indivíduo, conduzimos uma validação cruzada com K-*fold*, onde $K = 3$, com o objetivo de calcular um valor médio da métrica ROC AUC nos dados de validação e usar esse valor como resultado de uma das funções objetivo. É relevante notar que a ANN será treinada K vezes para obter uma avaliação mais consistente de sua qualidade de predição. A outra função objetivo, relativa à quantidade de parâmetros, é obtida no processo de construção da ANN, envolvendo todos os parâmetros ajustáveis da mesma.

Durante o processo de treinamento da ANN, empregou-se o otimizador Adam (KINGMA; BA, 2014), e 20% dos dados destinados ao treinamento da ANN, após a utilização do K-*fold*, foram alocados para um procedimento de parada de treinamento, caso a ANN apresentasse deterioração ao longo das últimas 10 épocas. Além disso, os otimizadores utilizados foram projetados para minimizar as funções objetivo. Portanto, foi necessário multiplicar a métrica ROC AUC por um fator de -1, a fim de converter a métrica que deve ser maximizada em uma equivalente que deve ser minimizada.

Realizamos 10 execuções independentes no processo de busca e calculamos as métricas de Hipervolume, Espaçamento e Cobertura. Em algumas análises, retiramos três soluções da frente de Pareto encontrada, sendo elas: a que apresentou o melhor resultado em métrica, aquela que apresentou o menor número de parâmetros e, por fim, a solução intermediária que está mais próxima da origem de acordo com a distância euclidiana. No experimento, para garantir a origem em (0, 0), realizamos a adição de 1 à métrica equivalente (- ROC AUC), com o intuito de tornar zero o melhor e menor valor que pode ser alcançado. Para esta segunda análise, calculamos as métricas ROC AUC e Quantidade de Parâmetros. Essas análises serão divididas nas seguintes subseções.

7.4.1 Análise da Frente de Pareto

As tabelas 45 e 46, presentes no Anexo D.1, apresentam os resultados sumarizados do quarto experimento numérico para os otimizadores NSGA-II e NSGA-III, bem como o tipo adaptativo da ANN. Elas exibem os valores mínimo, médio, mediano e máximo, além do desvio padrão, para as métricas Hipervolume e Espaçamento. Adicionalmente, as Figuras 108 e 109, também presentes no Anexo D.1, ilustram os *Violin Plots* das métricas Hipervolume e Espaçamento, respectivamente. Esses gráficos proporcionam uma visualização da distribuição das soluções encontradas.

É possível observar que, em relação à métrica Hipervolume, as estratégias adaptativas apresentam melhores valores em comparação com a estratégia Normal. Já em relação à métrica Espaçamento, é difícil afirmar qual estratégia é melhor, visto que o trio apresenta resultados bons e ruins em diferentes cenários. Por outro lado, é possível observar novamente que os resultados das estratégias adaptativas por Camada e Neurônio se igualaram em muitas comparações e se intercalaram na métrica Hipervolume, dependendo do otimizador utilizado. Mais uma vez, assim como já foi apresentado nos experimentos anteriores, não foi possível definir qual estratégia é melhor de forma geral. Ao analisarmos os algoritmos de otimização, ambos apresentam resultados comparáveis. Se torna ideal conduzir outras análises qualitativas dos resultados para indicar qual algoritmo apresenta melhor desempenho.

Para aprofundar a análise comparativa dos resultados, aplicamos o ANOVA com um intervalo de confiança de 95%, visando avaliar a hipótese nula de que não há diferença significativa entre as médias de cada conjunto (otimizador e tipo adaptativo). Os resultados obtidos indicam que, na maioria dos casos, o teste rejeitou a hipótese nula, revelando a existência de grupos formados pelos otimizadores e tipos adaptativos que apresentam desempenhos distintos. Entretanto, é importante ressaltar que o ANOVA não é capaz de identificar quais grupos apresentam diferenças médias.

Utilizamos, assim, o teste de Tukey com um intervalo de confiança de 95% para verificar as situações em que a diferença entre as médias é significativa. A Figura 71 exibe

uma contagem das situações em que os otimizadores e os tipos de estratégias adaptativas apresentaram diferença média significativa, de acordo com o teste de Tukey, para as duas métricas analisadas.

Ao analisar a Figura 71a, que exibe a métrica Hipervolume, é possível observar que, ao mantermos o otimizador fixo e compararmos os tipos adaptativos, identificamos uma diferença significativa entre a estratégia Normal e as estratégias Camada e Neurônio. Agora, ao compararmos a estratégia de Camada com a de Neurônio, observamos que não há diferença significativa. Ao examinarmos os algoritmos de otimização, notamos que a estratégia Normal com o NSGA-II não apresenta diferença significativa quando comparada com a estratégia Normal do NSGA-III. No entanto, existe uma diferença significativa entre as estratégias adaptativas do NSGA-III. Além disso, ao analisarmos a estratégia de Neurônio do NSGA-II em comparação com todos os tipos adaptativos do NSGA-III, encontramos diferença significativa em todos os cenários. Somando esses resultados aos anteriores, constatamos que essa métrica demonstrou melhores resultados em relação à métrica Hipervolume.

Agora, ao analisarmos a Figura 71b, que exibe a métrica Espaçamento, podemos observar que somente a estratégia Camada do NSGA-II demonstrou diferença significativa em comparação com a estratégia Normal do NSGA-III. As demais estratégias não apresentaram diferença significativa, sugerindo que são soluções semelhantes no que diz respeito à métrica Espaçamento.

Fizemos a agregação das 10 execuções independentes para formar uma única frente de Pareto em relação a cada conjunto de dados, algoritmo de otimização e estratégia adaptativa. Essas frentes de Pareto estão disponíveis na Figura 110, que se encontra no Anexo D.1. Com base nessas frentes de Pareto, podemos calcular a métrica de Cobertura, conforme detalhado na Tabela 16.

Tabela 16 – Comparação da métrica de Cobertura.

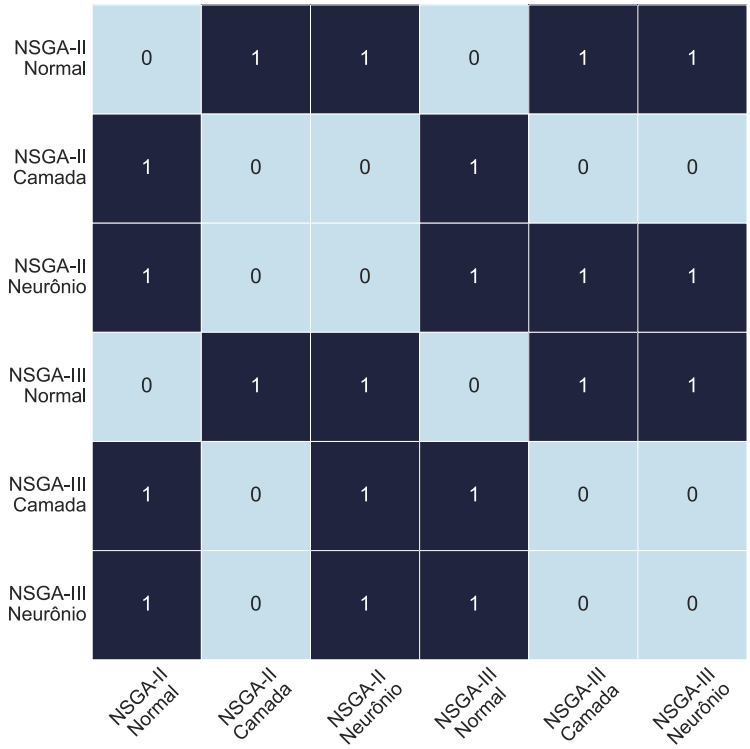
Conjunto A		Conjunto B					
		NSGA-II			NSGA-III		
		Normal	Camada	Neurônio	Normal	Camada	Neurônio
NSGA-II	Normal	0.000	**43.900**	**34.040**	**25.710**	**28.570**	**31.030**
	Camada	**60.610**	0.000	**53.190**	**48.570**	**17.860**	**48.280**
	Neurônio	**57.580**	**39.020**	0.000	**48.570**	**17.860**	**20.690**
NSGA-III	Normal	**42.420**	**46.340**	**36.170**	0.000	**32.140**	**37.930**
	Camada	**66.670**	**58.540**	**68.090**	**57.140**	0.000	**72.410**
	Neurônio	**60.610**	**31.710**	**36.170**	**54.290**	**14.290**	0.000

Fonte: Elaborada pelo autor (2023).

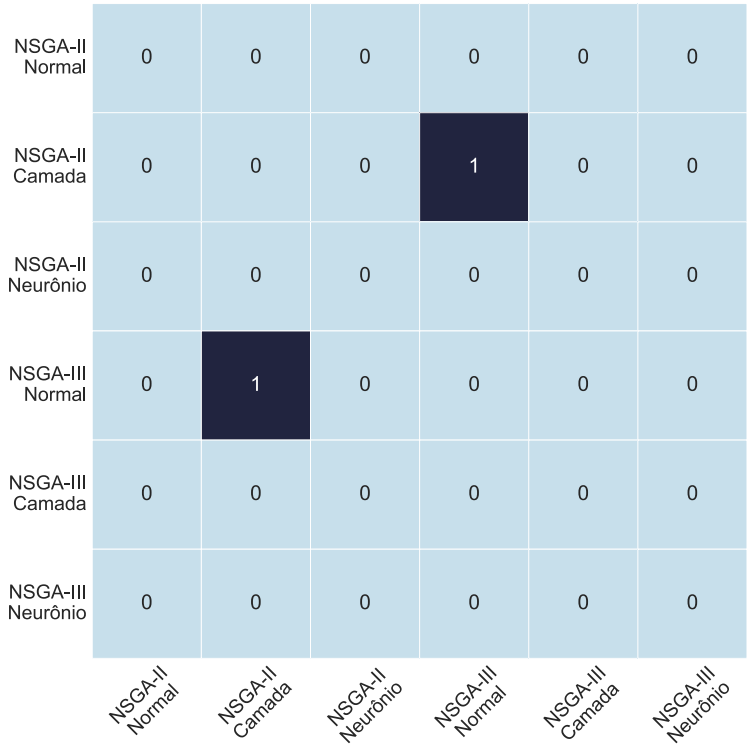
Observamos que, independentemente do algoritmo de otimização, as estratégias

Figura 71 – Contagem dos otimizadores e tipos adaptativos que apresentaram diferença média significativa de acordo com o teste de Tukey para o quarto experimento.

(a) Hipervolume



(b) Espaçamento



Fonte: Elaborada pelo autor (2023).

Camada e Neurônio tendem a possuir, de maneira geral, um valor maior de cobertura em comparação com a estratégia Normal, sobretudo a estratégia Normal do NSGA-II. Também é notável que, ao compararmos os otimizadores, a cobertura do NSGA-III em relação ao NSGA-II é maior do que a cobertura do NSGA-II em relação ao NSGA-III, quando consideramos tipos adaptativos equivalentes. Além disso, em todos os algoritmos de otimização, a estratégia Camada mostra-se menos coberta pelas outras estratégias, resultando em uma frente de Pareto menos dominada, especialmente a estratégia Camada do NSGA-III.

Ao reunir todas as análises efetuadas nesta subseção, é possível constatar que as estratégias adaptativas, seja por Camada ou por Neurônio, exibem resultados superiores em comparação com a estratégia Normal. Essa conclusão se torna ainda mais evidente ao observarmos a métrica Hipervolume. No caso da métrica Espaçamento, os tipos adaptativos apresentaram apenas um par com diferença significativa entre os grupos.

Ao compararmos a estratégia por Camada em relação à estratégia por Neurônio, percebemos que ambas demonstram um desempenho competitivo entre si, embora a estratégia por Camada apresente valores superiores para a métrica Cobertura. Essa observação nos leva a considerar a estratégia por Camada como potencialmente mais vantajosa em relação às outras.

7.4.2 Análise de Soluções da Frente de Pareto

As Tabelas 47 e 48, presentes no Anexo D.2, apresentam os resultados sumarizados do quarto experimento numérico para os otimizadores NSGA-II e NSGA-III, soluções extraídas da frente de Pareto e o tipo adaptativo da ANN, exibindo os valores mínimo, médio, mediano e máximo para as métricas ROC AUC e Quantidade de Parâmetros, além do desvio padrão para as 10 execuções independentes. Complementando, as Figuras 111 e 112, também presentes no Anexo D.2, representam os *Violin plots* das métricas ROC AUC e Quantidade de Parâmetros, respectivamente. Esses gráficos oferecem uma visualização da distribuição das soluções encontradas.

Novamente, apesar de esperado, é importante destacar que as soluções extraídas de uma execução independente faziam parte de uma frente de Pareto, ou seja, nenhuma dessas soluções dominava as outras. Desta forma, nos resultados, podemos observar que as soluções que foram extraídas por apresentarem o melhor resultado em relação à métrica ROC AUC serão aquelas que também apresentarão os melhores resultados para a mesma métrica. Isso se mantém também para a Quantidade de Parâmetros.

Considerando as soluções “Melhor em Métrica” e “Próximo da Origem” para a métrica ROC AUC, as estratégias adaptativas apresentam melhores valores em comparação com a estratégia Normal. Por outro lado, para a solução “Melhor em Parâmetro”, a estratégia Normal demonstra melhores resultados. No caso da métrica Quantidade de

Parâmetros, é difícil observar qual estratégia é superior, pois o trio apresenta resultados bons e ruins em diferentes cenários. Além disso, é evidente que os resultados da estratégia adaptativa por Neurônio, tanto para o NSGA-II quanto para o NSGA-III, revelaram os melhores desempenhos em relação à métrica ROC AUC. Isso sugere que essa estratégia é geralmente mais adequada para esta análise. Quando examinamos os algoritmos de otimização, ambos demonstram resultados comparáveis, tornando desejáveis análises qualitativas adicionais para determinar qual algoritmo apresenta o melhor desempenho.

Complementando as análises anteriores, empregamos o ANOVA com um intervalo de confiança de 95% para investigar a hipótese nula de que não há diferença significativa entre as médias de cada grupo (otimizador, solução extraída e tipo adaptativo). Os resultados obtidos indicaram que, em certos casos, o teste rejeitou a hipótese nula, revelando a existência de grupos compostos pelo otimizador, solução extraída e tipos adaptativos que exibem desempenhos distintos. Contudo, o ANOVA não consegue identificar especificamente quais grupos apresentam diferenças nas médias.

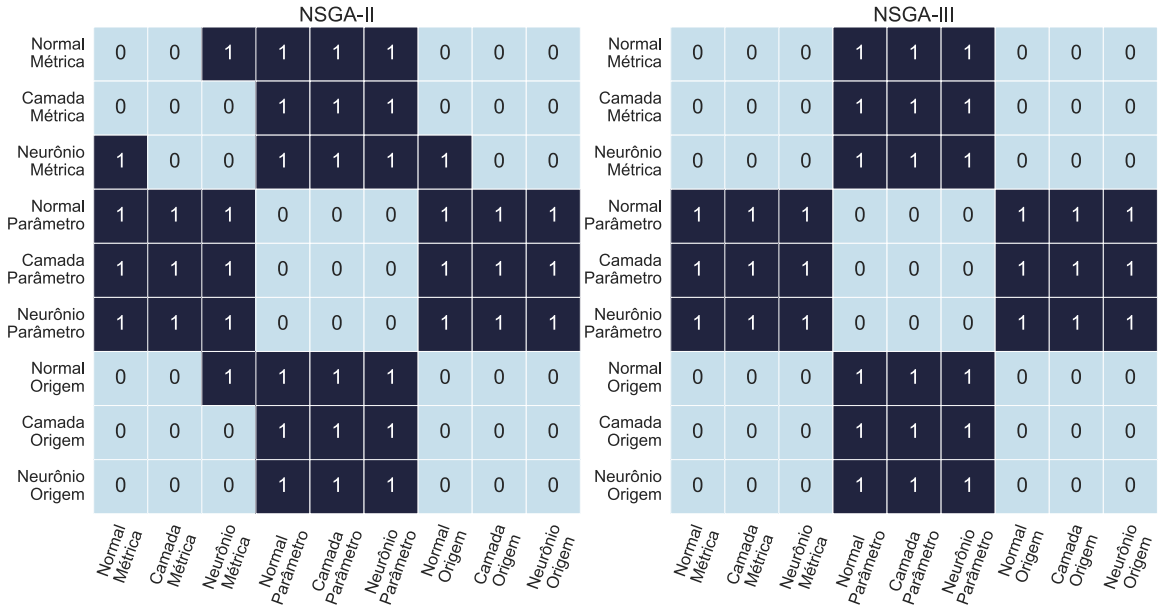
Utilizamos, portanto, o teste de Tukey, com um intervalo de confiança de 95%, para verificar onde a diferença entre as médias é significativa. A Figura 72 apresenta uma contagem das situações em que os otimizadores, soluções extraídas e tipos de funções adaptativas exibiram diferença média significativa de acordo com o teste de Tukey para as duas métricas estudadas.

Ao analisar a Figura 72a, que ilustra a métrica ROC AUC, podemos observar que entre as soluções de melhor métrica, para o otimizador NSGA-II, a única diferença significativa ocorreu entre os casos Normal e Neurônio no contexto adaptativo. Além disso, ao comparar as soluções extraídas que apresentam os melhores valores de ROC AUC com aquelas que estão próximas da origem, notamos que, para o otimizador NSGA-II, a única diferença significativa ocorreu entre o caso Normal próximo à origem e o caso Neurônio com melhor valor de métrica. Por fim, é observável que, em sua maioria, tanto para o NSGA-II quanto para o NSGA-III, as soluções próximas à origem não demonstraram diferença significativa em relação às soluções extraídas que apresentam melhores valores de ROC AUC. Nesse cenário, a escolha recomendada seria utilizar as soluções próximas à origem, pois elas possuem um menor número de parâmetros.

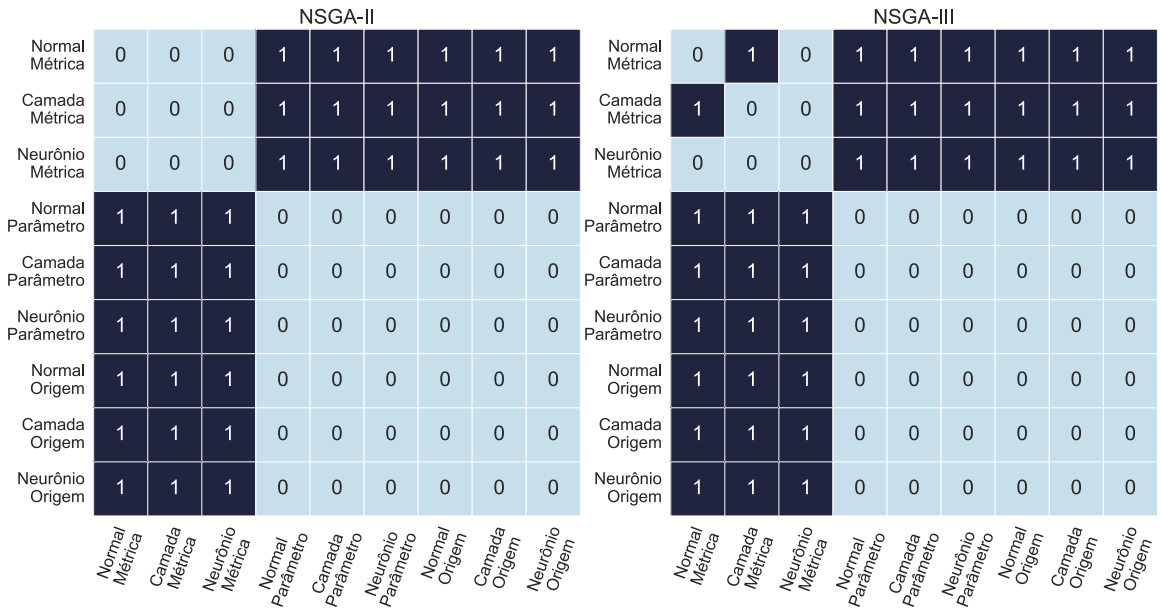
Ao analisar a Figura 72b, é evidente que as soluções extraídas pelo NSGA-II, que apresentam o melhor valor de ROC AUC, exibiram diferença significativa em apenas um par composto por Normal e o tipo adaptativo Camada. Quando examinamos os grupos de soluções extraídas da frente de Pareto, percebemos que as soluções com o melhor valor de métrica mostraram diferença significativa, em todos os casos, em comparação com os outros dois grupos. Por outro lado, as soluções próximas à origem não revelam diferença significativa, em todos os casos, em relação às soluções com o melhor número de parâmetros. Isso fortalece a recomendação de utilizar as soluções próximas à origem, visto

Figura 72 – Contagem dos otimizadores, soluções extraídas e tipos adaptativos que apresentaram diferença média significativa de acordo com o teste de Tukey para o terceiro experimento.

(a) ROC AUC



(b) Quantidade de Parâmetros



Fonte: Elaborada pelo autor (2023).

que elas possuem um menor número de parâmetros.

Por fim, com o propósito de avaliar a relevância do valor de n na AF Fourier (5.3.15), as Figuras 73 e 74 apresentam, para os otimizadores NSGA-II e NSGA-III, respectivamente, a frequência de seleção das AFs para os tipos adaptativos e para as soluções extraídas da frente de Pareto.

É possível observar que, independentemente do otimizador e do tipo de solução analisado, as AFs com $n = 4$ foram predominantemente mais selecionadas no processo de busca, seguidas na maioria das vezes pelas AFs com $n = 8$. Isso nos conduz a concluir que, para este experimento, as AFs com $n = 4$ são suficientes para atender a ambas as funções objetivo, ou seja, não aumentam consideravelmente a quantidade de parâmetros e ainda conseguem encontrar soluções satisfatórias.

É relevante destacar que o aumento de n para a estratégia Normal e por Camada não possui um impacto significativo, ao contrário do que ocorre para a estratégia Neurônio. Para as estratégias Normal e por Camada, quando $n = 4$, temos precisamente $3 \cdot n + 1 = 3 \cdot 4 + 1 = 13$ parâmetros, sejam eles ajustáveis ou não, independentemente da quantidade de neurônios dentro da camada. No entanto, ao analisarmos a estratégia Neurônio, esse número é multiplicado pela quantidade de neurônios dentro da camada interna, podendo chegar de 13 parâmetros a 1300 parâmetros se tivermos 100 neurônios.

A Tabela 17 proporciona uma análise comparativa das quantidades de neurônios para os diferentes tipos adaptativos, considerando todas as 10 frentes de Pareto. É perceptível que, em média, o caso Normal exibe a maior quantidade de neurônios, com uma média de 58.70 neurônios por solução. A sequência é seguida pelo tipo adaptativo Camada, com uma média de 48.78 neurônios, e o tipo adaptativo Neurônio, com 42.10 neurônios em média. Ademais, é evidente que os desvios padrão variam consideravelmente, indicando a dispersão dos dados ao redor das médias. Essa tabela possibilita a observação de que o tipo adaptativo Neurônio apresentou um número menor de neurônios, provavelmente devido à consideração de que a quantidade de parâmetros aumenta consideravelmente.

Tabela 17 – Análise comparativa das quantidades de neurônios para diferentes tipos adaptativos usando Frentes de Pareto.

Tipo Adaptativo	Quantidade Neurônios				
	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
Normal	1	58.70	157.37	12.0	1024
Camada	1	48.78	88.24	11.0	511
Neurônio	1	42.10	73.13	14.5	485

Fonte: Elaborada pelo autor (2023).

Ao compilar as análises efetuadas nesta subseção, percebe-se que as estratégias adaptativas, tanto por Camada quanto por Neurônio, embora apresentem melhores resultados em comparação com a estratégia Normal, não conseguiram gerar uma diferença significativamente notável. Apesar disso, torna-se evidente que a extração de uma solução intermediária é suficiente para assegurar um modelo com uma boa capacidade de inferência e, simultaneamente, com um número reduzido de parâmetros. Além disso, constatamos que as AFs Fourier, com valores baixos de n, já são amplamente suficientes para ajustar o

problema do modelo.

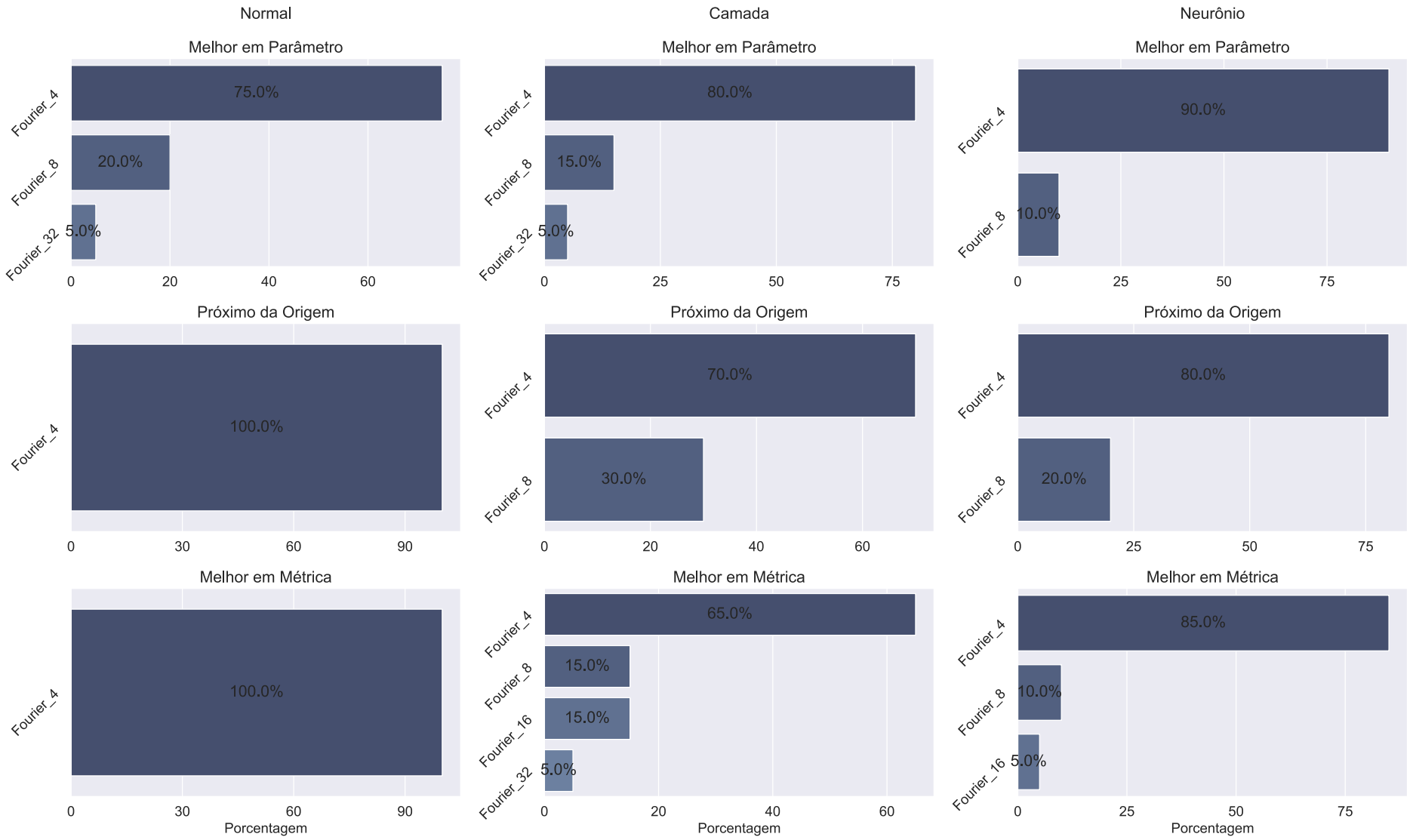
7.4.3 Pegadas de Carbono

A Tabela 49, presente no Anexo D, exibe as estimativas médias de emissão de CO_2, em quilogramas, associadas ao treinamento das três soluções retiradas da frente de Pareto encontrada, agregando todas as 10 execuções independentes do quarto experimento. Estas soluções são: aquela que apresentou o melhor resultado em métrica, aquela que apresentou o menor número de parâmetros e, por fim, a solução intermediária que está mais próxima da origem, de acordo com a distância euclidiana.

Podemos observar que o otimizador NSGA-III obteve resultados gerais inferiores em comparação com o NSGA-II. Em ambos os otimizadores, o tipo Normal apresentou consistentemente resultados mais favoráveis em relação às métricas de emissões de CO_2 quando comparado aos tipos adaptativos Camada e Neurônio. Além disso, no contexto do NSGA-II, o tipo adaptativo Camada resultou em emissões de CO_2 mais baixas do que o tipo adaptativo Neurônio, enquanto no NSGA-III, o padrão foi invertido, com Neurônio apresentando resultados ligeiramente melhores do que Camada. Em relação às soluções extraídas, as soluções que possuem menor número de parâmetros tiveram impactos variados nas emissões de CO_2.

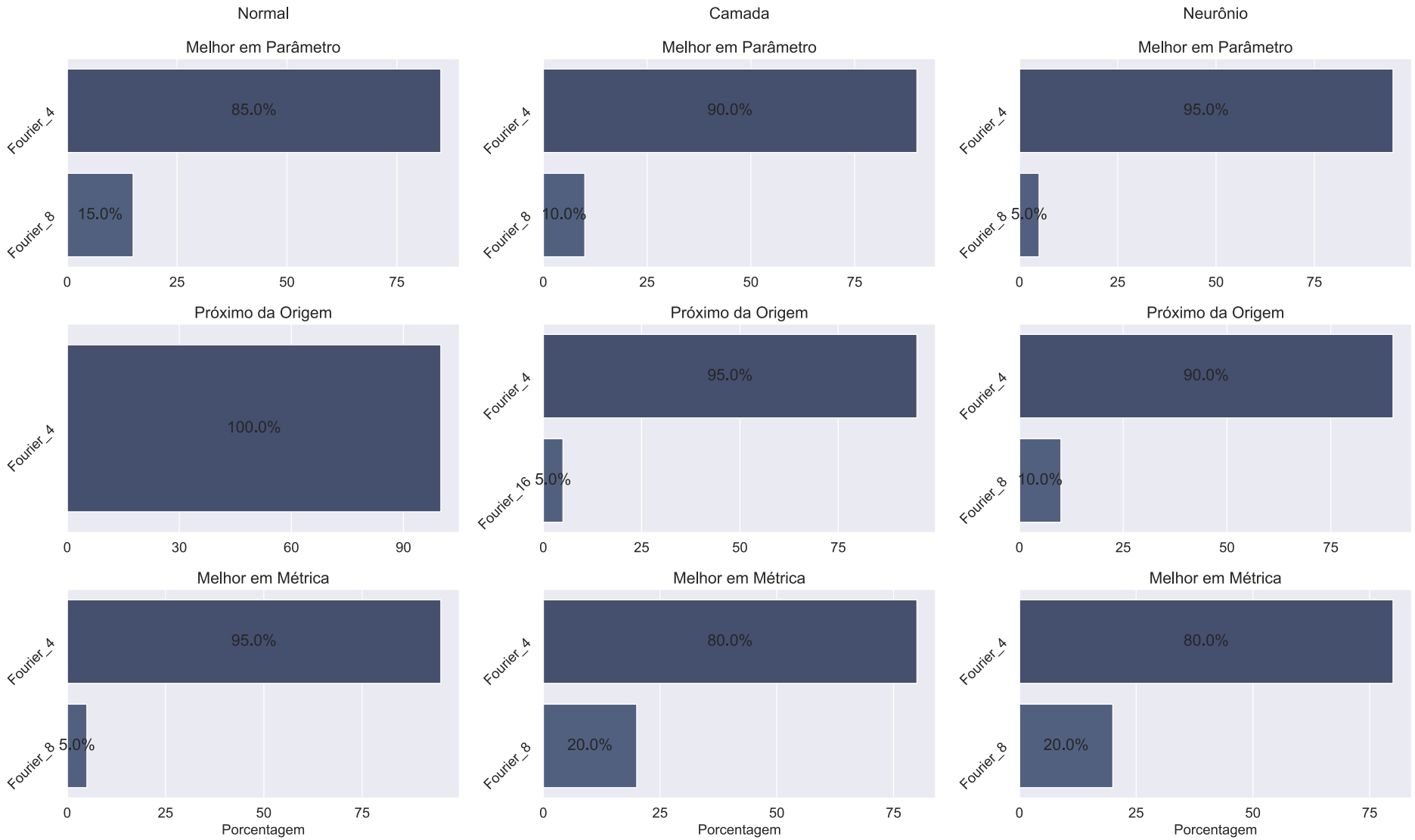
Apesar desta análise completa, o maior valor apresentado foi de $1.77e^{-5}$ kg, o que equivale a menos de 2.2% do que é gasto com a carga de um telefone.

Figura 73 – Frequência de seleção das AFs para o NSGA-II.



Fonte: Elaborada pelo autor (2023).

Figura 74 – Frequência de seleção das AFs para o NSGA-III.



Fonte: Elaborada pelo autor (2023).

8 CONCLUSÃO

Este trabalho teve como motivação investigar os desafios presentes no processo de criação de redes neurais artificiais e propor soluções que sejam capazes de melhorar a eficiência desse processo, tanto em questões de qualidade de predição quanto em outros aspectos, como limitações de *hardware* e impactos ambientais.

A hipótese assumida para definir nosso objetivo principal foi a de que as funções de ativação adaptativas seriam capazes de criar neurônios mais complexos na arquitetura, reduzindo a necessidade de seguir com abordagens de aprendizado profundo, onde os modelos se tornam mais complexos, pesados e com número de parâmetros ajustáveis consideravelmente maior.

Através desta motivação e hipótese, o objetivo principal deste trabalho consistiu em desenvolver uma estratégia evolutiva para a criação de redes neurais artificiais que utilizam as funções de ativação adaptativas em seu processo de treinamento, com o intuito de encontrar modelos que maximizassem o poder preditivo e minimizassem sua complexidade. Tais modelos, que atendem a esses dois critérios de otimização, revelam-se mais adequados para serem implantados em diversos tipos de *hardware*, incluindo aqueles com recursos computacionais limitados, ao mesmo tempo em que mantêm níveis equiparáveis de pegada de carbono.

Foram realizados quatro conjuntos de experimentos com o objetivo de avaliar nossa hipótese e proposta. Os dois primeiros experimentos, cada um composto por dez problemas, analisaram a eficácia das funções de ativação adaptativas em comparação com uma função de ativação tradicional, mantendo fixa a arquitetura e os parâmetros de treinamento em problemas de classificação e regressão. O terceiro experimento, com cinco problemas, concentrou-se na proposta central deste trabalho, que combina abordagens evolutivas com funções de ativação adaptativas para encontrar o melhor modelo em termos de métrica e complexidade para problemas de *Autoencoder*. Por fim, o quarto e último experimento também se concentrou na proposta central deste trabalho, mas com foco em um problema real de detecção de tumores cerebrais em imagens de ressonância magnética.

Ao analisar o primeiro experimento, relacionado aos problemas de classificação, concluímos que, para as bases estudadas, as funções de ativação adaptativas apresentaram resultados superiores em comparação com as funções de ativação tradicionais, maximizando a métrica F1 e minimizando o *Log Loss*. No entanto, não podemos afirmar, em geral, se a estratégia por Camada ou por Neurônio é melhor. Ao utilizar outra medida de desempenho, o Perfil de Desempenho, notamos que a vantagem das funções de ativação adaptativas se manteve. A estratégia por Neurônio é melhor, em média, na métrica F1, enquanto a estratégia por Camada é mais eficaz no melhor valor. Quanto à métrica *Log Loss*, a estratégia por Camada predominou, independentemente de se calcular a média ou o melhor

valor. Por fim, ao conduzir alguns testes estatísticos, observamos que as funções Cúbicas e Quadráticas proporcionaram maior benefício ao utilizar a versão adaptativa, ajustando-se melhor aos problemas, enquanto as funções Sigmoides e SoftPlus apresentaram um ganho menor, embora não tenham se deteriorado em relação à versão tradicional.

Na análise do segundo experimento, relacionado aos problemas de regressão, chegamos à mesma conclusão de que, para as bases estudadas, as funções de ativação adaptativas apresentaram melhores resultados em comparação com as funções de ativação tradicionais, maximizando a métrica R^2 e minimizando o RMSE. No entanto, não foi possível determinar, de forma geral, se a estratégia por Camada ou por Neurônio é superior. Ao utilizar o Perfil de Desempenho, notamos que a vantagem das funções de ativação adaptativas se manteve. Para a métrica R^2, a estratégia por Neurônio apresentou melhores resultados quando utilizado o melhor valor, enquanto para a média ambas tiveram resultados semelhantes. Quando analisamos a métrica RMSE, a estratégia por Neurônio foi predominante, independentemente de se calcular a média ou o melhor valor. Em conclusão, ao conduzir alguns testes estatísticos, observamos novamente que as funções Cúbicas e Quadráticas proporcionaram maior benefício ao utilizar a versão adaptativa, adaptando-se melhor aos problemas. Um número maior de funções apresentou um ganho menor, embora não tenham se deteriorado em relação à versão tradicional.

No terceiro experimento, adentramos em uma análise do objetivo do trabalho, que consistia em utilizar a abordagem evolutiva para encontrar as melhores arquiteturas de redes neurais artificiais e avaliar o impacto do uso das funções de ativação adaptativas no processo de busca. A conclusão obtida permanece a mesma: para as bases estudadas, as funções de ativação adaptativas apresentaram resultados superiores em comparação com as funções de ativação tradicionais. No entanto, a qualidade da solução final está diretamente relacionada ao otimizador utilizado no processo de busca. Ao compararmos o NSGA-II com o NSGA-III em relação à métrica Hipervolume, observamos que o NSGA-III apresenta melhores resultados. Porém, ao utilizarmos a métrica de Espaçamento, o comportamento é o oposto. É necessário determinar qual métrica é mais adequada para o problema em questão para, assim, determinar qual algoritmo é superior. Quando empregamos a métrica de Cobertura, que permite avaliar o quanto as soluções de uma estratégia dominam as outras, podemos concluir que a estratégia por Camada é a menos dominada, enquanto a estratégia Normal é a mais dominada.

No quarto e último experimento, exploramos o propósito do trabalho, que envolve a aplicação da abordagem evolutiva multiobjetivo na descoberta das estruturas mais eficazes de redes neurais artificiais, avaliando o impacto das funções de ativação adaptativas no processo de busca para um problema real. Os resultados obtidos indicam que, em relação à métrica Hipervolume, as estratégias adaptativas superam a estratégia Normal, embora a métrica Espaçamento não revele uma clara superioridade. Observou-se que a estratégia de Neurônio do NSGA-II tende a apresentar melhores resultados em termos de Hipervolume,

enquanto a estratégia Camada do NSGA-II se destaca na métrica Espaçamento. Além disso, as estratégias adaptativas, especialmente a estratégia por Neurônio, se destacaram na métrica ROC AUC, sugerindo sua preferência geral. Por outro lado, a estratégia Normal teve vantagens na métrica relacionada a um menor número de parâmetros, o que é coerente, uma vez que não possui os parâmetros adicionais das funções de ativação adaptativas. No entanto, soluções próximas à origem, que priorizam um equilíbrio entre as funções objetivos, geralmente apresentaram menos parâmetros e um bom desempenho na métrica ROC AUC. Ademais, a função de ativação Fourier com um número de parâmetros reduzidos já é capaz de gerar funções suficientes para o problema, sem a necessidade de aumentar o número de parâmetros de treinamento para performar.

Em todos os experimentos realizados, foi conduzida uma análise da Pegada de Carbono, demonstrando que, independentemente das condições experimentais, a discrepância entre as estratégias adaptativas foi insignificante. Isso sugere que não há justificativa para aplicar penalidades ao optar por estratégias adaptativas. Além disso, é importante observar que não houve uma redução significativa na Pegada de Carbono para os modelos analisados, possivelmente devido à simplicidade das arquiteturas utilizadas. No entanto, a estratégia adaptativa manteve uma Pegada de Carbono semelhante, ao mesmo tempo em que aumentou o desempenho da solução, induzindo novamente à utilização da estratégia.

Concluímos, com base nos experimentos realizados, que a utilização de funções de ativação adaptativas realmente proporciona benefícios para a rede neural artificial. Isso resulta em um melhor desempenho preditivo, mantendo, no pior cenário, o modelo com a mesma complexidade que teria se utilizado funções tradicionais. Essa constatação sugere que o uso de funções de ativação adaptativas deve ser incentivado sempre que possível. Embora em muitas situações as estratégias por Camada e por Neurônio possam competir para determinar a melhor abordagem, a estratégia por Camada demonstrou ser mais eficaz, especialmente na análise de Cobertura. Isso não exclui a possibilidade de que a estratégia por Neurônio possa obter melhores resultados em outros problemas ou contextos.

É importante destacar que mais estudos podem e devem ser realizados em problemas diferentes e mais complexos, a fim de complementar as conclusões obtidas ao longo deste trabalho.

8.1 TRABALHOS FUTUROS

Como trabalhos futuros, que dão continuidade ao apresentado aqui, se destacam:

- Aplicação em problemas de classificação e regressão, com foco ainda em MLP, com instancias de treinamento relativamente grandes (≥ 10.000);
- Aplicação em problemas de classificação e regressão, com foco ainda em MLP, com quantidade de características relativamente grandes (≥ 5.000);

- Aplicação em problemas de séries temporais, com foco ainda em MLP;
- Aplicar as funções de ativação adaptativas propostas em diferentes tipos de arquitetura, como Rede Neural Convolucional, Rede Neural Recorrente, Modelos de Disparos Neuronais e outras;
- Aplicar a estratégia evolutiva proposta em diferentes tipos de arquitetura, como Rede Neural Convolucional, Rede Neural Recorrente, Modelos de Disparos Neuronais e outras;
- Entender e evoluir a estratégia para aumentar a interpretabilidade dos modelos propostos, especialmente em cenários onde a explicabilidade do modelo é crucial;
- Realizar análises de sensibilidade para avaliar a robustez dos modelos em relação a perturbações nos dados de entrada e nos parâmetros do modelo;
- Testar e adaptar os modelos propostos para aplicações em domínios específicos, como finanças, saúde, manufatura, etc.;
- Investigar como técnicas avançadas de pré-processamento, como redução de dimensionalidade não linear ou técnicas de normalização específicas para determinados tipos de dados, podem impactar o desempenho dos modelos.

REFERÊNCIAS

AGOSTINELLI, Forest; HOFFMAN, Matthew; SADOWSKI, Peter; BALDI, Pierre. Learning activation functions to improve deep neural networks. **arXiv preprint arXiv:1412.6830**, 2014.

AHLBERG, J Harold; NILSON, Edwin Norman; WALSH, Joseph Leonard. **The Theory of Splines and Their Applications: Mathematics in Science and Engineering: A Series of Monographs and Textbooks, vol. 38**. [S.l.]: Elsevier, 2016. v. 38.

Alibaba Cloude. **Shortening Machine Learning Development Cycle with AutoML**. 2018. https://www.alibabacloud.com/blog/shortening-machine-learning-development-cycle-with-automl_594232, Último acesso em 30 de Outubro de 2022.

ALIBRAHIM, Hussain; LUDWIG, Simone A. Hyperparameter optimization: comparing genetic algorithm against grid search and bayesian optimization. In: IEEE. **2021 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.], 2021. p. 1551–1559.

ALPAYDIN, Ethem. **Introduction to machine learning**. [S.l.]: MIT press, 2020.

ALSOLAI, Hadeel; ROPER, Marc. Determining the best prediction accuracy of software maintainability models using auto-weka. In: SPRINGER. **International Conference on Computing**. [S.l.], 2019. p. 60–70.

AMAZON. **Amazon SageMaker – Accelerating Machine Learning**. 2017. https://aws.amazon.com/blogs/aws/sagemaker/, Último acesso em 30 de Outubro de 2022.

ARORA, Jasbir. **Introduction to optimum design**. [S.l.]: Academic Press, 1989.

BACK, Thomas. **Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms**. [S.l.]: Oxford university press, 1996.

Baidu. **EZDL**. 2018. http://ai.baidu.com/ezdl/, Último acesso em 30 de Outubro de 2022.

BAKER, Bowen; GUPTA, Otkrist; NAIK, Nikhil; RASKAR, Ramesh. Designing neural network architectures using reinforcement learning. **arXiv preprint arXiv:1611.02167**, 2016.

BALDEON-CALISTO, Maria; LAI-YUEN, Susana K. Adaresu-net: Multiobjective adaptive convolutional neural network for medical image segmentation. **Neurocomputing**, Elsevier, v. 392, p. 325–340, 2020.

BANZHAF, Wolfgang; NORDIN, Peter; KELLER, Robert E; FRANCONE, Frank D. **Genetic programming**. [S.l.]: Springer, 1998.

BAO, Yukun; LIU, Zhitao. A fast grid search method in support vector regression forecasting time series. In: SPRINGER. **International Conference on Intelligent Data Engineering and Automated Learning**. [S.l.], 2006. p. 504–511.

BARBOSA, Helio JC; BERNARDINO, Heder S; BARRETO, André MS. Using performance profiles to analyze the results of the 2006 cec constrained optimization competition. In: IEEE. **IEEE congress on evolutionary computation**. [S.l.], 2010. p. 1–8.

BAZARAA, Mokhtar S.; JARVIS, John J.; SHERALI, Hanif D. **Linear Programming and Network Flows**. USA: Wiley-Interscience, 2004. ISBN 0471485993.

BENGIO, Yoshua; GRANDVALET, Yves. No unbiased estimator of the variance of k-fold cross-validation. **Journal of machine learning research**, v. 5, n. Sep, p. 1089–1105, 2004.

BERGSTRA, James; BENGIO, Yoshua. Random search for hyper-parameter optimization. **Journal of machine learning research**, v. 13, n. Feb, p. 281–305, 2012.

BERGSTRA, James S; BARDENET, Rémi; BENGIO, Yoshua; KéGL, Balázs. Algorithms for hyper-parameter optimization. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2011. p. 2546–2554.

BIANCHI, Leonora; DORIGO, Marco; GAMBARDELLA, Luca Maria; GUTJAHR, Walter J. A survey on metaheuristics for stochastic combinatorial optimization. **Natural Computing**, Springer, v. 8, n. 2, p. 239–287, 2009.

BISHOP, Christopher M. **Pattern Recognition and Machine Learning**. New York: Springer, 2006. v. 1. 118 p.

BISHOP, Christopher M et al. **Neural networks for pattern recognition**. [S.l.]: Oxford university press, 1995.

BLUM, Christian; ROLI, Andrea. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM computing surveys (CSUR)**, Acm New York, NY, USA, v. 35, n. 3, p. 268–308, 2003.

BOLFARINE, Heleno; BUSSAB, Wilton de Oliveira. **Elementos de Amostragem**. [S.l.]: Editora Edgard Blucher, 2005.

BRADLEY, Andrew P. The use of the area under the roc curve in the evaluation of machine learning algorithms. **Pattern recognition**, Elsevier, v. 30, n. 7, p. 1145–1159, 1997.

BROWNLEE, Jason. Mean squared error: a risk measure for continuous probability distributions. **Journal of Artificial Intelligence Research**, AI Access Foundation, v. 44, p. 935–947, 2011.

BROWNLEE, Jason. **Master Machine Learning Algorithms: discover how they work and implement them from scratch**. [S.l.]: Machine Learning Mastery, 2016.

BURGES, C.J.C.; BOTTOU, L.; WEINBERGER, K.Q. **Gradient Methods in Machine Learning**. Cambridge, MA: MIT Press, 2010. ISBN 9780262018029.

BURKE, E. K.; GUSTAFSON, S.; KENDALL, G. Diversity in genetic programming: an analysis of measures and correlation with fitness. **IEEE Transactions on Evolutionary Computation**, v. 8, n. 1, p. 47–62, Feb 2004. ISSN 1089-778X.

CAI, Han; CHEN, Tianyao; ZHANG, Weinan; YU, Yong; WANG, Jun. Efficient architecture search by network transformation. In: **Thirty-Second AAAI conference on artificial intelligence**. [S.l.: s.n.], 2018.

ÇAM, Zehra Gülru; ÇIMEN, Sibel; YILDIRIM, Tülay. Learning parameter optimization of multi-layer perceptron using artificial bee colony, genetic algorithm and particle swarm optimization. In: IEEE. **2015 IEEE 13th International Symposium on Applied Machine Intelligence and Informatics (SAMI)**. [S.l.], 2015. p. 329–332.

CAMPOLUCCI, P; CAPPERELLI, F; GUARNIERI, S; PIAZZA, F; UNCINI, A. Neural networks with adaptive spline activation function. In: IEEE. **Proceedings of 8th Mediterranean Electrotechnical Conference on Industrial Applications in Power Systems, Computer Science and Telecommunications (MELECON 96)**. [S.l.], 1996. v. 3, p. 1442–1445.

CANZIANI, Alfredo; PASZKE, Adam; CULURCIELLO, Eugenio. An analysis of deep neural network models for practical applications. **arXiv preprint arXiv:1605.07678**, 2016.

CARUANA, Rich; NICULESCU-MIZIL, Alexandru; CREW, Geoff; KSIKES, Alex. Ensemble selection from libraries of models. In: **Proceedings of the twenty-first international conference on Machine learning**. [S.l.: s.n.], 2004. p. 18.

CATMULL, Edwin; ROM, Raphael. A class of local interpolating splines. In: **Computer aided geometric design**. [S.l.]: Elsevier, 1974. p. 317–326.

CHAMBERS, Michele; DINSMORE, Thomas W. **Advanced analytics methodologies: Driving business value with analytics**. [S.l.]: Pearson Education, 2014.

CHANKONG, Vira; HAIMES, Yacov Y. **Multiobjective decision making: theory and methodology**. [S.l.]: Courier Dover Publications, 2008.

CHAPELLE, Olivier; VAPNIK, Vladimir; BOUSQUET, Olivier; MUKHERJEE, Sayan. Choosing multiple parameters for support vector machines. **Machine learning**, Springer, v. 46, n. 1-3, p. 131–159, 2002.

CHEN, Boyu; LI, Peixia; LI, Chuming; LI, Baopu; BAI, Lei; LIN, Chen; SUN, Ming; YAN, Junjie; OUYANG, Wanli. Glit: Neural architecture search for global and local image transformer. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. [S.l.: s.n.], 2021. p. 12–21.

CHEN, Boyuan; WU, Harvey; MO, Warren; CHATTOPADHYAY, Ishanu; LIPSON, Hod. Autostacker: A compositional evolutionary learning system. In: **Proceedings of the Genetic and Evolutionary Computation Conference**. [S.l.: s.n.], 2018. p. 402–409.

CHEN, Chyi-Tsong; CHANG, Wei-Der. A feedforward neural network with function shape autotuning. **Neural networks**, Elsevier, v. 9, n. 4, p. 627–641, 1996.

CHEN, Guangyong; CHEN, Pengfei; SHI, Yujun; HSIEH, Chang-Yu; LIAO, Benben; ZHANG, Shengyu. Rethinking the usage of batch normalization and dropout in the training of deep neural networks. **arXiv preprint arXiv:1905.05928**, 2019.

CHEN, Minghao; PENG, Houwen; FU, Jianlong; LING, Haibin. Autoformer: Searching transformers for visual recognition. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. [S.l.: s.n.], 2021. p. 12270–12280.

CHEN, Peng-Wei; WANG, Jung-Ying; LEE, Hahn-Ming. Model selection of svms using ga approach. In: IEEE. **2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)**. [S.l.], 2004. v. 3, p. 2035–2040.

CHENG, Weiwei; KASNECI, Gjergji; GRAEPEL, Thore; STERN, David; HERBRICH, Ralf. Automated feature generation from structured knowledge. In: **Proceedings of the 20th ACM international conference on Information and knowledge management**. [S.l.: s.n.], 2011. p. 1395–1404.

COCHRAN, William G. **Sampling Techniques**. [S.l.]: John Wiley, 1977.

COELLO, Carlos A Coello; LAMONT, Gary B; VELDHUIZEN, David A Van et al. **Evolutionary algorithms for solving multi-objective problems**. [S.l.]: Springer, 2007. v. 5.

COHEN, Gilles; HILARIO, Mélanie; GEISSBUHLER, Antoine. Model selection for support vector classifiers via genetic algorithms. an application to medical decision support. In: SPRINGER. **International Symposium on Biological and Medical Data Analysis**. [S.l.], 2004. p. 200–211.

CSÁJI, Balázs Csanád et al. Approximation with artificial neural networks. **Faculty of Sciences, Etvs Lornd University, Hungary**, Citeseer, v. 24, n. 48, p. 7, 2001.

CYBENKO, George. Approximation by superpositions of a sigmoidal function. **Mathematics of control, signals and systems**, Springer, v. 2, n. 4, p. 303–314, 1989.

DARWIN, Charles. **On the Origin of Species by Means of Natural Selection**. London: Murray, 1859. Or the Preservation of Favored Races in the Struggle for Life.

DAS, Indraneel; DENNIS, John E. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. **SIAM journal on optimization**, SIAM, v. 8, n. 3, p. 631–657, 1998.

DAVIS, L. **Handbook of Genetic Algorithms.** [S.l.]: International Thomson Computer Press, 1996.

DEB, Kalyanmoy. **Multi-objective optimization using evolutionary algorithms**. [S.l.]: John Wiley & Sons, 2001. v. 16.

DEB, Kalyanmoy; AGRAWAL, Ram B. Simulated binary crossover for continuous search space. **Complex Systems**, p. 115–148, 1995.

DEB, Kalyanmoy; GOYAL, Mayank. A combined genetic adaptive search (geneas) for engineering design. **Computer Science and Informatics**, v. 26, p. 30–45, 1996.

DEB, Kalyanmoy; JAIN, Himanshu. Handling many-objective problems using an improved nsga-ii procedure. In: IEEE. **2012 IEEE Congress on Evolutionary Computation**. [S.l.], 2012. p. 1–8.

DEB, Kalyanmoy; JAIN, Himanshu. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. **IEEE transactions on evolutionary computation**, IEEE, v. 18, n. 4, p. 577–601, 2013.

DEB, Kalyanmoy; PRATAP, Amrit; AGARWAL, Sameer; MEYARIVAN, TAMT. A fast and elitist multiobjective genetic algorithm: Nsga-ii. **IEEE transactions on evolutionary computation**, IEEE, v. 6, n. 2, p. 182–197, 2002.

DEB, Kalyanmoy; SAXENA, D et al. Searching for pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In: **Proceedings of the world congress on computational intelligence (WCCI-2006)**. [S.l.: s.n.], 2006. p. 3352–3360.

DENG, Jia; DONG, Wei; SOCHER, Richard; LI, Li-Jia; LI, Kai; FEI-FEI, Li. **ImageNet**. 2009. http://www.image-net.org/.

DINH, Hung Nho; BAAN, Mirko Van der. A grid-search approach for 4d pressure-saturation discrimination. **Geophysics**, Society of Exploration Geophysicists, v. 84, n. 4, p. IM47–IM62, 2019.

DODGE, Jesse; PREWITT, Taylor; COMBES, Remi Tachet Des; ODMARK, Erika; SCHWARTZ, Roy; STRUBELL, Emma; LUCCIONI, Alexandra Sasha; SMITH, Noah A.; DECARIO, Nicole; BUCHANAN, Will. **Measuring the Carbon Intensity of AI in Cloud Instances**. 2022.

DOLAN, Elizabeth D; MORÉ, Jorge J. Benchmarking optimization software with performance profiles. **Mathematical programming**, Springer, v. 91, n. 2, p. 201–213, 2002.

DOMINGOS, Pedro. A few useful things to know about machine learning. **Communications of the ACM**, ACM New York, NY, USA, v. 55, n. 10, p. 78–87, 2012.

EFRON, Bradley. Estimating the error rate of a prediction rule: improvement on cross-validation. **Journal of the American statistical association**, Taylor & Francis, v. 78, n. 382, p. 316–331, 1983.

EGGENSPERGER, Katharina; FEURER, Matthias; HUTTER, Frank; BERGSTRA, James; SNOEK, Jasper; HOOS, Holger; LEYTON-BROWN, Kevin. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In: **NIPS workshop on Bayesian Optimization in Theory and Practice**. [S.l.: s.n.], 2013. v. 10, p. 3.

ELSKEN, Thomas; METZEN, Jan Hendrik; HUTTER, Frank. Efficient multi-objective neural architecture search via lamarckian evolution. **arXiv preprint arXiv:1804.09081**, 2018.

ERKAN, Ugur; TOKTAS, Abdurrahim; USTUN, Deniz. Hyperparameter optimization of deep cnn classifier for plant species identification using artificial bee colony algorithm. **Journal of Ambient Intelligence and Humanized Computing**, Springer, p. 1–12, 2022.

ESCALANTE, Hugo Jair; MONTES, Manuel; VILLASEÑOR, Luis. Particle swarm model selection for authorship verification. In: SPRINGER. **Iberoamerican Congress on Pattern Recognition**. [S.l.], 2009. p. 563–570.

ESHELMAN, Larry J; SCHAFFER, J David. Real-coded genetic algorithms and interval-schemata. In: **Foundations of genetic algorithms**. [S.l.]: Elsevier, 1993. v. 2, p. 187–202.

FAWCETT, Tom. An introduction to roc analysis. **Pattern recognition letters**, Elsevier, v. 27, n. 8, p. 861–874, 2006.

FELTRIN, Fernando. **Brain Tumor MRI Images**. 2023. https://www.kaggle.com/datasets/fernando2rad/brain-tumor-mri-images-44c. Acessado em: 03 de Abril de 2023.

FENG, Ju; SHEN, Wen Zhong. Solving the wind farm layout optimization problem using random search algorithm. **Renewable Energy**, Elsevier, v. 78, p. 182–192, 2015.

FERGUSON, James. Multivariable curve interpolation. **Journal of the ACM (JACM)**, ACM New York, NY, USA, v. 11, n. 2, p. 221–228, 1964.

FEURER, Matthias; KLEIN, Aaron; EGGENSPERGER, Katharina; SPRINGENBERG, Jost; BLUM, Manuel; HUTTER, Frank. Efficient and robust automated machine learning. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2015. p. 2962–2970.

FEURER, Matthias; KLEIN, Aaron; EGGENSPERGER, Katharina; SPRINGENBERG, Jost Tobias; BLUM, Manuel; HUTTER, Frank. Auto-sklearn: efficient and robust automated machine learning. In: **Automated Machine Learning**. [S.l.]: Springer, 2019. p. 113–134.

FISCHER, Hans. **A history of the central limit theorem: from classical to modern probability theory**. [S.l.]: Springer, 2011.

FISHER, Ronald A. Xv.—the correlation between relatives on the supposition of mendelian inheritance. **Earth and Environmental Science Transactions of the Royal Society of Edinburgh**, Royal Society of Edinburgh Scotland Foundation, v. 52, n. 2, p. 399–433, 1919.

FONSECA, Carlos M; FLEMING, Peter J. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. ii. application example. **IEEE Transactions on systems, man, and cybernetics-Part A: Systems and humans**, IEEE, v. 28, n. 1, p. 38–47, 1998.

FONSECA, Tales Lima. **Algoritmo Genético com Regressão: Busca Direcionada Através de Aprendizado de Máquina.** 2017. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, 2017.

FONSECA, Tales Lima; GORODETSKAYA, Yulia; TAVARES, Gisele Goulart; RIBEIRO, Celso Bandeira de Melo; FONSECA, Leonardo Goliatt da. A gradient boosting model optimized by a genetic algorithm for short-term riverflow forecas. **Revista Mundi Engenharia, Tecnologia e Gestão (ISSN: 2525-4782)**, v. 4, n. 3, 2019.

FRANCESCOMARINO, Chiara Di; DUMAS, Marlon; FEDERICI, Marco; GHIDINI, Chiara; MAGGI, Fabrizio Maria; RIZZI, Williams; SIMONETTO, Luca. Genetic algorithms for hyperparameter optimization in predictive business process monitoring. **Information Systems**, Elsevier, v. 74, p. 67–83, 2018.

FREITAS, João; LAVADO, Nuno; BERNARDINO, Jorge. Benchmarking auto-weka on a commodity machine. In: **DATA**. [S.l.: s.n.], 2018.

FRIDRICH, Martin. Hyperparameter optimization of artificial neural network in customer churn prediction using genetic algorithm. **Trends Economics and Management**, v. 11, n. 28, p. 9–21, 2017.

FRIEDMAN, Jerome; HASTIE, Trevor; TIBSHIRANI, Robert. **The elements of statistical learning**. [S.l.]: Springer series in statistics New York, 2001. v. 1.

GALE, David; KUHN, Harold William; TUCKER, Albert William. **Four Equivalent Linear-convex Problems**. [S.l.]: Princeton University, 1948.

GANDOMI, Amir Hossein; YANG, Xin-She; TALATAHARI, Siamak; ALAVI, Amir Hossein. **Metaheuristic Applications in Structures and Infrastructures**. [S.l.]: Elsevier Science, 2013.

GARSON, G. David. R-squared: What it is and how to use it. **The American Statistician**, Taylor Francis, v. 45, n. 3, p. 192–193, 1991.

GARZA-FABRE, Mario; PULIDO, Gregorio Toscano; COELLO, Carlos A Coello. Ranking methods for many-objective optimization. In: SPRINGER. **Mexican international conference on artificial intelligence**. [S.l.], 2009. p. 633–645.

GAUCI, Jason; STANLEY, Kenneth. Generating large-scale neural networks through discovering geometric regularities. In: **Proceedings of the 9th annual conference on Genetic and evolutionary computation**. [S.l.: s.n.], 2007. p. 997–1004.

GEMAN, Stuart; BIENENSTOCK, E; DOURSAT, R. Neural networks and the bias/variance dilemma. **Neural computation**, MIT Press, v. 4, n. 1, p. 1–58, 1992.

GLOROT, Xavier; BENGIO, Yoshua. Understanding the difficulty of training deep feedforward neural networks. **International conference on artificial intelligence and statistics**, p. 249–256, 2010.

GLOROT, Xavier; BORDES, Antoine; BENGIO, Yoshua. Deep sparse rectifier neural networks. In: **Proceedings of the fourteenth international conference on artificial intelligence and statistics**. [S.l.: s.n.], 2011. p. 315–323.

GODFREY, Luke Benjamin. **Parameterizing and Aggregating Activation Functions in Deep Neural Networks**. 2018. Tese (Doutorado) — University of Arkansas, 2018.

GODFREY, Luke B; GASHLER, Michael S. A continuum among logarithmic, linear, and exponential functions, and its potential to improve generalization in neural networks. In: IEEE. **2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)**. [S.l.], 2015. v. 1, p. 481–486.

GOLDBARG, Elizabeth; GOLDBARG, Marco; LUNA, Henrique. **Otimização Combinatória e Metaheurísticas: Algoritmos e Apliacações**. [S.l.]: Elsevier Brasil, 2017.

GOLDBERG, D.E. **Genetic Algorithms in Search, Optimization and Machine Learning.** [S.l.]: Addison-Wesley Publishing Co., 1989. Reading, Mass.

GOLDBERG, David E; DEB, Kalyanmoy. A comparative analysis of selection schemes used in genetic algorithms. In: **Foundations of genetic algorithms**. [S.l.]: Elsevier, 1991. v. 1, p. 69–93.

GOLDBERG, David E; DEB, Kalyanmoy; CLARK, James H. Genetic algorithms, noise, and the sizing of populations. **Urbana**, v. 51, p. 61801, 1991.

GOLDBERG, David E; DEB, Kalyanmoy; CLARK, James H. Accounting for noise in the sizing of populations. **Whitley**, v. 2419, p. 127–140, 2014.

GOLDBERG, David E et al. **Real-coded genetic algorithms, virtual alphabets and blocking**. [S.l.]: Citeseer, 1990.

GOLDBERG, David E; RICHARDSON, Jon et al. Genetic algorithms with sharing for multimodal function optimization. In: HILLSDALE, NJ: LAWRENCE ERLBAUM. **Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms**. [S.l.], 1987. p. 41–49.

GOLDBERG, Yoav. **Neural Network Methods for Natural Language Processing**. [S.l.]: Morgan & Claypool, 2017. v. 10.

GOLOVIN, Daniel; SOLNIK, Benjamin; MOITRA, Subhodeep; KOCHANSKI, Greg; KARRO, John; SCULLEY, D. Google vizier: A service for black-box optimization. In: **Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining**. [S.l.: s.n.], 2017. p. 1487–1495.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. [S.l.]: MIT Press, 2016. http://www.deeplearningbook.org.

GOOGLE. **Google Trends - AutoML**. 2022. https://trends.google.com.br/trends/explore?q=automl, Último acesso em 30 de Outubro de 2022.

GUARNIERI, Stefano; PIAZZA, Francesco; UNCINI, Aurelio. Multilayer feedforward networks with adaptive spline activation function. **IEEE Transactions on Neural Networks**, IEEE, v. 10, n. 3, p. 672–683, 1999.

GUDIVADA, Venkat; APON, Amy; DING, Junhua. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. **International Journal on Advances in Software**, v. 10, n. 1, p. 1–20, 2017.

GUERREIRO, Andreia P; FONSECA, Carlos M; PAQUETE, Luís. The hypervolume indicator: Problems and algorithms. **arXiv preprint arXiv:2005.00515**, 2020.

GUO, XC; YANG, JH; WU, CG; WANG, CY; LIANG, YC. A novel ls-svms hyper-parameter selection based on particle swarm optimization. **Neurocomputing**, Elsevier, v. 71, n. 16-18, p. 3211–3215, 2008.

H2O.AI. **H2O**. [S.l.], 2016. H2O version 3.10.0.8. Disponível em: https://github.com/h2oai/h2o-3.

HAACKE, Ewart M; BROWN, Robert W; THOMPSON, Michael R; VENKATESAN, Ramesh. Mri: physics, image reconstruction, and analysis. Wiley, 1999.

HAHNLOSER, Richard HR; SARPESHKAR, Rahul; MAHOWALD, Misha A; DOUGLAS, Rodney J; SEUNG, H Sebastian. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. **Nature**, Nature Publishing Group, v. 405, n. 6789, p. 947–951, 2000.

HAHNLOSER, Richard HR; SEUNG, H Sebastian. Permitted and forbidden sets in symmetric threshold-linear networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2001. p. 217–223.

HALL, Mark; FRANK, Eibe; HOLMES, Geoffrey; PFAHRINGER, Bernhard; REUTEMANN, Peter; WITTEN, Ian H. The weka data mining software: an update. **ACM SIGKDD explorations newsletter**, ACM New York, NY, USA, v. 11, n. 1, p. 10–18, 2009.

HAND, David J; TILL, Robert J. A simple generalisation of the area under the roc curve for multiple class classification problems. **Machine learning**, Springer, v. 45, p. 171–186, 2001.

HANLEY, James A; MCNEIL, Barbara J. The meaning and use of the area under a receiver operating characteristic (roc) curve. **Radiology**, v. 143, n. 1, p. 29–36, 1982.

HARP, Steven Alex; SAMAD, Tariq; GUHA, Aloke. Towards the genetic synthesis of neural network. In: **Proceedings of the Third International Conference on Genetic Algorithms**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989. p. 360–369. ISBN 1558600063.

HARTKE, Bernd. Global optimization. **Wiley Interdisciplinary Reviews: Computational Molecular Science**, Wiley Online Library, v. 1, n. 6, p. 879–887, 2011.

HAYKIN, Simon. **Neural Networks: A Comprehensive Foundation**. [S.l.]: Prentice Hall, 1994.

HAYKIN, Simon. **Neural Networks - A Comprehensive Foundation, Second Edition**. 2. ed. [S.l.]: Prentice Hall, 1998. ISBN 9780132733502,0132733501.

HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2015. p. 1026–1034.

HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep residual learning for image recognition. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2016. p. 770–778.

HE, Xin; ZHAO, Kaiyong; CHU, Xiaowen. Automl: A survey of the state-of-the-art. **Knowledge-Based Systems**, Elsevier BV, v. 212, p. 106622, Jan 2021. ISSN 0950-7051. Disponível em: http://dx.doi.org/10.1016/j.knosys.2020.106622.

HINTON, Geoffrey E; SALAKHUTDINOV, Ruslan R. Reducing the dimensionality of data with neural networks. **Science**, American Association for the Advancement of Science, v. 313, n. 5786, p. 504–507, 2006.

HOGG, Robert V; TANIS, Elliot A; ZIMMERMAN, Dale L. **Probability and statistical inference**. [S.l.]: Pearson/Prentice Hall Upper Saddle River, NJ, USA:, 2010.

HOLLAND, John Henry. **Adaptation in Natural and Artificial Systems**. [S.l.]: The University of Michigan Press, 1975.

HOLLAND, John Henry. **Hidden Order: How Adaptation Builds Complexity**. Basic Books, 1995. (Helix books). ISBN 9780201442304. Disponível em: https://books.google.com.br/books?id=ukc3sOTMfgUC.

HOWARD, Andrew G.; ZHU, Menglong; CHEN, Bo; KALENICHENKO, Dmitry; WANG, Weijun; WEYAND, Tobias; ANDREETTO, Marco; ADAM, Hartwig. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. p. 1386–1394.

HUANG, Guang-Bin; ZHU, Qin-Yu; SIEW, Chee-Kheong. Extreme learning machine: a new learning scheme of feedforward neural networks. **IEEE Transactions on Neural Networks**, IEEE, v. 17, n. 4, p. 975–986, 2006.

HUTTER, Frank; KOTTHOFF, Lars; VANSCHOREN, Joaquin (Ed.). **Automated Machine Learning: Methods, Systems, Challenges**. [S.l.]: Springer, 2018. In press, available at http://automl.org/book.

HUTTER, Frank; KOTTHOFF, Lars; VANSCHOREN, Joaquin. **Automated Machine Learning**. [S.l.]: Springer, 2019.

IANNONI, Ana Paula; MORABITO, Reinaldo. Modelo hipercubo integrado a um algoritmo genético para análise de sistemas médicos emergenciais em rodovias. **Gestão & Produção**, SciELO Brasil, v. 13, n. 1, p. 93–104, 2006.

IOFFE, Sergey; SZEGEDY, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: **International Conference on Machine Learning (ICML)**. [S.l.: s.n.], 2015.

JAGTAP, Ameya D; KAWAGUCHI, Kenji; KARNIADAKIS, George Em. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. **Journal of Computational Physics**, Elsevier, v. 404, p. 109136, 2020.

JAGTAP, Ameya D; SHIN, Yeonjong; KAWAGUCHI, Kenji; KARNIADAKIS, George Em. Deep kronecker neural networks: A general framework for neural networks with adaptive activation functions. **Neurocomputing**, Elsevier, v. 468, p. 165–180, 2022.

JAMES, Gareth; WITTEN, Daniela; HASTIE, Trevor; TIBSHIRANI, Robert. **An introduction to statistical learning**. [S.l.]: Springer, 2013. v. 112.

JIANG, Zhengyao; XU, Dixing; LIANG, Jinjun. **A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem**. 2017.

KANTER, James Max; VEERAMACHANENI, Kalyan. Deep feature synthesis: Towards automating data science endeavors. In: IEEE. **2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)**. [S.l.], 2015. p. 1–10.

KARLIK, Bekir; OLGAC, A Vehbi. Performance analysis of various activation functions in generalized mlp architectures of neural networks. **International Journal of Artificial Intelligence and Expert Systems**, v. 1, n. 4, p. 111–122, 2011.

KELLEHER, John D; NAMEE, Brian Mac; D'ARCY, Aoife. A practitioner's guide to cross-validation. **Journal of Chemical Information and Modeling**, ACS Publications, v. 55, n. 5, p. 1292–1304, 2015.

KIM, Dohyung; KOO, Jahwan; KIM, Ung-Mo. A survey on automated machine learning: Problems, methods and frameworks. In: SPRINGER. **International Conference on Human-Computer Interaction**. [S.l.], 2022. p. 57–70.

KIM, Youngeun; LI, Yuhang; PARK, Hyoungseob; VENKATESHA, Yeshwanth; PANDA, Priyadarshini. Neural architecture search for spiking neural networks. **arXiv preprint arXiv:2201.10355**, 2022.

KINGMA, Diederik P; BA, Jimmy. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KLEIN, Aaron; FALKNER, Stefan; BARTELS, Simon; HENNIG, Philipp; HUTTER, Frank. Fast bayesian optimization of machine learning hyperparameters on large datasets. **arXiv preprint arXiv:1605.07079**, 2016.

KOEHLER, Ann; SPRENT, Peter. A note on the use of the root mean squared error (rmse) in index-agreement studies. **Journal of applied ecology**, Wiley Online Library, v. 33, n. 3, p. 509–512, 1996.

KOHAVI, Ron et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: MONTREAL, CANADA. **Ijcai**. [S.l.], 1995. v. 14, n. 2, p. 1137–1145.

KOHAVI, Ron; WOLPERT, David H et al. Bias plus variance decomposition for zero-one loss functions. In: **ICML**. [S.l.: s.n.], 1996. v. 96, p. 275–83.

KOZA, John R. **Genetic programming: on the programming of computers by means of natural selection**. 1. ed. [S.l.]: MIT Press, 1992. (Complex adaptive systems). ISBN 0262111705,9780262111706.

KUHN, Max; JOHNSON, Kjell. **Applied predictive modeling**. [S.l.]: Springer, 2013. v. 26.

KUNC, Vladimír; KLÉMA, Jiří. On transformative adaptive activation functions in neural networks for gene expression inference. **bioRxiv**, Cold Spring Harbor Laboratory, p. 587287, 2019.

LARCHER, Celio H. N.; BARBOSA, Helio J. C. Auto-cve: A coevolutionary approach to evolve ensembles in automated machine learning. In: **Proceedings of the Genetic and Evolutionary Computation Conference**. New York, NY, USA: Association for Computing Machinery, 2019. (GECCO'19), p. 392–400. ISBN 9781450361118. Disponível em: https://doi.org/10.1145/3321707.3321844.

LARCHER, Celio H. N.; BARBOSA, Helio J. C. Evaluating models with dynamic sampling holdout. In: **Applications of Evolutionary Computation**. Cham: Springer International Publishing, 2021. (Evostar'21), p. 729–744. ISBN 9783030726997. Disponível em: https://doi.org/10.1007/978-3-030-72699-7_46.

LAVALLE, Steven M; BRANICKY, Michael S; LINDEMANN, Stephen R. On the relationship between classical grid search and probabilistic roadmaps. **The International Journal of Robotics Research**, SAGE Publications, v. 23, n. 7-8, p. 673–692, 2004.

LE, Trang T; FU, Weixuan; MOORE, Jason H. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. **Bioinformatics**, Oxford University Press, v. 36, n. 1, p. 250–256, 2020.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **Nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

LECUN, Yann; CORTES, Corinna. MNIST handwritten digit database. 2010. Disponível em: http://yann.lecun.com/exdb/mnist/.

LEMONGE, Afonso Celso de Castro. **Aplicação de Algoritmos Genéticos em Otimização Estrutural**. 1999. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, COPPE, 1999.

LI, Da; CHEN, Xinbo; BECCHI, Michela; ZONG, Ziliang. Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus. In: IEEE. **2016 IEEE international conferences on big data and cloud computing (BDCloud), social computing and networking (SocialCom), sustainable computing and communications (SustainCom)(BDCloud-SocialCom-SustainCom)**. [S.l.], 2016. p. 477–484.

LI, Xiang; CHEN, Shuo; HU, Xiaolin; YANG, Jian. Understanding the disharmony between dropout and batch normalization by variance shift. In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. [S.l.: s.n.], 2019. p. 2682–2690.

LIMA, Leandro Silva. **Uma Abordagem Multiobjetivo Usando Programação Genética Cartesiana Para Projeto de Circuitos Digitais Aproximados.** 2019. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, 2019.

LIN, Min; CHEN, Qiang; YAN, Shuicheng. Network in network. **arXiv preprint arXiv:1312.4400**, 2013.

LIU, Ming; YUAN, Fei; HU, Huosheng; LI, Bing. Semantic segmentation of ground-level images for autonomous vehicles: A review. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 19, n. 12, p. 3812–3825, 2017.

LIU, Yiqi. Adaptive just-in-time and relevant vector machine based soft-sensors with adaptive differential evolution algorithms for parameter optimization. **Chemical Engineering Science**, Elsevier, v. 172, p. 571–584, 2017.

LONG, Qiang; WU, Xue; WU, Changzhi. Non-dominated sorting methods for multi-objective optimization: Review and numerical comparison. **Journal of Industrial & Management Optimization**, American Institute of Mathematical Sciences, v. 17, n. 2, p. 1001, 2021.

LORENZO, Pablo Ribalta; NALEPA, Jakub; RAMOS, Luciano Sanchez; PASTOR, José Ranilla. Hyper-parameter selection in deep neural networks using parallel particle swarm optimization. In: **Proceedings of the Genetic and Evolutionary Computation Conference Companion**. [S.l.: s.n.], 2017. p. 1864–1871.

LOSHCHILOV, Ilya; HUTTER, Frank. Fixing weight decay regularization in adam. 2018.

LOTTICK, Kadan; SUSAI, Silvia; FRIEDLER, Sorelle A.; WILSON, Jonathan P. **Energy Usage Reports: Environmental awareness as part of algorithmic accountability**. 2019.

LU, Zhichao; WHALEN, Ian; BODDETI, Vishnu; DHEBAR, Yashesh; DEB, Kalyanmoy; GOODMAN, Erik; BANZHAF, Wolfgang. Nsga-net: neural architecture search using multi-objective genetic algorithm. In: **Proceedings of the genetic and evolutionary computation conference**. [S.l.: s.n.], 2019. p. 419–427.

LUXBURG, Ulrike Von; SCHÖLKOPF, Bernhard. Statistical learning theory: Models, concepts, and results. In: **Handbook of the History of Logic**. [S.l.]: Elsevier, 2011. v. 10, p. 651–706.

MAAS, Andrew L; HANNUN, Awni Y; NG, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In: **Proc. icml**. [S.l.: s.n.], 2013. v. 30, n. 1, p. 3.

MACLAURIN, Dougal; DUVENAUD, David; ADAMS, Ryan. Gradient-based hyperparameter optimization through reversible learning. In: **International Conference on Machine Learning**. [S.l.: s.n.], 2015. p. 2113–2122.

MALHOTRA, Akshay; DHALL, Abhinav; KUMAR, Vimal. A survey of deep learning techniques for brain tumor prediction from mri images. **Journal of King Saud University-Computer and Information Sciences**, Elsevier, 2020.

MANTOVANI, Rafael G; ROSSI, André LD; VANSCHOREN, Joaquin; BISCHL, Bernd; CARVALHO, André CPLF De. Effectiveness of random search in svm hyper-parameter tuning. In: IEEE. **2015 International Joint Conference on Neural Networks (IJCNN)**. [S.l.], 2015. p. 1–8.

MCCULLOCH, Warren S; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943.

MELLO, Rodrigo Fernandes de; PONTI, Moacir Antonelli. Machine learning: a practical approach on the statistical learning theory. **Cham: Springer International Publishing**, 2018.

MICHALEWICZ, Zbigniew. **Genetic Algorithms + Data Structures = Evolution Programs**. [S.l.]: Springer Berlin Heidelberg, 1992. (Artificial Intelligence). ISBN 978-3-662-02832-2,978-3-662-02830-8.

MINSKY, Marvin; PAPERT, Seymour. **Perceptrons: An Introduction to Computational Geometry**. Cambridge, MA, USA: MIT Press, 1969.

MITCHELL, Melanie. **An Introduction to Genetic Algorithms**. Third printing. [S.l.]: The MIT Press, 1998. (Complex Adaptive Systems). ISBN 9780262631853,0262631857,0262133164,9780262133166,9780585030944.

MUSTAFFA, Zuriani; YUSOF, Yuhanis. A hybridization of enhanced artificial bee colony-least squares support vector machines for price forecasting. **Journal of Computer Science**, v. 8, n. 10, p. 1680–1690, 2012.

NADKARNI, João; NEVES, Rui Ferreira. Combining neuroevolution and principal component analysis to trade in the financial markets. **Expert Systems with Applications**, Elsevier, v. 103, p. 184–195, 2018.

NASERI, Gita; KOFFAS, Mattheos AG. Application of combinatorial optimization strategies in synthetic biology. **Nature communications**, Nature Publishing Group, v. 11, n. 1, p. 1–14, 2020.

Nações Unidas no Brasil. **Objetivos de Desenvolvimento Sustentável**. 2022. https://brasil.un.org/pt-br/sdgs, Último acesso em 08 de Novembro de 2022.

NIETO, PJ García; GARCÍA-GONZALO, E; FERNÁNDEZ, JR Alonso; MUÑIZ, C Díaz. A hybrid wavelet kernel svm-based method using artificial bee colony algorithm for predicting the cyanotoxin content from experimental cyanobacteria concentrations in the trasona reservoir (northern spain). **Journal of Computational and Applied Mathematics**, Elsevier, v. 309, p. 587–602, 2017.

NIKBAKHT, Saeid; ANITESCU, Cosmin; RABCZUK, Timon. Optimizing the neural network hyperparameters utilizing genetic algorithm. **Journal of Zhejiang University-Science A**, Springer, v. 22, n. 6, p. 407–426, 2021.

OLIINYK, Andrii O; SKRUPSKY, S Yu; SUBBOTIN, Sergei A. Using parallel random search to train fuzzy neural networks. **Automatic Control and Computer Sciences**, Springer, v. 48, n. 6, p. 313–323, 2014.

OLSON, Randal S.; BARTLEY, Nathan; URBANOWICZ, Ryan J.; MOORE, Jason H. Evaluation of a tree-based pipeline optimization tool for automating data science. In: **Proceedings of the Genetic and Evolutionary Computation Conference 2016**. New York, NY, USA: ACM, 2016a. (GECCO '16), p. 485–492. ISBN 978-1-4503-4206-3. Disponível em: http://doi.acm.org/10.1145/2908812.2908918.

OLSON, Randal S.; CAVA, William La; ORZECHOWSKI, Patryk; URBANOWICZ, Ryan J.; MOORE, Jason H. PMLB: a large benchmark suite for machine learning evaluation and comparison. **BioData Mining**, v. 10, n. 1, p. 36, Dec 2017. ISSN 1756-0381.

OLSON, Randal S.; URBANOWICZ, Ryan J.; ANDREWS, Peter C.; LAVENDER, Nicole A.; KIDD, La Creis; MOORE, Jason H. Applications of evolutionary computation: 19th european conference, evoapplications 2016, porto, portugal, march 30 – april 1, 2016, proceedings, part i. In: _____. Springer International Publishing, 2016b. cap. Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, p. 123–137. ISBN 978-3-319-31204-0. Disponível em: http://dx.doi.org/10.1007/978-3-319-31204-0_9.

OpenAI. **GPT-3: Generative Pre-trained Transformer 3**. 2020. https://beta.openai.com/docs/models/gpt-3.

PAN, Sinno Jialin; YANG, Qiang. A survey on transfer learning. **IEEE Transactions on knowledge and data engineering**, IEEE, v. 22, n. 10, p. 1345–1359, 2009.

PANDA, Sashmita; PANDA, G. Fast and improved backpropagation learning of multi-layer artificial neural network using adaptive activation function. **Expert Systems**, 2020.

PEDREGOSA, Fabian. Hyperparameter optimization with approximate gradient. **arXiv preprint arXiv:1602.02355**, 2016.

PENG, Lu; LIU, Shan; LIU, Rui; WANG, Lin. Effective long short-term memory with differential evolution algorithm for electricity price prediction. **Energy**, Elsevier, v. 162, p. 1301–1314, 2018.

PHAM, Hieu; GUAN, Melody Y; ZOPH, Barret; LE, Quoc V; DEAN, Jeff. Efficient neural architecture search via parameter sharing. **arXiv preprint arXiv:1802.03268**, 2018.

PHAM, Van Thinh; NGUYEN, Hong-Tham T; NGUYEN, Duyen Thi Cam; LE, Hanh TN; NGUYEN, Thuong Thi; LE, Nhan Thi Hong; LIM, Kwon Teak; NGUYEN, Trinh Duy; TRAN, Thuan Van; BACH, Long Giang et al. Process optimization by a response surface methodology for adsorption of congo red dye onto exfoliated graphite-decorated mnfe2o4 nanocomposite: The pivotal role of surface chemistry. **Processes**, Multidisciplinary Digital Publishing Institute, v. 7, n. 5, p. 305, 2019.

PIAZZA, F; UNCINI, A; ZENOBI, M. Artificial neural networks with adaptive polynomial activation function. Citeseer, 1992.

PÓLYA, Georg. Über den zentralen grenzwertsatz der wahrscheinlichkeitsrechnung und das momentenproblem. **Mathematische Zeitschrift**, Springer, v. 8, n. 3, p. 171–181, 1920.

POWERS, David MW. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. **Journal of Machine Learning Technologies**, Citeseer, v. 2, n. 1, p. 37–63, 2011.

QIAN, Sheng; LIU, Hua; LIU, Cheng; WU, Si; WONG, Hau San. Adaptive activation functions in convolutional neural networks. **Neurocomputing**, Elsevier, v. 272, p. 204–212, 2018.

RAMACHANDRAN, Prajit; ZOPH, Barret; LE, Quoc V. Searching for activation functions. **arXiv preprint arXiv:1710.05941**, 2017.

RANJBAR, Mostafa; MARBURG, Steffen. Vibroacoustic optimization of mechanical structures: a controlled random search approach. In: TRANS TECH PUBL. **Advanced Materials Research**. [S.l.], 2013. v. 622, p. 158–161.

RAO, Singiresu S; RAO, Singiresu S. **Engineering optimization: theory and practice**. [S.l.]: John Wiley & Sons, 2009.

RÄTSCH, Gunnar; ONODA, Takashi; MÜLLER, K-R. Soft margins for adaboost. **Machine learning**, Springer, v. 42, n. 3, p. 287–320, 2001.

REAL, Esteban; MOORE, Sherry; SELLE, Andrew; SAXENA, Saurabh; SUEMATSU, Yutaka Leon; TAN, Jie; LE, Quoc V; KURAKIN, Alexey. Large-scale evolution of image classifiers. In: JMLR. ORG. **Proceedings of the 34th International Conference on Machine Learning-Volume 70**. [S.l.], 2017. p. 2902–2911.

RISI, Sebastian; STANLEY, Kenneth O. An enhanced hypercube-based encoding for evolving the placement, density, and connectivity of neurons. **Artificial life**, MIT Press, v. 18, n. 4, p. 331–363, 2012.

RODI, William. Grid-search event location with non-gaussian error models. **Physics of the Earth and Planetary Interiors**, Elsevier, v. 158, n. 1, p. 55–66, 2006.

RONALD, Edmund; SCHOENAUER, Marc. Genetic lander: An experiment in accurate neuro-genetic control. In: SPRINGER. **International Conference on Parallel Problem Solving from Nature**. [S.l.], 1994. p. 452–461.

ROSENBLATT, Frank. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

RUDIN, Walter. **Real and complex analysis**. [S.l.]: McGraw-hill, 1987.

RUDOLPH, Günter. **Evolutionary search under partially ordered fitness sets**. [S.l.]: Citeseer, 2001.

RUMELHART, David E; HINTON, Geoffrey E; WILLIAMS, Ronald J. Learning representations by back-propagating errors. **Nature**, Nature Research, v. 323, n. 6088, p. 533, 1986.

RUSSELL, Stuart; NORVIG, Peter. Artificial intelligence: a modern approach. 2002.

RUSSELL, Stuart; NORVIG, Peter. **Artificial Intelligence: A Modern Approach**. 4th. ed. [S.l.]: Pearson, 2021.

RUSSO, Igor Lucas De Souza. **Otimização Multiobjetivo e Programação Genética para Descoberta de Conhecimento em Engenharia.** 2017. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, 2017.

SALIMANS, Tim; KINGMA, Durk P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. **Advances in neural information processing systems**, v. 29, 2016.

SALVADOR, Manuel Martin; BUDKA, Marcin; GABRYS, Bogdan. Adapting multicomponent predictive systems using hybrid adaptation strategies with auto-weka in process industry. In: **Workshop on Automatic Machine Learning**. [S.l.: s.n.], 2016. p. 48–57.

SAMANTA, B. Gear fault detection using artificial neural networks and support vector machines with genetic algorithms. **Mechanical systems and signal processing**, Elsevier, v. 18, n. 3, p. 625–644, 2004.

SANTOS, Carlos Eduardo da Silva; SAMPAIO, Renato Coral; COELHO, Leandro dos Santos; BESTARD, Guillermo Alvarez; LLANOS, Carlos Humberto. Multi-objective adaptive differential evolution for svm/svr hyperparameters selection. **Pattern Recognition**, Elsevier, v. 110, p. 107649, 2021.

SAPORETTI, Camila Martins; DUARTE, Grasiele Regina; FONSECA, Tales Lima; FONSECA, Leonardo Goliatt da; PEREIRA, Egberto. Extreme learning machine combined with a differential evolution algorithm for lithology identification. **Revista de Informática Teórica e Aplicada**, v. 25, n. 4, p. 43–56, 2018.

SCHAFFER, J David; WHITLEY, Darrell; ESHELMAN, Larry J. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In: IEEE. **[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks**. [S.l.], 1992. p. 1–37.

SCHMIDT, Victor; COURTY, Benoit; SABONI, Amine. **CodeCarbon**. 2020. https://mlco2.github.io/codecarbon/index.html#.

SCHOTT, Jason Ramon. **Fault tolerant design using single and multicriteria genetic algorithm optimization**. 1995. Tese (Doutorado) — Massachusetts Institute of Technology, 1995.

SCHULER, Friedmann. Decision trees for business intelligence and data mining. **Business Intelligence and Data Mining**, Wiley Online Library, v. 2, n. 1, p. 1–9, 2002.

SCHWARTZ, Roy; DODGE, Jesse; SMITH, Noah A; ETZIONI, Oren. Green ai. **arXiv preprint arXiv:1907.10597**, 2019.

SCHWARTZ, Roy; DODGE, Jesse; SMITH, Noah A; ETZIONI, Oren. Green ai. **Communications of the ACM**, ACM New York, NY, USA, v. 63, n. 12, p. 54–63, 2020.

SINDHYA, Karthik; MIETTINEN, Kaisa; DEB, Kalyanmoy. A hybrid framework for evolutionary multi-objective optimization. **IEEE Transactions on Evolutionary Computation**, IEEE, v. 17, n. 4, p. 495–511, 2012.

SINGH, Pratibha; CHAUDHURY, Santanu; PANIGRAHI, Bijaya Ketan. Hybrid mpso-cnn: Multi-level particle swarm optimized hyperparameters of convolutional neural network. **Swarm and Evolutionary Computation**, Elsevier, v. 63, p. 100863, 2021.

SNOEK, Jasper; LAROCHELLE, Hugo; ADAMS, Ryan P. Practical bayesian optimization of machine learning algorithms. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2012. p. 2951–2959.

SOKOLOVA, Maria; LAPALME, Guy. A systematic analysis of performance measures for classification tasks. **Information Processing Management**, Elsevier, v. 45, n. 4, p. 427–437, 2009.

SOLAZZI, Mirko; UNCINI, Aurelio. Adaptive multidimensional spline neural network for digital equalization. In: IEEE. **Neural Networks for Signal Processing X. Proceedings of the 2000 IEEE Signal Processing Society Workshop (Cat. No. 00TH8501)**. [S.l.], 2000. v. 2, p. 729–735.

SOLAZZI, Mirko; UNCINI, Aurelio. Artificial neural networks with adaptive multidimensional spline activation functions. In: IEEE. **Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium**. [S.l.], 2000. v. 3, p. 471–476.

SOUZA, Bruno Feres De; CARVALHO, Andre CPLF De; CALVO, Rodrigo; ISHII, Renato Porfirio. Multiclass svm model selection using particle swarm optimization. In: IEEE. **2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06)**. [S.l.], 2006. p. 31–31.

SRINIVAS, Nidamarthi; DEB, Kalyanmoy. Muiltiobjective optimization using nondominated sorting in genetic algorithms. **Evolutionary computation**, MIT Press, v. 2, n. 3, p. 221–248, 1994.

SRIVASTAVA, Nitish; HINTON, Geoffrey; KRIZHEVSKY, Alex; SUTSKEVER, Ilya; SALAKHUTDINOV, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, n. 1, p. 1929–1958, 2014.

STANLEY, Kenneth O; BRYANT, Bobby D; MIIKKULAINEN, Risto. Real-time neuroevolution in the nero video game. **IEEE transactions on evolutionary computation**, IEEE, v. 9, n. 6, p. 653–668, 2005.

STANLEY, Kenneth O; MIIKKULAINEN, Risto. Evolving neural networks through augmenting topologies. **Evolutionary computation**, MIT Press, v. 10, n. 2, p. 99–127, 2002.

STECK, Harald; BALTRUNAS, Linas; ELAHI, Ehtsham; LIANG, Dawen; RAIMOND, Yves; BASILICO, Justin. Deep learning for recommender systems: A netflix case study. **AI Magazine**, v. 42, n. 3, p. 7–18, 2021.

STEIN, Elias M; SHAKARCHI, Rami. **Fourier analysis: an introduction**. [S.l.]: Princeton University Press, 2011. v. 1.

STONE, M. Cross-validatory choice and assessment of statistical predictions. **Journal of the Royal Statistical Society. Series B (Methodological)**, JSTOR, v. 36, n. 2, p. 111–147, 1974.

STRUBELL, Emma; GANESH, Ananya; MCCALLUM, Andrew. Energy and policy considerations for deep learning in nlp. **arXiv preprint arXiv:1906.02243**, 2019.

SUCH, Felipe Petroski; MADHAVAN, Vashisht; CONTI, Edoardo; LEHMAN, Joel; STANLEY, Kenneth O; CLUNE, Jeff. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. **arXiv preprint arXiv:1712.06567**, 2017.

SUN, Yuting; DING, Shifei; ZHANG, Zichen; JIA, Weikuan. An improved grid search algorithm to optimize svr for prediction. **Soft Computing**, Springer, v. 25, n. 7, p. 5633–5644, 2021.

SWERSKY, Kevin; SNOEK, Jasper; ADAMS, Ryan P. Multi-task bayesian optimization. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2013. p. 2004–2012.

TEZEL, Gülay; ÖZBAY, Yüksel. A new neural network with adaptive activation function for classification of ecg arrhythmias. In: SPRINGER. **International Conference on Knowledge-Based and Intelligent Information and Engineering Systems**. [S.l.], 2007. p. 1–8.

THORNTON, Chris; HUTTER, Frank; HOOS, Holger H; LEYTON-BROWN, Kevin. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In: **Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.: s.n.], 2013. p. 847–855.

TOAL, David JJ; BRESSLOFF, NW; KEANE, AJ; HOLDEN, CME. The development of a hybridized particle swarm for kriging hyperparameter tuning. **Engineering optimization**, Taylor & Francis, v. 43, n. 6, p. 675–699, 2011.

TUKEY, John W. **Exploratory data analysis**. [S.l.]: Reading, MA, 1977. v. 2.

VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N; KAISER, Łukasz; POLOSUKHIN, Illia. Attention is all you need. **arXiv preprint arXiv:1706.03762**, 2017.

VATIN, NI; IVANOV, A Yu; RUTMAN, Yu L; CHERNOGORSKIY, SA; SHVETSOV, KV. Earthquake engineering optimization of structures by economic criterion. **Magazine of Civil Engineering**, v. 76, n. 8, 2017.

VECCI, Lorenzo; CAMPOLUCCI, Paolo; PIAZZA, Francesco; UNCINI, Aurelio. Approximation capabilities of adaptive spline neural networks. In: IEEE. **Proceedings of International Conference on Neural Networks (ICNN'97)**. [S.l.], 1997. v. 1, p. 260–265.

VECCI, Lorenzo; PIAZZA, Francesco; UNCINI, Aurelio. Learning and approximation capabilities of adaptive spline activation function neural networks. **Neural Networks**, Elsevier, v. 11, n. 2, p. 259–270, 1998.

VERBANCSICS, Phillip; HARGUESS, Josh. **Generative NeuroEvolution for Deep Learning**. 2013.

VILLALOBOS-ARIAS, Leonardo; QUESADA-LÓPEZ, Christian. Comparative study of random search hyper-parameter tuning for software effort estimation. In: **Proceedings of the 17th International Conference on Predictive Models and Data Analytics in Software Engineering**. [S.l.: s.n.], 2021. p. 21–29.

VOLTERRA, Vito. **Sopra le funzioni che dipendono da altre funzioni**. [S.l.]: Tip. della R. Accademia dei Lincei, 1887.

VOLTERRA, Vito. Theory of functionals and of integral and integro-differential equations. Dover, 1959.

WANG, Xiaofei; HAN, Yiwen; LEUNG, Victor CM; NIYATO, Dusit; YAN, Xueqiang; CHEN, Xu. Convergence of edge computing and deep learning: A comprehensive survey. **IEEE Communications Surveys & Tutorials**, IEEE, v. 22, n. 2, p. 869–904, 2020.

WIERSTRA, Daan; GOMEZ, Faustino J; SCHMIDHUBER, Jürgen. Modeling systems with internal state using evolino. In: **Proceedings of the 7th annual conference on Genetic and evolutionary computation**. [S.l.: s.n.], 2005. p. 1795–1802.

WIERZBICKI, Andrzej P. The use of reference objectives in multiobjective optimization. In: **Multiple criteria decision making theory and application**. [S.l.]: Springer, 1980. p. 468–486.

WILLSEY, Matthew S; NASON-TOMASZEWSKI, Samuel R; ENSEL, Scott R; TEMMAR, Hisham; MENDER, Matthew J; COSTELLO, Joseph T; PATIL, Parag G; CHESTEK, Cynthia A. Real-time brain-machine interface in non-human primates achieves high-velocity prosthetic finger movements using a shallow feedforward neural

network decoder. **Nature Communications**, Nature Publishing Group UK London, v. 13, n. 1, p. 6899, 2022.

WITTEN, Ian H.; FRANK, Eibe; HALL, Mark A.; PAL, Christopher J. **Data Mining: Practical Machine Learning Tools and Techniques**. [S.l.]: Morgan Kaufmann, 2016. (Morgan Kaufmann Series in Data Management Systems).

XU, Jingjing; ZHAO, Liang; LIN, Junyang; GAO, Rundong; SUN, Xu; YANG, Hongxia. Knas: green neural architecture search. In: PMLR. **International Conference on Machine Learning**. [S.l.], 2021. p. 11613–11625.

YI, Xinyang; WANG, Yuyi; SUN, Xiaoming; SUN, Le. Uout: An uncertain dropout method. **CoRR**, abs/2007.13756, 2020. Disponível em: https://arxiv.org/abs/2007.13756.

YU, Lean; WANG, Shouyang; LAI, Kin Keung. Data preparation in neural network data analysis. **Foreign-Exchange-Rate Forecasting With Artificial Neural Networks**, Springer, p. 39–62, 2007.

YULIYONO, Andary; GIRSANG, Ganda. Artificial bee colony-optimized lstm for bitcoin price prediction. **Advances in Science, Technology and Engineering Systems Journal**, v. 4, 01 2019.

ZAHEDINASAB, Roxana; MOHSENI, Hadis. Neuroevolutionary based convolutional neural network with adaptive activation functions. **Neurocomputing**, Elsevier, v. 381, p. 306–313, 2020.

ZHANG, Chiyuan; BENGIO, Samy; HARDT, Moritz; RECHT, Benjamin; VINYALS, Oriol. Combining batch and weight normalization for training deep neural networks. In: **International Conference on Learning Representations**. [S.l.: s.n.], 2018.

ZHANG, Jingjing; NIU, Qun; LI, Kang; IRWIN, George W. Model selection in svms using differential evolution. **IFAC Proceedings Volumes**, Elsevier, v. 44, n. 1, p. 14717–14722, 2011.

ZHANG, Qingfu; LI, Hui. Moea/d: A multiobjective evolutionary algorithm based on decomposition. **IEEE Transactions on evolutionary computation**, IEEE, v. 11, n. 6, p. 712–731, 2007.

ZHANG, Yongli; YANG, Yuhong. Cross-validation for selecting a model selection procedure. **Journal of Econometrics**, Elsevier, v. 187, n. 1, p. 95–112, 2015.

ZHENG, Alice; CASARI, Amanda. **Feature engineering for machine learning: principles and techniques for data scientists**. [S.l.]: "O'Reilly Media, Inc.", 2018.

ZHOU, Aimin; ZHANG, Qingfu; JIN, Yaochu. Approximating the set of pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm. **IEEE transactions on evolutionary computation**, IEEE, v. 13, n. 5, p. 1167–1189, 2009.

ZITZLER, Eckart. **Evolutionary algorithms for multiobjective optimization: Methods and applications**. [S.l.]: Shaker Ithaca, 1999. v. 63.

ZITZLER, Eckart; DEB, Kalyanmoy; THIELE, Lothar. Comparison of multiobjective evolutionary algorithms: Empirical results. **Evolutionary computation**, MIT Press, v. 8, n. 2, p. 173–195, 2000.

ZOPH, Barret; LE, Quoc V. Neural architecture search with reinforcement learning. **arXiv preprint arXiv:1611.01578**, 2016.

ZöRNIG, Peter. **Nonlinear programming**. [S.l.]: De Gruyter, 2014. (de Gruyter Textbook). ISBN 3110315270,978-3-11-031527-1,978-3-11-031528-8.

APÊNDICE A – COMPLEMENTO DO EXPERIMENTO I

Todas as tabelas e figuras apresentadas aqui utilizam um mapa de cores com brilho linearmente decrescente (ou crescente). Isso implica que as informações serão mantidas caso sejam impressas em preto e branco ou visualizadas por pessoas com deficiência de daltonismo.

A.1 Análise das Métricas de Desempenho e Visualização de Distribuição

As Tabelas 18, 19, 20, 21, 22, 23, 24, 25, 26 e 27 apresentam os resultados sumarizados do primeiro experimento numérico. Para cada função de ativação e tipo adaptativo da ANN, exibimos os valores mínimo, médio, mediano e máximo das métricas F1 e *Log Loss*, juntamente com o desvio padrão para as 30 execuções independentes. Os valores marcados com a cor **"Azul Escuro"** representam os melhores resultados encontrados, enquanto os valores destacados em **"Azul Claro"** representam os piores resultados encontrados.

As Figuras 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87 e 88 representam os *Violin plots* das métricas F1 e *Log Loss* para os problemas do Experimento 1, considerando diferentes tipos adaptativos das funções de ativação em 30 execuções independentes. A escolha dos *Violin plots* se deve à sua capacidade de combinar as informações contidas em um *boxplot* com a representação de densidade.

Tabela 18 – Resultados do Experimento 1 para a Base de Dados 1 (analcatdata_lawsuit).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	0.048	0.763	0.231	0.844	0.956	0.328	0.581	0.106	0.573	0.796
	Camada	0.048	0.762	0.232	0.844	0.956	0.315	0.579	0.109	0.575	0.797
	Neurônio	**0.596**	**0.854**	**0.072**	**0.879**	**0.966**	**0.301**	**0.517**	**0.085**	**0.512**	**0.677**
Cúbico	Normal	0.011	0.598	0.423	**0.888**	0.931	0.472	0.643	0.103	0.607	0.856
	Camada	0.008	0.800	0.252	**0.888**	**0.962**	0.450	0.595	0.096	0.579	0.846
	Neurônio	**0.194**	**0.867**	**0.128**	**0.888**	0.956	**0.407**	**0.533**	**0.072**	**0.511**	**0.706**
Linear	Normal	0.048	0.772	0.208	0.836	**0.982**	0.358	0.580	0.106	0.581	0.814
	Camada	0.011	0.794	0.199	0.854	**0.982**	0.322	0.554	0.117	0.552	0.813
	Neurônio	**0.760**	**0.879**	**0.049**	**0.879**	**0.982**	**0.309**	**0.499**	**0.085**	**0.502**	**0.658**
Mish	Normal	0.011	0.730	0.268	0.850	0.927	0.427	0.611	0.098	0.619	0.835
	Camada	0.011	0.729	0.271	0.854	0.927	0.408	0.608	0.101	0.619	0.832
	Neurônio	**0.469**	**0.845**	**0.109**	**0.888**	**0.936**	**0.382**	**0.540**	**0.076**	**0.546**	**0.709**
PELU	Normal	0.083	0.758	0.236	0.844	0.956	0.375	0.594	0.101	0.602	0.823
	Camada	0.048	0.756	0.247	0.844	0.956	0.338	0.589	0.107	0.599	0.821
	Neurônio	**0.541**	**0.852**	**0.085**	**0.888**	**0.966**	**0.334**	**0.526**	**0.081**	**0.525**	**0.696**
PReLU	Normal	0.011	0.727	0.284	0.879	0.888	0.403	0.595	0.109	0.586	0.835
	Camada	0.011	0.729	0.283	0.879	0.888	0.392	0.591	0.109	0.584	0.821
	Neurônio	**0.469**	**0.847**	**0.113**	**0.888**	**0.946**	**0.367**	**0.528**	**0.084**	**0.522**	**0.703**
Quadrática	Normal	0.011	0.632	0.380	**0.888**	0.913	0.476	0.643	0.105	0.616	0.857
	Camada	0.011	0.806	0.229	**0.888**	0.910	0.466	0.602	0.095	0.599	0.849

Continua na próxima página

Tabela 18 – Resultados do Experimento 1 para a Base de Dados 1 (analcatdata_lawsuit).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**0.048**	**0.862**	**0.154**	**0.888**	**0.927**	**0.420**	**0.540**	**0.074**	**0.524**	**0.704**
SELU	Normal	0.151	0.789	0.168	**0.850**	**0.982**	**0.343**	**0.569**	0.109	0.570	**0.804**
	Camada	**0.083**	**0.788**	**0.179**	**0.850**	**0.982**	0.318	0.564	**0.116**	**0.572**	**0.804**
	Neurônio	**0.697**	**0.866**	**0.067**	**0.879**	**0.982**	**0.311**	**0.506**	**0.090**	**0.508**	**0.685**
Sigmóide	Normal	**0.011**	0.810	0.243	**0.888**	**0.888**	**0.336**	**0.508**	0.119	**0.472**	0.878
	Camada	**0.011**	**0.805**	**0.255**	**0.888**	**0.888**	0.320	0.501	**0.129**	0.455	**0.889**
	Neurônio	**0.118**	**0.861**	**0.141**	**0.888**	**0.888**	**0.319**	**0.457**	**0.093**	**0.429**	**0.741**
SoftExponential	Normal	**0.151**	**0.764**	**0.227**	**0.849**	0.980	**0.332**	**0.577**	0.107	**0.558**	**0.795**
	Camada	**0.151**	0.791	0.208	0.863	**0.982**	0.302	0.545	**0.118**	0.552	0.789
	Neurônio	**0.760**	**0.876**	**0.041**	**0.888**	**0.966**	**0.288**	**0.493**	**0.085**	**0.497**	**0.654**
SoftPlus	Normal	0.350	0.863	0.107	**0.888**	**0.927**	**0.285**	**0.428**	0.106	**0.399**	0.759
	Camada	**0.011**	**0.846**	**0.176**	**0.888**	**0.927**	**0.269**	0.423	**0.122**	0.388	**0.814**
	Neurônio	**0.827**	**0.885**	**0.012**	**0.888**	**0.888**	0.272	**0.394**	**0.087**	**0.370**	**0.660**
SoftSign	Normal	**0.011**	0.742	0.236	**0.823**	**0.927**	**0.436**	**0.613**	0.094	**0.612**	**0.848**
	Camada	**0.011**	**0.739**	**0.244**	**0.823**	**0.927**	0.427	0.610	**0.097**	**0.612**	0.842
	Neurônio	**0.377**	**0.850**	**0.110**	**0.888**	**0.982**	**0.396**	**0.541**	**0.073**	**0.540**	**0.719**
Swish	Normal	**0.011**	**0.725**	**0.281**	**0.860**	**0.927**	**0.441**	**0.617**	0.097	**0.616**	**0.844**
	Camada	**0.011**	**0.725**	**0.281**	**0.860**	**0.927**	0.432	0.615	**0.099**	0.614	0.839
	Neurônio	**0.377**	**0.842**	**0.129**	**0.888**	**0.936**	**0.400**	**0.545**	**0.075**	**0.545**	**0.716**
Tangente Hiperbólica	Normal	**0.011**	0.761	0.222	**0.823**	**0.980**	**0.407**	**0.595**	0.099	0.600	**0.832**

Continua na próxima página

Tabela 18 – Resultados do Experimento 1 para a Base de Dados 1 (analcatdata_lawsuit).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**0.011**	**0.759**	**0.227**	0.831	**0.980**	0.383	0.591	**0.104**	**0.602**	0.829
	Neurônio	**0.552**	**0.854**	**0.080**	**0.884**	**0.982**	**0.366**	**0.527**	**0.079**	**0.532**	**0.707**

Fonte: Elaborada pelo autor (2023).

Tabela 19 – Resultados do Experimento 1 para a Base de Dados 2 (australian).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	**0.734**	0.831	0.038	**0.840**	0.899	0.386	0.491	**0.044**	0.492	0.606
	Camada	**0.734**	**0.832**	0.038	**0.840**	0.899	**0.373**	**0.485**	0.046	**0.486**	0.605
	Neurônio	0.725	0.829	**0.037**	0.835	**0.906**	0.379	0.486	0.045	**0.486**	**0.604**
Cúbico	Normal	0.268	0.402	**0.076**	0.416	0.587	0.679	0.692	**0.006**	0.691	**0.705**
	Camada	0.299	**0.558**	0.150	**0.539**	**0.891**	**0.567**	0.690	0.058	0.678	0.861
	Neurônio	**0.416**	0.547	0.101	0.528	0.796	0.623	**0.682**	0.045	**0.668**	0.798
Linear	Normal	0.740	0.836	0.037	0.832	**0.899**	0.387	0.488	**0.043**	0.486	0.604
	Camada	0.740	**0.839**	**0.036**	**0.839**	0.897	**0.360**	**0.474**	0.048	**0.469**	0.602
	Neurônio	**0.745**	0.834	**0.036**	**0.839**	0.897	0.377	0.481	0.045	0.476	**0.601**
Mish	Normal	**0.565**	0.784	**0.070**	0.804	**0.877**	0.493	0.585	**0.036**	0.584	0.665
	Camada	0.559	**0.786**	0.071	**0.805**	**0.877**	**0.469**	**0.579**	0.041	**0.578**	0.670
	Neurônio	0.559	0.780	0.073	0.800	**0.877**	0.485	0.580	0.037	**0.578**	**0.664**
PELU	Normal	0.680	0.821	0.046	0.826	0.891	0.423	0.523	**0.040**	0.522	0.612
	Camada	**0.705**	**0.822**	**0.043**	0.826	0.891	**0.381**	**0.507**	0.049	**0.505**	**0.609**
	Neurônio	0.692	**0.822**	0.045	**0.827**	**0.906**	0.411	0.516	0.042	0.513	**0.609**
PReLU	Normal	0.477	0.759	0.077	0.767	**0.861**	0.518	0.605	**0.034**	0.607	0.675
	Camada	0.477	**0.760**	0.077	**0.774**	0.854	**0.498**	**0.599**	0.038	**0.601**	0.678
	Neurônio	**0.490**	0.756	**0.075**	0.771	0.860	0.511	0.600	0.035	0.602	**0.674**
Quadrática	Normal	0.268	0.446	0.088	0.443	0.651	0.673	0.692	0.011	0.690	0.716
	Camada	0.268	**0.490**	0.125	**0.465**	**0.797**	**0.659**	0.686	0.011	0.684	0.704

Continua na próxima página

Tabela 19 – Resultados do Experimento 1 para a Base de Dados 2 (australian).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**0.400**	0.452	**0.061**	0.439	0.716	0.664	**0.683**	**0.010**	**0.681**	**0.703**
SELU	Normal	0.762	0.838	0.034	**0.833**	**0.899**	0.369	0.466	**0.041**	0.466	0.561
	Camada	0.762	**0.840**	0.034	**0.833**	**0.899**	**0.343**	**0.451**	0.047	**0.450**	**0.557**
	Neurônio	**0.768**	0.839	**0.032**	0.832	0.898	0.360	0.459	0.043	0.459	0.558
Sigmóide	Normal	**0.396**	0.459	**0.065**	**0.441**	0.651	0.621	0.661	**0.017**	0.661	0.695
	Camada	**0.396**	**0.462**	0.067	0.435	**0.661**	**0.616**	**0.660**	0.019	**0.660**	0.695
	Neurônio	**0.396**	0.459	0.066	0.431	0.651	0.619	**0.660**	**0.017**	**0.660**	**0.694**
SoftExponential	Normal	**0.755**	**0.838**	0.035	0.833	0.891	0.385	0.489	**0.043**	0.489	0.602
	Camada	**0.755**	**0.838**	**0.034**	0.833	**0.898**	**0.371**	**0.483**	0.046	0.482	**0.601**
	Neurônio	0.753	0.835	0.035	**0.839**	**0.898**	0.378	0.484	0.045	**0.477**	**0.601**
SoftPlus	Normal	**0.486**	0.707	**0.073**	0.717	0.852	0.520	0.604	**0.032**	0.605	**0.662**
	Camada	0.457	**0.716**	0.081	**0.730**	**0.868**	**0.492**	**0.598**	0.039	**0.599**	0.671
	Neurônio	0.472	0.711	0.077	0.726	0.867	0.513	0.601	0.034	0.602	0.663
SoftSign	Normal	**0.629**	**0.779**	**0.068**	**0.796**	0.891	0.517	0.593	**0.032**	0.591	0.653
	Camada	**0.629**	**0.779**	**0.068**	**0.796**	0.891	**0.491**	**0.585**	0.037	**0.583**	0.657
	Neurônio	0.624	0.776	0.071	0.788	**0.899**	0.508	0.588	0.033	0.587	**0.650**
Swish	Normal	0.508	**0.763**	**0.081**	**0.775**	0.877	0.522	0.606	**0.033**	0.605	0.684
	Camada	0.503	**0.763**	0.082	**0.775**	0.877	**0.502**	**0.601**	0.037	0.601	0.687
	Neurônio	**0.513**	0.760	0.083	0.769	**0.891**	0.514	**0.601**	0.034	**0.600**	**0.682**
Tangente Hiperbólica	Normal	0.741	0.816	0.045	0.817	0.891	0.440	0.532	**0.037**	0.534	0.602

Continua na próxima página

Tabela 19 – Resultados do Experimento 1 para a Base de Dados 2 (australian).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**0.748**	**0.819**	**0.045**	**0.820**	0.899	**0.408**	**0.519**	**0.044**	**0.520**	**0.600**
	Neurônio	**0.733**	0.818	**0.044**	0.818	**0.906**	0.429	0.525	0.039	0.529	**0.600**

Fonte: Elaborada pelo autor (2023).

Tabela 20 – Resultados do Experimento 1 para a Base de Dados 3 (breast_w).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	0.858	0.942	**0.022**	**0.946**	**0.971**	0.218	0.318	0.059	0.311	0.447
	Camada	0.858	**0.943**	0.023	**0.946**	**0.971**	**0.200**	**0.296**	0.062	**0.286**	**0.428**
	Neurônio	**0.865**	0.938	0.023	0.943	**0.971**	0.210	0.299	**0.057**	0.294	0.429
Cúbico	Normal	0.140	0.497	**0.093**	0.521	0.537	0.646	0.668	**0.015**	0.665	0.712
	Camada	**0.537**	**0.775**	0.105	**0.783**	**0.942**	**0.343**	**0.551**	0.092	**0.576**	0.660
	Neurônio	**0.537**	0.701	0.104	0.698	0.871	0.396	0.572	0.061	0.591	**0.644**
Linear	Normal	**0.875**	**0.943**	0.020	**0.950**	**0.972**	0.213	0.316	0.057	0.306	0.434
	Camada	0.874	0.941	**0.019**	0.943	0.971	**0.172**	**0.272**	0.061	**0.257**	**0.391**
	Neurônio	0.873	0.937	0.020	0.942	0.971	0.192	0.291	**0.055**	0.282	0.407
Mish	Normal	0.828	0.919	0.035	0.924	**0.964**	0.344	0.461	**0.058**	0.464	0.557
	Camada	**0.845**	**0.922**	**0.032**	**0.930**	**0.964**	**0.289**	**0.432**	0.073	**0.433**	0.548
	Neurônio	0.791	0.894	0.045	0.900	**0.964**	0.312	0.440	0.059	0.442	**0.542**
PELU	Normal	**0.858**	0.937	0.024	0.943	**0.972**	0.252	0.361	**0.059**	0.354	0.477
	Camada	**0.858**	**0.940**	**0.023**	**0.946**	0.965	**0.188**	**0.302**	0.073	**0.287**	**0.437**
	Neurônio	0.857	0.934	0.024	0.935	0.964	0.224	0.334	**0.059**	0.328	0.456
PReLU	Normal	0.742	0.889	0.062	**0.901**	0.964	0.364	0.495	**0.057**	0.513	0.597
	Camada	**0.752**	**0.898**	**0.056**	**0.901**	**0.972**	**0.319**	**0.468**	0.073	**0.481**	0.609
	Neurônio	0.675	0.869	0.070	0.882	0.964	0.335	0.474	0.058	0.494	**0.572**
Quadrática	Normal	0.124	0.513	0.103	0.521	0.650	0.635	0.670	**0.017**	0.673	0.721
	Camada	**0.521**	**0.617**	0.129	**0.537**	**0.919**	**0.493**	**0.626**	0.047	0.645	0.682

Continua na próxima página

Tabela 20 – Resultados do Experimento 1 para a Base de Dados 3 (breast_w).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**0.521**	0.556	**0.069**	**0.521**	0.791	0.578	0.630	0.020	**0.637**	**0.657**
SELU	Normal	0.891	0.944	0.017	**0.950**	**0.965**	**0.188**	**0.282**	0.050	**0.270**	**0.405**
	Camada	**0.899**	**0.946**	**0.016**	**0.950**	**0.965**	**0.155**	**0.241**	**0.051**	**0.225**	**0.366**
	Neurônio	**0.874**	**0.943**	**0.020**	**0.950**	**0.972**	0.173	0.261	**0.048**	0.250	0.381
Sigmóide	Normal	**0.521**	0.535	0.048	**0.521**	0.752	**0.538**	**0.597**	**0.033**	**0.592**	0.676
	Camada	**0.521**	**0.541**	**0.065**	**0.521**	**0.810**	**0.518**	**0.592**	**0.039**	**0.587**	**0.678**
	Neurônio	**0.521**	**0.533**	**0.045**	**0.521**	**0.742**	0.528	0.594	0.034	0.590	**0.671**
SoftExponential	Normal	**0.873**	**0.941**	**0.019**	**0.943**	**0.971**	**0.213**	**0.315**	0.058	**0.307**	**0.421**
	Camada	**0.873**	0.940	**0.020**	**0.942**	**0.971**	**0.193**	**0.290**	**0.060**	**0.280**	**0.396**
	Neurônio	**0.873**	**0.936**	**0.020**	**0.942**	**0.971**	0.203	0.296	**0.056**	0.289	0.402
SoftPlus	Normal	**0.521**	**0.754**	0.086	**0.757**	0.919	**0.384**	**0.489**	**0.051**	**0.491**	0.579
	Camada	**0.521**	**0.780**	**0.091**	**0.787**	**0.935**	**0.312**	**0.461**	**0.072**	**0.455**	**0.599**
	Neurônio	**0.521**	0.762	**0.085**	**0.757**	**0.912**	0.357	0.477	0.056	0.478	**0.576**
SoftSign	Normal	**0.854**	0.915	0.034	0.911	**0.965**	**0.345**	**0.472**	**0.057**	**0.470**	0.565
	Camada	0.850	**0.920**	**0.032**	**0.919**	**0.972**	**0.275**	**0.439**	**0.077**	**0.433**	**0.577**
	Neurônio	**0.775**	**0.894**	**0.049**	**0.887**	0.971	0.317	0.452	0.061	0.452	**0.556**
Swish	Normal	0.772	0.899	0.047	0.901	**0.964**	**0.378**	**0.499**	**0.055**	**0.503**	0.578
	Camada	**0.810**	**0.906**	**0.044**	**0.913**	**0.964**	**0.328**	**0.476**	**0.070**	**0.480**	**0.584**
	Neurônio	**0.720**	**0.870**	**0.059**	**0.880**	**0.957**	0.347	0.479	0.057	0.481	**0.564**
Tangente Hiperbólica	Normal	0.874	0.936	0.023	0.939	**0.971**	**0.255**	**0.374**	**0.058**	**0.365**	**0.471**

Continua na próxima página

Tabela 20 – Resultados do Experimento 1 para a Base de Dados 3 (breast_w).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**0.882**	**0.940**	**0.021**	**0.942**	**0.971**	**0.198**	**0.329**	**0.072**	**0.315**	**0.445**
	Neurônio	**0.865**	**0.930**	**0.025**	**0.934**	**0.964**	0.230	0.351	0.060	0.343	0.451

Fonte: Elaborada pelo autor (2023).

Tabela 21 – Resultados do Experimento 1 para a Base de Dados 4 (buggyCrx).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	**0.616**	**0.731**	0.053	**0.736**	**0.826**	**0.487**	**0.573**	**0.040**	**0.572**	0.651
	Camada	**0.616**	0.732	**0.055**	**0.736**	**0.826**	**0.478**	0.570	**0.043**	0.568	**0.654**
	Neurônio	**0.619**	**0.735**	**0.050**	**0.736**	**0.826**	0.481	**0.569**	0.041	**0.567**	**0.649**
Cúbico	Normal	0.268	**0.388**	**0.071**	**0.416**	**0.535**	**0.681**	**0.691**	**0.006**	**0.689**	**0.705**
	Camada	**0.249**	0.487	**0.127**	**0.508**	**0.762**	**0.641**	0.689	**0.034**	0.682	**0.799**
	Neurônio	**0.400**	**0.511**	0.076	0.504	0.712	0.646	**0.682**	0.030	**0.677**	0.796
Linear	Normal	0.613	**0.739**	**0.055**	**0.754**	**0.827**	**0.490**	**0.571**	**0.040**	**0.566**	0.653
	Camada	**0.603**	**0.739**	0.054	**0.750**	**0.827**	**0.472**	**0.564**	**0.046**	**0.556**	**0.661**
	Neurônio	**0.621**	**0.738**	**0.052**	**0.750**	**0.819**	0.482	0.565	0.042	0.559	**0.652**
Mish	Normal	0.476	**0.670**	0.075	**0.678**	**0.807**	**0.562**	**0.636**	**0.032**	**0.637**	0.701
	Camada	**0.460**	0.671	**0.082**	0.683	**0.815**	**0.551**	0.633	**0.035**	0.635	**0.702**
	Neurônio	**0.498**	**0.681**	**0.074**	**0.694**	**0.807**	0.558	**0.632**	**0.032**	**0.633**	**0.693**
PELU	Normal	0.585	**0.717**	**0.064**	**0.729**	**0.826**	**0.512**	**0.596**	**0.038**	**0.595**	0.673
	Camada	**0.576**	**0.717**	**0.065**	**0.732**	**0.819**	**0.489**	**0.589**	**0.045**	0.591	**0.678**
	Neurônio	**0.598**	**0.721**	**0.064**	**0.732**	**0.826**	0.505	0.591	0.039	**0.590**	**0.670**
PReLU	Normal	0.451	**0.637**	0.076	0.639	**0.789**	**0.579**	**0.649**	**0.029**	**0.647**	0.711
	Camada	**0.437**	0.642	**0.077**	**0.637**	**0.789**	**0.570**	0.647	**0.031**	0.646	**0.712**
	Neurônio	**0.519**	**0.647**	**0.068**	**0.644**	**0.789**	0.577	**0.646**	**0.029**	**0.642**	**0.704**
Quadrática	Normal	**0.262**	**0.418**	0.078	**0.416**	0.647	**0.677**	**0.694**	**0.011**	**0.692**	**0.713**
	Camada	**0.262**	0.431	**0.093**	**0.416**	**0.685**	0.672	0.689	**0.011**	0.689	0.711

Continua na próxima página

Tabela 21 – Resultados do Experimento 1 para a Base de Dados 4 (buggyCrx).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**0.398**	**0.448**	**0.063**	**0.416**	0.644	**0.670**	**0.688**	**0.010**	**0.687**	**0.706**
SELU	Normal	**0.618**	0.739	**0.046**	0.744	**0.812**	0.472	0.556	**0.039**	0.557	0.638
	Camada	0.611	0.739	0.048	**0.748**	**0.812**	**0.455**	**0.550**	0.045	**0.551**	0.648
	Neurônio	0.600	**0.740**	0.051	0.744	**0.812**	0.464	0.551	0.041	0.554	**0.637**
Sigmóide	Normal	**0.400**	0.449	0.056	**0.438**	0.576	**0.646**	**0.676**	**0.015**	**0.677**	**0.703**
	Camada	**0.400**	**0.452**	0.060	**0.438**	**0.608**	0.647	**0.676**	**0.015**	**0.677**	**0.703**
	Neurônio	**0.400**	0.445	**0.054**	0.428	0.563	**0.646**	**0.676**	**0.015**	**0.677**	**0.703**
SoftExponential	Normal	0.611	0.736	**0.053**	0.742	**0.827**	0.485	0.571	**0.042**	0.575	0.654
	Camada	0.603	0.736	0.055	0.743	**0.827**	**0.477**	0.568	0.045	0.570	0.658
	Neurônio	**0.614**	**0.739**	**0.053**	**0.747**	0.819	0.480	**0.567**	0.043	**0.569**	**0.653**
SoftPlus	Normal	0.416	0.608	**0.094**	0.608	**0.806**	0.576	0.645	**0.030**	0.648	0.701
	Camada	0.416	**0.613**	0.098	0.613	**0.806**	**0.567**	**0.643**	0.033	**0.646**	0.702
	Neurônio	**0.431**	**0.613**	0.095	**0.616**	**0.806**	0.574	0.644	**0.030**	0.647	**0.699**
SoftSign	Normal	**0.460**	0.659	0.092	0.671	0.824	0.576	0.641	**0.030**	0.639	0.701
	Camada	0.448	0.659	0.096	0.669	**0.831**	**0.565**	0.638	0.033	0.636	0.702
	Neurônio	0.455	**0.665**	**0.090**	**0.698**	0.823	0.572	**0.637**	0.031	**0.635**	**0.699**
Swish	Normal	0.444	0.649	0.083	0.648	0.782	0.583	0.648	**0.029**	0.650	0.707
	Camada	0.425	0.648	0.086	0.651	**0.790**	**0.574**	0.646	0.032	0.648	0.708
	Neurônio	**0.494**	**0.659**	**0.079**	**0.663**	**0.790**	0.579	**0.644**	**0.029**	**0.646**	**0.700**
Tangente Hiperbólica	Normal	0.529	0.708	**0.070**	0.718	**0.826**	0.524	0.602	**0.037**	0.599	0.682

Continua na próxima página

Tabela 21 – Resultados do Experimento 1 para a Base de Dados 4 (buggyCrx).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**0.520**	**0.707**	**0.072**	**0.718**	**0.812**	**0.507**	**0.597**	**0.042**	0.596	**0.686**
	Neurônio	**0.563**	**0.715**	**0.070**	**0.735**	**0.826**	0.517	0.598	0.038	**0.594**	**0.681**

Fonte: Elaborada pelo autor (2023).

Tabela 22 – Resultados do Experimento 1 para a Base de Dados 5 (dna).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	0.883	0.902	**0.009**	0.903	0.920	0.338	0.366	**0.014**	0.362	0.395
	Camada	**0.893**	**0.914**	**0.009**	**0.913**	**0.937**	**0.251**	**0.274**	**0.014**	**0.272**	**0.302**
	Neurônio	0.883	0.907	0.010	0.906	0.928	0.291	0.317	**0.014**	0.316	0.345
Cúbico	Normal	**0.354**	0.354	**0.000**	0.354	0.354	1.032	1.033	**0.001**	1.032	1.035
	Camada	**0.354**	**0.817**	0.136	**0.869**	**0.892**	**0.314**	**0.465**	0.177	**0.395**	1.001
	Neurônio	**0.354**	0.354	**0.000**	0.354	0.354	0.972	0.987	0.006	0.987	**0.998**
Linear	Normal	0.883	0.906	0.010	0.907	0.928	0.349	0.379	0.013	0.380	0.398
	Camada	**0.914**	**0.927**	**0.006**	**0.926**	**0.945**	**0.193**	**0.221**	**0.011**	**0.221**	**0.239**
	Neurônio	0.890	0.911	0.010	0.913	0.933	0.289	0.317	0.012	0.318	0.338
Mish	Normal	0.802	0.844	0.019	0.844	0.880	0.516	0.553	**0.017**	0.550	0.584
	Camada	**0.865**	**0.896**	**0.012**	**0.897**	**0.916**	**0.287**	**0.319**	0.018	**0.317**	**0.359**
	Neurônio	0.819	0.861	0.017	0.863	0.892	0.449	0.491	0.018	0.487	0.523
PELU	Normal	0.867	0.894	0.011	0.896	0.915	0.393	0.424	0.015	0.420	0.452
	Camada	**0.914**	**0.924**	**0.007**	**0.924**	**0.937**	**0.177**	**0.200**	**0.014**	**0.202**	**0.227**
	Neurônio	0.879	0.900	0.011	0.902	0.919	0.333	0.363	0.015	0.360	0.392
PReLU	Normal	0.762	0.790	0.017	0.786	0.825	0.577	0.617	**0.017**	0.615	0.647
	Camada	**0.861**	**0.891**	**0.013**	**0.893**	**0.912**	**0.335**	**0.371**	0.020	**0.369**	**0.416**
	Neurônio	0.786	0.820	0.016	0.817	0.845	0.516	0.562	0.019	0.561	0.597
Quadrática	Normal	**0.354**	0.354	**0.000**	**0.354**	0.354	1.020	1.023	**0.001**	1.023	1.025
	Camada	**0.354**	**0.437**	0.141	**0.354**	**0.770**	**0.655**	**0.913**	0.121	**0.968**	1.026

Continua na próxima página

Tabela 22 – Resultados do Experimento 1 para a Base de Dados 5 (dna).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**0.354**	0.354	**0.000**	**0.354**	0.354	1.015	1.019	**0.001**	1.019	**1.021**
SELU	Normal	0.889	0.910	0.010	0.909	0.933	0.316	0.346	0.012	0.346	0.363
	Camada	**0.914**	**0.929**	**0.007**	**0.928**	**0.944**	**0.189**	**0.208**	**0.010**	**0.209**	**0.226**
	Neurônio	0.895	0.915	0.010	0.914	0.940	0.270	0.297	0.011	0.296	0.313
Sigmóide	Normal	0.354	0.358	**0.006**	0.354	0.378	0.868	0.890	**0.010**	0.890	0.906
	Camada	**0.442**	**0.605**	0.072	**0.607**	**0.715**	**0.702**	**0.758**	0.035	**0.768**	**0.817**
	Neurônio	0.354	0.363	0.010	0.360	0.392	0.852	0.878	0.012	0.879	0.896
SoftExponential	Normal	0.885	0.903	0.011	0.904	0.922	0.337	0.364	0.014	0.363	0.389
	Camada	**0.902**	**0.916**	**0.008**	**0.917**	**0.937**	**0.243**	**0.268**	**0.012**	**0.266**	**0.292**
	Neurônio	0.894	0.910	0.010	0.911	0.929	0.288	0.314	0.013	0.314	0.340
SoftPlus	Normal	0.774	0.800	0.017	0.795	0.841	0.578	0.619	**0.016**	0.619	0.648
	Camada	**0.870**	**0.900**	**0.011**	**0.902**	**0.918**	**0.258**	**0.293**	0.017	**0.291**	**0.333**
	Neurônio	0.788	0.833	0.017	0.835	0.868	0.513	0.561	0.019	0.561	0.593
SoftSign	Normal	0.816	0.865	0.018	0.870	0.886	0.556	0.594	**0.016**	0.593	0.626
	Camada	**0.876**	**0.903**	**0.012**	**0.902**	**0.936**	**0.288**	**0.318**	**0.016**	**0.317**	**0.350**
	Neurônio	0.816	0.862	0.017	0.866	0.890	0.482	0.524	0.018	0.523	0.559
Swish	Normal	0.768	0.808	0.018	0.807	0.840	0.581	0.619	**0.017**	0.616	0.648
	Camada	**0.851**	**0.884**	**0.013**	**0.885**	**0.907**	**0.324**	**0.364**	0.020	**0.363**	**0.408**
	Neurônio	0.784	0.832	0.018	0.833	0.863	0.515	0.559	0.019	0.557	0.592
Tangente Hiperbólica	Normal	0.864	0.894	0.013	0.896	0.920	0.418	0.452	0.014	0.454	0.479

Continua na próxima página

Tabela 22 – Resultados do Experimento 1 para a Base de Dados 5 (dna).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**0.896**	**0.917**	**0.009**	**0.917**	**0.939**	**0.237**	**0.260**	**0.012**	**0.259**	**0.285**
	Neurônio	0.870	0.898	**0.014**	0.902	0.923	0.352	0.383	**0.015**	0.384	0.410

Fonte: Elaborada pelo autor (2023).

Tabela 23 – Resultados do Experimento 1 para a Base de Dados 6 (hypothyroid).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.193**	**0.210**	**0.008**	**0.212**	**0.223**
	Camada	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.167**	**0.182**	**0.008**	**0.182**	0.203
	Neurônio	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	0.176	0.188	**0.006**	0.188	**0.201**
Cúbico	Normal	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.264**	**0.273**	**0.005**	**0.273**	**0.281**
	Camada	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.184**	**0.188**	**0.001**	**0.188**	**0.190**
	Neurônio	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	0.196	0.200	0.002	0.200	0.202
Linear	Normal	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.216**	**0.229**	**0.009**	**0.228**	**0.251**
	Camada	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.172**	**0.181**	**0.006**	**0.180**	**0.193**
	Neurônio	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	0.181	0.191	**0.006**	0.191	0.206
Mish	Normal	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.213**	**0.226**	**0.007**	**0.227**	**0.246**
	Camada	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.176**	**0.192**	**0.007**	**0.192**	0.207
	Neurônio	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	0.186	0.196	**0.005**	0.196	**0.205**
PELU	Normal	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.211**	**0.225**	0.007	**0.225**	**0.239**
	Camada	**0.929**	**0.930**	**0.001**	**0.929**	**0.935**	**0.162**	**0.182**	**0.010**	**0.183**	**0.205**
	Neurônio	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	0.184	0.194	**0.005**	0.194	**0.205**
PReLU	Normal	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.188**	**0.201**	**0.008**	**0.203**	**0.221**
	Camada	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.178**	**0.190**	0.007	**0.190**	0.204
	Neurônio	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	0.179	**0.190**	**0.006**	0.191	**0.199**
Quadrática	Normal	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.255**	**0.264**	**0.005**	**0.264**	**0.273**
	Camada	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.183**	**0.189**	**0.002**	**0.189**	**0.193**

Continua na próxima página

Tabela 23 – Resultados do Experimento 1 para a Base de Dados 6 (hypothyroid).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
SELU	Neurônio	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	0.196	0.200	**0.002**	0.200	0.205
	Normal	**0.929**	**0.929**	**0.000**	**0.929**	0.929	**0.206**	**0.223**	0.010	**0.220**	**0.250**
	Camada	**0.929**	**0.929**	0.001	**0.929**	**0.932**	**0.168**	**0.188**	0.012	**0.187**	0.222
	Neurônio	**0.929**	**0.929**	**0.000**	**0.929**	0.929	0.181	0.193	**0.007**	0.193	**0.215**
Sigmóide	Normal	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.188**	**0.192**	**0.002**	**0.192**	**0.196**
	Camada	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.188**	**0.192**	**0.002**	**0.192**	**0.196**
	Neurônio	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.188**	**0.192**	**0.002**	**0.192**	**0.196**
SoftExponential	Normal	0.928	**0.930**	0.001	**0.929**	**0.932**	**0.195**	**0.209**	**0.008**	**0.210**	**0.227**
	Camada	**0.929**	0.929	**0.000**	**0.929**	0.929	**0.168**	**0.178**	0.007	**0.178**	0.200
	Neurônio	**0.929**	0.929	**0.000**	**0.929**	0.929	0.175	0.185	**0.006**	0.185	**0.196**
SoftPlus	Normal	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.182**	**0.188**	**0.003**	**0.188**	**0.194**
	Camada	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.181**	**0.187**	**0.003**	**0.188**	**0.193**
	Neurônio	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.182**	**0.187**	**0.003**	**0.188**	**0.193**
SoftSign	Normal	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.243**	**0.254**	**0.006**	**0.254**	**0.270**
	Camada	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.201**	0.209	**0.006**	0.209	0.227
	Neurônio	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	0.202	**0.207**	**0.003**	**0.207**	**0.217**
Swish	Normal	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.223**	**0.235**	0.006	**0.235**	**0.252**
	Camada	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.182**	**0.196**	**0.007**	**0.197**	0.209
	Neurônio	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	0.191	0.199	**0.004**	0.199	**0.207**
Tangente Hiperbólica	Normal	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.230**	**0.241**	0.008	**0.238**	**0.264**

Continua na próxima página

Tabela 23 – Resultados do Experimento 1 para a Base de Dados 6 (hypothyroid).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	**0.186**	**0.200**	**0.009**	**0.200**	0.227
	Neurônio	**0.929**	**0.929**	**0.000**	**0.929**	**0.929**	0.194	0.202	**0.005**	0.201	**0.217**

Fonte: Elaborada pelo autor (2023).

Tabela 24 – Resultados do Experimento 1 para a Base de Dados 7 (monk1).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	**0.333**	0.487	0.071	0.499	0.586	**0.674**	0.708	0.022	0.703	0.774
	Camada	**0.333**	0.488	0.072	0.500	0.594	**0.674**	0.707	**0.021**	0.702	**0.770**
	Neurônio	0.329	**0.496**	**0.069**	**0.510**	**0.607**	**0.674**	**0.705**	**0.021**	**0.700**	0.772
Cúbico	Normal	0.325	0.343	**0.036**	0.333	0.489	0.693	0.697	**0.005**	0.695	0.712
	Camada	0.301	0.391	0.079	0.353	**0.571**	**0.685**	0.696	0.006	0.695	0.707
	Neurônio	**0.333**	**0.406**	0.076	**0.372**	0.560	0.686	**0.694**	0.006	**0.694**	**0.705**
Linear	Normal	**0.333**	0.490	0.072	0.506	**0.644**	0.680	0.708	0.020	0.703	0.766
	Camada	**0.333**	0.490	0.071	0.510	**0.644**	**0.679**	0.706	**0.019**	0.702	**0.760**
	Neurônio	0.329	**0.499**	**0.065**	**0.516**	0.603	**0.679**	**0.705**	**0.019**	**0.699**	0.764
Mish	Normal	**0.333**	0.468	0.084	0.477	**0.609**	0.673	0.702	0.016	0.701	0.740
	Camada	**0.333**	0.469	0.083	0.473	**0.609**	0.673	0.701	0.016	0.700	**0.738**
	Neurônio	0.329	**0.483**	**0.076**	**0.500**	0.607	**0.672**	**0.700**	**0.015**	**0.699**	0.739
PELU	Normal	**0.333**	0.481	0.077	0.482	**0.617**	0.674	0.706	0.019	0.703	0.756
	Camada	**0.333**	0.481	0.076	0.482	**0.617**	0.674	0.705	**0.018**	0.702	**0.751**
	Neurônio	**0.333**	**0.494**	**0.070**	**0.513**	0.598	**0.673**	**0.703**	**0.018**	**0.700**	0.754
PReLU	Normal	0.321	0.470	0.078	**0.489**	**0.597**	0.667	0.701	0.017	0.701	0.747
	Camada	0.321	0.471	0.077	**0.489**	**0.597**	0.667	0.700	0.017	0.700	0.744
	Neurônio	**0.331**	**0.477**	**0.072**	0.481	0.594	**0.665**	**0.699**	**0.016**	**0.699**	**0.739**
Quadrática	Normal	0.326	0.365	**0.054**	0.333	0.501	0.689	0.697	0.006	0.695	0.716
	Camada	**0.333**	0.367	0.060	0.333	0.522	**0.688**	0.696	0.005	0.695	0.707

Continua na próxima página

Tabela 24 – Resultados do Experimento 1 para a Base de Dados 7 (monk1).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	0.329	**0.381**	0.068	**0.343**	**0.581**	**0.688**	**0.695**	**0.004**	**0.694**	**0.702**
SELU	Normal	**0.333**	0.495	0.072	0.515	**0.636**	0.676	0.710	0.021	0.704	0.773
	Camada	**0.333**	0.495	0.074	0.514	**0.636**	**0.675**	0.708	**0.020**	0.703	**0.765**
	Neurônio	0.325	**0.499**	**0.068**	**0.524**	0.621	**0.675**	**0.707**	**0.020**	**0.700**	0.770
Sigmóide	Normal	0.317	0.428	**0.087**	0.420	0.568	0.684	0.696	0.007	0.695	0.719
	Camada	0.317	0.425	**0.087**	0.420	0.568	0.684	0.696	0.007	0.695	0.720
	Neurônio	**0.324**	**0.444**	0.088	**0.458**	**0.581**	**0.683**	**0.695**	**0.006**	**0.694**	**0.710**
SoftExponential	Normal	**0.333**	0.487	0.074	0.494	0.636	0.679	0.708	**0.020**	0.703	0.767
	Camada	**0.333**	0.493	0.073	0.501	**0.644**	**0.678**	0.707	**0.020**	0.702	**0.764**
	Neurônio	**0.333**	**0.500**	**0.066**	**0.517**	0.593	**0.678**	**0.705**	**0.020**	**0.700**	0.766
SoftPlus	Normal	0.343	0.498	0.070	0.518	0.604	**0.673**	0.697	**0.012**	0.695	0.731
	Camada	**0.365**	0.497	**0.069**	0.515	0.603	0.674	**0.696**	**0.012**	0.695	**0.729**
	Neurônio	0.327	**0.501**	**0.069**	**0.522**	**0.614**	**0.673**	**0.696**	**0.012**	**0.694**	0.730
SoftSign	Normal	**0.333**	0.469	0.082	0.487	**0.606**	0.677	0.701	0.013	0.700	0.728
	Camada	**0.333**	0.470	0.083	0.488	**0.606**	0.677	0.701	0.013	0.699	0.727
	Neurônio	**0.333**	**0.486**	**0.074**	**0.496**	0.597	**0.676**	**0.699**	**0.012**	**0.697**	**0.726**
Swish	Normal	**0.329**	0.464	0.090	0.485	**0.617**	0.674	0.701	0.014	0.699	0.733
	Camada	**0.329**	0.463	0.091	0.487	**0.617**	0.674	0.700	0.014	0.699	**0.731**
	Neurônio	**0.329**	**0.483**	**0.079**	**0.505**	0.607	**0.672**	**0.699**	**0.013**	**0.698**	**0.731**
Tangente Hiperbólica	Normal	**0.333**	0.489	0.076	0.502	**0.613**	0.676	0.705	0.017	0.703	0.746

Continua na próxima página

Tabela 24 – Resultados do Experimento 1 para a Base de Dados 7 (monk1).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**0.333**	0.490	**0.077**	0.514	**0.613**	**0.676**	0.704	**0.016**	0.702	**0.743**
	Neurônio	**0.333**	**0.501**	**0.067**	**0.521**	0.605	**0.675**	**0.703**	**0.016**	**0.700**	0.745

Fonte: Elaborada pelo autor (2023).

Tabela 25 – Resultados do Experimento 1 para a Base de Dados 8 (penguins).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	**0.257**	0.620	0.116	**0.627**	**0.842**	**0.754**	**0.898**	**0.081**	**0.905**	1.074
	Camada	**0.257**	**0.616**	**0.120**	**0.627**	**0.842**	**0.734**	0.895	**0.089**	0.900	**1.075**
	Neurônio	**0.257**	**0.629**	**0.107**	**0.645**	**0.805**	0.735	**0.887**	0.085	**0.894**	**1.072**
Cúbico	Normal	**0.059**	**0.221**	**0.068**	**0.262**	**0.369**	**1.064**	**1.090**	**0.018**	**1.084**	**1.132**
	Camada	0.155	0.350	**0.172**	**0.262**	**0.707**	**0.914**	1.054	**0.046**	1.069	1.118
	Neurônio	**0.189**	**0.377**	0.158	**0.310**	**0.707**	0.955	**1.050**	0.031	**1.060**	**1.100**
Linear	Normal	**0.213**	**0.618**	**0.125**	**0.614**	**0.849**	**0.757**	**0.900**	**0.082**	**0.900**	1.086
	Camada	**0.168**	**0.618**	**0.138**	0.617	**0.849**	**0.714**	0.890	**0.098**	0.889	**1.091**
	Neurônio	0.188	**0.631**	0.134	**0.619**	**0.896**	0.725	**0.884**	0.088	**0.880**	**1.084**
Mish	Normal	**0.074**	**0.467**	0.183	**0.473**	0.720	**0.870**	**1.012**	**0.067**	**1.019**	1.170
	Camada	**0.074**	0.469	**0.189**	0.489	**0.750**	0.855	1.009	**0.072**	1.018	**1.172**
	Neurônio	**0.070**	**0.493**	**0.173**	**0.521**	**0.707**	**0.853**	**1.001**	**0.067**	**1.001**	**1.146**
PELU	Normal	0.163	0.568	0.150	**0.580**	**0.811**	**0.801**	**0.950**	**0.076**	**0.952**	1.097
	Camada	**0.098**	**0.559**	**0.161**	0.581	**0.811**	**0.768**	0.943	**0.092**	0.941	**1.106**
	Neurônio	**0.216**	**0.589**	**0.135**	**0.604**	**0.799**	0.782	**0.934**	0.080	**0.932**	**1.092**
PReLU	Normal	**0.128**	**0.442**	**0.174**	**0.394**	**0.707**	**0.901**	**1.026**	0.065	**1.033**	**1.203**
	Camada	0.115	0.444	0.177	**0.394**	**0.707**	0.891	1.024	**0.068**	**1.033**	1.199
	Neurônio	**0.097**	**0.470**	**0.179**	**0.472**	**0.707**	**0.883**	**1.016**	**0.064**	**1.026**	**1.178**
Quadrática	Normal	**0.072**	**0.239**	**0.082**	**0.253**	**0.460**	**1.047**	**1.090**	0.024	**1.084**	**1.159**
	Camada	**0.072**	0.283	**0.116**	**0.262**	**0.639**	**0.982**	1.077	**0.026**	1.075	1.117

Continua na próxima página

Tabela 25 – Resultados do Experimento 1 para a Base de Dados 8 (penguins).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**0.152**	**0.294**	0.110	**0.262**	0.577	1.021	**1.069**	**0.018**	**1.067**	**1.105**
SELU	Normal	**0.262**	0.646	**0.108**	0.637	**0.857**	**0.726**	**0.871**	**0.083**	**0.877**	1.072
	Camada	**0.246**	**0.642**	**0.114**	**0.636**	0.868	**0.697**	0.862	**0.099**	0.863	**1.079**
	Neurônio	0.257	**0.657**	0.112	**0.640**	**0.896**	0.709	**0.856**	0.089	**0.862**	**1.070**
Sigmóide	Normal	0.102	0.307	0.143	**0.262**	**0.707**	**1.008**	**1.070**	**0.034**	**1.069**	1.148
	Camada	**0.090**	**0.306**	**0.145**	**0.262**	**0.707**	1.006	**1.070**	**0.034**	**1.069**	**1.149**
	Neurônio	**0.116**	**0.314**	**0.138**	**0.262**	**0.694**	**1.005**	**1.063**	**0.032**	**1.062**	**1.132**
SoftExponential	Normal	**0.223**	**0.616**	**0.120**	**0.621**	0.826	**0.743**	**0.899**	**0.081**	**0.894**	1.083
	Camada	**0.223**	**0.616**	**0.127**	**0.625**	**0.821**	**0.715**	0.893	**0.088**	0.885	**1.085**
	Neurônio	**0.197**	**0.628**	**0.120**	**0.621**	**0.865**	0.723	**0.886**	0.084	**0.882**	**1.081**
SoftPlus	Normal	**0.142**	0.448	0.184	0.476	**0.725**	**0.911**	**1.014**	**0.055**	**1.013**	1.117
	Camada	0.144	**0.447**	**0.189**	**0.478**	**0.725**	**0.896**	1.013	**0.061**	1.012	**1.127**
	Neurônio	**0.151**	**0.460**	**0.180**	**0.469**	**0.755**	0.897	**1.006**	0.057	**1.005**	**1.112**
SoftSign	Normal	**0.000**	**0.458**	0.201	**0.480**	**0.792**	**0.901**	**1.028**	**0.059**	**1.025**	**1.152**
	Camada	**0.000**	0.460	**0.203**	0.487	0.777	**0.884**	1.024	**0.063**	1.024	**1.152**
	Neurônio	**0.068**	**0.473**	**0.183**	**0.503**	**0.706**	0.888	**1.017**	0.060	**1.016**	**1.143**
Swish	Normal	**0.014**	0.434	0.189	**0.425**	**0.707**	**0.900**	**1.032**	**0.061**	**1.038**	**1.182**
	Camada	**0.014**	**0.432**	**0.194**	**0.425**	**0.707**	0.887	1.029	**0.065**	1.037	1.180
	Neurônio	**0.014**	**0.452**	**0.181**	**0.468**	**0.707**	**0.884**	**1.021**	**0.061**	**1.021**	**1.159**
Tangente Hiperbólica	Normal	**0.104**	0.551	**0.166**	0.566	0.783	**0.824**	**0.967**	**0.073**	**0.962**	1.119

Continua na próxima página

Tabela 25 – Resultados do Experimento 1 para a Base de Dados 8 (penguins).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**0.071**	**0.545**	**0.174**	**0.553**	**0.753**	**0.799**	0.961	**0.083**	0.959	**1.128**
	Neurônio	0.088	**0.568**	0.170	**0.603**	**0.827**	0.808	**0.954**	0.076	**0.952**	**1.112**

Fonte: Elaborada pelo autor (2023).

Tabela 26 – Resultados do Experimento 1 para a Base de Dados 9 (sonar).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	0.441	0.575	0.081	0.569	**0.733**	0.606	**0.672**	**0.040**	**0.672**	**0.752**
	Camada	**0.462**	0.576	0.080	0.569	**0.733**	**0.605**	**0.672**	0.041	0.673	0.753
	Neurônio	0.441	**0.577**	**0.077**	**0.578**	0.713	**0.605**	**0.672**	0.041	**0.672**	**0.752**
Cúbico	Normal	**0.307**	0.351	**0.044**	0.354	0.490	0.691	0.694	**0.002**	0.694	**0.700**
	Camada	**0.307**	0.453	0.077	0.456	**0.657**	0.674	**0.689**	0.009	**0.688**	0.706
	Neurônio	0.297	**0.471**	0.080	**0.470**	**0.657**	**0.672**	0.690	0.009	**0.688**	0.706
Linear	Normal	**0.429**	**0.578**	**0.080**	**0.583**	0.706	0.615	**0.671**	**0.040**	**0.663**	**0.753**
	Camada	**0.429**	0.577	0.081	**0.583**	0.706	**0.613**	**0.671**	0.041	0.664	0.755
	Neurônio	**0.429**	0.577	0.084	0.579	**0.733**	0.614	**0.671**	**0.040**	**0.663**	**0.753**
Mish	Normal	**0.370**	**0.545**	**0.094**	0.545	**0.710**	0.632	0.682	**0.027**	**0.679**	**0.732**
	Camada	**0.370**	**0.545**	**0.094**	0.545	**0.710**	**0.630**	0.682	**0.027**	**0.679**	0.733
	Neurônio	**0.370**	0.544	0.095	**0.553**	**0.710**	0.631	**0.681**	**0.027**	**0.679**	**0.732**
PELU	Normal	**0.451**	**0.568**	**0.088**	**0.570**	**0.710**	0.615	**0.675**	**0.036**	**0.670**	**0.749**
	Camada	**0.451**	0.566	0.089	0.569	**0.710**	**0.612**	**0.675**	0.037	**0.670**	0.751
	Neurônio	**0.451**	0.566	0.089	0.569	**0.710**	0.614	**0.675**	**0.036**	**0.670**	**0.749**
PReLU	Normal	**0.393**	0.545	0.089	0.553	**0.691**	0.631	0.683	**0.025**	**0.684**	**0.722**
	Camada	**0.393**	0.547	0.087	0.554	**0.691**	**0.630**	0.683	**0.025**	**0.684**	**0.722**
	Neurônio	**0.393**	**0.550**	**0.085**	**0.564**	0.690	0.631	**0.682**	**0.025**	**0.684**	**0.722**
Quadrática	Normal	**0.286**	0.416	0.095	0.400	**0.651**	**0.684**	0.694	**0.005**	**0.693**	0.705
	Camada	**0.286**	0.412	**0.093**	0.385	0.636	**0.684**	**0.693**	**0.005**	**0.693**	**0.704**

Continua na próxima página

Tabela 26 – Resultados do Experimento 1 para a Base de Dados 9 (sonar).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
Tangente Hiperbólica	Neurônio	0.251	**0.427**	0.097	**0.433**	0.630	**0.684**	**0.693**	**0.005**	**0.693**	**0.704**
SELU	Normal	**0.429**	0.582	**0.079**	0.589	0.710	0.608	0.669	**0.043**	**0.661**	**0.769**
	Camada	**0.429**	**0.583**	0.081	0.589	0.733	**0.607**	0.669	**0.043**	**0.661**	0.771
	Neurônio	**0.429**	**0.583**	0.080	**0.592**	**0.736**	**0.607**	**0.668**	**0.043**	**0.661**	**0.769**
Sigmóide	Normal	**0.279**	0.488	0.094	**0.496**	**0.714**	**0.676**	**0.692**	**0.011**	**0.690**	**0.711**
	Camada	**0.279**	**0.489**	0.094	**0.496**	**0.714**	**0.676**	**0.692**	**0.011**	**0.690**	**0.711**
	Neurônio	**0.279**	0.486	**0.092**	**0.496**	0.690	**0.676**	**0.692**	**0.011**	**0.690**	**0.711**
SoftExponential	Normal	**0.453**	**0.574**	0.083	**0.570**	**0.713**	0.610	**0.672**	**0.040**	**0.666**	**0.750**
	Camada	**0.453**	0.573	0.082	**0.570**	**0.713**	**0.609**	**0.672**	0.041	**0.666**	0.751
	Neurônio	**0.453**	**0.574**	**0.077**	0.569	0.710	**0.609**	**0.672**	**0.040**	**0.666**	0.751
SoftPlus	Normal	**0.352**	0.539	0.092	0.535	**0.714**	0.643	**0.683**	**0.023**	**0.680**	**0.725**
	Camada	**0.352**	0.537	**0.090**	0.534	**0.714**	**0.641**	**0.683**	**0.023**	**0.680**	0.726
	Neurônio	**0.352**	**0.540**	0.091	**0.545**	**0.714**	0.642	**0.683**	**0.023**	**0.680**	**0.725**
SoftSign	Normal	**0.378**	0.533	**0.090**	**0.553**	**0.684**	0.646	**0.683**	**0.026**	**0.674**	**0.737**
	Camada	**0.378**	0.533	**0.090**	**0.553**	**0.684**	**0.645**	**0.683**	**0.026**	**0.674**	0.738
	Neurônio	**0.378**	**0.534**	**0.090**	0.532	**0.684**	0.646	**0.683**	**0.026**	**0.674**	**0.737**
Swish	Normal	**0.393**	0.530	**0.086**	0.521	**0.706**	0.640	**0.684**	**0.023**	0.682	**0.726**
	Camada	**0.393**	0.530	**0.086**	0.521	**0.706**	**0.639**	**0.684**	0.024	**0.681**	**0.726**
	Neurônio	0.383	**0.536**	0.089	**0.541**	**0.706**	0.640	**0.684**	**0.023**	**0.681**	**0.726**
	Normal	**0.404**	0.557	**0.087**	**0.579**	**0.688**	0.624	**0.675**	**0.035**	0.665	**0.751**

Continua na próxima página

Tabela 26 – Resultados do Experimento 1 para a Base de Dados 9 (sonar).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**0.404**	**0.557**	**0.088**	**0.579**	**0.688**	**0.622**	**0.675**	**0.036**	**0.664**	**0.753**
	Neurônio	**0.404**	**0.558**	**0.088**	**0.579**	**0.688**	0.623	**0.675**	**0.035**	**0.664**	**0.751**

Fonte: Elaborada pelo autor (2023).

Tabela 27 – Resultados do Experimento 1 para a Base de Dados 10 (threeOf9).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	0.474	0.621	0.061	0.632	0.727	0.564	0.644	**0.036**	**0.640**	0.722
	Camada	0.474	0.621	**0.060**	0.635	0.727	**0.560**	0.644	0.037	**0.640**	**0.721**
	Neurônio	**0.485**	**0.627**	**0.060**	**0.640**	**0.732**	0.562	**0.643**	**0.036**	**0.640**	**0.721**
Cúbico	Normal	0.296	0.355	**0.049**	0.372	0.539	0.689	0.694	**0.005**	0.692	0.712
	Camada	0.296	0.450	0.107	0.414	**0.689**	**0.660**	0.688	0.011	0.691	0.703
	Neurônio	**0.317**	**0.461**	0.081	**0.469**	0.625	0.661	**0.687**	0.008	**0.688**	**0.700**
Linear	Normal	0.483	0.628	**0.057**	0.636	0.708	0.563	0.642	**0.037**	0.635	0.719
	Camada	0.485	0.628	0.059	0.638	**0.719**	**0.556**	0.641	0.039	0.637	0.718
	Neurônio	**0.495**	**0.633**	0.058	**0.650**	**0.719**	0.560	**0.639**	0.038	**0.634**	**0.716**
Mish	Normal	0.443	0.573	**0.070**	0.577	**0.707**	0.611	0.673	**0.025**	0.672	0.723
	Camada	0.443	0.572	0.071	0.581	**0.707**	**0.606**	0.672	0.026	0.672	0.722
	Neurônio	**0.464**	**0.579**	0.071	**0.595**	**0.707**	0.608	**0.671**	**0.025**	**0.671**	**0.721**
PELU	Normal	0.464	0.606	**0.063**	0.611	**0.737**	0.578	0.656	**0.032**	0.651	0.724
	Camada	0.464	0.605	0.065	0.611	0.728	**0.569**	0.655	0.034	0.651	0.722
	Neurônio	**0.474**	**0.612**	0.064	**0.619**	**0.737**	0.574	**0.653**	0.033	**0.650**	**0.721**
PReLU	Normal	0.377	0.541	**0.069**	**0.546**	0.661	0.624	0.679	**0.022**	0.677	0.720
	Camada	0.377	0.543	0.070	**0.546**	0.669	**0.619**	0.678	0.023	0.677	**0.719**
	Neurônio	**0.388**	**0.548**	0.072	0.543	**0.698**	0.621	**0.677**	**0.022**	**0.676**	**0.719**
Quadrática	Normal	**0.296**	0.378	**0.050**	**0.372**	0.528	0.687	0.694	0.005	0.694	0.712
	Camada	**0.296**	0.396	0.060	**0.372**	0.543	**0.680**	0.692	0.005	0.692	0.704

Continua na próxima página

Tabela 27 – Resultados do Experimento 1 para a Base de Dados 10 (threeOf9).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**0.296**	**0.415**	0.083	**0.372**	**0.612**	0.685	**0.691**	**0.003**	**0.691**	**0.699**
SELU	Normal	0.477	0.637	0.055	0.645	0.699	0.551	0.635	**0.037**	0.629	0.714
	Camada	0.477	0.634	**0.054**	0.642	0.695	**0.545**	0.635	0.039	0.630	0.715
	Neurônio	**0.492**	**0.640**	0.055	**0.654**	**0.706**	0.548	**0.632**	0.038	**0.628**	**0.711**
Sigmóide	Normal	0.332	0.449	**0.088**	0.420	**0.680**	0.664	0.686	**0.011**	**0.687**	0.708
	Camada	0.324	0.448	**0.088**	0.420	**0.680**	0.665	0.686	**0.011**	**0.687**	0.708
	Neurônio	**0.350**	**0.454**	**0.088**	**0.442**	0.660	**0.663**	**0.685**	**0.011**	**0.687**	**0.707**
SoftExponential	Normal	**0.491**	0.627	**0.060**	0.643	0.738	0.565	0.642	**0.037**	**0.634**	0.718
	Camada	**0.491**	0.629	0.062	0.644	**0.748**	**0.561**	0.641	0.038	**0.634**	**0.717**
	Neurônio	0.481	**0.633**	0.065	**0.648**	0.728	0.562	**0.640**	**0.037**	**0.634**	**0.717**
SoftPlus	Normal	0.373	0.544	0.072	0.556	0.679	0.632	0.674	**0.020**	**0.674**	0.715
	Camada	0.365	0.542	0.074	**0.561**	0.679	**0.627**	0.674	**0.020**	0.675	**0.713**
	Neurônio	**0.391**	**0.548**	**0.068**	0.551	**0.686**	0.629	**0.673**	**0.020**	**0.674**	**0.713**
SoftSign	Normal	0.428	0.558	0.083	0.559	**0.697**	0.612	0.673	**0.025**	0.673	0.721
	Camada	0.428	0.558	0.083	0.554	**0.697**	**0.606**	0.673	0.026	0.674	0.720
	Neurônio	**0.436**	**0.574**	**0.074**	**0.573**	0.696	0.609	**0.671**	**0.025**	**0.672**	**0.718**
Swish	Normal	0.388	0.557	0.080	**0.575**	**0.707**	0.622	0.677	**0.022**	0.677	0.720
	Camada	0.388	0.557	0.080	**0.575**	**0.707**	**0.618**	0.677	0.023	**0.676**	0.720
	Neurônio	**0.403**	**0.562**	**0.076**	0.572	**0.707**	0.620	**0.676**	**0.022**	**0.676**	**0.719**
Tangente Hiperbólica	Normal	0.451	0.601	0.064	0.606	**0.718**	0.581	0.658	**0.032**	0.655	0.723

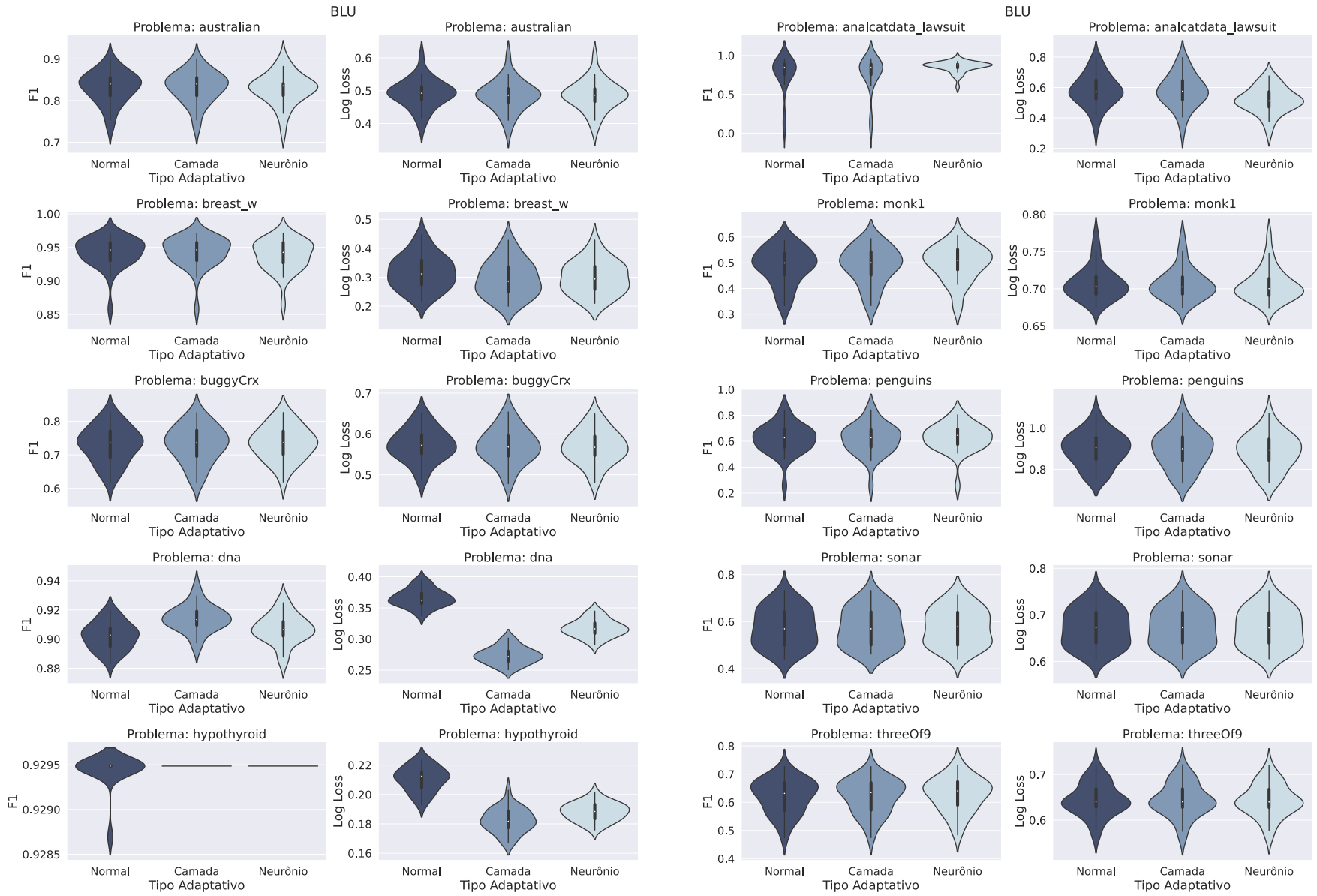
Continua na próxima página

Tabela 27 – Resultados do Experimento 1 para a Base de Dados 10 (threeOf9).

Função de Ativação	Tipo Adaptativo	F1					Log Loss				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**0.440**	**0.598**	**0.066**	**0.600**	**0.718**	**0.574**	0.657	**0.034**	**0.655**	0.722
	Neurônio	**0.473**	**0.607**	**0.056**	**0.611**	0.698	0.578	**0.656**	0.033	**0.654**	**0.720**

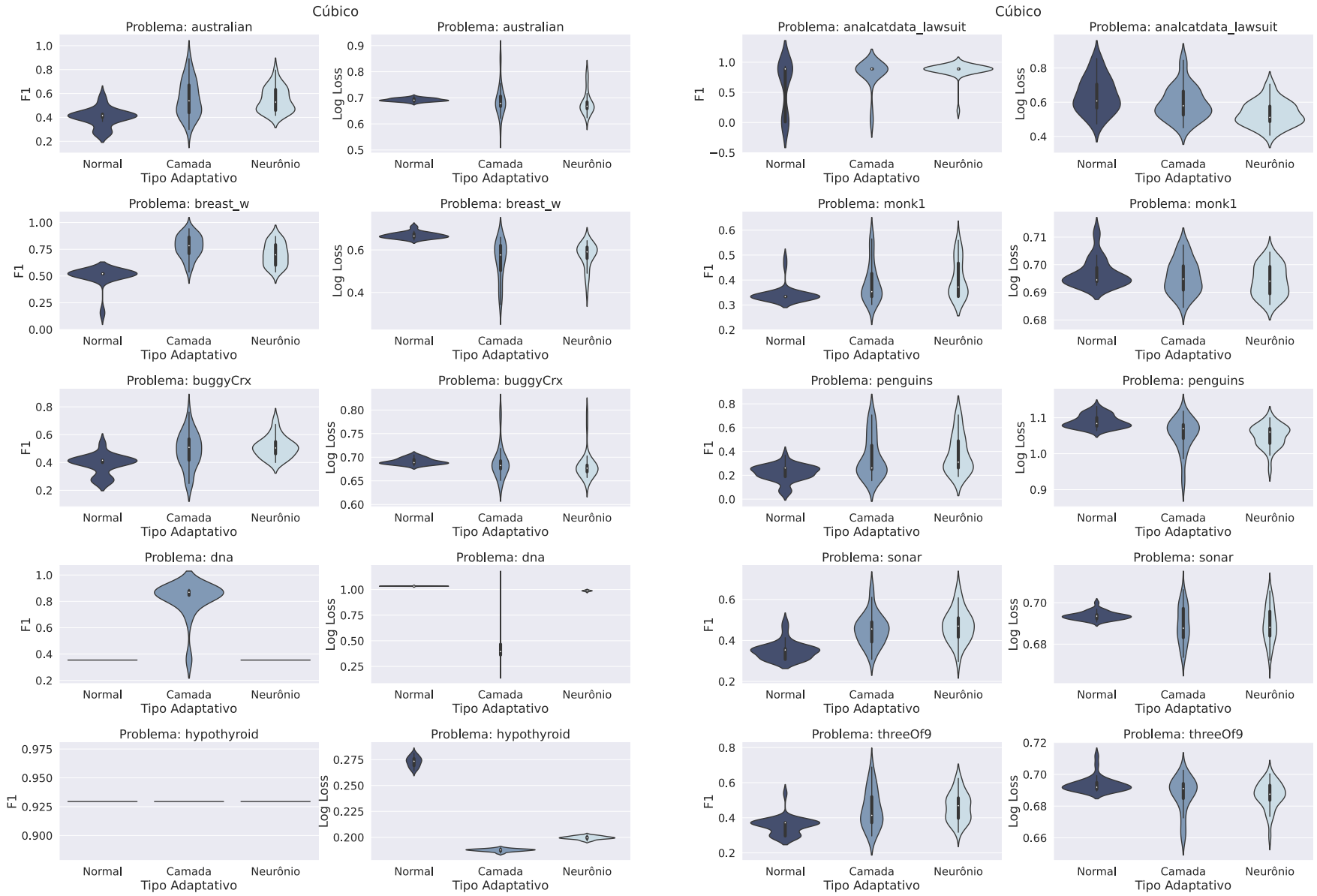
Fonte: Elaborada pelo autor (2023).

Figura 75 – *Violin Plots* do Experimento 1 para função de ativação BLU.



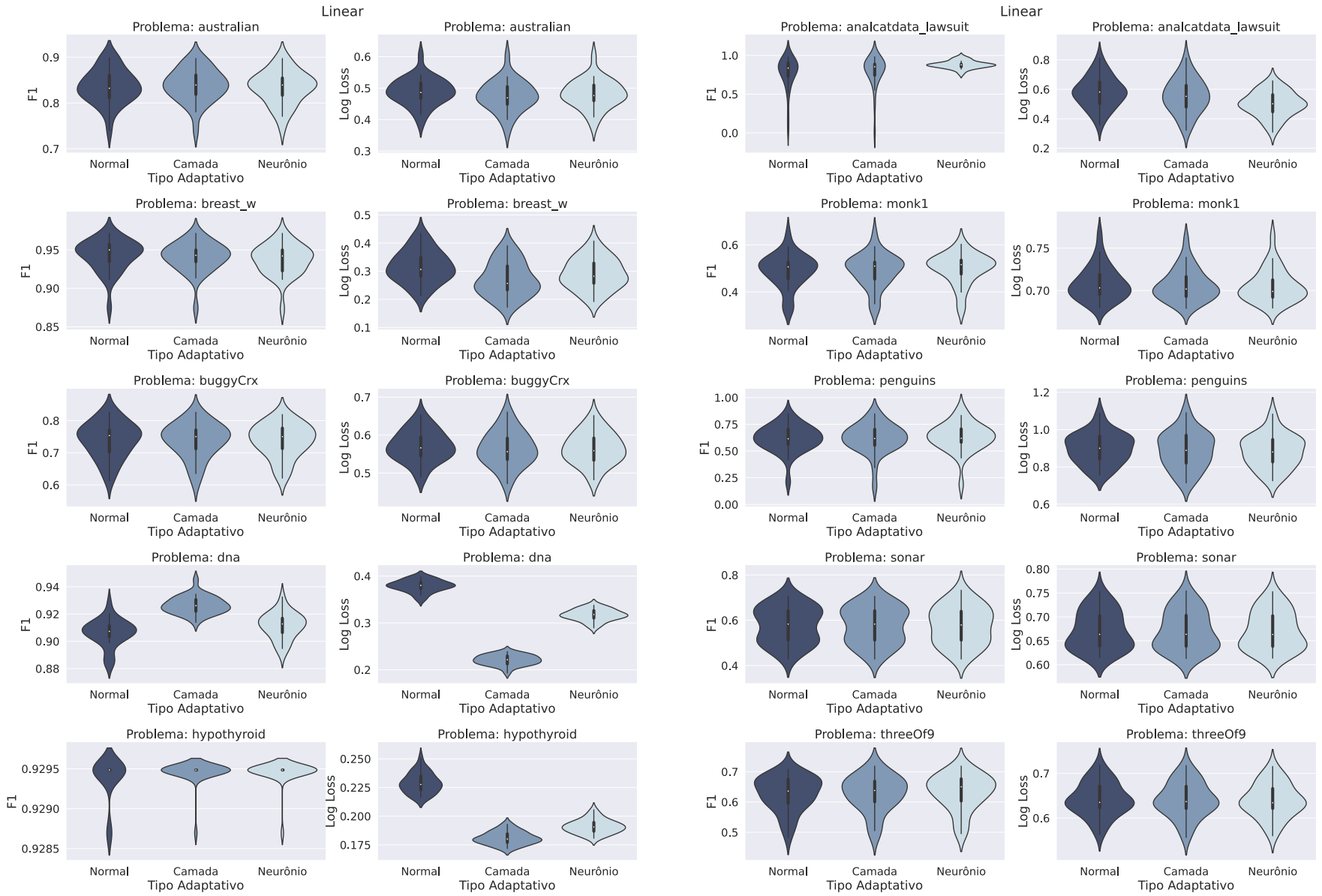
Fonte: Elaborada pelo autor (2023).

Figura 76 – *Violin Plots* do Experimento 1 para função de ativação Cúbico.



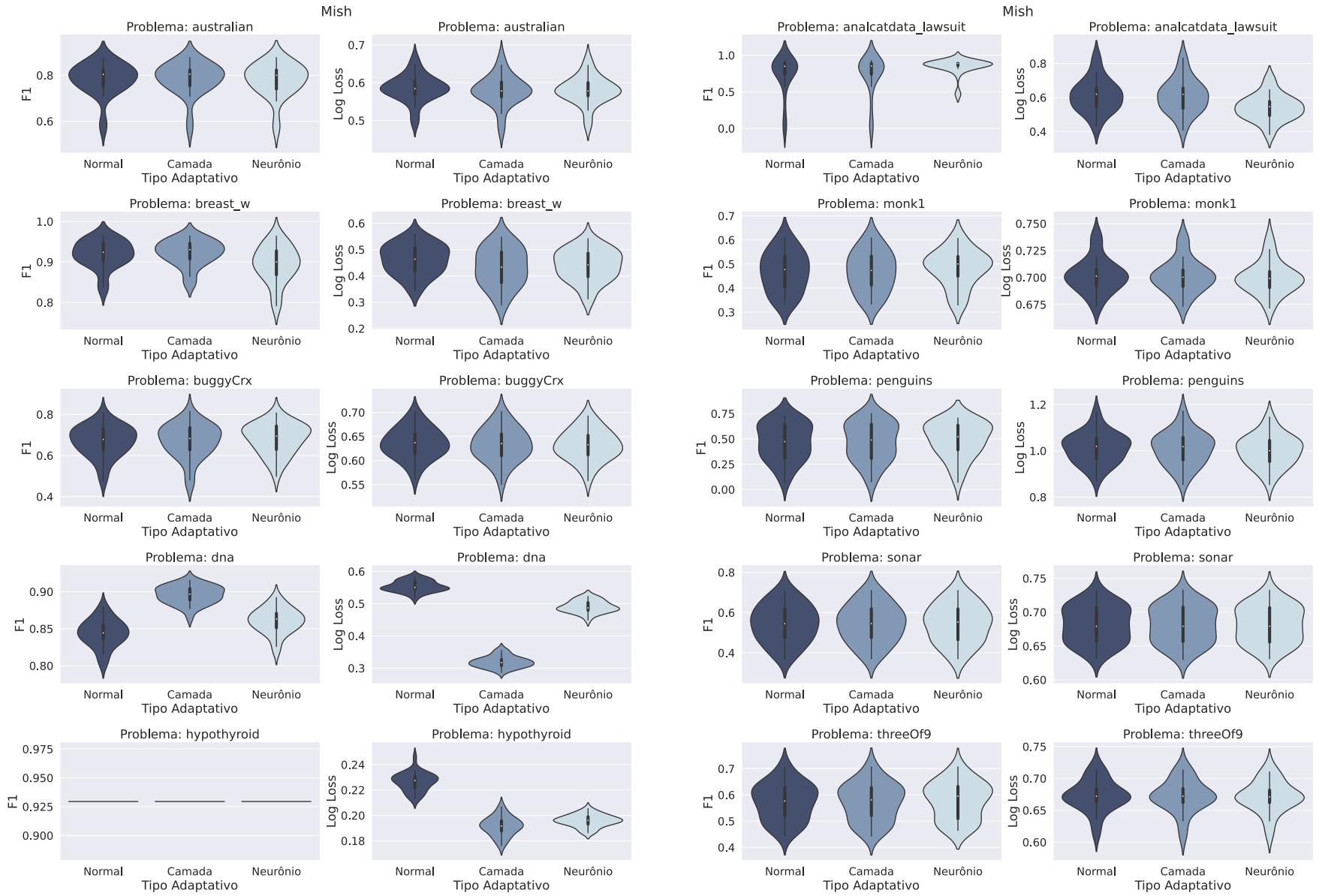
Fonte: Elaborada pelo autor (2023).

Figura 77 – *Violin Plots* do Experimento 1 para função de ativação Linear.



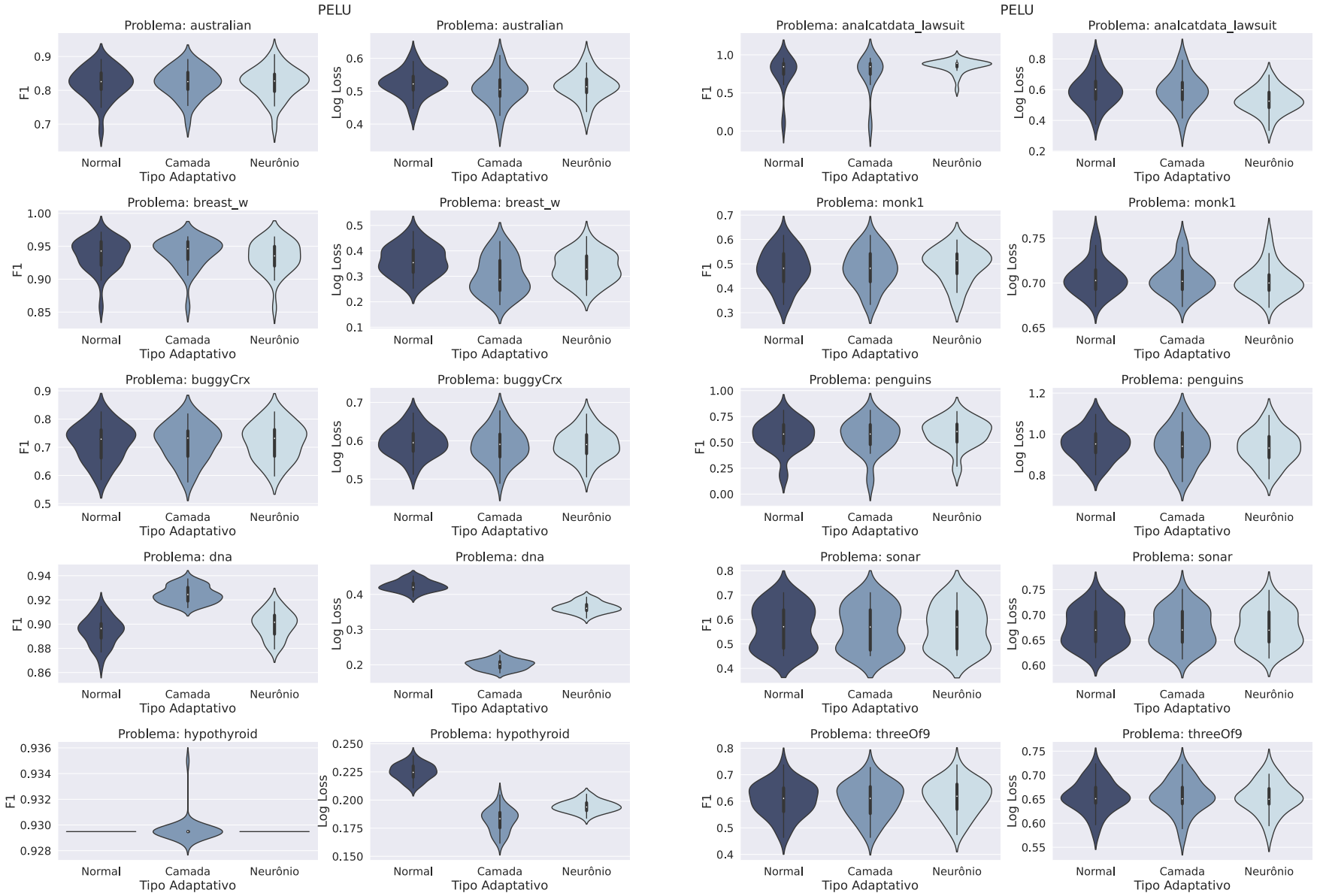
Fonte: Elaborada pelo autor (2023).

Figura 78 – *Violin Plots* do Experimento 1 para função de ativação Mish.



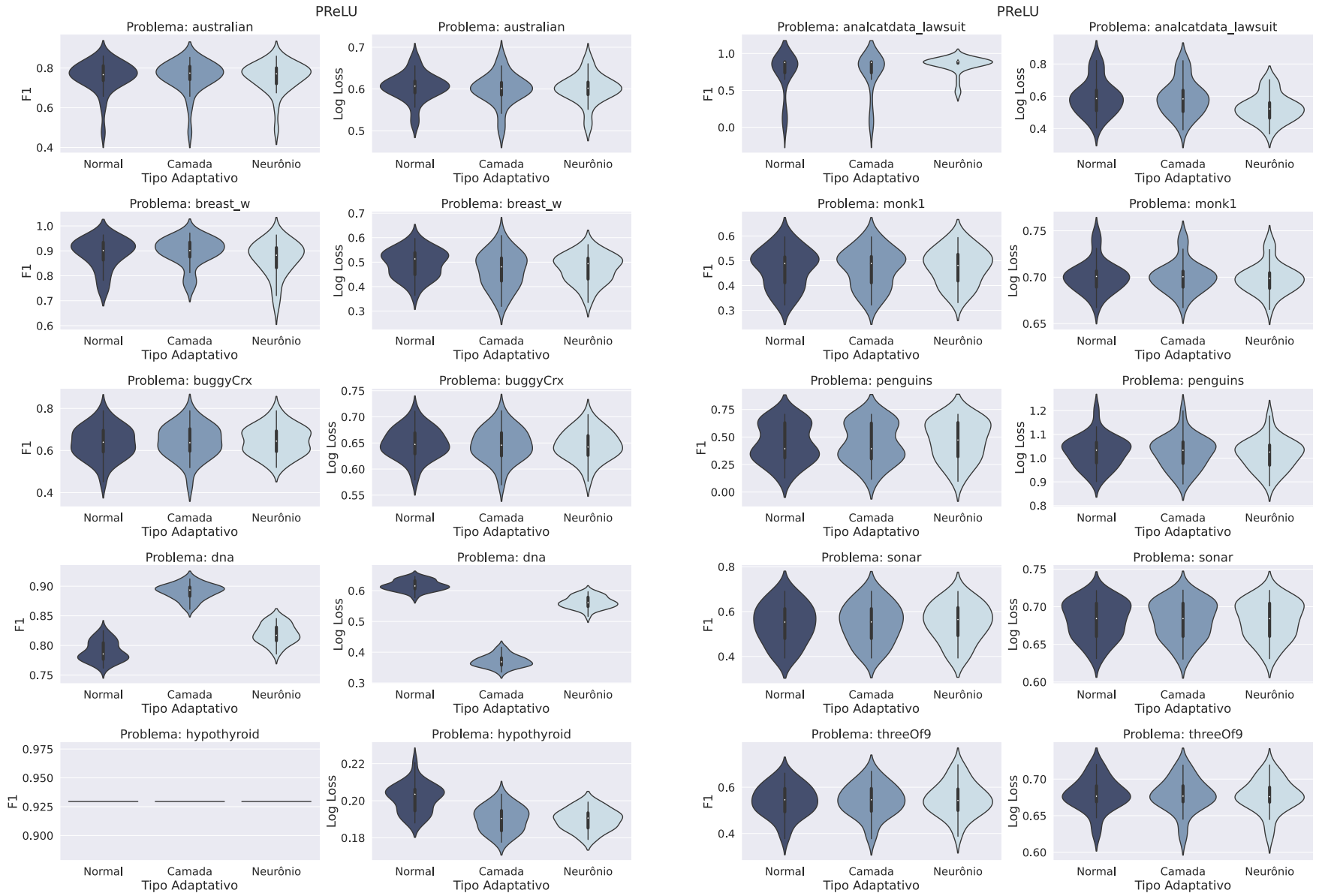
Fonte: Elaborada pelo autor (2023).

Figura 79 – *Violin Plots* do Experimento 1 para função de ativação PELU.



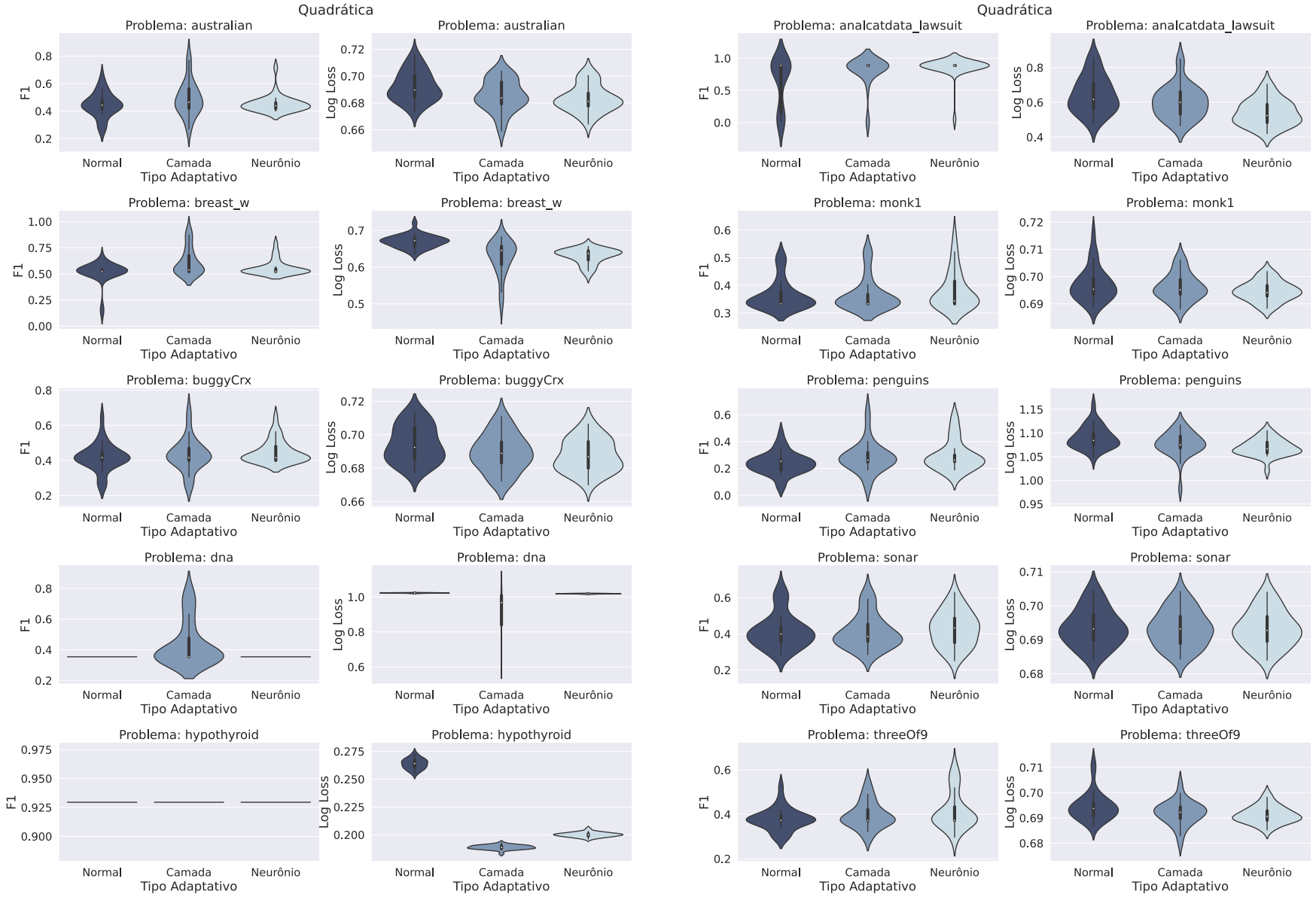
Fonte: Elaborada pelo autor (2023).

Figura 80 – *Violin Plots* do Experimento 1 para função de ativação PReLU.



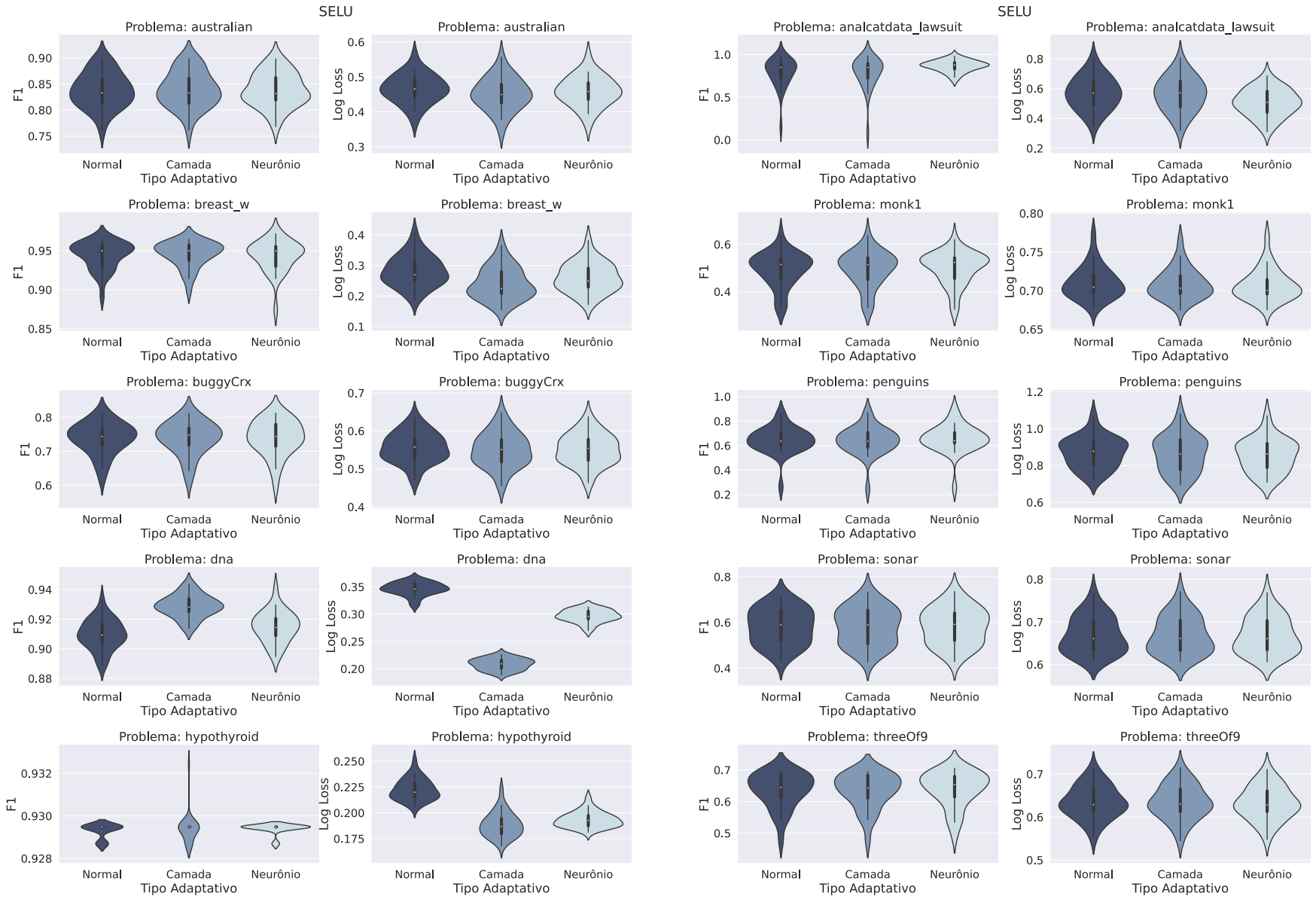
Fonte: Elaborada pelo autor (2023).

Figura 81 – *Violin Plots* do Experimento 1 para função de ativação Quadrática.



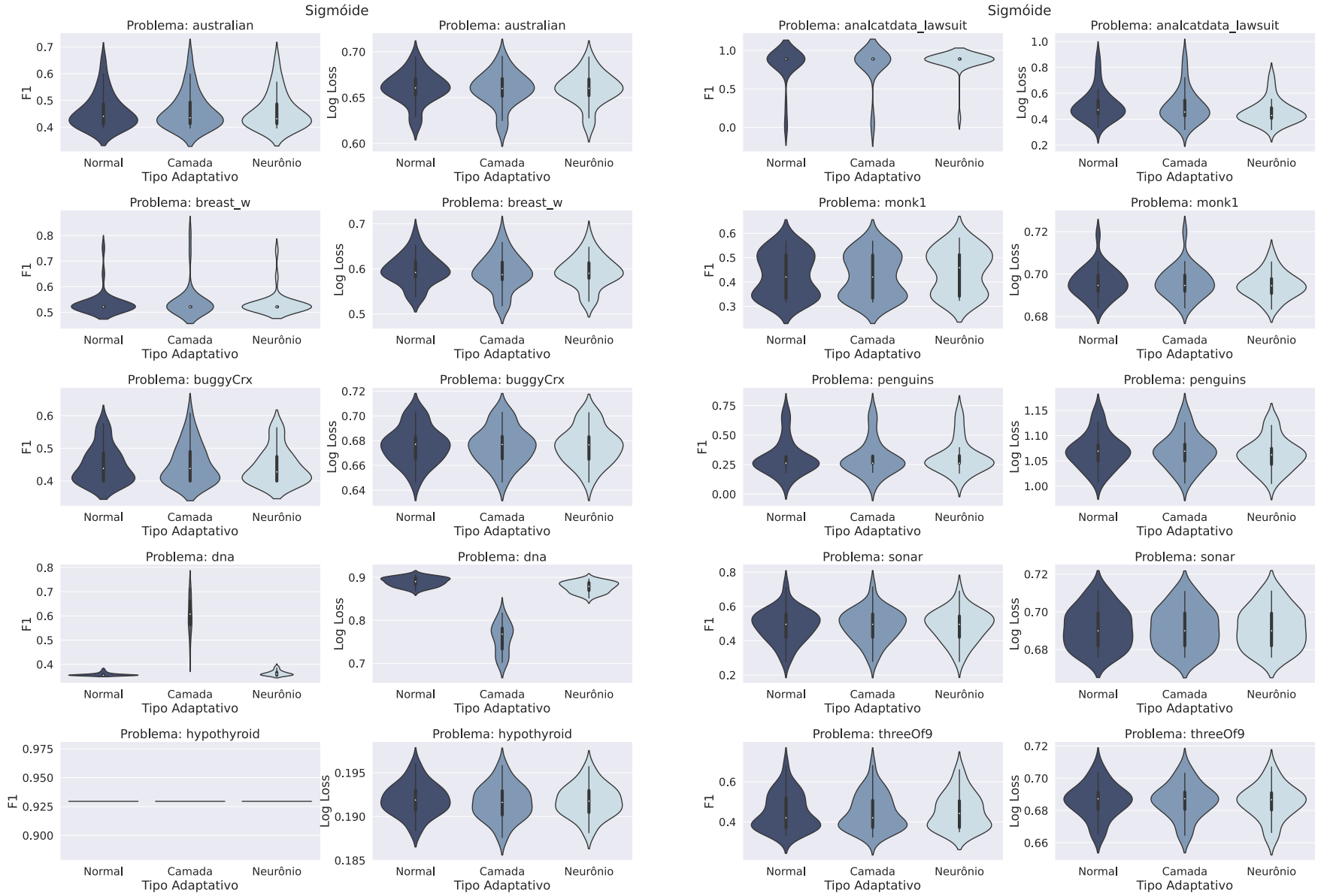
Fonte: Elaborada pelo autor (2023).

Figura 82 – *Violin Plots* do Experimento 1 para função de ativação SELU.



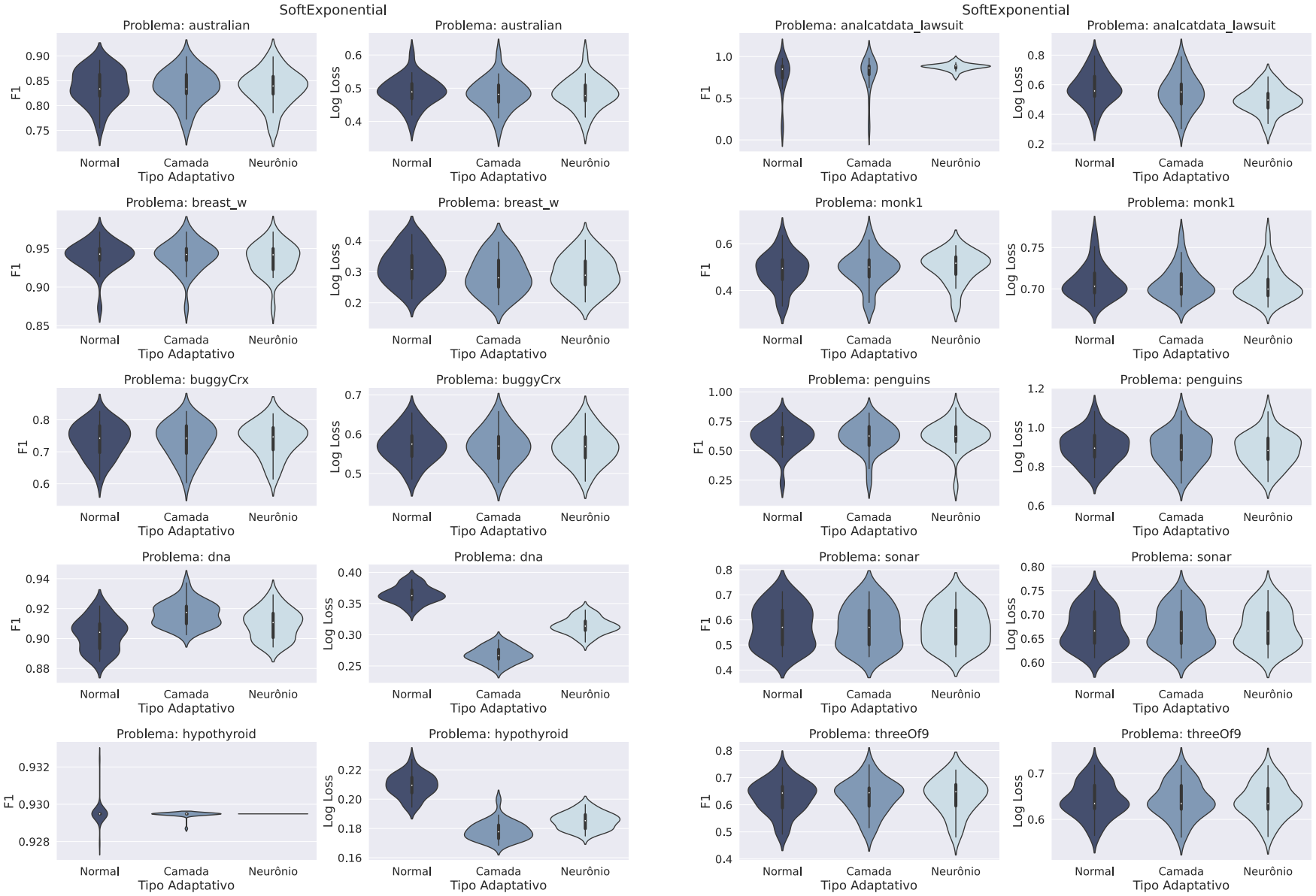
Fonte: Elaborada pelo autor (2023).

Figura 83 – *Violin Plots* do Experimento 1 para função de ativação Sigmóide.



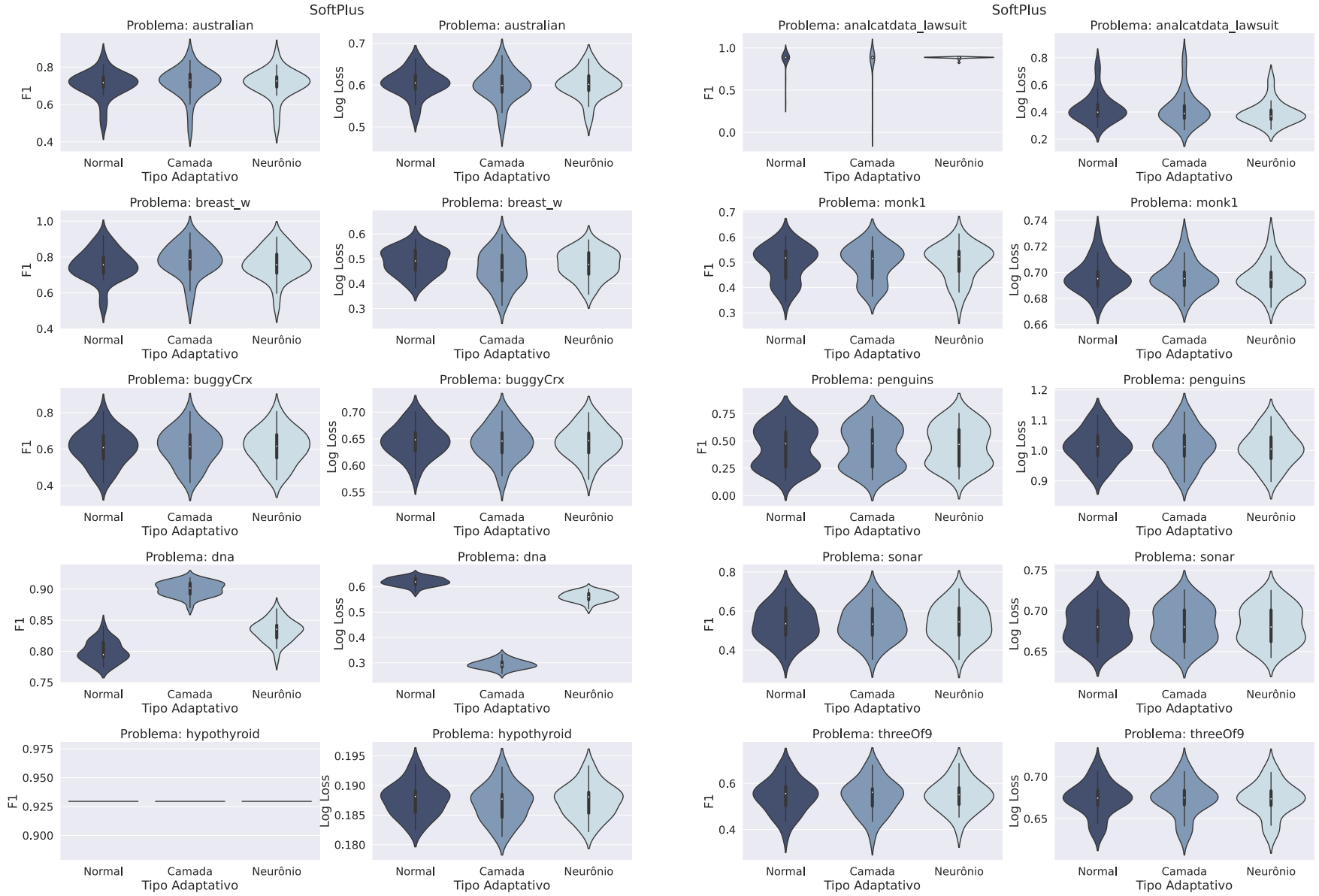
Fonte: Elaborada pelo autor (2023).

Figura 84 – *Violin Plots* do Experimento 1 para função de ativação SoftExponential.



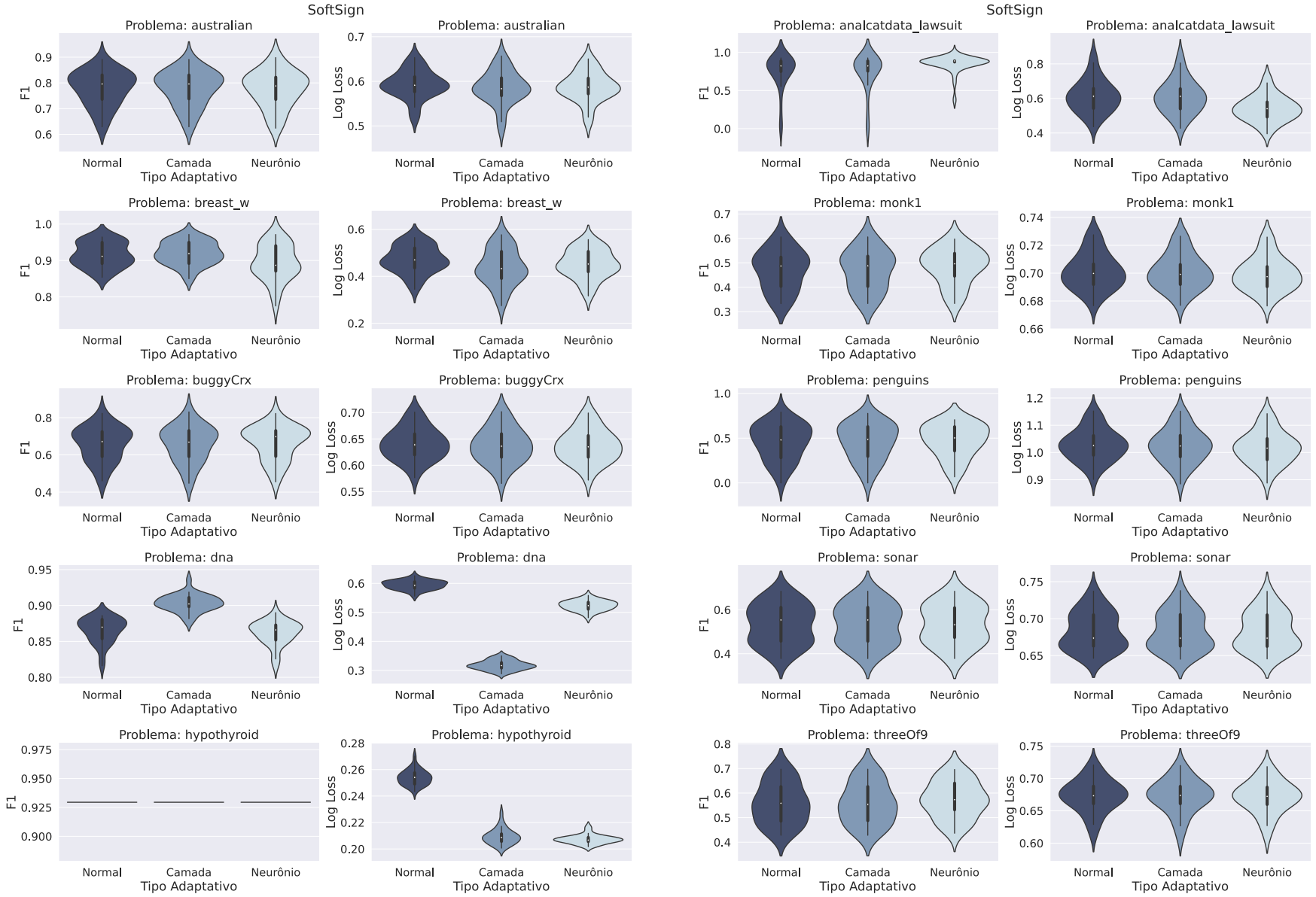
Fonte: Elaborada pelo autor (2023).

Figura 85 – *Violin Plots* do Experimento 1 para função de ativação SoftPlus.



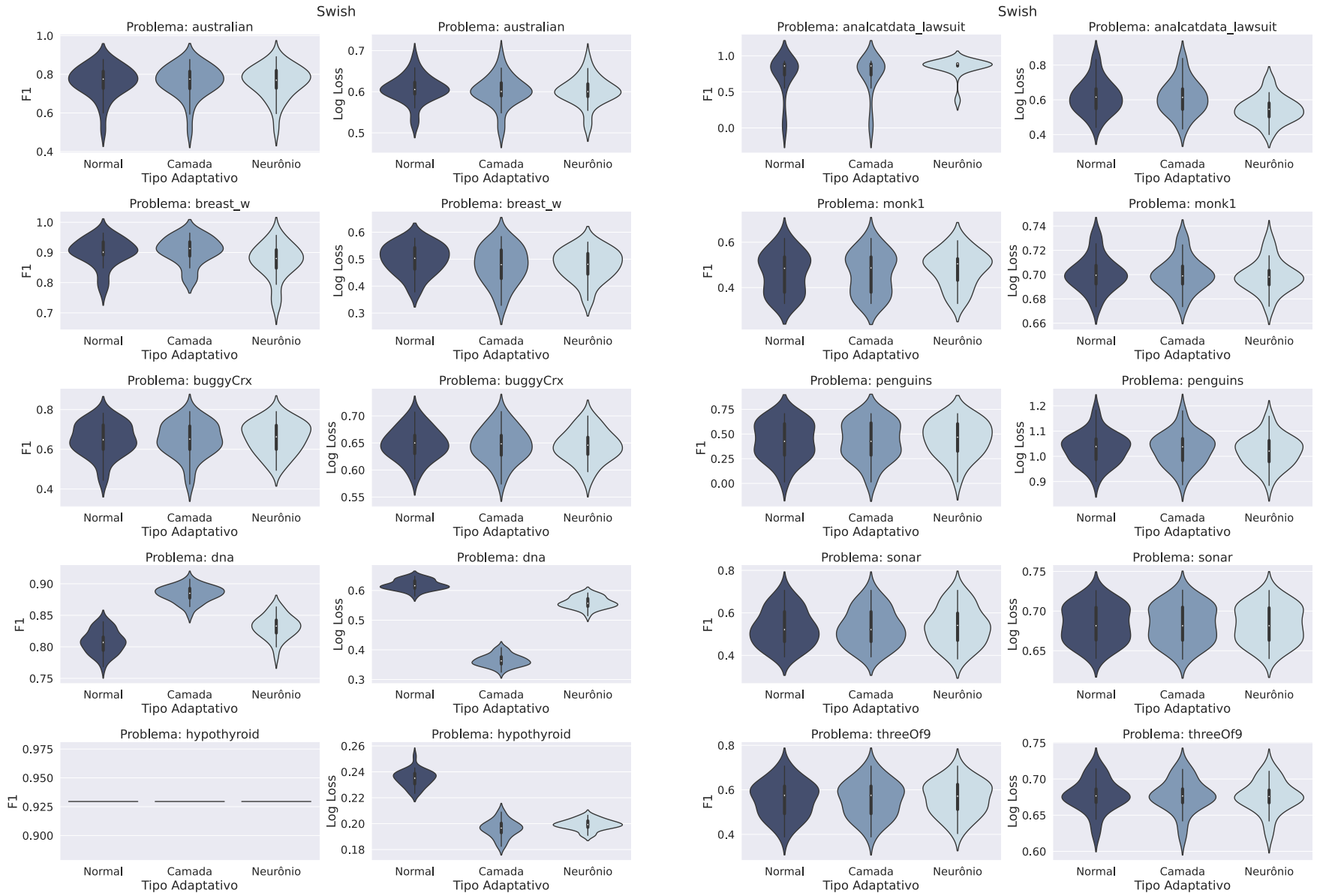
Fonte: Elaborada pelo autor (2023).

Figura 86 – *Violin Plots* do Experimento 1 para função de ativação SoftSign.



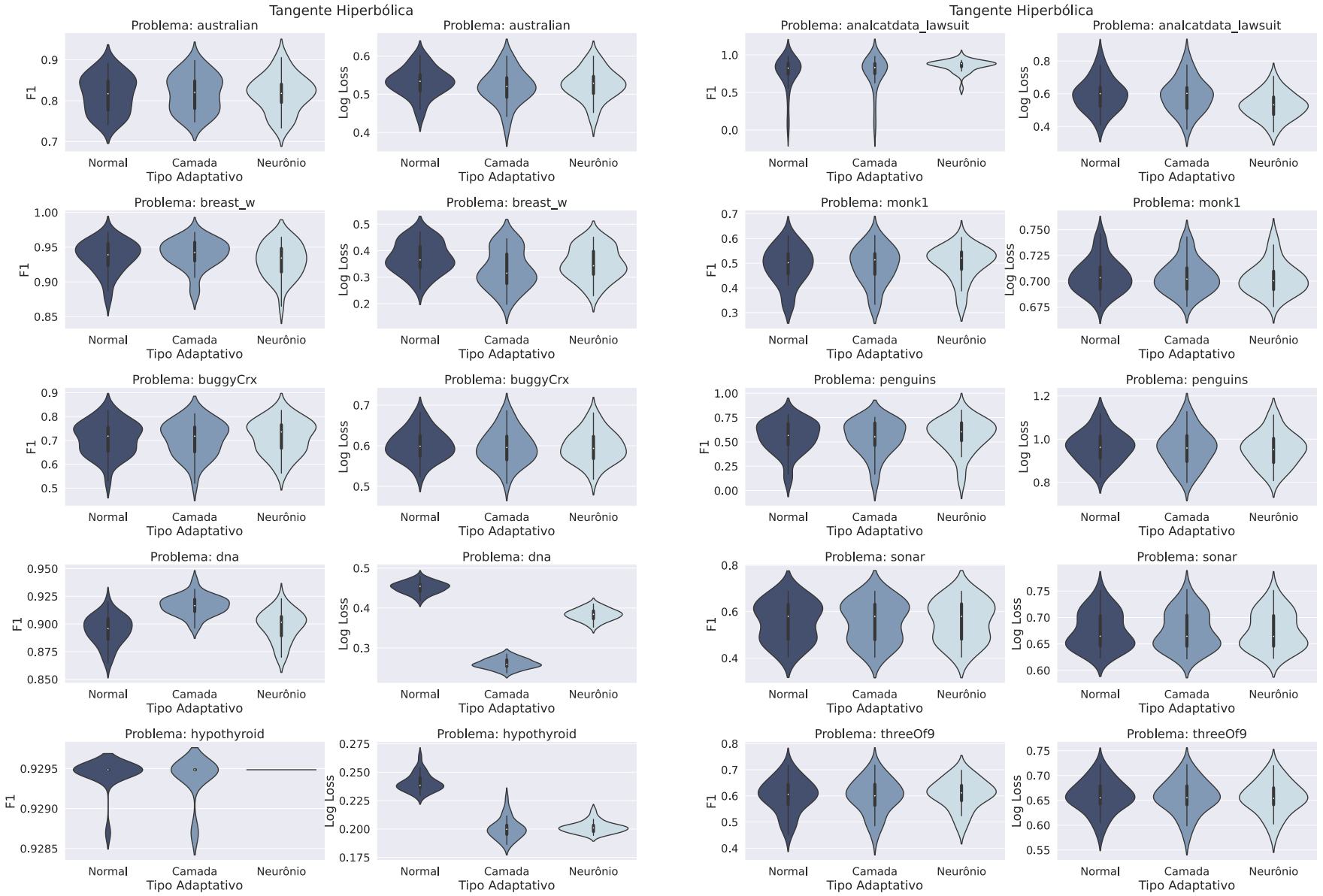
Fonte: Elaborada pelo autor (2023).

Figura 87 – *Violin Plots* do Experimento 1 para função de ativação Swish.



Fonte: Elaborada pelo autor (2023).

Figura 88 – *Violin Plots* do Experimento 1 para função de ativação Tangente Hiperbólica.



Fonte: Elaborada pelo autor (2023).

A.2 Pegada de Carbono

A Tabela 28 apresenta as estimativas médias de emissão de CO_2, em quilograma, associadas ao treinamento de cada modelo no primeiro experimento numérico. Essa média é obtida a partir de cinco execuções focadas na estimativa da emissão de CO_2. Os valores destacados com a cor **"Azul Escuro"** representam os melhores resultados encontrados, enquanto os valores marcados com a cor **"Azul Claro"** representam os resultados menos favoráveis.

Tabela 28 – Estimativas de Emissões de CO_2, em quilograma, do Experimento 1.

Base de Dados	Função de Ativação	Normal	Camada	Neurônio
1	BLU	**2.96e-06**	3.54e-06	**3.66e-06**
	Cúbico	**2.90e-06**	**3.60e-06**	3.59e-06
	Linear	**2.82e-06**	**3.24e-06**	3.19e-06
	Mish	2.95e-06	**2.79e-06**	**3.52e-06**
	PELU	**3.07e-06**	**3.78e-06**	3.73e-06
	PReLU	**2.91e-06**	3.34e-06	**3.35e-06**
	Quadrática	**2.80e-06**	3.38e-06	**3.40e-06**
	SELU	**2.98e-06**	3.56e-06	**3.59e-06**
	Sigmóide	**2.90e-06**	3.44e-06	**3.46e-06**
	SoftExponential	**2.88e-06**	3.43e-06	**4.53e-06**
	SoftPlus	2.90e-06	**2.75e-06**	**3.41e-06**
	SoftSign	**2.97e-06**	3.42e-06	**3.42e-06**
	Swish	**2.97e-06**	**3.52e-06**	3.51e-06
	Tangente Hiperbólica	**2.96e-06**	3.72e-06	**3.76e-06**
2	BLU	**6.89e-06**	8.54e-06	**8.71e-06**
	Cúbico	**6.81e-06**	8.70e-06	**8.76e-06**
	Linear	**6.61e-06**	7.69e-06	**7.71e-06**
	Mish	**6.96e-06**	**8.43e-06**	8.42e-06
	PELU	**7.33e-06**	9.15e-06	**9.28e-06**
	PReLU	**7.01e-06**	**8.07e-06**	8.06e-06
	Quadrática	**5.37e-06**	8.15e-06	**8.20e-06**
	SELU	**7.29e-06**	8.63e-06	**8.73e-06**
	Sigmóide	**6.92e-06**	8.32e-06	**8.36e-06**
	SoftExponential	**6.85e-06**	8.34e-06	**1.12e-05**
	SoftPlus	**6.88e-06**	**8.18e-06**	8.15e-06
	SoftSign	**6.85e-06**	8.30e-06	**8.34e-06**
	Swish	**6.91e-06**	8.43e-06	**8.52e-06**

Continua na próxima página

Base de Dados	Função de Ativação	Normal	Camada	Neurônio
	Tangente Hiperbólica	**7.05e-06**	9.15e-06	**9.22e-06**
3	BLU	**6.91e-06**	**8.54e-06**	6.93e-06
	Cúbico	**6.82e-06**	8.68e-06	**8.70e-06**
	Linear	**6.69e-06**	**7.78e-06**	7.77e-06
	Mish	7.02e-06	**8.39e-06**	**6.78e-06**
	PELU	**7.27e-06**	**9.14e-06**	7.51e-06
	PReLU	**7.07e-06**	8.08e-06	**8.19e-06**
	Quadrática	**6.74e-06**	8.19e-06	**8.20e-06**
	SELU	**7.16e-06**	8.61e-06	**8.64e-06**
	Sigmóide	**6.90e-06**	8.36e-06	**8.47e-06**
	SoftExponential	**6.93e-06**	8.48e-06	**8.95e-06**
	SoftPlus	6.95e-06	**8.19e-06**	**6.63e-06**
	SoftSign	6.87e-06	**8.28e-06**	**4.98e-06**
	Swish	**6.95e-06**	8.41e-06	**8.49e-06**
	Tangente Hiperbólica	**7.16e-06**	9.14e-06	**9.14e-06**
4	BLU	**5.52e-06**	8.51e-06	**8.65e-06**
	Cúbico	**6.83e-06**	8.65e-06	**8.72e-06**
	Linear	**6.66e-06**	**7.78e-06**	7.71e-06
	Mish	**6.92e-06**	**8.52e-06**	8.42e-06
	PELU	**7.34e-06**	9.19e-06	**9.21e-06**
	PReLU	**7.07e-06**	8.03e-06	**8.05e-06**
	Quadrática	**6.75e-06**	8.15e-06	**8.21e-06**
	SELU	**7.21e-06**	**8.62e-06**	8.61e-06
	Sigmóide	**6.91e-06**	8.30e-06	**8.38e-06**
	SoftExponential	**6.87e-06**	8.39e-06	**1.13e-05**
	SoftPlus	**6.90e-06**	**8.24e-06**	8.22e-06
	SoftSign	**6.90e-06**	8.24e-06	**8.28e-06**
	Swish	**5.53e-06**	8.47e-06	**8.50e-06**
	Tangente Hiperbólica	**7.07e-06**	9.09e-06	**9.13e-06**
5	BLU	**3.66e-05**	4.36e-05	**4.43e-05**
	Cúbico	**3.49e-05**	**4.52e-05**	3.64e-05
	Linear	**3.28e-05**	3.79e-05	**3.86e-05**
	Mish	**4.02e-05**	4.92e-05	**4.97e-05**
	PELU	**3.90e-05**	5.02e-05	**5.08e-05**
	PReLU	**3.68e-05**	4.21e-05	**4.23e-05**
	Quadrática	**3.39e-05**	4.12e-05	**4.16e-05**

Continua na próxima página

Base de Dados	Função de Ativação	Normal	Camada	Neurônio
	SELU	**3.81e-05**	4.62e-05	**4.67e-05**
	Sigmóide	**3.47e-05**	4.28e-05	**4.32e-05**
	SoftExponential	**3.46e-05**	4.31e-05	**6.42e-05**
	SoftPlus	**3.54e-05**	4.24e-05	**4.27e-05**
	SoftSign	**3.46e-05**	4.21e-05	**4.24e-05**
	Swish	**3.52e-05**	4.36e-05	**4.39e-05**
	Tangente Hiperbólica	**3.66e-05**	4.85e-05	**4.86e-05**
	BLU	**3.00e-05**	3.70e-05	**3.73e-05**
	Cúbico	**2.95e-05**	3.79e-05	**3.79e-05**
	Linear	**2.84e-05**	3.31e-05	**3.33e-05**
	Mish	**3.01e-05**	3.68e-05	**3.69e-05**
	PELU	**3.16e-05**	4.02e-05	**4.03e-05**
	PReLU	**3.03e-05**	3.49e-05	**3.50e-05**
6	Quadrática	**2.90e-05**	3.54e-05	**3.57e-05**
	SELU	**3.11e-05**	**3.77e-05**	3.77e-05
	Sigmóide	**2.98e-05**	3.61e-05	**3.64e-05**
	SoftExponential	**2.96e-05**	3.62e-05	**4.92e-05**
	SoftPlus	**2.96e-05**	3.56e-05	**3.58e-05**
	SoftSign	**2.98e-05**	**3.56e-05**	3.55e-05
	Swish	**2.97e-05**	3.67e-05	**3.70e-05**
	Tangente Hiperbólica	**3.05e-05**	3.96e-05	**4.00e-05**
	BLU	**5.49e-06**	6.71e-06	**6.74e-06**
	Cúbico	**5.45e-06**	6.74e-06	**6.83e-06**
	Linear	**5.21e-06**	6.08e-06	**6.19e-06**
	Mish	**5.48e-06**	**6.72e-06**	6.62e-06
	PELU	**5.74e-06**	**7.15e-06**	7.14e-06
	PReLU	**5.59e-06**	**6.38e-06**	6.30e-06
7	Quadrática	**5.42e-06**	6.35e-06	**6.44e-06**
	SELU	**5.79e-06**	**6.75e-06**	6.75e-06
	Sigmóide	**5.46e-06**	6.50e-06	**6.55e-06**
	SoftExponential	5.47e-06	**5.30e-06**	**6.91e-06**
	SoftPlus	**5.49e-06**	6.41e-06	**6.49e-06**
	SoftSign	**5.44e-06**	6.42e-06	**6.45e-06**
	Swish	**5.54e-06**	6.60e-06	**6.67e-06**
	Tangente Hiperbólica	**5.52e-06**	7.03e-06	**7.14e-06**
	BLU	**3.61e-06**	4.42e-06	**4.51e-06**

Continua na próxima página

Base de Dados	Função de Ativação	Normal	Camada	Neurônio
	Cúbico	**3.56e-06**	**4.50e-06**	4.47e-06
	Linear	**3.43e-06**	4.02e-06	**4.07e-06**
	Mish	**3.66e-06**	4.37e-06	**4.38e-06**
	PELU	**3.76e-06**	**4.72e-06**	4.71e-06
	PReLU	**3.66e-06**	**4.16e-06**	4.13e-06
8	Quadrática	**3.56e-06**	4.25e-06	**4.40e-06**
	SELU	**3.79e-06**	4.41e-06	**4.44e-06**
	Sigmóide	**3.58e-06**	4.32e-06	**4.36e-06**
	SoftExponential	**3.60e-06**	4.27e-06	**5.74e-06**
	SoftPlus	**3.59e-06**	4.29e-06	**4.31e-06**
	SoftSign	**3.56e-06**	4.26e-06	**4.34e-06**
	Swish	**3.58e-06**	4.32e-06	**4.37e-06**
	Tangente Hiperbólica	3.71e-06	**1.91e-06**	**4.64e-06**
	BLU	**2.55e-06**	3.14e-06	**3.17e-06**
	Cúbico	2.54e-06	**1.93e-06**	**3.23e-06**
	Linear	**2.46e-06**	2.81e-06	**2.84e-06**
	Mish	**2.65e-06**	**3.19e-06**	3.14e-06
	PELU	**2.75e-06**	**3.49e-06**	3.45e-06
	PReLU	**2.67e-06**	2.97e-06	**3.05e-06**
9	Quadrática	2.49e-06	**2.38e-06**	**3.03e-06**
	SELU	**2.74e-06**	**3.21e-06**	3.21e-06
	Sigmóide	**2.58e-06**	3.07e-06	**3.13e-06**
	SoftExponential	**2.60e-06**	3.09e-06	**4.21e-06**
	SoftPlus	**2.57e-06**	**3.07e-06**	3.02e-06
	SoftSign	**2.56e-06**	**3.04e-06**	3.03e-06
	Swish	**2.58e-06**	3.10e-06	**3.13e-06**
	Tangente Hiperbólica	**2.62e-06**	3.40e-06	**3.44e-06**
	BLU	**5.13e-06**	6.27e-06	**6.45e-06**
	Cúbico	**5.03e-06**	6.39e-06	**6.47e-06**
	Linear	**4.89e-06**	5.71e-06	**5.73e-06**
	Mish	**5.07e-06**	6.11e-06	**6.23e-06**
	PELU	**5.38e-06**	6.73e-06	**6.73e-06**
	PReLU	**5.23e-06**	5.89e-06	**5.91e-06**
10	Quadrática	**4.00e-06**	6.03e-06	**6.08e-06**
	SELU	**5.37e-06**	6.33e-06	**6.39e-06**
	Sigmóide	5.20e-06	**6.11e-06**	**3.75e-06**

Continua na próxima página

Base de Dados	Função de Ativação	Normal	Camada	Neurônio
	SoftExponential	**5.13e-06**	6.19e-06	**8.11e-06**
	SoftPlus	**5.04e-06**	**6.06e-06**	6.05e-06
	SoftSign	**5.10e-06**	6.00e-06	**6.08e-06**
	Swish	**5.08e-06**	6.15e-06	**6.22e-06**
	Tangente Hiperbólica	**5.23e-06**	**6.64e-06**	6.62e-06

Fonte: Elaborada pelo autor (2023).

APÊNDICE B – COMPLEMENTO DO EXPERIMENTO II

Todas as tabelas e figuras apresentadas aqui utilizam um mapa de cores com brilho linearmente decrescente (ou crescente). Isso implica que as informações serão mantidas caso sejam impressas em preto e branco ou visualizadas por pessoas com deficiência de daltonismo.

B.1 Análise das Métricas de Desempenho e Visualização de Distribuição

As Tabelas 29, 30, 31, 32, 33, 34, 35, 36, 37 e 38 apresentam os resultados sumarizados do segundo experimento numérico. Para cada função de ativação e tipo adaptativo da ANN, exibimos os valores mínimo, médio, mediano e máximo das métricas R^2 e RMSE, juntamente com o desvio padrão para as 30 execuções independentes. Os valores marcados com a cor **"Azul Escuro"** representam os melhores resultados encontrados, enquanto os valores destacados em **"Azul Claro"** representam os piores resultados encontrados.

As Figuras 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101 e 102 representam os *Violin plots* das métricas R^2 e RMSE para os problemas do Experimento 2, considerando diferentes tipos adaptativos das funções de ativação em 30 execuções independentes. A escolha dos *Violin plots* se deve à sua capacidade de combinar as informações contidas em um *boxplot* com a representação de densidade.

Tabela 29 – Resultados do Experimento 2 para a Base de Dados 1 (1027_ESL).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	-0.688	0.049	0.319	0.098	0.734	0.646	1.205	0.211	1.191	1.629
	Camada	**-0.334**	**0.177**	**0.275**	**0.141**	**0.785**	**0.581**	**1.119**	**0.208**	**1.162**	**1.448**
	Neurônio	**-0.334**	**0.177**	**0.275**	**0.141**	**0.785**	**0.581**	**1.119**	**0.208**	**1.162**	**1.448**
Cúbico	Normal	-0.428	-0.061	**0.111**	-0.005	0.046	1.225	1.290	**0.065**	1.257	1.498
	Camada	**-0.014**	**0.309**	0.238	**0.276**	**0.728**	**0.654**	**1.025**	0.189	**1.066**	**1.262**
	Neurônio	**-0.014**	**0.309**	0.238	**0.276**	**0.728**	**0.654**	**1.025**	0.189	**1.066**	**1.262**
Linear	Normal	-0.554	0.064	0.315	0.119	0.772	0.599	1.195	**0.214**	1.177	1.563
	Camada	**-0.198**	**0.209**	**0.285**	**0.177**	**0.798**	**0.563**	**1.093**	0.226	**1.137**	**1.372**
	Neurônio	**-0.198**	**0.209**	**0.285**	**0.177**	**0.798**	**0.563**	**1.093**	0.226	**1.137**	**1.372**
Mish	Normal	-0.423	0.046	0.246	0.052	0.647	0.745	1.214	**0.166**	1.221	1.496
	Camada	**-0.140**	**0.175**	**0.235**	**0.098**	**0.707**	**0.678**	**1.125**	0.180	**1.191**	**1.339**
	Neurônio	**-0.140**	**0.175**	**0.235**	**0.098**	**0.707**	**0.678**	**1.125**	0.180	**1.191**	**1.339**
PELU	Normal	-0.515	0.059	0.289	0.103	0.742	0.637	1.201	0.197	1.187	1.543
	Camada	**-0.148**	**0.196**	**0.254**	**0.146**	**0.750**	**0.627**	**1.107**	**0.196**	**1.159**	**1.344**
	Neurônio	**-0.148**	**0.196**	**0.254**	**0.146**	**0.750**	**0.627**	**1.107**	**0.196**	**1.159**	**1.344**
PReLU	Normal	-0.434	0.031	0.232	0.014	0.607	0.786	1.225	**0.154**	1.245	1.501
	Camada	**-0.213**	**0.159**	**0.230**	**0.084**	**0.740**	**0.639**	**1.137**	0.174	**1.200**	**1.381**
	Neurônio	**-0.213**	**0.159**	**0.230**	**0.084**	**0.740**	**0.639**	**1.137**	0.174	**1.200**	**1.381**
Quadrática	Normal	-0.439	-0.047	**0.118**	-0.002	0.091	1.195	1.281	**0.070**	1.255	1.504
	Camada	**0.008**	**0.313**	0.246	**0.270**	**0.794**	**0.568**	**1.020**	0.200	**1.071**	**1.249**

Continua na próxima página

Tabela 29 – Resultados do Experimento 2 para a Base de Dados 1 (1027_ESL).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**0.008**	**0.313**	0.246	**0.270**	**0.794**	**0.568**	**1.020**	0.200	**1.071**	**1.249**
SELU	Normal	-0.655	0.060	0.330	0.123	0.744	0.634	1.196	0.219	1.174	1.613
	Camada	**-0.218**	**0.196**	**0.274**	**0.169**	**0.775**	**0.595**	**1.104**	**0.213**	**1.143**	**1.384**
	Neurônio	**-0.218**	**0.196**	**0.274**	**0.169**	**0.775**	**0.595**	**1.104**	**0.213**	**1.143**	**1.384**
Sigmóide	Normal	-0.370	-0.003	0.161	0.007	0.474	0.909	1.251	0.104	1.249	1.468
	Camada	**-0.041**	**0.099**	0.188	**0.029**	**0.762**	**0.612**	**1.181**	0.147	**1.235**	**1.279**
	Neurônio	**-0.041**	**0.099**	0.188	**0.029**	**0.762**	**0.612**	**1.181**	0.147	**1.235**	**1.279**
SoftExponential	Normal	-0.580	0.059	0.312	0.113	0.782	0.585	1.198	**0.213**	1.181	1.576
	Camada	**-0.304**	**0.187**	**0.287**	**0.166**	**0.796**	**0.566**	**1.110**	0.220	**1.145**	**1.431**
	Neurônio	**-0.304**	**0.187**	**0.287**	**0.166**	**0.796**	**0.566**	**1.110**	0.220	**1.145**	**1.431**
SoftPlus	Normal	-0.505	0.014	0.253	0.019	0.715	0.669	1.233	0.173	1.242	1.538
	Camada	**-0.101**	**0.136**	**0.214**	**0.083**	**0.720**	**0.664**	**1.154**	**0.164**	**1.201**	**1.316**
	Neurônio	**-0.101**	**0.136**	**0.214**	**0.083**	**0.720**	**0.664**	**1.154**	**0.164**	**1.201**	**1.316**
SoftSign	Normal	-0.495	0.046	**0.223**	0.072	0.669	0.721	1.215	**0.152**	1.208	1.533
	Camada	**-0.083**	**0.184**	0.240	**0.118**	**0.754**	**0.621**	**1.117**	0.187	**1.177**	**1.305**
	Neurônio	**-0.083**	**0.184**	0.240	**0.118**	**0.754**	**0.621**	**1.117**	0.187	**1.177**	**1.305**
Swish	Normal	-0.408	0.043	0.229	0.048	0.635	0.757	1.217	**0.155**	1.224	1.488
	Camada	**-0.116**	**0.170**	**0.228**	**0.089**	**0.703**	**0.684**	**1.129**	0.175	**1.197**	**1.325**
	Neurônio	**-0.116**	**0.170**	**0.228**	**0.089**	**0.703**	**0.684**	**1.129**	0.175	**1.197**	**1.325**
Tangente Hiperbólica	Normal	-0.533	0.060	0.267	0.102	0.695	0.692	1.202	**0.181**	1.188	1.552

Continua na próxima página

Tabela 29 – Resultados do Experimento 2 para a Base de Dados 1 (1027_ESL).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**-0.129**	**0.187**	**0.245**	**0.150**	**0.743**	**0.636**	**1.115**	0.188	**1.156**	**1.332**
	Neurônio	**-0.129**	**0.187**	**0.245**	**0.150**	**0.743**	**0.636**	**1.115**	0.188	**1.156**	**1.332**

Fonte: Elaborada pelo autor (2023).

Tabela 30 – Resultados do Experimento 2 para a Base de Dados 2 (1028_SWD).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	**-0.057**	0.119	**0.083**	0.115	0.286	0.682	0.756	**0.036**	0.759	**0.829**
	Camada	-0.063	**0.123**	0.086	**0.118**	0.289	0.680	**0.755**	0.037	**0.758**	0.832
	Neurônio	-0.063	**0.123**	0.086	0.116	**0.291**	**0.679**	**0.755**	0.037	0.759	0.832
Cúbico	Normal	**-0.007**	0.002	**0.008**	-0.000	0.039	0.791	0.806	**0.003**	0.807	**0.809**
	Camada	-0.019	0.115	0.086	0.104	0.266	0.691	0.758	0.037	0.764	0.815
	Neurônio	-0.044	**0.125**	0.086	**0.116**	**0.300**	**0.675**	**0.754**	0.037	**0.759**	0.824
Linear	Normal	**-0.046**	**0.125**	**0.086**	**0.124**	**0.289**	**0.680**	**0.754**	**0.037**	**0.755**	**0.825**
	Camada	-0.049	**0.125**	0.087	0.117	0.283	0.683	**0.754**	0.038	0.758	0.826
	Neurônio	-0.056	**0.125**	0.087	0.116	0.285	0.682	**0.754**	**0.037**	0.759	0.829
Mish	Normal	**-0.053**	0.103	**0.073**	0.098	0.257	0.695	0.763	**0.031**	0.766	**0.828**
	Camada	-0.055	**0.115**	0.081	**0.109**	**0.285**	**0.682**	**0.758**	0.035	**0.762**	0.829
	Neurônio	-0.055	0.114	0.078	0.105	0.279	0.685	0.759	0.034	0.763	0.829
PELU	Normal	**-0.054**	**0.119**	**0.082**	**0.117**	0.282	0.684	**0.756**	0.036	**0.758**	**0.828**
	Camada	-0.060	**0.119**	0.085	0.114	**0.297**	**0.676**	**0.756**	0.037	0.760	0.831
	Neurônio	-0.061	0.118	**0.082**	0.110	0.296	0.677	0.757	**0.035**	0.761	0.831
PReLU	Normal	-0.063	0.080	**0.067**	0.060	0.242	0.703	0.773	**0.029**	0.782	0.832
	Camada	-0.063	0.101	0.085	**0.071**	**0.285**	**0.682**	0.764	0.037	**0.778**	0.832
	Neurônio	**-0.058**	**0.103**	0.085	**0.071**	0.283	0.683	**0.763**	0.037	**0.778**	**0.830**
Quadrática	Normal	**-0.011**	0.005	**0.011**	0.003	0.033	0.794	0.805	**0.005**	0.806	**0.811**
	Camada	-0.015	0.117	0.089	0.110	0.270	0.689	0.757	0.038	0.761	0.813

Continua na próxima página

Tabela 30 – Resultados do Experimento 2 para a Base de Dados 2 (1028_SWD).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
Tangente Hiperbólica	Neurônio	-0.051	**0.129**	0.089	**0.126**	**0.297**	**0.677**	**0.752**	0.039	**0.754**	0.827
	Normal	**-0.045**	**0.124**	**0.084**	**0.126**	**0.289**	**0.680**	**0.754**	**0.037**	**0.754**	**0.825**
SELU	Camada	-0.059	0.122	0.086	0.118	0.285	0.682	0.755	**0.037**	0.758	0.830
	Neurônio	-0.059	0.122	0.085	0.120	0.285	0.682	0.755	**0.037**	0.757	0.830
	Normal	-0.055	0.026	**0.044**	0.030	0.165	0.737	0.796	**0.018**	0.795	0.829
Sigmóide	Camada	-0.043	0.049	0.077	0.009	0.253	**0.697**	0.786	0.033	0.803	**0.824**
	Neurônio	**-0.042**	**0.062**	0.078	**0.038**	**0.254**	**0.697**	**0.781**	0.033	**0.791**	**0.824**
	Normal	**-0.056**	0.124	**0.087**	**0.118**	**0.289**	**0.680**	**0.754**	**0.038**	**0.758**	**0.829**
SoftExponential	Camada	-0.059	0.124	0.088	0.116	0.284	0.683	**0.754**	**0.038**	**0.758**	0.830
	Neurônio	-0.061	**0.125**	0.088	**0.118**	0.284	0.683	**0.754**	**0.038**	**0.758**	0.831
	Normal	-0.066	0.056	**0.073**	0.053	0.248	0.700	0.783	**0.031**	0.785	0.833
SoftPlus	Camada	**-0.052**	0.064	0.084	0.038	**0.270**	**0.690**	0.780	0.036	0.791	**0.828**
	Neurônio	-0.055	**0.075**	0.081	**0.060**	0.267	0.691	**0.775**	0.035	**0.782**	0.829
	Normal	**-0.046**	0.107	**0.073**	0.107	0.262	0.693	0.762	**0.031**	0.762	**0.825**
SoftSign	Camada	-0.059	**0.118**	0.085	**0.115**	**0.277**	**0.686**	**0.757**	0.037	**0.759**	0.830
	Neurônio	-0.059	**0.118**	0.082	0.109	0.271	0.689	**0.757**	0.036	0.761	0.830
	Normal	**-0.049**	0.100	**0.071**	0.095	0.254	0.697	0.765	**0.031**	0.768	**0.826**
Swish	Camada	-0.052	**0.113**	0.080	**0.108**	**0.279**	**0.685**	**0.759**	0.035	**0.762**	0.828
	Neurônio	-0.053	0.112	0.077	0.104	0.273	0.688	0.760	0.033	0.764	0.828
	Normal	**-0.049**	**0.118**	**0.079**	**0.119**	**0.266**	**0.691**	**0.757**	**0.034**	**0.757**	**0.826**

Continua na próxima página

Tabela 30 – Resultados do Experimento 2 para a Base de Dados 2 (1028_SWD).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	-0.059	**0.118**	**0.083**	0.118	**0.266**	**0.691**	**0.757**	**0.036**	0.758	0.830
	Neurônio	**-0.060**	**0.118**	0.081	**0.114**	**0.265**	**0.692**	**0.757**	0.035	**0.759**	**0.831**

Fonte: Elaborada pelo autor (2023).

Tabela 31 – Resultados do Experimento 2 para a Base de Dados 3 (1029_LEV).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	-0.085	0.028	**0.107**	-0.015	0.370	0.734	0.909	**0.053**	0.931	0.962
	Camada	**-0.052**	**0.035**	0.122	**-0.014**	**0.385**	**0.725**	**0.905**	0.062	**0.930**	**0.948**
	Neurônio	**-0.052**	**0.035**	0.122	**-0.014**	**0.385**	**0.725**	**0.905**	0.062	**0.930**	**0.948**
Cúbico	Normal	-0.092	-0.032	**0.019**	-0.032	0.001	0.924	0.938	**0.009**	0.939	0.966
	Camada	**-0.037**	**0.056**	0.130	**-0.003**	**0.390**	**0.722**	**0.895**	0.066	**0.925**	**0.941**
	Neurônio	**-0.037**	**0.056**	0.130	**-0.003**	**0.390**	**0.722**	**0.895**	0.066	**0.925**	**0.941**
Linear	Normal	-0.078	0.032	**0.109**	**-0.009**	0.386	**0.724**	0.908	**0.054**	**0.928**	0.959
	Camada	**-0.043**	**0.034**	0.121	-0.018	**0.387**	**0.724**	**0.906**	0.061	0.932	**0.944**
	Neurônio	**-0.043**	**0.034**	0.121	-0.018	**0.387**	**0.724**	**0.906**	0.061	0.932	**0.944**
Mish	Normal	-0.090	0.020	**0.098**	**-0.022**	0.336	0.753	0.914	**0.048**	**0.934**	0.964
	Camada	**-0.047**	**0.032**	0.120	-0.024	**0.386**	**0.724**	**0.907**	0.060	0.935	**0.946**
	Neurônio	**-0.047**	**0.032**	0.120	-0.024	**0.386**	**0.724**	**0.907**	0.060	0.935	**0.946**
PELU	Normal	-0.084	0.027	**0.106**	**-0.014**	0.369	0.734	0.910	**0.052**	**0.930**	0.962
	Camada	**-0.042**	**0.031**	0.118	-0.021	**0.375**	**0.730**	**0.908**	0.059	0.933	**0.943**
	Neurônio	**-0.042**	**0.031**	0.118	-0.021	**0.375**	**0.730**	**0.908**	0.059	0.933	**0.943**
PReLU	Normal	-0.097	0.006	**0.086**	-0.030	0.308	0.769	0.920	**0.042**	0.938	0.968
	Camada	**-0.061**	**0.029**	0.118	**-0.024**	**0.379**	**0.728**	**0.908**	0.059	**0.935**	**0.952**
	Neurônio	**-0.061**	**0.029**	0.118	**-0.024**	**0.379**	**0.728**	**0.908**	0.059	**0.935**	**0.952**
Quadrática	Normal	-0.084	-0.029	**0.019**	-0.031	0.018	0.915	0.937	**0.008**	0.938	0.962
	Camada	**-0.039**	**0.054**	0.133	**-0.009**	**0.395**	**0.719**	**0.896**	0.067	**0.928**	**0.942**

Continua na próxima página

Tabela 31 – Resultados do Experimento 2 para a Base de Dados 3 (1029_LEV).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**-0.039**	**0.054**	0.133	**-0.009**	**0.395**	**0.719**	**0.896**	0.067	**0.928**	**0.942**
SELU	Normal	-0.073	0.031	**0.109**	**-0.008**	**0.376**	**0.730**	0.908	**0.054**	**0.927**	0.957
	Camada	**-0.045**	**0.034**	0.118	-0.015	0.373	0.731	**0.906**	0.059	0.931	**0.944**
	Neurônio	**-0.045**	**0.034**	0.118	-0.015	0.373	0.731	**0.906**	0.059	0.931	**0.944**
Sigmóide	Normal	-0.099	-0.024	**0.054**	**-0.032**	0.235	0.808	0.935	**0.026**	**0.938**	0.969
	Camada	**-0.040**	**0.000**	0.085	**-0.032**	**0.371**	**0.733**	**0.923**	0.043	0.939	**0.942**
	Neurônio	**-0.040**	**0.000**	0.085	**-0.032**	**0.371**	**0.733**	**0.923**	0.043	0.939	**0.942**
SoftExponential	Normal	-0.086	0.029	**0.106**	**-0.012**	**0.384**	**0.725**	0.909	**0.053**	**0.929**	0.963
	Camada	**-0.051**	**0.035**	0.119	-0.017	**0.384**	**0.725**	**0.906**	0.060	0.932	**0.947**
	Neurônio	**-0.051**	**0.035**	0.119	-0.017	**0.384**	**0.725**	**0.906**	0.060	0.932	**0.947**
SoftPlus	Normal	-0.121	-0.015	**0.080**	**-0.031**	0.353	0.743	0.930	**0.039**	**0.938**	0.978
	Camada	**-0.043**	**0.015**	0.097	**-0.031**	**0.365**	**0.736**	**0.916**	0.048	**0.938**	**0.944**
	Neurônio	**-0.043**	**0.015**	0.097	**-0.031**	**0.365**	**0.736**	**0.916**	0.048	**0.938**	**0.944**
SoftSign	Normal	-0.089	0.021	**0.101**	**-0.021**	**0.372**	**0.732**	0.913	**0.050**	**0.934**	0.964
	Camada	**-0.040**	**0.031**	0.117	-0.022	0.371	0.733	**0.908**	0.059	**0.934**	**0.942**
	Neurônio	**-0.040**	**0.031**	0.117	-0.022	0.371	0.733	**0.908**	0.059	**0.934**	**0.942**
Swish	Normal	-0.091	0.018	**0.095**	**-0.021**	0.336	0.753	0.914	**0.047**	**0.933**	0.965
	Camada	**-0.046**	**0.032**	0.120	-0.024	**0.386**	**0.724**	**0.907**	0.060	0.935	**0.945**
	Neurônio	**-0.046**	**0.032**	0.120	-0.024	**0.386**	**0.724**	**0.907**	0.060	0.935	**0.945**
Tangente Hiperbólica	Normal	-0.079	0.027	**0.104**	**-0.013**	**0.366**	**0.736**	0.910	**0.052**	**0.930**	0.960

Continua na próxima página

Tabela 31 – Resultados do Experimento 2 para a Base de Dados 3 (1029_LEV).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**-0.041**	**0.031**	0.114	-0.019	0.365	**0.736**	**0.908**	0.057	0.933	**0.942**
	Neurônio	**-0.041**	**0.031**	0.114	-0.019	0.365	**0.736**	**0.908**	0.057	0.933	**0.942**

Fonte: Elaborada pelo autor (2023).

Tabela 32 – Resultados do Experimento 2 para a Base de Dados 4 (1030_ERA).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	-0.099	0.038	**0.077**	0.008	0.275	1.733	1.994	**0.082**	2.026	2.133
	Camada	**-0.026**	**0.048**	0.085	**0.010**	**0.303**	**1.699**	**1.983**	0.093	**2.025**	**2.061**
	Neurônio	**-0.026**	**0.048**	0.085	**0.010**	**0.303**	**1.699**	**1.983**	0.093	**2.025**	**2.061**
Cúbico	Normal	-0.028	-0.002	**0.011**	-0.002	0.038	1.996	2.036	**0.011**	2.036	2.063
	Camada	**-0.004**	**0.083**	0.101	**0.043**	**0.338**	**1.656**	**1.946**	0.112	**1.991**	**2.039**
	Neurônio	**-0.004**	**0.083**	0.101	**0.043**	**0.338**	**1.656**	**1.946**	0.112	**1.991**	**2.039**
Linear	Normal	-0.071	0.041	**0.082**	0.009	0.332	1.663	1.991	**0.089**	2.026	2.106
	Camada	**-0.016**	**0.050**	0.092	**0.015**	**0.333**	**1.662**	**1.981**	0.101	**2.020**	**2.051**
	Neurônio	**-0.016**	**0.050**	0.092	**0.015**	**0.333**	**1.662**	**1.981**	0.101	**2.020**	**2.051**
Mish	Normal	-0.067	0.030	**0.062**	0.004	0.193	1.828	2.003	**0.066**	2.031	2.102
	Camada	**-0.016**	**0.044**	0.079	**0.006**	**0.252**	**1.759**	**1.988**	0.085	**2.029**	**2.051**
	Neurônio	**-0.016**	**0.044**	0.079	**0.006**	**0.252**	**1.759**	**1.988**	0.085	**2.029**	**2.051**
PELU	Normal	-0.071	0.036	**0.072**	**0.008**	0.264	1.745	1.996	**0.077**	**2.027**	2.106
	Camada	**-0.015**	**0.045**	0.082	0.006	**0.265**	**1.744**	**1.986**	0.089	2.028	**2.050**
	Neurônio	**-0.015**	**0.045**	0.082	0.006	**0.265**	**1.744**	**1.986**	0.089	2.028	**2.050**
PReLU	Normal	-0.075	0.019	**0.051**	0.003	0.151	1.875	2.014	**0.053**	2.032	2.110
	Camada	**-0.026**	**0.041**	0.079	**0.008**	**0.274**	**1.734**	**1.991**	0.086	**2.026**	**2.061**
	Neurônio	**-0.026**	**0.041**	0.079	**0.008**	**0.274**	**1.734**	**1.991**	0.086	**2.026**	**2.061**
Quadrática	Normal	-0.057	-0.005	**0.018**	-0.002	0.026	2.008	2.039	**0.018**	2.036	2.092
	Camada	**-0.005**	**0.076**	0.099	**0.027**	**0.338**	**1.656**	**1.953**	0.110	**2.007**	**2.039**

Continua na próxima página

Tabela 32 – Resultados do Experimento 2 para a Base de Dados 4 (1030_ERA).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**-0.005**	**0.076**	0.099	**0.027**	**0.338**	**1.656**	**1.953**	0.110	**2.007**	**2.039**
SELU	Normal	-0.072	0.038	**0.078**	**0.008**	**0.290**	**1.714**	1.994	**0.084**	2.027	2.107
	Camada	**-0.017**	**0.045**	0.084	**0.008**	0.287	1.718	**1.986**	0.091	**2.026**	**2.052**
	Neurônio	**-0.017**	**0.045**	0.084	**0.008**	0.287	1.718	**1.986**	0.091	**2.026**	**2.052**
Sigmóide	Normal	-0.059	0.004	**0.040**	**0.000**	0.194	1.827	2.030	**0.042**	**2.034**	2.093
	Camada	**-0.008**	**0.019**	0.059	-0.002	**0.293**	**1.711**	**2.014**	0.064	2.036	**2.043**
	Neurônio	**-0.008**	**0.019**	0.059	-0.002	**0.293**	**1.711**	**2.014**	0.064	2.036	**2.043**
SoftExponential	Normal	-0.094	0.038	**0.079**	**0.012**	**0.323**	**1.675**	1.994	**0.086**	**2.022**	2.128
	Camada	**-0.024**	**0.048**	0.086	0.011	0.314	1.685	**1.983**	0.094	2.024	**2.059**
	Neurônio	**-0.024**	**0.048**	0.086	0.011	0.314	1.685	**1.983**	0.094	2.024	**2.059**
SoftPlus	Normal	-0.128	0.006	**0.056**	-0.001	0.246	1.766	2.028	**0.060**	2.035	2.161
	Camada	**-0.013**	**0.027**	0.063	**0.001**	**0.261**	**1.749**	**2.006**	0.067	**2.034**	**2.048**
	Neurônio	**-0.013**	**0.027**	0.063	**0.001**	**0.261**	**1.749**	**2.006**	0.067	**2.034**	**2.048**
SoftSign	Normal	-0.040	0.030	**0.065**	**0.006**	0.277	1.731	2.003	**0.070**	**2.029**	2.075
	Camada	**-0.008**	**0.042**	0.079	0.005	**0.292**	**1.712**	**1.990**	0.085	2.030	**2.043**
	Neurônio	**-0.008**	**0.042**	0.079	0.005	**0.292**	**1.712**	**1.990**	0.085	2.030	**2.043**
Swish	Normal	-0.062	0.029	**0.061**	**0.005**	0.196	1.825	2.004	**0.064**	2.030	2.096
	Camada	**-0.014**	**0.043**	0.078	**0.005**	**0.250**	**1.762**	**1.988**	0.085	**2.029**	**2.049**
	Neurônio	**-0.014**	**0.043**	0.078	**0.005**	**0.250**	**1.762**	**1.988**	0.085	**2.029**	**2.049**
Tangente Hiperbólica	Normal	-0.046	0.034	**0.070**	**0.008**	0.280	1.727	1.998	**0.075**	**2.026**	2.081

Continua na próxima página

Tabela 32 – Resultados do Experimento 2 para a Base de Dados 4 (1030_ERA).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**-0.011**	**0.042**	0.077	**0.008**	**0.283**	**1.723**	**1.990**	0.084	2.027	**2.046**
	Neurônio	**-0.011**	**0.042**	0.077	**0.008**	**0.283**	**1.723**	**1.990**	0.084	2.027	**2.046**

Fonte: Elaborada pelo autor (2023).

Tabela 33 – Resultados do Experimento 2 para a Base de Dados 5 (1089_USCrime).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	**-1.635**	**-0.263**	**0.485**	**-0.204**	**0.490**	**24.857**	**38.440**	**7.322**	**38.187**	**56.490**
	Camada	-1.489	**-0.227**	**0.454**	-0.143	**0.487**	**24.929**	**37.924**	**7.011**	37.201	54.903
	Neurônio	**-1.474**	**-0.227**	**0.454**	**-0.138**	0.489	24.883	37.931	7.037	**37.124**	**54.744**
Cúbico	Normal	**-0.402**	**-0.118**	**0.127**	**-0.053**	**0.016**	**34.515**	**36.739**	**2.037**	**35.718**	**41.205**
	Camada	-0.292	-0.048	0.150	-0.047	0.327	28.551	35.542	2.615	35.612	39.556
	Neurônio	**-0.285**	**-0.024**	**0.154**	**-0.010**	**0.341**	**28.257**	**35.119**	**2.695**	**34.970**	**39.458**
Linear	Normal	**-1.411**	**-0.218**	**0.433**	**-0.130**	**0.469**	**25.360**	**37.835**	**6.738**	**36.990**	**54.035**
	Camada	-1.260	-0.171	**0.402**	-0.117	0.527	23.927	37.126	**6.476**	36.780	52.324
	Neurônio	**-1.245**	**-0.164**	0.410	**-0.114**	**0.541**	**23.585**	**36.974**	6.643	**36.736**	**52.142**
Mish	Normal	**-1.104**	**-0.166**	**0.325**	**-0.133**	**0.324**	**28.622**	**37.244**	**5.057**	**37.048**	**50.475**
	Camada	**-0.979**	-0.134	**0.298**	-0.125	0.343	28.211	36.759	**4.763**	36.912	48.961
	Neurônio	**-0.979**	**-0.130**	0.302	**-0.123**	**0.345**	**28.168**	**36.684**	4.830	**36.885**	**48.954**
PELU	Normal	**-1.323**	**-0.191**	**0.396**	**-0.149**	**0.444**	**25.958**	**37.496**	**6.160**	**37.300**	**53.042**
	Camada	**-1.143**	-0.142	**0.356**	-0.130	**0.489**	24.874	36.748	**5.773**	36.987	**50.946**
	Neurônio	-1.144	**-0.132**	0.364	**-0.121**	**0.489**	**24.873**	**36.564**	5.924	**36.846**	50.961
PReLU	Normal	**-1.097**	**-0.195**	**0.317**	**-0.146**	**0.223**	**30.675**	**37.748**	**4.831**	**37.260**	**50.396**
	Camada	-0.989	-0.165	**0.295**	**-0.150**	0.256	30.026	37.290	**4.602**	**37.320**	49.081
	Neurônio	**-0.980**	**-0.158**	0.297	-0.149	**0.298**	**29.166**	**37.163**	4.661	37.299	**48.972**
Quadrática	Normal	**-0.435**	**-0.125**	**0.137**	-0.068	**0.050**	**33.912**	**36.850**	**2.205**	35.958	**41.694**
	Camada	-0.367	-0.089	**0.114**	**-0.072**	0.143	32.214	36.270	**1.881**	**36.035**	40.691

Continua na próxima página

Tabela 33 – Resultados do Experimento 2 para a Base de Dados 5 (1089_USCrime).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**-0.356**	**-0.078**	0.121	**-0.037**	**0.161**	**31.879**	**36.073**	2.003	**35.443**	**40.522**
SELU	Normal	**-1.528**	**-0.236**	**0.464**	-0.138	**0.485**	**24.986**	**38.043**	**7.167**	37.119	**55.337**
	Camada	-1.360	-0.185	**0.424**	**-0.141**	0.569	22.850	37.290	**6.805**	**37.177**	53.462
	Neurônio	**-1.350**	**-0.175**	0.428	**-0.132**	**0.571**	**22.788**	**37.105**	6.908	**37.023**	**53.345**
Sigmóide	Normal	**-0.700**	**-0.194**	**0.201**	-0.150	**0.072**	**33.533**	**37.909**	**3.094**	37.313	**45.370**
	Camada	-0.591	-0.167	0.171	**-0.156**	**0.095**	**33.101**	37.509	2.685	**37.413**	43.901
	Neurônio	**-0.580**	**-0.164**	**0.169**	**-0.145**	0.093	33.135	**37.452**	**2.667**	**37.235**	**43.751**
SoftExponential	Normal	**-1.502**	**-0.234**	**0.462**	**-0.150**	**0.490**	**24.850**	**38.025**	**7.090**	**37.321**	**55.043**
	Camada	-1.367	-0.200	**0.440**	**-0.119**	0.503	24.530	37.507	**6.939**	**36.808**	53.545
	Neurônio	**-1.351**	**-0.199**	0.445	-0.124	**0.512**	**24.300**	**37.471**	7.056	36.890	**53.364**
SoftPlus	Normal	**-1.160**	**-0.300**	**0.334**	**-0.223**	**0.168**	**31.747**	**39.386**	**4.927**	**38.482**	**51.152**
	Camada	-0.944	-0.234	**0.288**	**-0.189**	0.247	30.208	38.408	**4.424**	**37.948**	48.522
	Neurônio	**-0.930**	**-0.228**	0.290	-0.196	**0.259**	**29.958**	**38.317**	4.480	38.062	**48.343**
SoftSign	Normal	**-0.854**	**-0.119**	**0.255**	**-0.089**	**0.296**	**29.196**	**36.591**	**4.100**	**36.315**	**47.390**
	Camada	-0.768	-0.089	**0.244**	**-0.086**	**0.352**	**28.012**	36.105	**4.026**	**36.274**	46.271
	Neurônio	**-0.767**	**-0.083**	0.246	-0.088	0.351	28.027	**35.993**	4.068	36.306	**46.261**
Swish	Normal	**-0.980**	**-0.149**	**0.291**	**-0.117**	**0.296**	**29.199**	**37.032**	**4.564**	**36.785**	**48.966**
	Camada	**-0.870**	-0.121	**0.268**	-0.114	0.315	28.803	36.597	**4.321**	36.726	**47.596**
	Neurônio	-0.871	**-0.118**	0.271	**-0.112**	**0.316**	**28.773**	**36.546**	4.360	**36.704**	47.599
Tangente Hiperbólica	Normal	**-1.127**	**-0.149**	**0.338**	**-0.136**	**0.387**	**27.243**	**36.934**	**5.361**	**37.087**	**50.752**

Continua na próxima página

Tabela 33 – Resultados do Experimento 2 para a Base de Dados 5 (1089_USCrime).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	-1.008	-0.113	**0.316**	**-0.106**	**0.439**	**26.066**	36.361	**5.188**	**36.607**	49.315
	Neurônio	**-1.005**	**-0.105**	0.320	-0.111	0.438	26.092	**36.223**	5.258	36.685	**49.284**

Fonte: Elaborada pelo autor (2023).

Tabela 34 – Resultados do Experimento 2 para a Base de Dados 6 (1096_FacultySalaries).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	-1.741	-0.371	0.553	-0.210	0.499	3.037	4.935	0.970	4.722	7.107
	Camada	**-1.343**	**-0.257**	**0.466**	**-0.126**	**0.535**	**2.927**	**4.738**	**0.864**	**4.556**	**6.570**
	Neurônio	**-1.343**	**-0.257**	**0.466**	**-0.126**	**0.535**	**2.927**	**4.738**	**0.864**	**4.556**	**6.570**
Cúbico	Normal	-1.051	-0.262	0.300	-0.136	-0.000	4.293	4.794	0.544	4.576	6.148
	Camada	**-0.571**	**-0.104**	**0.187**	**-0.029**	**0.133**	**3.998**	**4.496**	**0.370**	**4.355**	**5.380**
	Neurônio	**-0.571**	**-0.104**	**0.187**	**-0.029**	**0.133**	**3.998**	**4.496**	**0.370**	**4.355**	**5.380**
Linear	Normal	-1.455	-0.333	0.483	-0.229	0.519	2.978	4.880	0.881	4.759	6.726
	Camada	**-1.032**	**-0.189**	**0.388**	**-0.094**	**0.579**	**2.784**	**4.618**	**0.769**	**4.490**	**6.120**
	Neurônio	**-1.032**	**-0.189**	**0.388**	**-0.094**	**0.579**	**2.784**	**4.618**	**0.769**	**4.490**	**6.120**
Mish	Normal	-1.377	-0.308	0.411	-0.175	0.313	3.559	4.855	0.739	4.653	6.618
	Camada	**-1.020**	**-0.200**	**0.325**	**-0.112**	**0.361**	**3.431**	**4.663**	**0.620**	**4.527**	**6.101**
	Neurônio	**-1.020**	**-0.200**	**0.325**	**-0.112**	**0.361**	**3.431**	**4.663**	**0.620**	**4.527**	**6.101**
PELU	Normal	-1.460	-0.318	0.460	-0.198	0.448	3.190	4.859	0.834	4.699	6.733
	Camada	**-1.055**	**-0.191**	**0.368**	**-0.130**	**0.535**	**2.927**	**4.631**	**0.721**	**4.563**	**6.154**
	Neurônio	**-1.055**	**-0.191**	**0.368**	**-0.130**	**0.535**	**2.927**	**4.631**	**0.721**	**4.563**	**6.154**
PReLU	Normal	-1.498	-0.318	0.417	-0.192	0.256	3.703	4.876	0.735	4.687	6.785
	Camada	**-1.165**	**-0.219**	**0.340**	**-0.118**	**0.304**	**3.582**	**4.699**	**0.629**	**4.539**	**6.316**
	Neurônio	**-1.165**	**-0.219**	**0.340**	**-0.118**	**0.304**	**3.582**	**4.699**	**0.629**	**4.539**	**6.316**
Quadrática	Normal	-1.077	-0.272	0.309	-0.130	0.002	4.288	4.809	0.560	4.563	6.187
	Camada	**-0.639**	**-0.127**	**0.189**	**-0.040**	**0.083**	**4.110**	**4.543**	**0.367**	**4.377**	**5.495**

Continua na próxima página

Tabela 34 – Resultados do Experimento 2 para a Base de Dados 6 (1096_FacultySalaries).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
SELU	Neurônio	**-0.639**	**-0.127**	**0.189**	**-0.040**	**0.083**	**4.110**	**4.543**	**0.367**	**4.377**	**5.495**
	Normal	-1.485	-0.348	0.528	-0.186	0.576	2.795	4.893	0.958	4.674	6.767
	Camada	**-1.120**	**-0.217**	**0.436**	**-0.102**	**0.649**	**2.543**	**4.661**	**0.856**	**4.506**	**6.251**
	Neurônio	**-1.120**	**-0.217**	**0.436**	**-0.102**	**0.649**	**2.543**	**4.661**	**0.856**	**4.506**	**6.251**
Sigmóide	Normal	-1.489	-0.272	0.423	-0.142	0.215	3.803	4.787	0.736	4.588	6.773
	Camada	**-1.065**	**-0.170**	**0.311**	**-0.073**	**0.260**	**3.693**	**4.609**	**0.575**	**4.446**	**6.169**
	Neurônio	**-1.065**	**-0.170**	**0.311**	**-0.073**	**0.260**	**3.693**	**4.609**	**0.575**	**4.446**	**6.169**
SoftExponential	Normal	-1.630	-0.338	0.510	-0.223	0.517	2.985	4.883	0.920	4.748	6.962
	Camada	**-1.186**	**-0.210**	**0.419**	**-0.102**	**0.549**	**2.883**	**4.654**	**0.810**	**4.506**	**6.347**
	Neurônio	**-1.186**	**-0.210**	**0.419**	**-0.102**	**0.549**	**2.883**	**4.654**	**0.810**	**4.506**	**6.347**
SoftPlus	Normal	-2.115	-0.374	0.615	-0.113	0.311	3.564	4.932	1.016	4.529	7.577
	Camada	**-1.521**	**-0.233**	**0.454**	**-0.081**	**0.335**	**3.500**	**4.699**	**0.811**	**4.463**	**6.816**
	Neurônio	**-1.521**	**-0.233**	**0.454**	**-0.081**	**0.335**	**3.500**	**4.699**	**0.811**	**4.463**	**6.816**
SoftSign	Normal	-1.090	-0.252	0.345	-0.161	0.330	3.513	4.761	0.644	4.626	6.207
	Camada	**-0.752**	**-0.148**	**0.270**	**-0.103**	**0.392**	**3.347**	**4.569**	**0.538**	**4.508**	**5.682**
	Neurônio	**-0.752**	**-0.148**	**0.270**	**-0.103**	**0.392**	**3.347**	**4.569**	**0.538**	**4.508**	**5.682**
Swish	Normal	-1.280	-0.295	0.383	-0.157	0.269	3.671	4.838	0.694	4.618	6.482
	Camada	**-0.941**	**-0.191**	**0.298**	**-0.104**	**0.311**	**3.563**	**4.651**	**0.571**	**4.510**	**5.981**
	Neurônio	**-0.941**	**-0.191**	**0.298**	**-0.104**	**0.311**	**3.563**	**4.651**	**0.571**	**4.510**	**5.981**
Tangente Hiperbólica	Normal	-1.211	-0.280	0.408	-0.196	0.458	3.160	4.799	0.761	4.694	6.383

Continua na próxima página

Tabela 34 – Resultados do Experimento 2 para a Base de Dados 6 (1096_FacultySalaries).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	**-0.886**	**-0.168**	**0.331**	**-0.124**	**0.529**	**2.947**	**4.592**	**0.665**	**4.551**	**5.895**
	Neurônio	**-0.886**	**-0.168**	**0.331**	**-0.124**	**0.529**	**2.947**	**4.592**	**0.665**	**4.551**	**5.895**

Fonte: Elaborada pelo autor (2023).

Tabela 35 – Resultados do Experimento 2 para a Base de Dados 7 (294_satellite_image).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	**0.581**	**0.700**	**0.042**	**0.697**	**0.779**	**1.036**	**1.206**	**0.084**	**1.215**	**1.428**
	Camada	0.664	0.724	0.031	0.726	0.781	1.031	1.156	0.066	1.155	1.277
	Neurônio	**0.735**	**0.763**	**0.017**	**0.768**	**0.797**	**0.994**	**1.072**	**0.039**	**1.063**	**1.136**
Cúbico	Normal	**0.071**	**0.142**	**0.044**	**0.146**	**0.247**	**1.913**	**2.042**	**0.052**	**2.038**	**2.125**
	Camada	0.706	0.739	0.017	0.738	0.779	1.037	1.126	0.037	1.129	1.197
	Neurônio	**0.749**	**0.781**	**0.015**	**0.780**	**0.810**	**0.961**	**1.031**	**0.034**	**1.033**	**1.104**
Linear	Normal	**0.647**	**0.678**	**0.014**	**0.682**	**0.699**	**1.211**	**1.251**	**0.027**	**1.244**	**1.310**
	Camada	0.649	**0.679**	**0.014**	**0.682**	**0.700**	**1.209**	**1.249**	**0.027**	**1.243**	1.306
	Neurônio	**0.650**	**0.679**	**0.014**	**0.682**	**0.699**	**1.209**	**1.249**	**0.027**	**1.243**	**1.305**
Mish	Normal	**0.597**	**0.704**	**0.042**	**0.712**	**0.776**	**1.043**	**1.197**	**0.084**	**1.183**	**1.401**
	Camada	0.604	0.719	0.035	0.718	0.781	1.032	1.168	0.070	1.172	1.387
	Neurônio	**0.704**	**0.754**	**0.019**	**0.754**	**0.791**	**1.008**	**1.093**	**0.043**	**1.093**	**1.200**
PELU	Normal	**0.639**	**0.709**	0.032	**0.710**	**0.783**	**1.028**	**1.187**	0.067	**1.188**	**1.326**
	Camada	0.646	0.723	**0.034**	0.723	0.787	1.018	1.158	**0.071**	1.160	1.313
	Neurônio	**0.665**	**0.737**	**0.028**	**0.745**	**0.789**	**1.013**	**1.130**	**0.059**	**1.113**	**1.277**
PReLU	Normal	**0.470**	**0.678**	**0.067**	**0.704**	**0.753**	**1.096**	**1.246**	**0.123**	**1.200**	**1.606**
	Camada	0.654	0.719	0.031	0.720	0.779	1.036	1.167	0.065	1.167	1.297
	Neurônio	**0.725**	**0.758**	**0.019**	**0.761**	**0.792**	**1.006**	**1.083**	**0.042**	**1.078**	**1.157**
Quadrática	Normal	**0.137**	**0.278**	**0.071**	**0.273**	**0.449**	**1.637**	**1.872**	**0.093**	**1.880**	**2.048**
	Camada	0.689	0.717	0.019	0.714	0.752	1.098	1.172	0.039	1.179	1.230

Continua na próxima página

Tabela 35 – Resultados do Experimento 2 para a Base de Dados 7 (294_satellite_image).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**0.725**	**0.755**	**0.016**	**0.755**	**0.785**	**1.022**	**1.092**	**0.035**	**1.093**	**1.157**
SELU	Normal	0.657	0.705	0.019	0.705	0.756	1.089	1.196	**0.038**	1.198	1.292
	Camada	0.699	0.735	0.023	0.731	0.785	1.021	1.134	0.050	1.144	1.210
	Neurônio	**0.719**	**0.762**	**0.018**	**0.763**	**0.797**	**0.994**	**1.076**	0.040	**1.075**	**1.168**
Sigmóide	Normal	0.570	0.646	0.030	0.655	0.703	1.203	1.310	0.055	1.295	1.447
	Camada	**0.696**	0.727	0.019	**0.730**	**0.761**	**1.078**	1.151	0.040	**1.146**	**1.217**
	Neurônio	0.694	**0.729**	**0.017**	0.728	0.759	1.082	**1.148**	**0.037**	1.149	1.219
SoftExponential	Normal	0.625	0.680	**0.019**	0.684	0.722	1.162	1.247	**0.037**	1.240	1.351
	Camada	0.672	0.706	0.026	0.698	0.773	1.051	1.194	0.054	1.211	1.263
	Neurônio	**0.702**	**0.732**	0.021	**0.726**	**0.780**	**1.033**	**1.141**	0.045	**1.154**	**1.203**
SoftPlus	Normal	0.561	0.673	0.043	0.677	0.767	1.065	1.258	0.082	1.253	1.462
	Camada	0.586	0.700	0.042	0.697	0.777	1.040	1.204	0.084	1.215	1.419
	Neurônio	**0.609**	**0.708**	**0.037**	**0.712**	**0.780**	**1.035**	**1.189**	**0.075**	**1.184**	**1.379**
SoftSign	Normal	0.678	0.722	**0.017**	0.724	0.746	1.111	1.162	**0.035**	1.158	1.252
	Camada	0.703	0.734	0.019	0.735	0.768	1.063	1.138	0.041	1.135	1.202
	Neurônio	**0.710**	**0.743**	**0.017**	**0.744**	**0.770**	**1.059**	**1.118**	0.036	**1.115**	**1.188**
Swish	Normal	0.585	0.700	0.043	0.707	0.776	1.045	1.205	0.085	1.194	1.420
	Camada	0.668	0.718	0.029	0.715	0.780	1.034	1.170	0.060	1.177	1.271
	Neurônio	**0.700**	**0.752**	**0.019**	**0.752**	**0.790**	**1.012**	**1.098**	**0.042**	**1.098**	**1.208**
Tangente Hiperbólica	Normal	0.691	0.715	**0.013**	0.715	0.743	1.117	1.176	**0.027**	1.178	1.226

Continua na próxima página

Tabela 35 – Resultados do Experimento 2 para a Base de Dados 7 (294_satellite_image).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	0.702	0.735	**0.019**	0.734	0.764	1.071	1.135	**0.040**	1.137	1.203
	Neurônio	**0.704**	**0.740**	0.017	**0.740**	**0.769**	**1.060**	**1.123**	0.038	**1.124**	**1.199**

Fonte: Elaborada pelo autor (2023).

Tabela 36 – Resultados do Experimento 2 para a Base de Dados 8 (4544_GeographicalOriginalofMusic).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	0.474	0.575	**0.038**	0.584	0.624	0.560	0.595	**0.026**	0.589	0.662
	Camada	0.492	**0.596**	0.041	0.605	**0.654**	**0.537**	**0.580**	0.029	0.574	0.651
	Neurônio	**0.493**	0.594	0.039	**0.607**	0.646	0.543	0.581	0.027	**0.573**	**0.650**
Cúbico	Normal	-0.006	0.022	**0.014**	0.022	0.063	0.884	0.903	**0.006**	0.903	0.916
	Camada	**0.416**	**0.535**	0.049	0.535	0.626	0.559	**0.622**	0.033	0.622	**0.698**
	Neurônio	0.215	0.524	0.079	**0.538**	**0.647**	**0.542**	0.628	0.049	**0.621**	0.809
Linear	Normal	0.479	0.589	**0.043**	0.597	0.648	0.542	0.585	**0.030**	0.580	0.659
	Camada	0.486	**0.604**	0.045	**0.620**	**0.669**	**0.525**	**0.574**	0.032	**0.563**	0.655
	Neurônio	**0.495**	0.603	**0.043**	0.617	0.668	0.526	**0.574**	0.031	0.565	**0.649**
Mish	Normal	0.458	0.538	**0.038**	0.550	0.585	0.588	0.620	**0.025**	0.612	0.672
	Camada	0.470	**0.576**	0.042	**0.586**	**0.634**	**0.553**	**0.594**	0.029	**0.588**	0.665
	Neurônio	**0.484**	0.564	0.039	0.573	0.630	0.555	0.602	0.027	0.597	**0.656**
PELU	Normal	0.484	0.574	**0.038**	0.582	0.624	0.560	0.595	**0.026**	0.591	0.656
	Camada	0.492	**0.588**	0.041	**0.600**	**0.651**	**0.540**	**0.585**	0.029	**0.578**	0.651
	Neurônio	**0.495**	**0.588**	0.040	0.598	0.650	**0.540**	**0.585**	0.028	0.579	**0.649**
PReLU	Normal	0.412	0.497	0.042	0.509	0.567	0.601	0.647	0.026	0.640	0.700
	Camada	0.476	**0.578**	0.041	**0.585**	**0.631**	**0.554**	**0.593**	0.028	**0.588**	0.661
	Neurônio	**0.479**	0.555	**0.034**	0.560	0.608	0.572	0.608	**0.023**	0.606	**0.659**
Quadrática	Normal	0.042	0.116	**0.031**	0.119	0.173	0.830	0.858	**0.015**	0.857	0.894
	Camada	0.375	0.557	0.053	0.564	**0.638**	**0.549**	0.607	0.035	0.603	0.722

Continua na próxima página

Tabela 36 – Resultados do Experimento 2 para a Base de Dados 8 (4544_GeographicalOriginalofMusic).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**0.494**	**0.576**	0.039	**0.588**	0.632	0.554	**0.594**	0.027	**0.586**	**0.649**
SELU	Normal	0.485	0.591	0.041	0.593	0.645	0.544	0.584	**0.029**	0.583	0.655
	Camada	0.493	0.602	0.042	**0.605**	**0.659**	**0.533**	**0.575**	0.030	**0.574**	0.650
	Neurônio	**0.499**	**0.603**	**0.040**	**0.605**	0.658	0.534	**0.575**	**0.029**	**0.574**	**0.646**
Sigmóide	Normal	0.290	0.404	0.050	0.393	0.488	0.653	0.704	0.030	0.712	0.770
	Camada	**0.428**	**0.518**	0.040	**0.518**	**0.600**	**0.578**	**0.633**	0.026	**0.634**	**0.691**
	Neurônio	0.417	0.505	**0.039**	0.501	0.586	0.587	0.642	**0.025**	0.645	0.697
SoftExponential	Normal	0.487	0.588	**0.042**	0.591	0.650	0.540	0.585	**0.029**	0.584	0.654
	Camada	0.490	0.599	**0.042**	0.608	0.663	0.530	0.578	0.030	0.572	0.652
	Neurônio	**0.501**	**0.606**	0.043	**0.618**	**0.675**	**0.520**	**0.573**	0.031	**0.564**	**0.645**
SoftPlus	Normal	0.408	0.499	0.038	0.502	0.558	0.607	0.646	**0.024**	0.645	0.703
	Camada	**0.461**	**0.539**	0.039	**0.552**	**0.598**	**0.579**	**0.619**	0.026	**0.611**	**0.671**
	Neurônio	0.460	0.533	**0.037**	0.547	0.586	0.587	0.623	**0.024**	0.615	**0.671**
SoftSign	Normal	0.437	0.545	0.042	0.557	0.602	0.576	0.615	**0.028**	0.608	0.685
	Camada	0.461	**0.588**	0.045	**0.601**	**0.645**	**0.544**	**0.586**	0.031	**0.577**	0.670
	Neurônio	**0.479**	0.580	**0.041**	0.590	0.636	0.551	0.591	0.029	0.585	**0.659**
Swish	Normal	0.448	0.529	**0.038**	0.543	0.580	0.592	0.626	**0.025**	0.617	0.678
	Camada	0.465	**0.567**	0.042	**0.577**	0.623	**0.560**	**0.600**	0.029	**0.594**	0.668
	Neurônio	**0.476**	0.559	0.040	0.570	**0.624**	**0.560**	0.606	0.027	0.599	**0.661**
Tangente Hiperbólica	Normal	0.457	0.570	**0.043**	0.580	0.630	0.555	0.598	**0.030**	0.592	0.673

Continua na próxima página

Tabela 36 – Resultados do Experimento 2 para a Base de Dados 8 (4544_GeographicalOriginalofMusic).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	0.463	**0.586**	**0.045**	**0.597**	**0.649**	**0.541**	**0.587**	**0.031**	**0.580**	0.669
	Neurônio	**0.467**	0.585	0.044	0.595	0.646	0.543	0.588	**0.031**	0.581	**0.667**

Fonte: Elaborada pelo autor (2023).

Tabela 37 – Resultados do Experimento 2 para a Base de Dados 9 (503_wind).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	**0.226**	**0.531**	**0.116**	**0.572**	**0.679**	**3.791**	**4.547**	**0.544**	**4.376**	**5.885**
	Camada	0.243	0.556	**0.118**	0.591	0.696	3.687	4.420	**0.569**	4.277	5.820
	Neurônio	**0.244**	**0.558**	**0.118**	**0.592**	**0.698**	**3.673**	**4.409**	**0.569**	**4.270**	**5.816**
Cúbico	Normal	**0.013**	**0.134**	**0.080**	**0.140**	**0.283**	**5.664**	**6.215**	**0.290**	**6.201**	**6.644**
	Camada	0.245	0.557	**0.117**	0.592	0.696	3.686	4.414	**0.566**	4.271	5.813
	Neurônio	**0.250**	**0.561**	**0.117**	**0.598**	**0.699**	**3.671**	**4.396**	**0.566**	**4.240**	**5.790**
Linear	Normal	**0.240**	**0.555**	**0.118**	**0.591**	**0.696**	**3.688**	**4.426**	**0.571**	**4.276**	**5.829**
	Camada	**0.242**	**0.555**	**0.119**	**0.591**	**0.696**	**3.688**	**4.423**	**0.572**	**4.277**	**5.822**
	Neurônio	**0.242**	**0.555**	**0.118**	**0.591**	**0.696**	**3.688**	4.424	**0.572**	**4.277**	**5.822**
Mish	Normal	**0.213**	**0.499**	**0.110**	**0.527**	**0.649**	**3.964**	**4.709**	**0.501**	**4.598**	**5.933**
	Camada	0.242	0.556	**0.118**	**0.591**	0.696	3.687	4.420	**0.570**	4.279	5.823
	Neurônio	**0.243**	**0.558**	**0.118**	**0.591**	**0.697**	**3.681**	**4.413**	0.568	**4.278**	**5.819**
PELU	Normal	**0.234**	**0.536**	**0.112**	**0.575**	**0.685**	**3.751**	**4.527**	**0.529**	**4.361**	**5.855**
	Camada	**0.237**	**0.544**	0.115	**0.582**	0.688	3.738	**4.486**	**0.549**	**4.325**	**5.841**
	Neurônio	0.236	0.543	**0.116**	0.581	**0.689**	**3.732**	4.488	**0.549**	4.329	5.845
PReLU	Normal	**0.142**	**0.446**	**0.125**	**0.478**	**0.617**	**4.140**	**4.947**	**0.545**	**4.832**	**6.196**
	Camada	**0.245**	**0.556**	0.118	**0.590**	0.696	3.686	4.421	**0.569**	**4.282**	**5.811**
	Neurônio	0.242	**0.556**	**0.117**	**0.590**	**0.698**	**3.677**	**4.419**	0.567	4.284	5.822
Quadrática	Normal	**0.003**	**0.133**	**0.078**	**0.117**	**0.302**	**5.587**	**6.221**	**0.285**	**6.284**	**6.679**
	Camada	0.241	0.556	**0.118**	0.591	0.696	3.687	4.419	**0.572**	4.277	5.827

Continua na próxima página

Tabela 37 – Resultados do Experimento 2 para a Base de Dados 9 (503_wind).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Neurônio	**0.245**	**0.559**	0.118	**0.595**	**0.699**	**3.671**	**4.404**	0.571	**4.255**	**5.811**
SELU	Normal	0.246	0.548	**0.113**	0.586	0.691	3.715	4.467	**0.538**	4.305	5.809
	Camada	**0.248**	0.549	**0.113**	0.586	0.692	3.714	4.459	0.540	4.304	5.800
	Neurônio	**0.248**	**0.551**	**0.113**	**0.587**	**0.695**	**3.692**	**4.448**	0.540	**4.296**	**5.799**
Sigmóide	Normal	0.127	0.480	0.149	0.548	0.628	4.078	4.778	0.660	4.496	6.249
	Camada	**0.233**	0.541	**0.114**	0.584	0.673	3.822	4.499	**0.541**	4.312	**5.857**
	Neurônio	0.221	**0.545**	0.116	**0.585**	**0.688**	**3.737**	**4.478**	0.553	**4.310**	5.904
SoftExponential	Normal	0.240	0.553	**0.118**	0.589	0.695	3.696	4.436	**0.570**	4.289	5.832
	Camada	**0.242**	0.556	**0.118**	0.591	0.696	3.688	4.421	0.572	4.277	5.822
	Neurônio	**0.242**	**0.557**	**0.118**	**0.592**	**0.697**	**3.682**	**4.416**	0.571	**4.273**	**5.821**
SoftPlus	Normal	0.158	0.507	0.127	0.556	0.643	3.997	4.664	0.578	4.456	6.138
	Camada	0.221	0.528	0.116	0.565	0.664	3.878	4.562	**0.541**	4.412	5.904
	Neurônio	**0.225**	**0.531**	**0.115**	**0.569**	**0.680**	**3.780**	**4.551**	**0.541**	**4.388**	**5.886**
SoftSign	Normal	0.237	0.532	**0.109**	0.570	0.659	3.903	4.547	**0.511**	4.387	5.841
	Camada	0.244	**0.556**	0.118	**0.592**	0.696	3.688	**4.419**	0.570	**4.273**	5.815
	Neurônio	**0.246**	**0.556**	0.117	0.591	**0.697**	**3.680**	4.420	0.568	4.277	**5.809**
Swish	Normal	0.214	0.498	**0.110**	0.529	0.643	3.997	4.711	**0.500**	4.590	5.930
	Camada	0.241	0.556	0.118	**0.591**	0.696	3.687	4.420	0.570	4.279	5.825
	Neurônio	**0.243**	**0.557**	0.118	**0.591**	**0.697**	**3.679**	**4.414**	0.568	**4.277**	**5.820**
Tangente Hiperbólica	Normal	0.242	0.537	**0.108**	0.575	0.663	3.880	4.522	**0.512**	4.358	5.821

Continua na próxima página

Tabela 37 – Resultados do Experimento 2 para a Base de Dados 9 (503_wind).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	0.244	**0.552**	**0.116**	**0.588**	0.690	3.721	**4.440**	**0.561**	**4.292**	5.814
	Neurônio	**0.245**	**0.552**	**0.116**	**0.588**	**0.691**	**3.718**	4.444	0.559	4.294	**5.813**

Fonte: Elaborada pelo autor (2023).

Tabela 38 – Resultados do Experimento 2 para a Base de Dados 10 (573_cpu_act).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
BLU	Normal	**0.143**	**0.418**	**0.100**	**0.424**	**0.630**	**10.531**	**13.155**	1.129	**13.143**	**16.029**
	Camada	0.204	0.463	**0.095**	0.469	0.631	10.519	12.638	**1.105**	12.616	15.449
	Neurônio	**0.267**	**0.508**	**0.095**	**0.514**	**0.681**	**9.775**	**12.086**	**1.165**	**12.069**	**14.823**
Cúbico	Normal	**-0.109**	**0.038**	**0.041**	**0.040**	**0.109**	**16.337**	**16.976**	**0.354**	**16.961**	**18.229**
	Camada	**0.187**	0.458	0.108	0.489	0.651	10.230	12.683	1.240	12.379	**15.609**
	Neurônio	0.167	**0.484**	**0.118**	**0.501**	**0.654**	**10.185**	**12.365**	**1.383**	**12.229**	15.796
Linear	Normal	**0.185**	**0.445**	**0.094**	**0.451**	**0.627**	**10.572**	**12.849**	**1.083**	**12.831**	**15.628**
	Camada	**0.185**	**0.446**	**0.094**	**0.454**	**0.621**	10.651	**12.843**	**1.086**	**12.795**	**15.631**
	Neurônio	**0.185**	**0.446**	**0.094**	**0.454**	**0.621**	**10.652**	12.844	**1.086**	**12.795**	**15.631**
Mish	Normal	**0.093**	**0.391**	**0.110**	**0.398**	**0.628**	**10.554**	**13.454**	**1.215**	**13.430**	**16.486**
	Camada	0.120	0.450	0.103	0.463	0.632	10.498	12.781	1.177	12.689	16.243
	Neurônio	**0.259**	**0.495**	**0.088**	**0.492**	**0.662**	**10.058**	**12.257**	**1.067**	**12.338**	**14.905**
PELU	Normal	**0.165**	**0.432**	**0.096**	**0.450**	0.638	10.415	**13.007**	**1.088**	**12.842**	**15.822**
	Camada	0.202	0.442	**0.091**	0.456	**0.637**	**10.423**	12.889	**1.053**	12.771	15.460
	Neurônio	**0.219**	**0.456**	0.092	**0.472**	**0.653**	**10.203**	**12.726**	1.077	**12.584**	**15.294**
PReLU	Normal	**0.114**	**0.364**	**0.114**	**0.375**	**0.622**	**10.645**	**13.752**	**1.262**	**13.686**	**16.297**
	Camada	**0.221**	0.449	0.096	0.458	0.627	10.570	12.803	**1.115**	12.748	**15.278**
	Neurônio	0.220	**0.484**	**0.094**	**0.476**	**0.672**	**9.908**	**12.387**	1.133	**12.536**	15.293
Quadrática	Normal	**0.055**	**0.109**	**0.041**	**0.097**	**0.208**	**15.404**	**16.337**	**0.378**	**16.447**	**16.827**
	Camada	0.131	0.455	**0.101**	0.456	0.628	10.556	12.729	**1.160**	12.764	16.136

Continua na próxima página

Tabela 38 – Resultados do Experimento 2 para a Base de Dados 10 (573_cpu_act).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
SELU	Neurônio	**0.176**	**0.491**	0.097	**0.502**	**0.661**	**10.079**	**12.299**	1.146	**12.222**	**15.716**
	Normal	0.237	0.450	0.083	0.455	0.625	10.601	12.797	0.967	12.776	15.123
	Camada	0.284	0.467	**0.079**	0.483	0.628	10.562	12.610	**0.927**	12.447	14.644
	Neurônio	**0.288**	**0.505**	0.085	**0.512**	**0.698**	**9.521**	**12.139**	1.042	**12.097**	**14.605**
Sigmóide	Normal	0.144	0.399	0.091	0.383	0.536	11.790	13.383	**1.012**	13.602	16.018
	Camada	0.228	0.475	0.088	**0.506**	0.648	10.272	12.497	1.028	**12.172**	15.205
	Neurônio	**0.235**	**0.480**	**0.086**	0.505	**0.650**	**10.247**	**12.439**	1.021	12.176	**15.139**
SoftExponential	Normal	0.195	0.432	0.091	0.434	0.610	10.804	13.006	**1.040**	13.024	15.527
	Camada	0.179	0.448	0.098	0.455	0.625	10.604	12.812	1.128	12.782	15.686
	Neurônio	**0.230**	**0.468**	**0.088**	**0.467**	**0.664**	**10.032**	**12.583**	1.056	**12.639**	**15.187**
SoftPlus	Normal	0.124	0.395	**0.099**	0.389	0.602	10.919	13.420	**1.102**	13.529	16.206
	Camada	0.147	0.409	0.101	0.409	0.611	10.797	13.265	1.133	13.313	15.990
	Neurônio	**0.159**	**0.418**	0.100	**0.418**	**0.619**	**10.680**	**13.160**	1.136	**13.201**	**15.872**
SoftSign	Normal	0.216	0.449	**0.080**	0.452	0.601	10.940	12.819	**0.921**	12.816	15.327
	Camada	0.230	0.481	0.089	0.503	0.642	10.352	12.433	1.057	12.208	15.191
	Neurônio	**0.256**	**0.491**	0.086	**0.517**	**0.647**	**10.285**	**12.311**	1.027	**12.031**	**14.935**
Swish	Normal	0.089	0.379	0.107	0.381	0.610	10.811	13.589	1.177	13.614	16.519
	Camada	0.117	0.445	0.105	0.444	0.626	10.590	12.847	1.193	12.903	16.270
	Neurônio	**0.251**	**0.491**	**0.090**	**0.498**	**0.660**	**10.090**	**12.300**	**1.091**	**12.268**	**14.984**
Tangente Hiperbólica	Normal	0.238	0.461	**0.082**	0.467	0.628	10.554	12.678	**0.955**	12.635	15.107

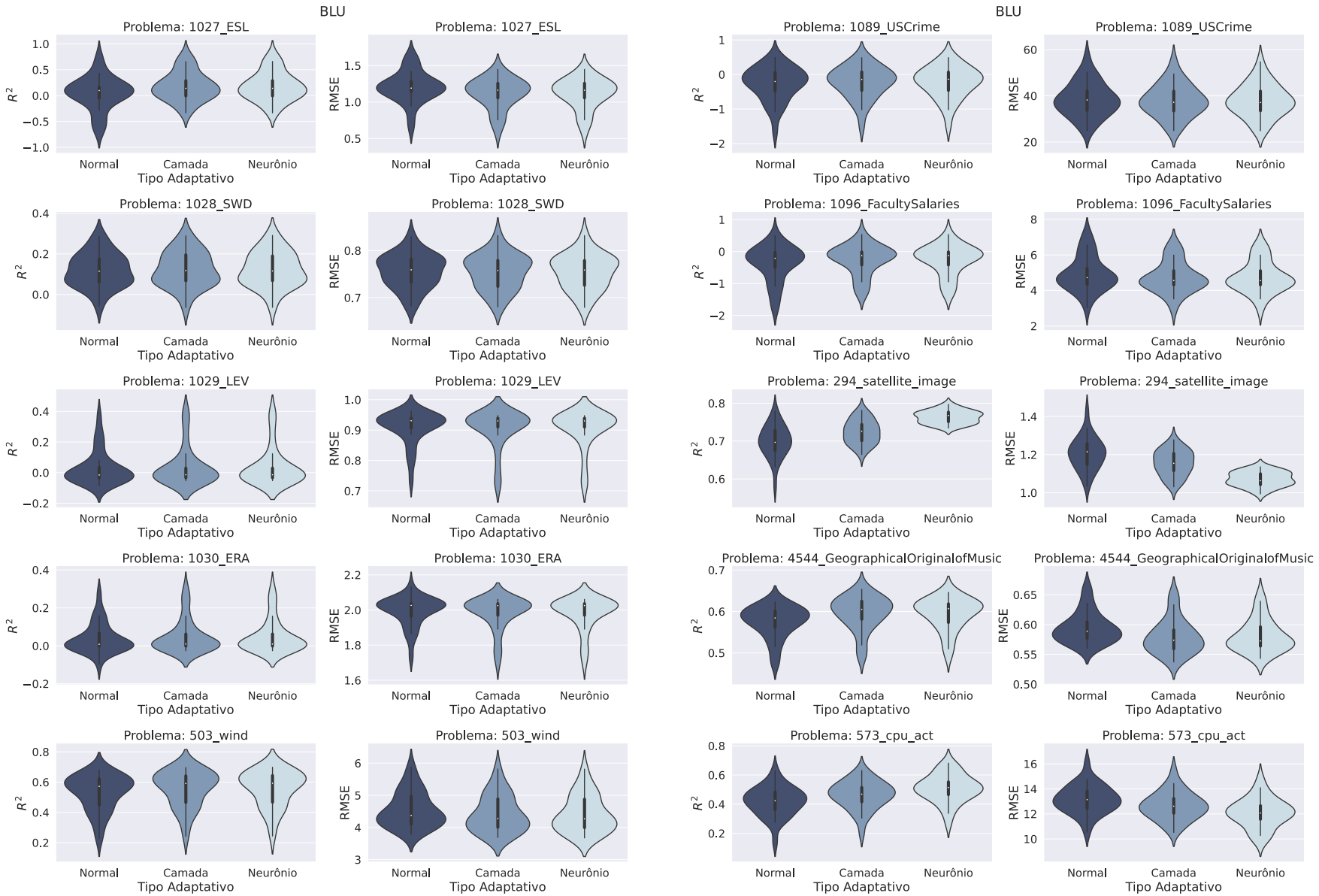
Continua na próxima página

Tabela 38 – Resultados do Experimento 2 para a Base de Dados 10 (573_cpu_act).

Função de Ativação	Tipo Adaptativo	R^2					RMSE				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
	Camada	0.242	0.481	**0.088**	0.504	0.650	10.234	12.432	**1.037**	12.191	15.067
	Neurônio	**0.267**	**0.489**	0.085	**0.512**	**0.651**	**10.230**	**12.337**	1.014	**12.098**	**14.821**

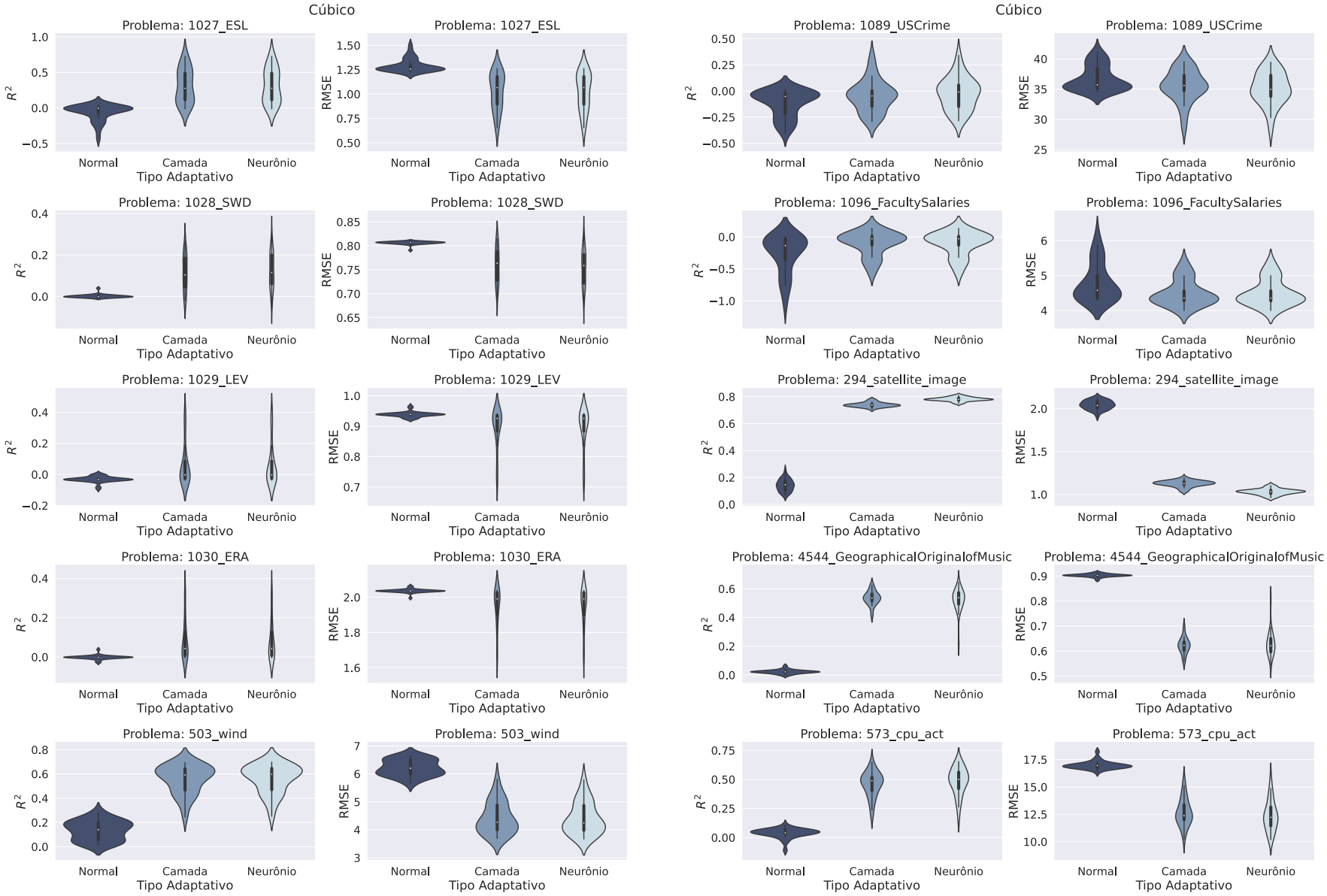
Fonte: Elaborada pelo autor (2023).

Figura 89 – *Violin Plots* do Experimento 2 para função de ativação BLU.



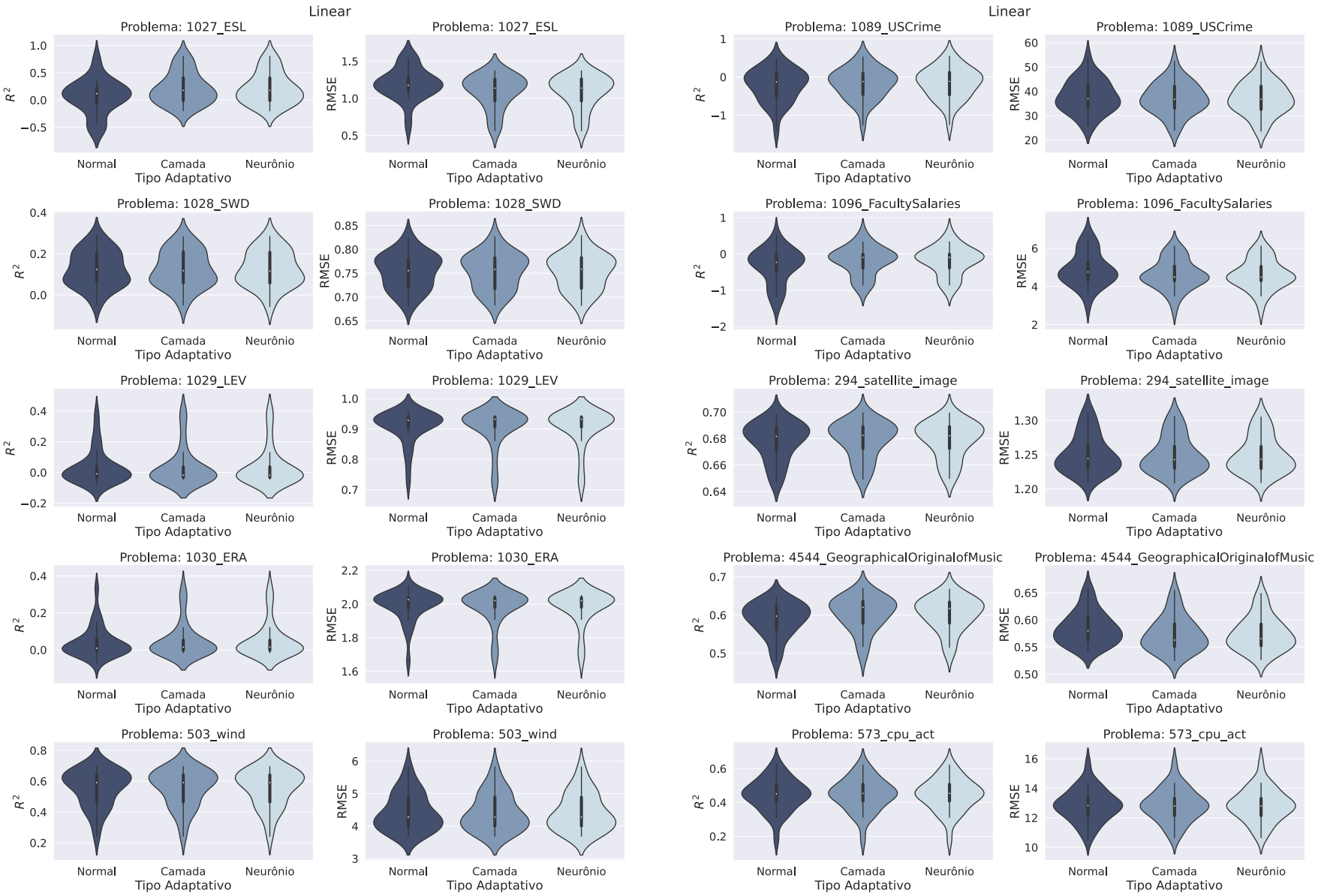
Fonte: Elaborada pelo autor (2023).

Figura 90 – *Violin Plots* do Experimento 2 para função de ativação Cúbico.



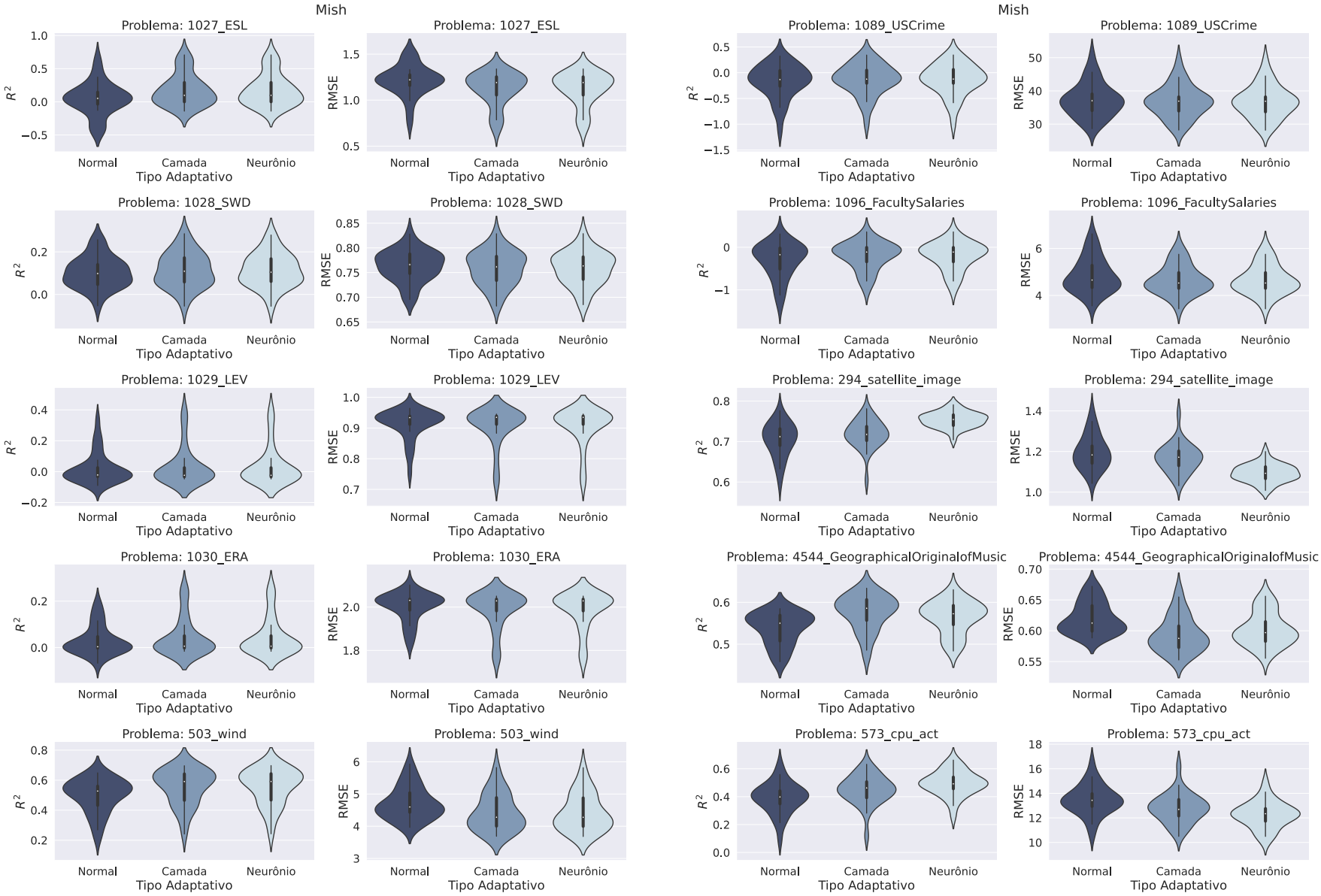
Fonte: Elaborada pelo autor (2023).

Figura 91 – *Violin Plots* do Experimento 2 para função de ativação Linear.



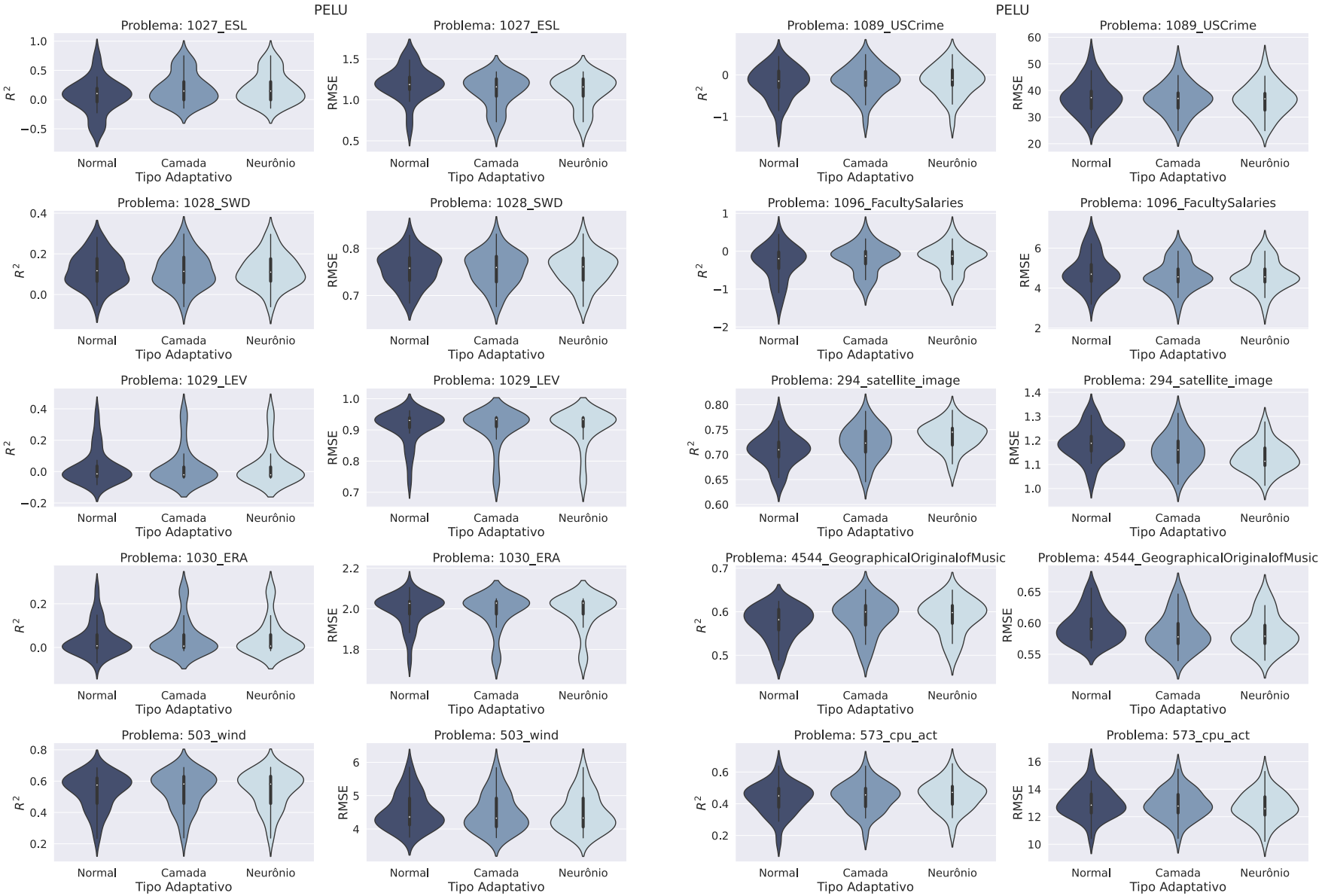
Fonte: Elaborada pelo autor (2023).

Figura 92 – *Violin Plots* do Experimento 2 para função de ativação Mish.



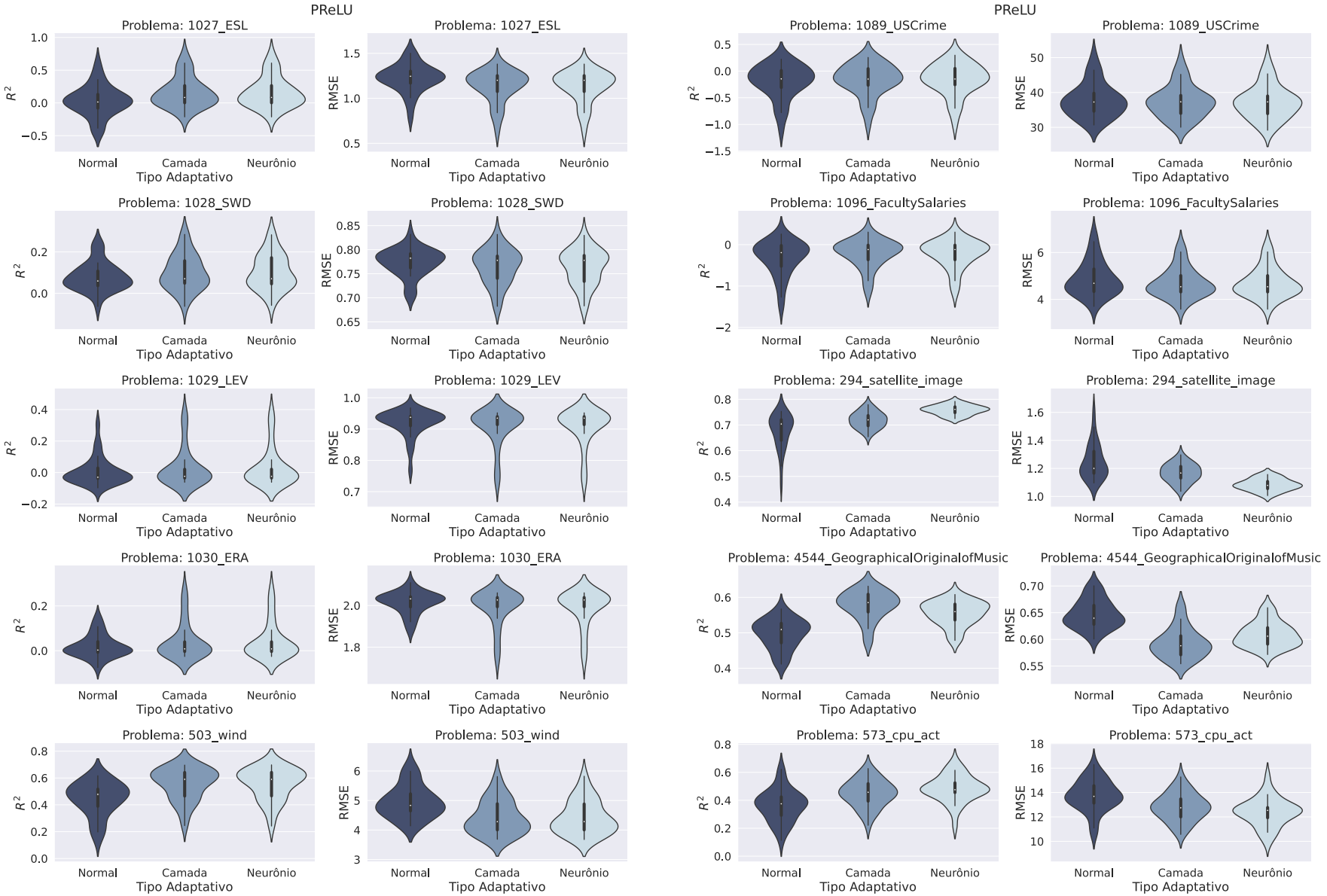
Fonte: Elaborada pelo autor (2023).

Figura 93 – *Violin Plots* do Experimento 2 para função de ativação PELU.



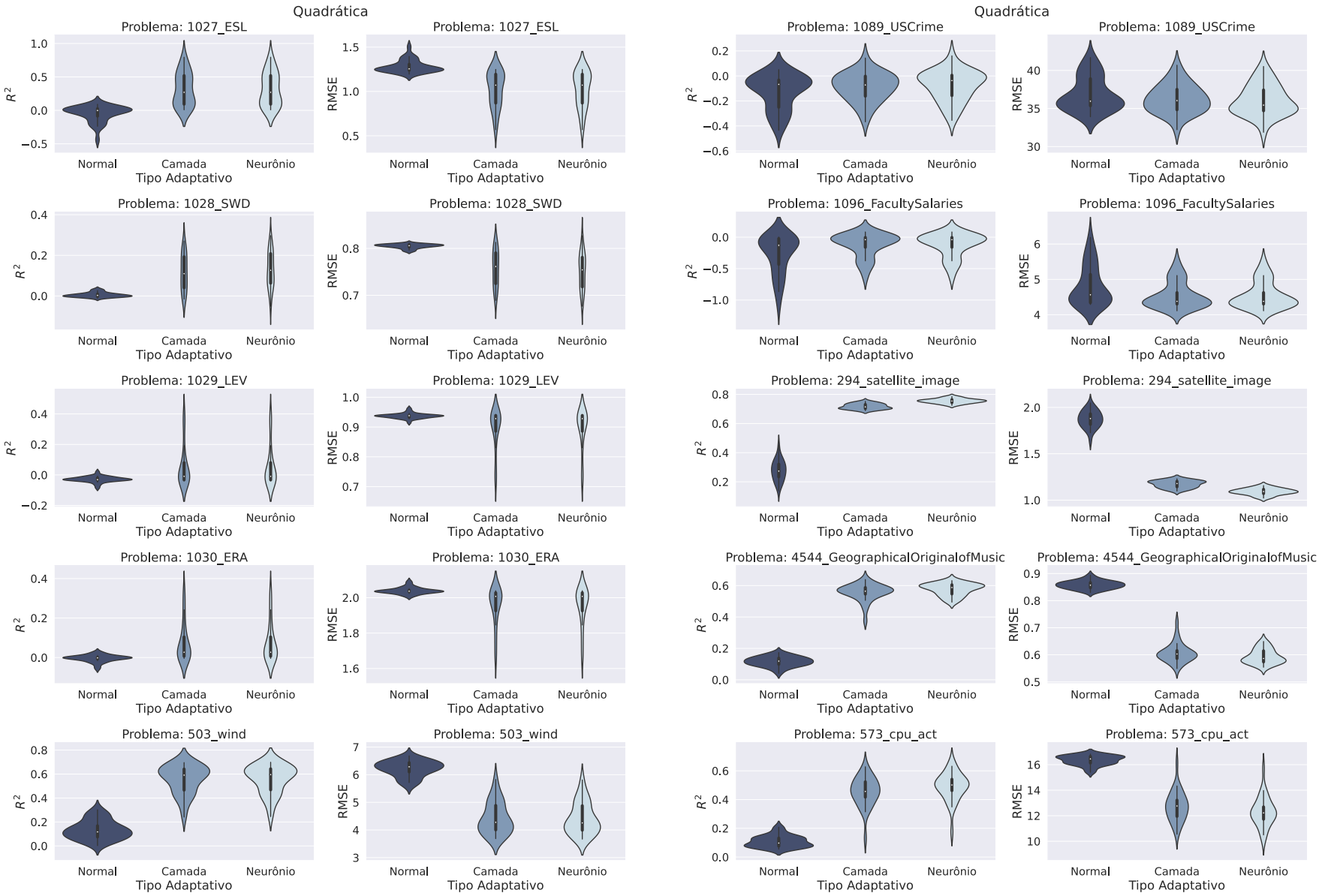
Fonte: Elaborada pelo autor (2023).

Figura 94 – *Violin Plots* do Experimento 2 para função de ativação PReLU.



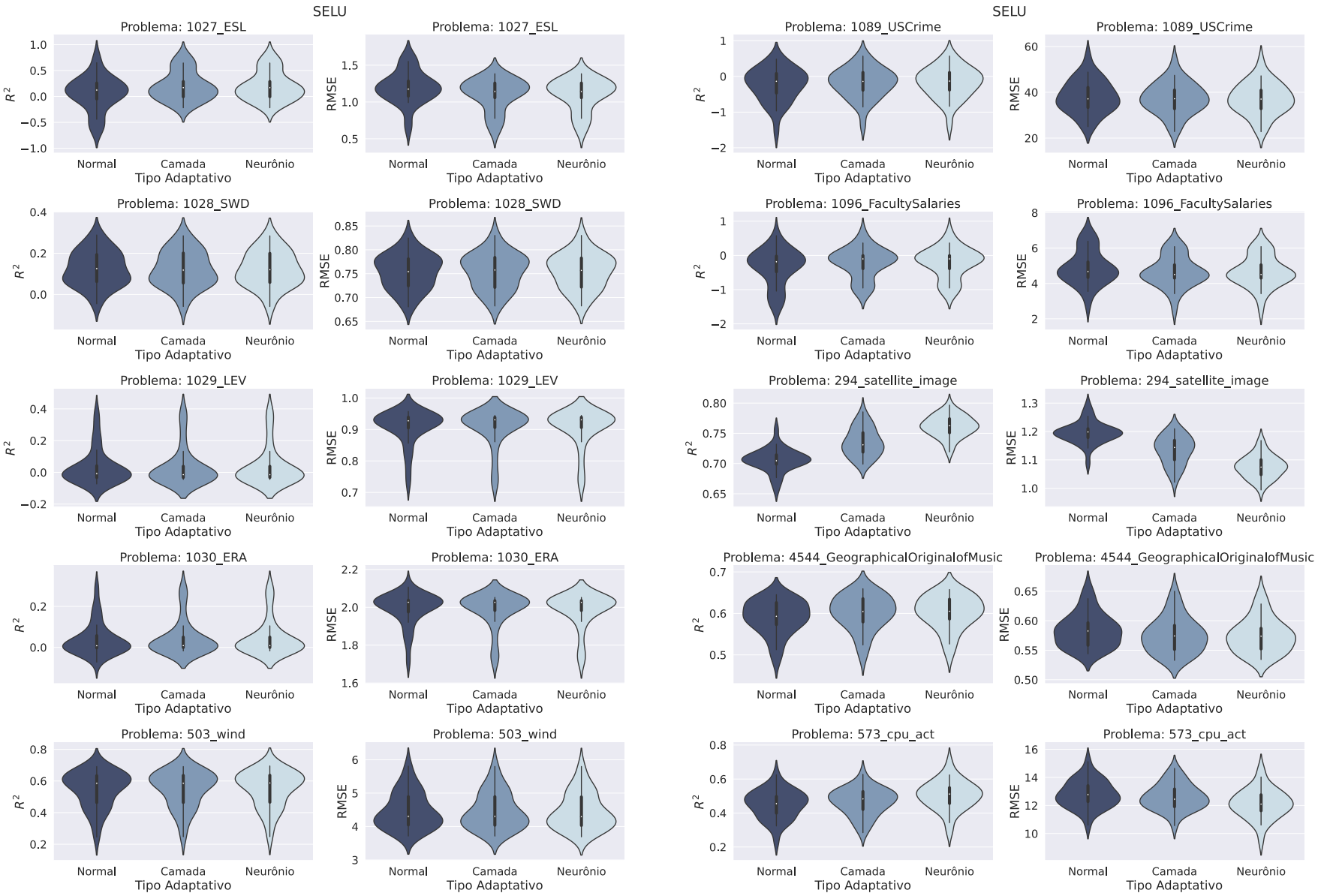
Fonte: Elaborada pelo autor (2023).

Figura 95 – *Violin Plots* do Experimento 2 para função de ativação Quadrática.



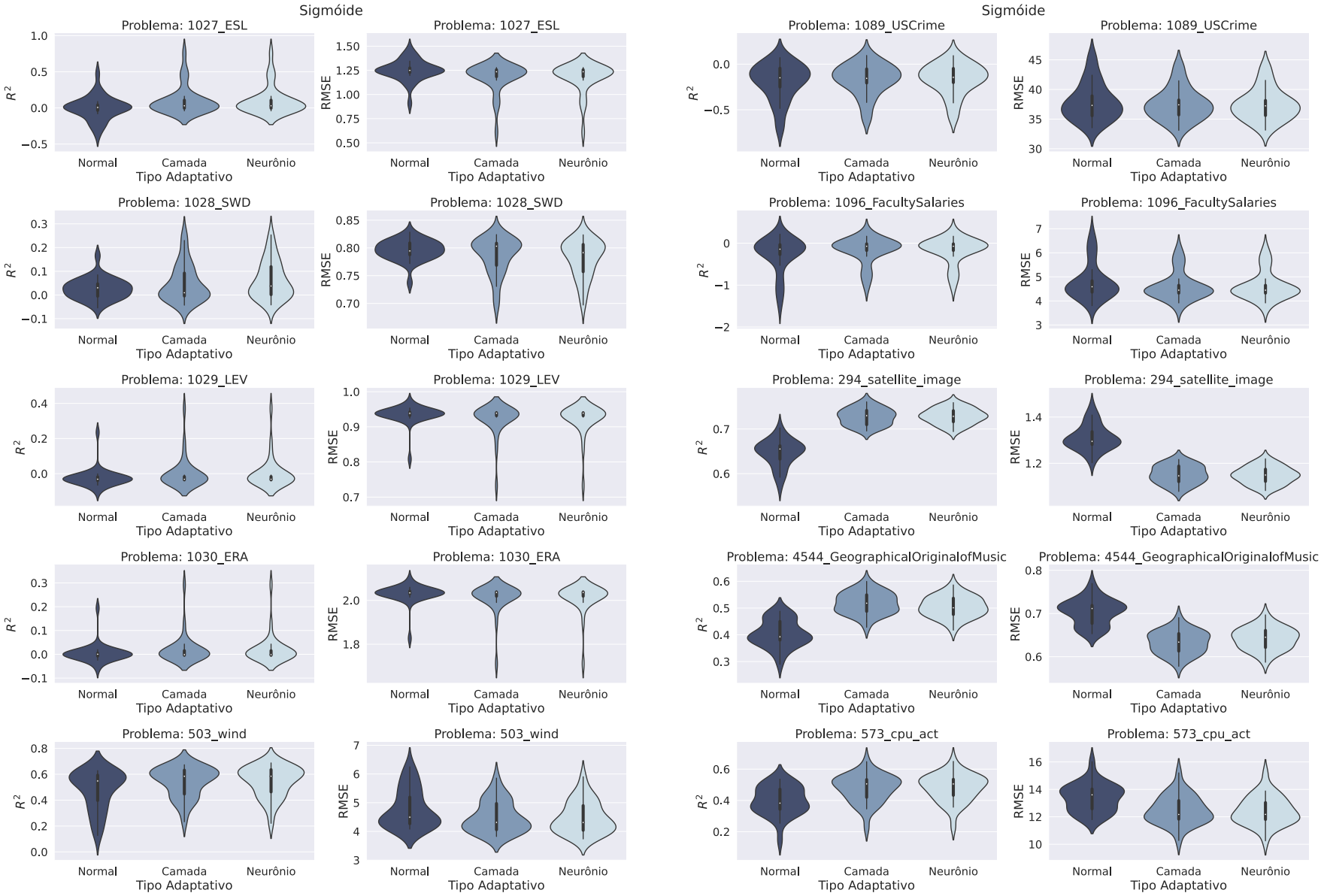
Fonte: Elaborada pelo autor (2023).

Figura 96 – *Violin Plots* do Experimento 2 para função de ativação SELU.



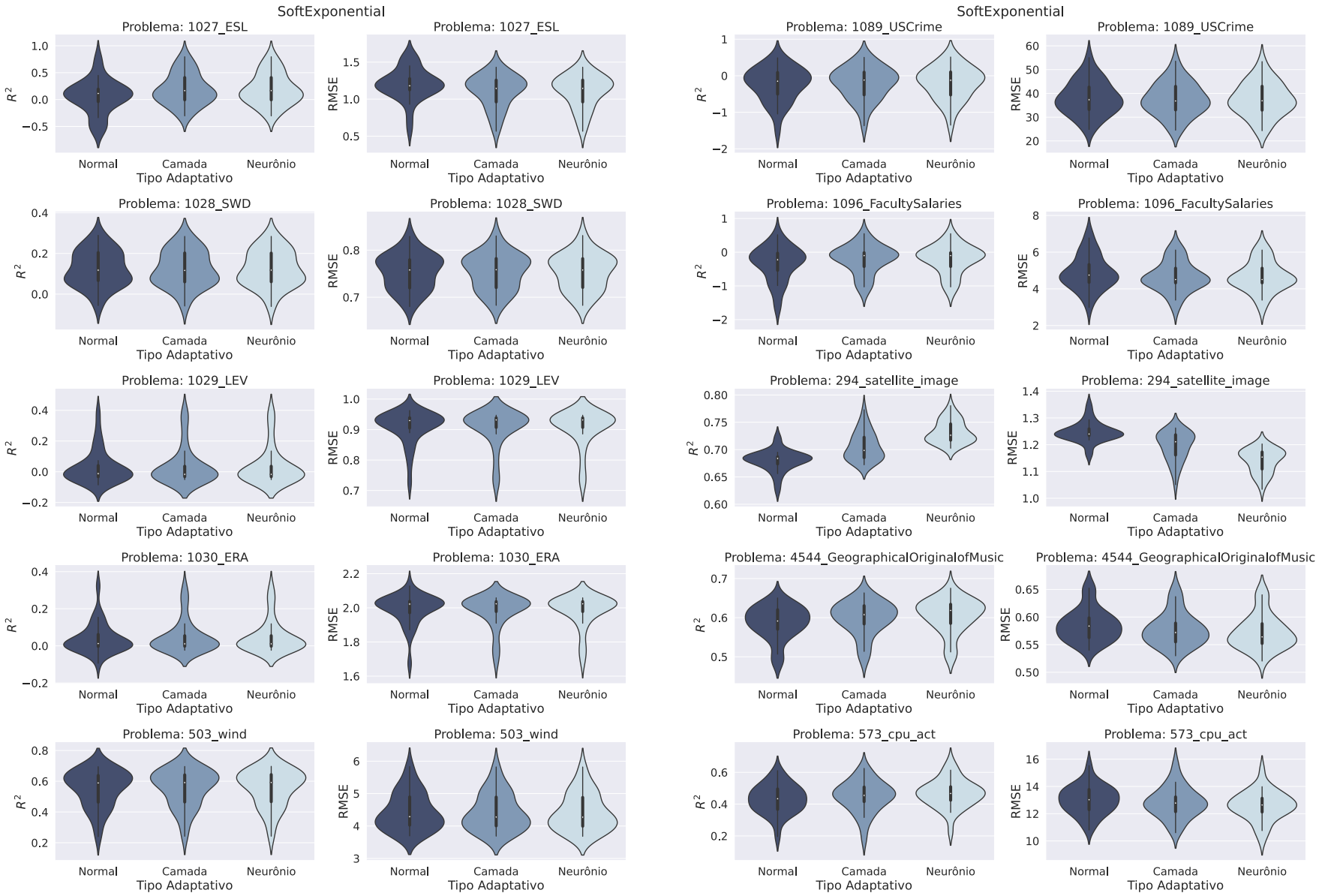
Fonte: Elaborada pelo autor (2023).

Figura 97 – *Violin Plots* do Experimento 2 para função de ativação Sigmóide.



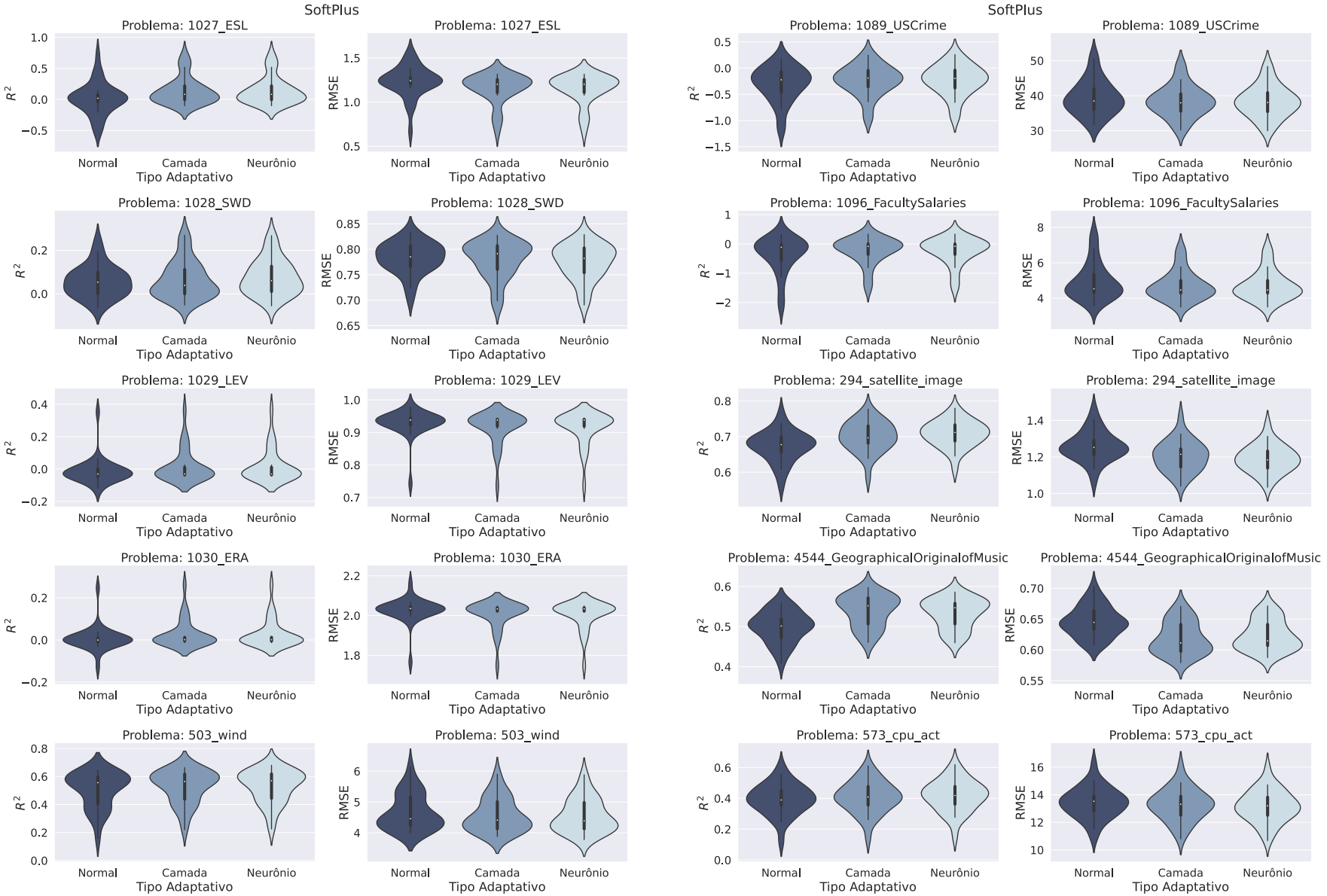
Fonte: Elaborada pelo autor (2023).

Figura 98 – *Violin Plots* do Experimento 2 para função de ativação SoftExponential.



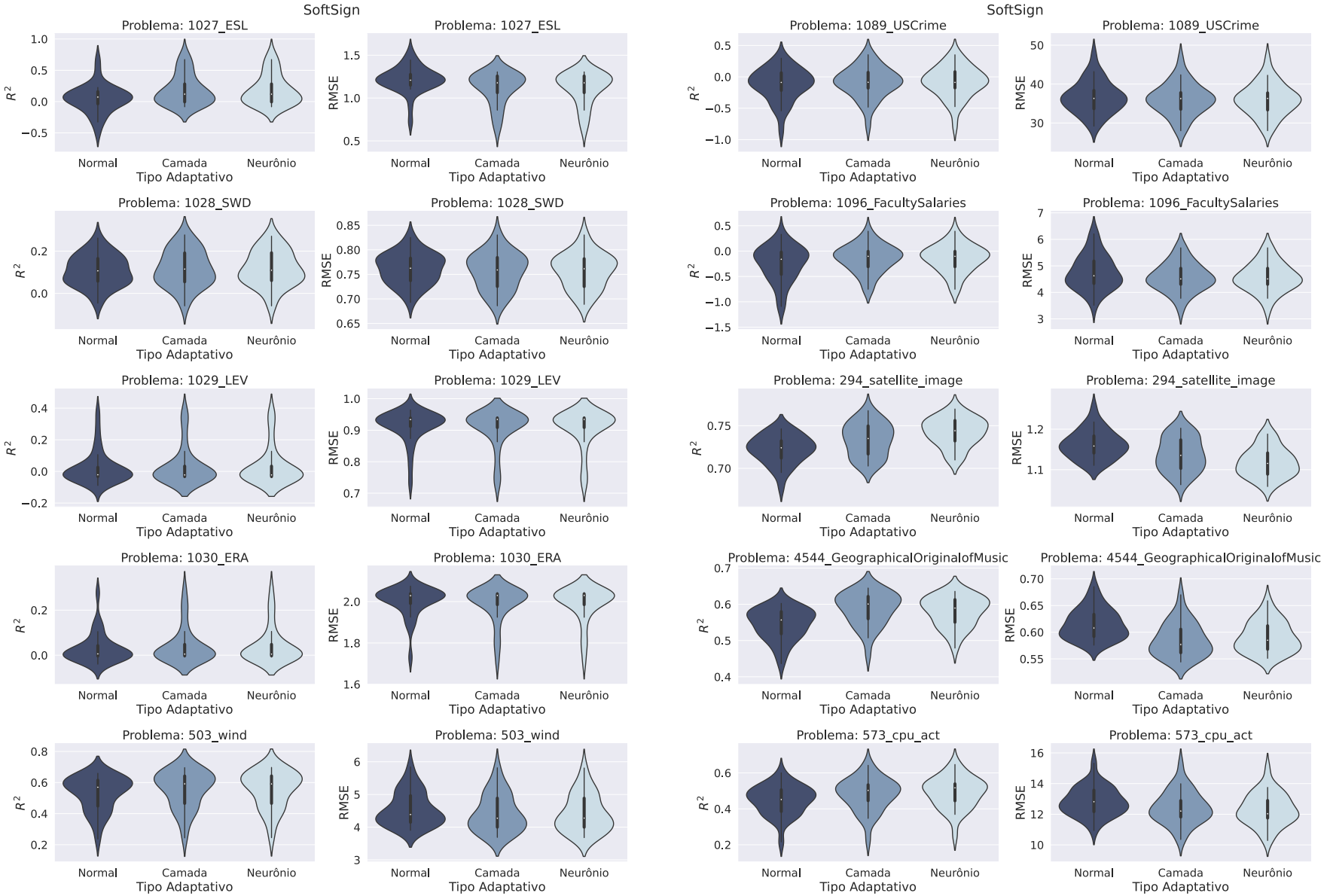
Fonte: Elaborada pelo autor (2023).

Figura 99 – *Violin Plots* do Experimento 2 para função de ativação SoftPlus.



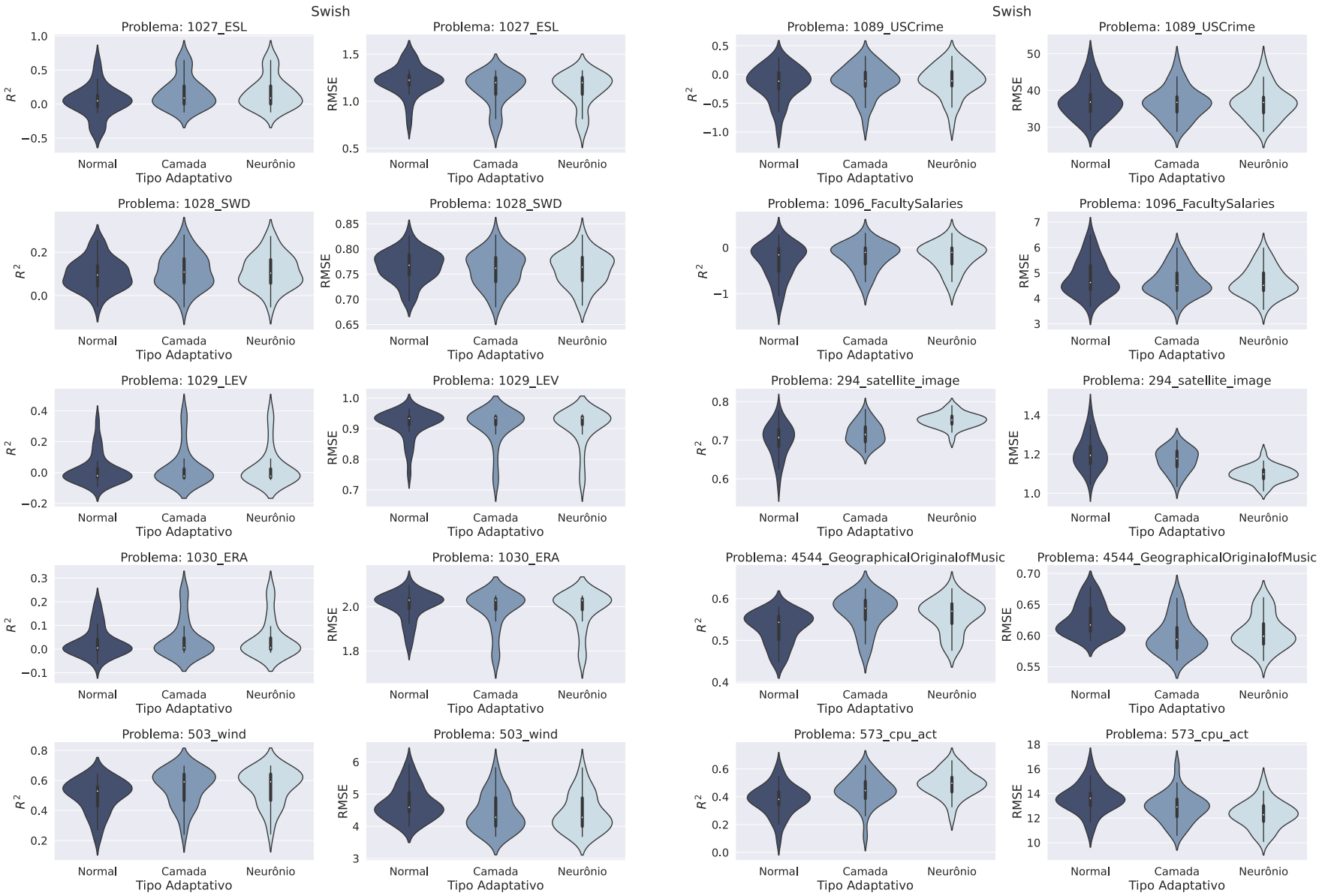
Fonte: Elaborada pelo autor (2023).

Figura 100 – *Violin Plots* do Experimento 2 para função de ativação SoftSign.



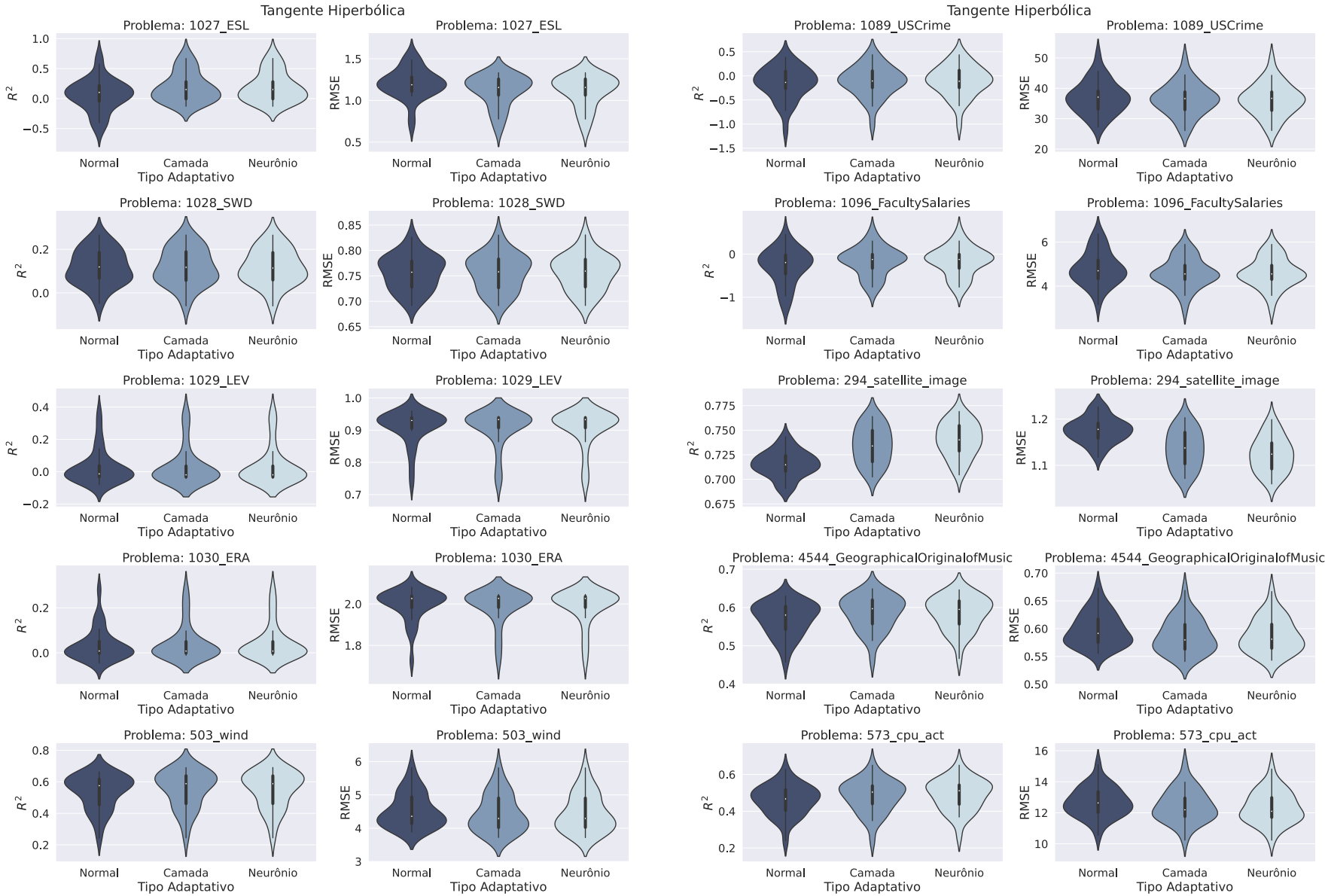
Fonte: Elaborada pelo autor (2023).

Figura 101 – *Violin Plots* do Experimento 2 para função de ativação Swish.



Fonte: Elaborada pelo autor (2023).

Figura 102 – *Violin Plots* do Experimento 2 para função de ativação Tangente Hiperbólica.



Fonte: Elaborada pelo autor (2023).

B.2 Pegada de Carbono

A Tabela 39 apresenta as estimativas médias de emissão de CO_2, em quilograma, associadas ao treinamento de cada modelo no segundo experimento numérico. Essa média é obtida a partir de cinco execuções focadas na estimativa da emissão de CO_2. Os valores destacados com a cor **"Azul Escuro"** representam os melhores resultados encontrados, enquanto os valores marcados com a cor **"Azul Claro"** representam os resultados menos favoráveis.

Tabela 39 – Estimativas de Emissões de CO_2, em quilograma, do Experimento 2.

Base de Dados	Função de Ativação	Normal	Camada	Neurônio
1	BLU	**5.04e-06**	**6.60e-06**	6.55e-06
	Cúbico	**4.95e-06**	**7.09e-06**	7.07e-06
	Linear	**4.92e-06**	6.09e-06	**6.12e-06**
	Mish	**4.94e-06**	**6.56e-06**	6.48e-06
	PELU	**5.25e-06**	6.88e-06	**6.94e-06**
	PReLU	**5.06e-06**	6.06e-06	**6.13e-06**
	Quadrática	**4.95e-06**	**6.59e-06**	6.53e-06
	SELU	**5.14e-06**	**6.63e-06**	6.61e-06
	Sigmóide	**4.90e-06**	**6.56e-06**	6.42e-06
	SoftExponential	**4.96e-06**	6.46e-06	**7.90e-06**
	SoftPlus	**4.98e-06**	6.29e-06	**6.42e-06**
	SoftSign	**4.91e-06**	6.36e-06	**6.43e-06**
	Swish	**4.96e-06**	**6.56e-06**	6.45e-06
	Tangente Hiperbólica	**5.02e-06**	6.92e-06	**6.95e-06**
2	BLU	**9.50e-06**	1.28e-05	**1.28e-05**
	Cúbico	**9.52e-06**	**1.35e-05**	1.34e-05
	Linear	**9.07e-06**	**1.17e-05**	1.16e-05
	Mish	**9.59e-06**	**1.26e-05**	1.24e-05
	PELU	**9.86e-06**	1.32e-05	**1.34e-05**
	PReLU	**9.54e-06**	1.17e-05	**1.18e-05**
	Quadrática	**9.15e-06**	1.25e-05	**1.26e-05**
	SELU	**9.70e-06**	1.27e-05	**1.28e-05**
	Sigmóide	**9.49e-06**	1.25e-05	**1.25e-05**
	SoftExponential	**9.61e-06**	1.27e-05	**1.52e-05**
	SoftPlus	**9.58e-06**	1.22e-05	**1.24e-05**
	SoftSign	**9.53e-06**	1.23e-05	**1.24e-05**
	Swish	**9.60e-06**	1.23e-05	**1.26e-05**

Continua na próxima página

Base de Dados	Função de Ativação	Normal	Camada	Neurônio
	Tangente Hiperbólica	**9.62e-06**	1.33e-05	**1.33e-05**
3	BLU	**9.41e-06**	**1.26e-05**	1.25e-05
	Cúbico	**9.32e-06**	**1.35e-05**	1.33e-05
	Linear	**9.14e-06**	**1.17e-05**	1.17e-05
	Mish	**9.61e-06**	1.22e-05	**1.26e-05**
	PELU	**9.93e-06**	**1.32e-05**	1.31e-05
	PReLU	**9.41e-06**	**1.16e-05**	1.15e-05
	Quadrática	**9.27e-06**	**1.25e-05**	1.24e-05
	SELU	**9.56e-06**	**1.28e-05**	1.24e-05
	Sigmóide	**9.50e-06**	**1.24e-05**	1.24e-05
	SoftExponential	**9.32e-06**	1.23e-05	**1.50e-05**
	SoftPlus	**9.44e-06**	1.19e-05	**1.23e-05**
	SoftSign	**9.25e-06**	**1.23e-05**	1.21e-05
	Swish	**9.53e-06**	1.24e-05	**1.25e-05**
	Tangente Hiperbólica	**9.83e-06**	**1.31e-05**	1.31e-05
4	BLU	**9.56e-06**	1.25e-05	**1.27e-05**
	Cúbico	**9.35e-06**	**1.34e-05**	1.34e-05
	Linear	**9.15e-06**	1.15e-05	**1.18e-05**
	Mish	**9.46e-06**	**1.25e-05**	1.23e-05
	PELU	**9.87e-06**	**1.33e-05**	1.32e-05
	PReLU	**9.59e-06**	**1.18e-05**	1.16e-05
	Quadrática	**9.31e-06**	1.24e-05	**1.25e-05**
	SELU	**9.78e-06**	1.26e-05	**1.28e-05**
	Sigmóide	**9.51e-06**	1.21e-05	**1.24e-05**
	SoftExponential	**9.51e-06**	1.23e-05	**1.48e-05**
	SoftPlus	**9.50e-06**	**1.22e-05**	1.22e-05
	SoftSign	**9.49e-06**	**1.22e-05**	1.20e-05
	Swish	**9.51e-06**	1.23e-05	**1.24e-05**
	Tangente Hiperbólica	**9.61e-06**	1.30e-05	**1.31e-05**
5	BLU	**1.02e-06**	1.27e-06	**1.28e-06**
	Cúbico	**1.01e-06**	1.34e-06	**1.34e-06**
	Linear	**9.86e-07**	**1.22e-06**	1.20e-06
	Mish	**9.89e-07**	1.28e-06	**1.28e-06**
	PELU	**1.10e-06**	**1.34e-06**	1.33e-06
	PReLU	**1.04e-06**	**1.20e-06**	1.18e-06
	Quadrática	**9.94e-07**	**1.30e-06**	1.27e-06

Continua na próxima página

Base de Dados	Função de Ativação	Normal	Camada	Neurônio
	SELU	**1.07e-06**	1.26e-06	**1.29e-06**
	Sigmóide	**9.98e-07**	**1.25e-06**	1.24e-06
	SoftExponential	**1.04e-06**	1.24e-06	**1.46e-06**
	SoftPlus	**9.92e-07**	**1.24e-06**	1.22e-06
	SoftSign	**1.01e-06**	**1.25e-06**	1.24e-06
	Swish	**1.01e-06**	1.28e-06	**1.29e-06**
	Tangente Hiperbólica	**1.05e-06**	1.31e-06	**1.34e-06**
	BLU	**1.07e-06**	1.28e-06	**1.28e-06**
	Cúbico	**1.05e-06**	1.32e-06	**1.34e-06**
	Linear	**1.01e-06**	**1.25e-06**	1.22e-06
	Mish	**1.00e-06**	**1.26e-06**	1.25e-06
	PELU	**1.06e-06**	1.31e-06	**1.33e-06**
	PReLU	**1.08e-06**	1.20e-06	**1.21e-06**
6	Quadrática	**1.02e-06**	1.26e-06	**1.27e-06**
	SELU	**1.09e-06**	1.28e-06	**1.30e-06**
	Sigmóide	**1.01e-06**	**1.26e-06**	1.25e-06
	SoftExponential	**1.01e-06**	1.22e-06	**1.46e-06**
	SoftPlus	**1.05e-06**	**1.23e-06**	1.23e-06
	SoftSign	**9.98e-07**	**1.25e-06**	1.24e-06
	Swish	**1.01e-06**	1.26e-06	**1.27e-06**
	Tangente Hiperbólica	**1.08e-06**	1.30e-06	**1.33e-06**
	BLU	**5.98e-05**	**8.05e-05**	8.00e-05
	Cúbico	**5.82e-05**	8.48e-05	**8.49e-05**
	Linear	**5.78e-05**	7.33e-05	**7.39e-05**
	Mish	**5.97e-05**	7.81e-05	**7.92e-05**
	PELU	**6.18e-05**	**8.48e-05**	8.39e-05
	PReLU	**6.07e-05**	**7.39e-05**	7.37e-05
7	Quadrática	**5.82e-05**	**7.96e-05**	7.87e-05
	SELU	**6.12e-05**	**8.06e-05**	7.99e-05
	Sigmóide	**5.96e-05**	**7.83e-05**	6.28e-05
	SoftExponential	**6.01e-05**	7.85e-05	**9.66e-05**
	SoftPlus	**5.96e-05**	7.67e-05	**7.83e-05**
	SoftSign	**5.96e-05**	7.59e-05	**7.74e-05**
	Swish	**5.97e-05**	7.82e-05	**7.91e-05**
	Tangente Hiperbólica	**5.99e-05**	**8.33e-05**	8.32e-05
	BLU	**1.17e-05**	1.55e-05	**1.57e-05**

Continua na próxima página

Base de Dados	Função de Ativação	Normal	Camada	Neurônio
8	Cúbico	**1.16e-05**	1.65e-05	**1.68e-05**
	Linear	**1.11e-05**	1.43e-05	**1.43e-05**
	Mish	**1.17e-05**	1.54e-05	**1.56e-05**
	PELU	**1.22e-05**	1.65e-05	**1.66e-05**
	PReLU	**1.18e-05**	1.43e-05	**1.44e-05**
	Quadrática	**1.15e-05**	1.53e-05	**1.54e-05**
	SELU	**1.20e-05**	1.55e-05	**1.58e-05**
	Sigmóide	**1.17e-05**	**1.52e-05**	1.22e-05
	SoftExponential	**1.15e-05**	1.50e-05	**1.90e-05**
	SoftPlus	**1.16e-05**	1.49e-05	**1.51e-05**
	SoftSign	**1.16e-05**	1.49e-05	**1.51e-05**
	Swish	**1.16e-05**	1.53e-05	**1.53e-05**
	Tangente Hiperbólica	**1.17e-05**	1.62e-05	**1.64e-05**
9	BLU	**6.00e-05**	8.12e-05	**8.18e-05**
	Cúbico	**4.80e-05**	8.65e-05	**9.28e-05**
	Linear	**5.90e-05**	7.49e-05	**7.50e-05**
	Mish	**6.12e-05**	**8.04e-05**	8.00e-05
	PELU	**6.34e-05**	8.67e-05	**8.68e-05**
	PReLU	**6.18e-05**	**7.65e-05**	7.59e-05
	Quadrática	**5.86e-05**	7.96e-05	**8.30e-05**
	SELU	**6.23e-05**	**8.16e-05**	8.16e-05
	Sigmóide	**6.08e-05**	8.02e-05	**8.21e-05**
	SoftExponential	**5.96e-05**	8.09e-05	**9.98e-05**
	SoftPlus	**6.08e-05**	**7.94e-05**	7.89e-05
	SoftSign	**6.01e-05**	**7.86e-05**	7.85e-05
	Swish	**6.03e-05**	8.10e-05	**8.11e-05**
	Tangente Hiperbólica	**6.18e-05**	8.61e-05	**9.06e-05**
10	BLU	**7.56e-05**	**1.02e-04**	1.02e-04
	Cúbico	**7.51e-05**	1.07e-04	**1.07e-04**
	Linear	**7.41e-05**	9.36e-05	**9.41e-05**
	Mish	**7.59e-05**	**1.01e-04**	1.00e-04
	PELU	**7.87e-05**	1.07e-04	**1.08e-04**
	PReLU	**7.65e-05**	9.41e-05	**9.45e-05**
	Quadrática	**7.39e-05**	1.00e-04	**1.01e-04**
	SELU	**7.82e-05**	1.02e-04	**1.02e-04**
	Sigmóide	**7.53e-05**	9.98e-05	**9.98e-05**

Continua na próxima página

Base de Dados	Função de Ativação	Normal	Camada	Neurônio
	SoftExponential	**7.58e-05**	1.06e-04	**1.24e-04**
	SoftPlus	**7.47e-05**	9.74e-05	**9.76e-05**
	SoftSign	**7.52e-05**	**9.79e-05**	9.78e-05
	Swish	**7.51e-05**	**1.01e-04**	9.97e-05
	Tangente Hiperbólica	**7.70e-05**	1.07e-04	**1.07e-04**

Fonte: Elaborada pelo autor (2023).

APÊNDICE C – COMPLEMENTO DO EXPERIMENTO III

Todas as tabelas e figuras apresentadas aqui utilizam um mapa de cores com brilho linearmente decrescente (ou crescente). Isso implica que as informações serão mantidas caso sejam impressas em preto e branco ou visualizadas por pessoas com deficiência de daltonismo.

As informações estão separadas em duas seções: uma referente aos resultados das frentes de Pareto e a outra às soluções extraídas das frentes de Pareto.

C.1 Análise da Frente de Pareto

As Tabelas 40 e 41 apresentam os resultados sumarizados do terceiro experimento numérico para a análise da frente de Pareto. Para cada base de dados, algoritmo de otimização e tipo adaptativo da ANN, exibimos, para as métricas Hipervolume e Espaçamento, o valor mínimo, médio, mediano e máximo, além do desvio padrão para as 30 execuções independentes. Os valores apresentados com a cor **"Azul Escuro"** representam os melhores valores encontrados, enquanto os valores apresentados com a cor **"Azul Claro"** representam os piores valores encontrados.

As Figuras 103 e 104 representam os *Violin plots* das métricas Hipervolume e Espaçamento, respectivamente, para os problemas do Experimento 3 para as 30 execuções independentes. Optamos por utilizar os *Violin plots* devido à sua capacidade de combinar as informações presentes em um *boxplot* com um gráfico de densidade.

Por fim, a Figura 105 apresenta as frentes de Pareto encontradas, sumarizando as soluções obtidas em 30 execuções independentes para diferentes bases de dados, tipos de estratégias (adaptativas ou não) e algoritmos de otimização.

Tabela 40 – Resultados do Experimento 3, para o otimizador NSGA-II, para as frentes de Pareto encontradas.

Base de Dados	Tipo Adaptativo	Hipervolume					Espaçamento				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
1	Normal	**0.481**	**0.620**	**0.058**	**0.626**	**0.706**	**6.096**	**11.925**	4.647	**10.100**	25.336
	Camada	0.572	0.681	0.051	0.682	0.753	5.166	11.443	**5.345**	**9.376**	**26.032**
	Neurônio	**0.590**	**0.685**	**0.045**	**0.691**	**0.754**	**4.808**	**10.433**	**3.790**	9.857	**21.975**
2	Normal	**0.155**	**0.453**	**0.097**	**0.454**	**0.633**	**1.869**	15.575	**10.872**	**11.064**	44.551
	Camada	**0.383**	**0.566**	0.107	**0.574**	**0.768**	**3.898**	**16.687**	10.496	**13.485**	**46.281**
	Neurônio	0.316	0.556	**0.123**	0.553	0.767	2.320	**13.938**	**8.482**	12.972	**43.823**
3	Normal	0.551	**0.632**	**0.039**	**0.636**	**0.701**	**5.723**	**18.735**	**7.817**	**18.213**	**43.446**
	Camada	**0.591**	**0.686**	0.041	**0.695**	**0.740**	**6.248**	**13.432**	**5.506**	**13.039**	**31.405**
	Neurônio	**0.493**	0.682	**0.045**	0.687	0.734	5.850	15.524	7.012	14.020	37.479
4	Normal	**0.503**	**0.582**	**0.034**	**0.584**	**0.645**	**8.768**	**13.880**	3.884	13.197	23.856
	Camada	0.535	0.623	**0.047**	0.627	**0.702**	8.748	**13.686**	**3.362**	**13.218**	**21.919**
	Neurônio	**0.552**	**0.640**	0.039	**0.647**	0.691	**6.727**	13.875	**4.802**	**12.806**	**29.182**
5	Normal	0.584	**0.677**	0.043	**0.670**	**0.743**	**5.814**	**27.662**	**13.853**	**24.901**	**63.125**
	Camada	**0.645**	0.710	**0.037**	0.709	0.764	**13.041**	**24.082**	**8.462**	21.804	**44.077**
	Neurônio	**0.555**	**0.736**	**0.050**	**0.752**	**0.794**	9.285	25.721	12.293	**20.820**	57.609

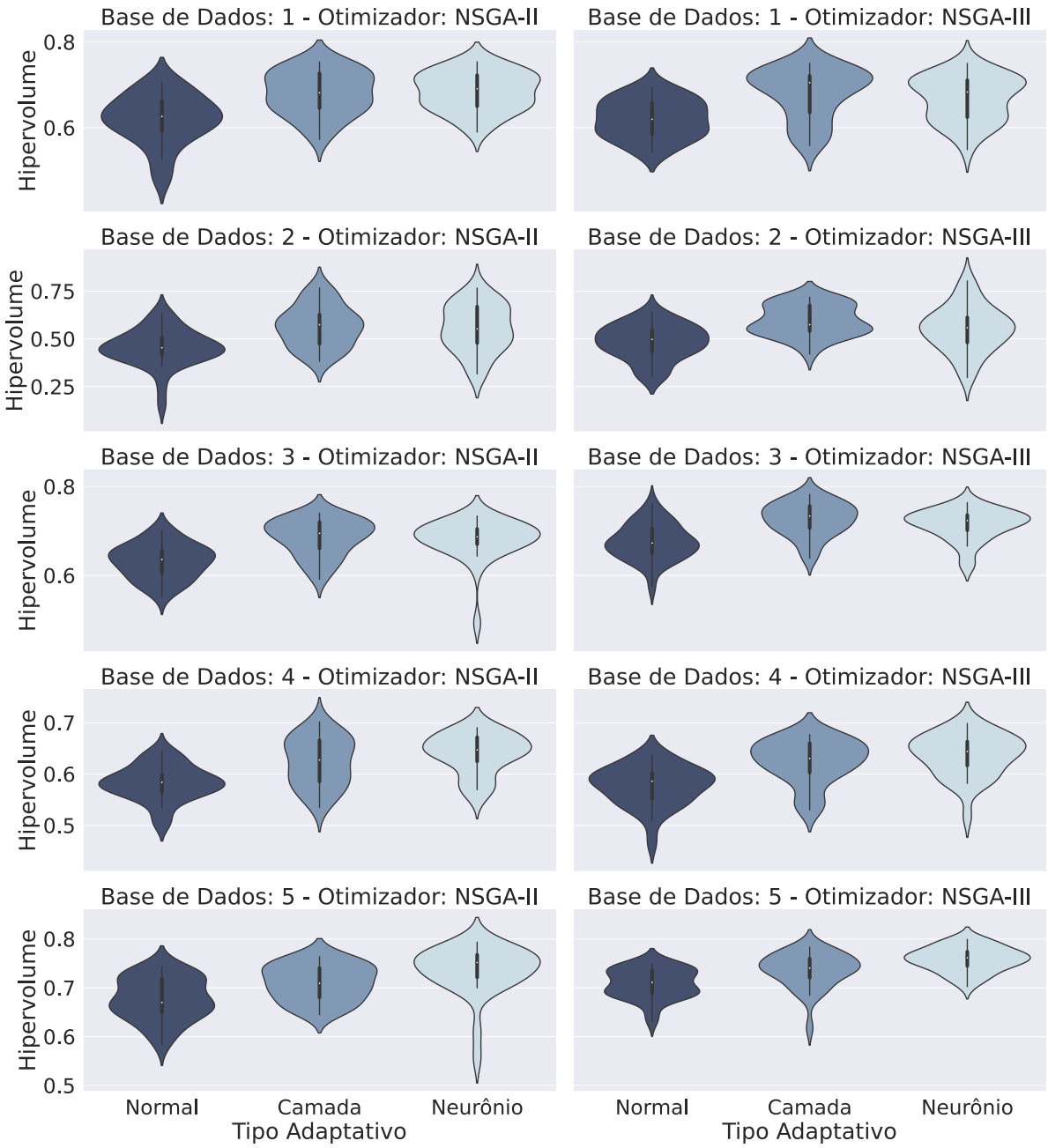
Fonte: Elaborada pelo autor (2023).

Tabela 41 – Resultados do Experimento 3, para o otimizador NSGA-III, para as frentes de Pareto encontradas.

Base de Dados	Tipo Adaptativo	Hipervolume					Espaçamento				
		$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
1	Normal	**0.542**	**0.620**	**0.045**	**0.620**	**0.695**	**6.685**	**15.296**	**7.047**	12.922	29.289
	Camada	**0.558**	**0.680**	**0.058**	**0.705**	**0.751**	**7.198**	**13.406**	**5.358**	**12.311**	**25.956**
	Neurônio	0.549	0.668	0.052	0.683	0.750	7.168	15.247	5.743	**14.687**	**33.353**
2	Normal	0.303	**0.483**	0.090	**0.498**	**0.639**	**5.132**	14.988	8.853	**13.277**	35.545
	Camada	**0.420**	**0.596**	**0.081**	**0.575**	0.719	**3.577**	**15.237**	**9.615**	12.750	**39.629**
	Neurônio	**0.298**	0.552	**0.119**	0.559	**0.804**	3.819	**11.625**	**4.997**	**10.515**	**24.038**
3	Normal	**0.575**	**0.675**	**0.040**	**0.673**	**0.761**	**7.455**	**22.549**	**9.299**	20.240	**38.605**
	Camada	**0.638**	**0.726**	0.037	**0.734**	**0.782**	**9.343**	22.359	9.062	**22.838**	37.684
	Neurônio	0.622	0.714	**0.034**	0.723	0.764	9.137	**20.443**	**7.789**	**18.514**	**35.874**
4	Normal	**0.465**	**0.577**	**0.039**	**0.586**	**0.636**	**9.633**	19.469	**6.696**	**16.997**	32.434
	Camada	**0.530**	0.625	**0.042**	0.630	0.677	**10.200**	**21.636**	6.264	**20.314**	**37.769**
	Neurônio	0.518	**0.637**	0.041	**0.644**	**0.699**	9.854	**18.859**	**4.707**	18.834	**26.739**
5	Normal	0.631	**0.708**	0.030	**0.711**	**0.750**	**11.889**	**39.294**	**16.958**	38.062	65.187
	Camada	**0.618**	0.736	**0.035**	0.740	0.783	12.922	39.805	**12.449**	**41.776**	**60.493**
	Neurônio	**0.702**	**0.758**	**0.025**	**0.761**	**0.799**	**16.285**	**40.214**	15.214	**36.954**	**73.154**

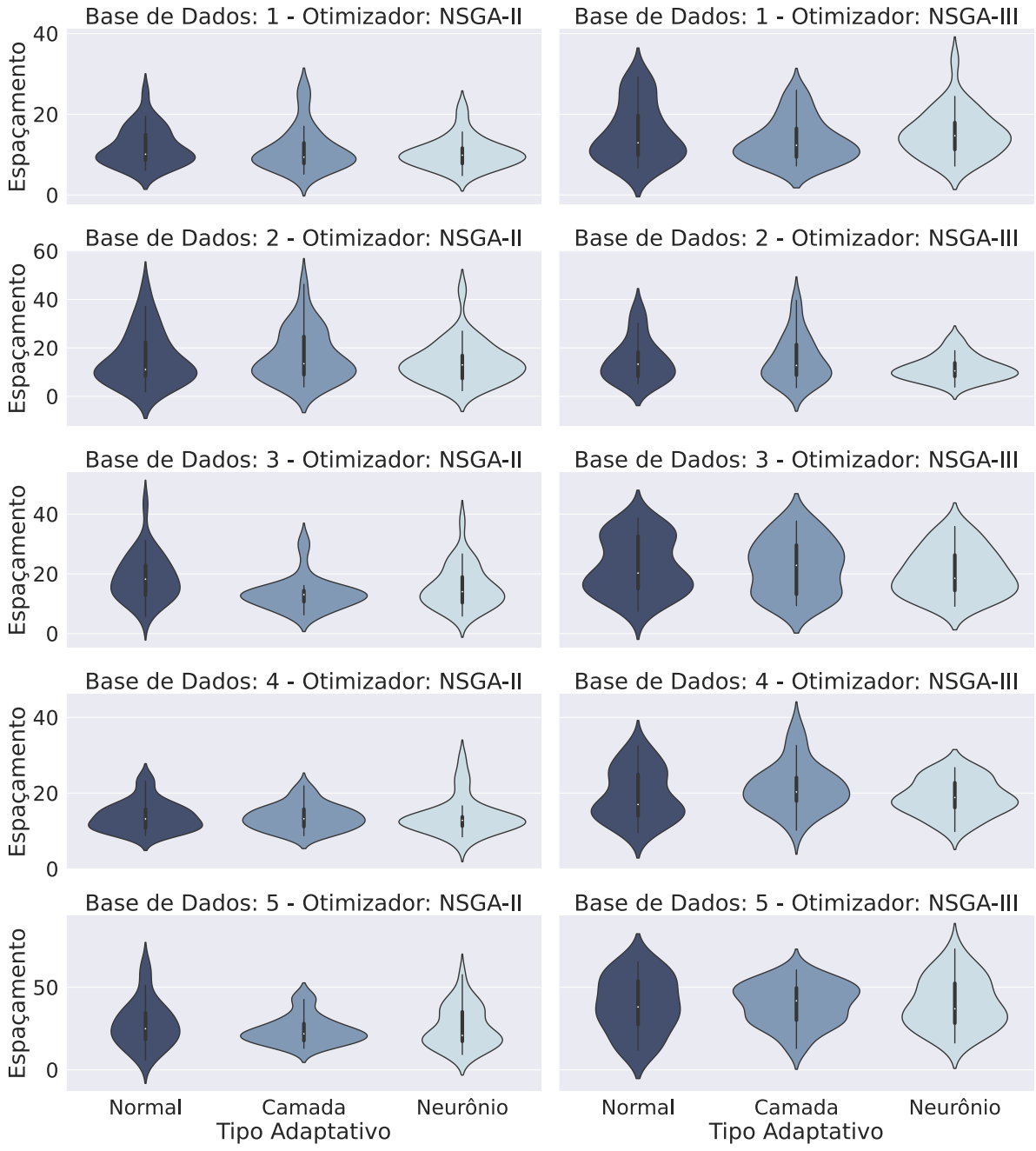
Fonte: Elaborada pelo autor (2023).

Figura 103 – *Violin Plots* do Experimento 3 para a métrica Hipervolume.



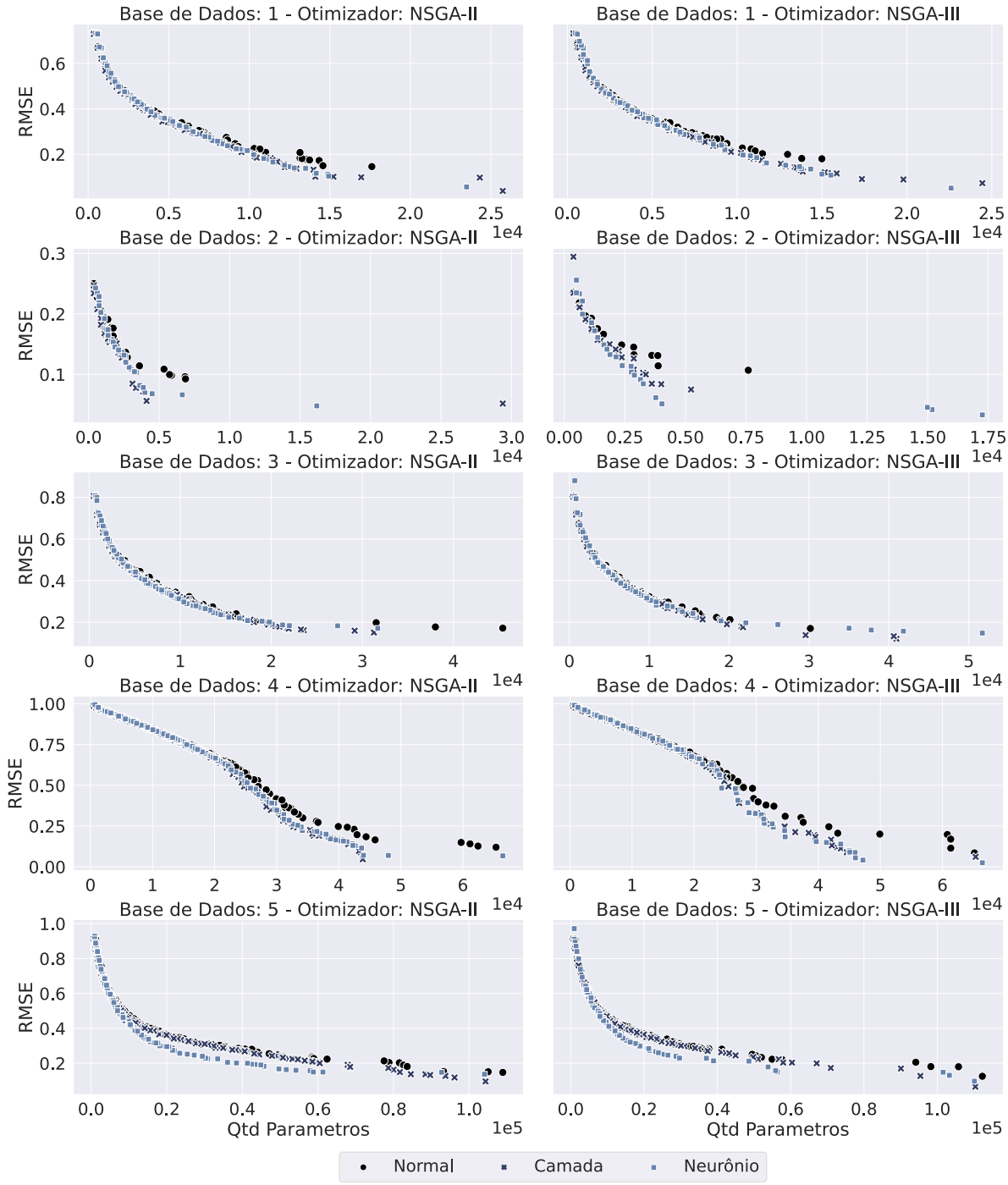
Fonte: Elaborada pelo autor (2023).

Figura 104 – *Violin Plots* do Experimento 3 para a métrica Espaçamento.



Fonte: Elaborada pelo autor (2023).

Figura 105 – Frentes de Pareto do Experimento 3.



Fonte: Elaborada pelo autor (2023).

C.2 Análise de Soluções da Frente de Pareto

As Tabelas 42 e 43 apresentam os resultados sumarizados do terceiro experimento numérico para a análise de soluções extraídas da frente de Pareto. Para cada base de dados, algoritmo de otimização, tipo de solução extraída e tipo adaptativo da ANN, exibimos os valores mínimos, médios, medianos e máximos, além do desvio padrão para as 30 execuções independentes das métricas RMSE e Quantidade de Parâmetros. Os valores apresentados com a cor **"Azul Escuro"** representam os melhores resultados encontrados, enquanto os valores apresentados com a cor **"Azul Claro"** representam os piores resultados encontrados.

As Figuras 106 e 107 apresentam os *Violin plots* das métricas RMSE e Quantidade de Parâmetros, respectivamente, para os problemas do Experimento 3 para as 30 execuções independentes. Optamos por utilizar os *Violin plots* devido à sua capacidade de combinar as informações presentes em um *boxplot* com um gráfico de densidade.

Tabela 42 – Resultados do Experimento 3, para o otimizador NSGA-II, para diferentes soluções da frente de Pareto.

Base de Dados	Pareto	Tipo Adaptativo	RMSE					Qtd Parâmetros				
			$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
1	Melhor em Métrica	Normal	**0.265**	**0.431**	**0.131**	**0.398**	**0.709**	11493	**17970**	5156	**15130**	**27377**
		Camada	**0.068**	0.261	0.125	0.226	0.509	**9600**	**17609**	**5977**	15193	27850
		Neurônio	0.075	**0.240**	**0.100**	**0.217**	**0.498**	**12062**	17737	**4917**	**15345**	**28075**
	Melhor em Parâmetro	Normal	**1.207**	**1.305**	**0.044**	**1.293**	1.425	**352**	**388**	**82**	**353**	**587**
		Camada	1.187	**1.274**	0.045	**1.275**	**1.450**	354	403	91	356	594
		Neurônio	**1.145**	1.283	**0.053**	1.281	**1.378**	**470**	**561**	**127**	**474**	**945**
	Próximo da Origem	Normal	0.342	**0.747**	**0.158**	**0.796**	**0.954**	**2937**	**5254**	**1678**	**4856**	**9321**
		Camada	**0.352**	0.693	0.146	0.729	0.920	3201	5394	1446	5156	9049
		Neurônio	**0.340**	**0.666**	**0.137**	**0.691**	**0.854**	**3683**	**5595**	**1286**	**5518**	**8443**
2	Melhor em Métrica	Normal	**0.092**	**0.143**	**0.034**	**0.141**	0.238	**2624**	12567	6899	**12934**	**28738**
		Camada	0.045	**0.106**	0.035	0.103	**0.190**	3389	**13900**	**9463**	10376	**31127**
		Neurônio	**0.041**	0.108	**0.044**	**0.091**	**0.240**	**3635**	**10980**	**6875**	**8560**	30232
	Melhor em Parâmetro	Normal	0.211	**0.332**	0.072	**0.304**	0.570	**373**	424	**127**	**373**	**871**
		Camada	**0.196**	0.329	**0.095**	**0.284**	**0.593**	375	**418**	112	376	886
		Neurônio	**0.272**	**0.304**	**0.048**	0.285	**0.473**	**498**	**594**	**112**	**622**	**998**
	Próximo da Origem	Normal	**0.107**	**0.196**	**0.056**	**0.190**	0.313	**623**	**2508**	**1257**	**2194**	**4408**
		Camada	**0.063**	0.165	**0.089**	0.165	**0.535**	**875**	**3215**	**1844**	**3252**	**8220**
		Neurônio	0.069	**0.157**	**0.056**	**0.148**	**0.286**	750	3022	1592	2821	7651
3	Melhor em Métrica	Normal	**11.649**	**20.215**	**3.619**	**20.597**	27.150	**17234**	**32694**	8832	**34101**	**45423**
		Camada	**9.204**	16.006	**4.208**	15.785	**25.025**	**10680**	**25831**	**8621**	**23408**	**43307**

Continua na próxima página

Tabela 42 – Resultados do Experimento 3, para o otimizador NSGA-II, para diferentes soluções da frente de Pareto.

Base de Dados	Pareto	Tipo Adaptativo	RMSE					Qtd Parâmetros				
			$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
4		Neurônio	10.153	**15.916**	3.721	**15.204**	**30.090**	16484	30083	**9366**	31824	43893
	Melhor em Parâmetro	Normal	**50.450**	**64.283**	**3.943**	**64.635**	73.728	**505**	635	**261**	**507**	**1688**
		Camada	**64.240**	**65.866**	**2.944**	64.768	**76.437**	507	**525**	**51**	508	**677**
		Neurônio	53.390	64.622	3.184	**64.823**	**68.528**	**674**	**810**	162	**842**	1353
	Próximo da Origem	Normal	19.057	**28.656**	4.168	**29.074**	**35.507**	4898	**9492**	**2658**	**9008**	14194
		Camada	**22.579**	28.409	**3.705**	27.856	37.534	**3222**	**7649**	2387	**7090**	**12640**
		Neurônio	**18.448**	**26.020**	**4.303**	**25.658**	**38.652**	**4924**	8801	**2296**	8979	**14362**
	Melhor em Métrica	Normal	**0.051**	**0.103**	0.035	**0.091**	**0.185**	**40653**	**50615**	**8670**	**45948**	65520
		Camada	**0.019**	0.071	**0.036**	0.071	0.142	41203	**48563**	**6045**	45976	**63896**
		Neurônio	0.027	**0.056**	**0.026**	**0.048**	**0.114**	**43245**	50260	8308	**46222**	**66780**
	Melhor em Parâmetro	Normal	**0.420**	**0.425**	**0.002**	**0.425**	**0.430**	**541**	**621**	**176**	**543**	**1266**
		Camada	0.417	**0.424**	**0.003**	**0.425**	**0.430**	544	678	242	546	1449
		Neurônio	**0.416**	**0.425**	**0.003**	**0.425**	**0.430**	**722**	**1084**	**281**	**906**	**1806**
	Próximo da Origem	Normal	**0.132**	**0.229**	**0.055**	**0.249**	**0.310**	**18327**	**26312**	**4921**	**24984**	35557
		Camada	0.122	0.189	**0.055**	0.167	**0.288**	20402	27384	3260	28278	**32764**
		Neurônio	**0.086**	**0.161**	**0.047**	**0.158**	0.292	**20868**	**29130**	**3149**	**29022**	**36572**
5	Melhor em Métrica	Normal	**0.309**	**0.620**	0.153	**0.642**	**0.889**	**25110**	80990	**27013**	85344	115917
		Camada	**0.219**	0.517	**0.171**	0.521	**0.802**	**45939**	**83143**	**24016**	**90431**	**114722**
		Neurônio	0.283	**0.459**	**0.128**	**0.444**	0.839	31208	**72562**	24917	**61580**	**116872**
	Melhor em Parâmetro	Normal	1.926	2.204	**0.071**	**2.212**	**2.385**	**721**	**850**	**235**	**723**	**1686**

Continua na próxima página

Tabela 42 – Resultados do Experimento 3, para o otimizador NSGA-II, para diferentes soluções da frente de Pareto.

Base de Dados	Pareto	Tipo Adaptativo	RMSE					Qtd Parâmetros				
			$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
		Camada	**1.932**	**2.207**	0.083	2.206	2.362	723	853	274	726	1692
		Neurônio	**1.885**	**2.155**	**0.086**	**2.167**	**2.282**	**962**	**1372**	**402**	**1203**	**2660**
	Próximo da Origem	Normal	**0.773**	**0.941**	0.111	**0.923**	**1.242**	**7485**	18478	5062	17162	**26562**
		Camada	0.746	0.877	**0.082**	0.852	1.050	**12321**	**19709**	**5232**	**19243**	30008
		Neurônio	**0.571**	**0.794**	**0.114**	**0.797**	**1.030**	10377	**18077**	**5060**	**16308**	**32172**

Fonte: Elaborada pelo autor (2023).

Tabela 43 – Resultados do Experimento 3, para o otimizador NSGA-III, para diferentes soluções da frente de Pareto.

Base de Dados	Pareto	Tipo Adaptativo	RMSE					Qtd Parâmetros				
			$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
1	Melhor em Métrica	Normal	0.264	0.432	**0.118**	0.418	0.664	11490	18712	5889	15221	27963
		Camada	0.083	0.297	0.155	0.255	0.655	**6936**	**15692**	**5023**	**14604**	**26902**
		Neurônio	**0.065**	**0.286**	0.127	**0.243**	**0.545**	10109	17591	5362	15110	28431
	Melhor em Parâmetro	Normal	1.269	1.319	0.064	1.293	**1.517**	**352**	**368**	**51**	**352**	**586**
		Camada	1.210	**1.296**	**0.062**	**1.277**	1.543	354	372	60	356	594
		Neurônio	**1.168**	1.306	0.078	1.289	1.562	470	582	148	588	1082
	Próximo da Origem	Normal	0.577	0.763	**0.104**	0.788	0.929	2467	**5032**	1526	**4374**	7701
		Camada	**0.387**	0.712	0.172	0.770	**0.987**	**2001**	5037	**1711**	4820	**9126**
		Neurônio	0.441	**0.711**	0.120	**0.722**	**0.920**	2969	5207	**1245**	**4824**	**7638**
2	Melhor em Métrica	Normal	0.092	0.141	0.037	0.134	0.230	**1754**	14901	9726	13811	31248
		Camada	0.066	**0.111**	**0.030**	**0.112**	**0.161**	2618	**10432**	7979	**7674**	31128
		Neurônio	**0.033**	0.123	0.044	0.123	0.249	3900	11339	**6714**	8498	**28832**
	Melhor em Parâmetro	Normal	0.248	0.343	0.068	0.332	0.557	**373**	**394**	**66**	**373**	**622**
		Camada	0.243	0.336	0.105	**0.281**	0.609	375	427	193	376	1407
		Neurônio	**0.191**	**0.315**	**0.057**	0.301	**0.486**	498	570	126	500	878
	Próximo da Origem	Normal	0.103	0.189	0.046	0.185	0.301	**626**	2779	1835	**2132**	7902
		Camada	0.091	**0.156**	**0.039**	**0.160**	**0.248**	875	**2602**	**997**	2554	**4236**
		Neurônio	**0.055**	0.175	0.055	0.167	0.311	1010	3157	1642	2574	7322
3	Melhor em Métrica	Normal	13.554	20.857	**3.657**	20.951	28.394	20732	32768	**8270**	36046	43560
		Camada	**7.941**	**15.020**	4.366	**14.186**	26.032	16491	31495	8705	34208	**43308**

Continua na próxima página

Tabela 43 – Resultados do Experimento 3, para o otimizador NSGA-III, para diferentes soluções da frente de Pareto.

Base de Dados	Pareto	Tipo Adaptativo	RMSE					Qtd Parâmetros				
			$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
4		Neurônio	9.361	15.746	4.351	15.244	**25.605**	**15508**	**29441**	10508	**26097**	51693
	Melhor em Parâmetro	Normal	59.191	64.946	**1.520**	64.675	**68.241**	**505**	635	191	**507**	**1179**
		Camada	**52.294**	**63.835**	3.400	64.553	68.634	508	**576**	180	509	1351
		Neurônio	56.712	65.009	2.385	**64.382**	71.853	674	799	**159**	842	1350
	Próximo da Origem	Normal	22.679	28.478	**3.129**	28.485	36.057	6435	9041	**2021**	8980	15084
		Camada	17.889	26.309	3.707	**25.904**	**34.279**	**4228**	8801	2522	9141	15258
		Neurônio	**17.881**	**26.226**	4.261	26.309	35.125	4911	**8655**	2510	**8494**	**14616**
	Melhor em Métrica	Normal	0.038	0.105	0.038	0.105	0.199	**40833**	52001	9063	46770	66060
		Camada	0.017	0.070	**0.029**	0.066	**0.137**	42421	50999	8546	46336	**65705**
		Neurônio	**0.010**	**0.062**	0.034	**0.053**	0.161	40870	**48083**	**7308**	**45798**	66420
	Melhor em Parâmetro	Normal	0.420	0.426	0.003	**0.425**	0.434	**541**	681	183	**543**	1088
		Camada	0.420	0.426	**0.002**	**0.425**	**0.430**	544	**618**	**122**	546	**908**
		Neurônio	**0.418**	**0.425**	0.003	**0.425**	**0.430**	722	1013	254	905	1629
	Próximo da Origem	Normal	0.147	0.233	0.056	0.238	0.332	17967	**26164**	5138	**26280**	39381
		Camada	**0.085**	0.183	0.064	0.167	**0.303**	**17692**	28300	5046	29426	39390
		Neurônio	0.095	**0.163**	**0.048**	**0.157**	0.316	19416	28941	**3079**	29333	**35288**
5	Melhor em Métrica	Normal	0.313	0.584	0.137	0.581	0.857	**29703**	81926	28278	94764	**113750**
		Camada	**0.135**	0.455	0.182	0.468	0.942	42813	88208	**24325**	97146	116164
		Neurônio	0.219	**0.426**	**0.096**	**0.425**	**0.674**	31692	**80613**	25752	**83013**	117360
	Melhor em Parâmetro	Normal	**1.577**	**2.177**	0.144	2.207	2.382	**721**	955	591	**722**	3628

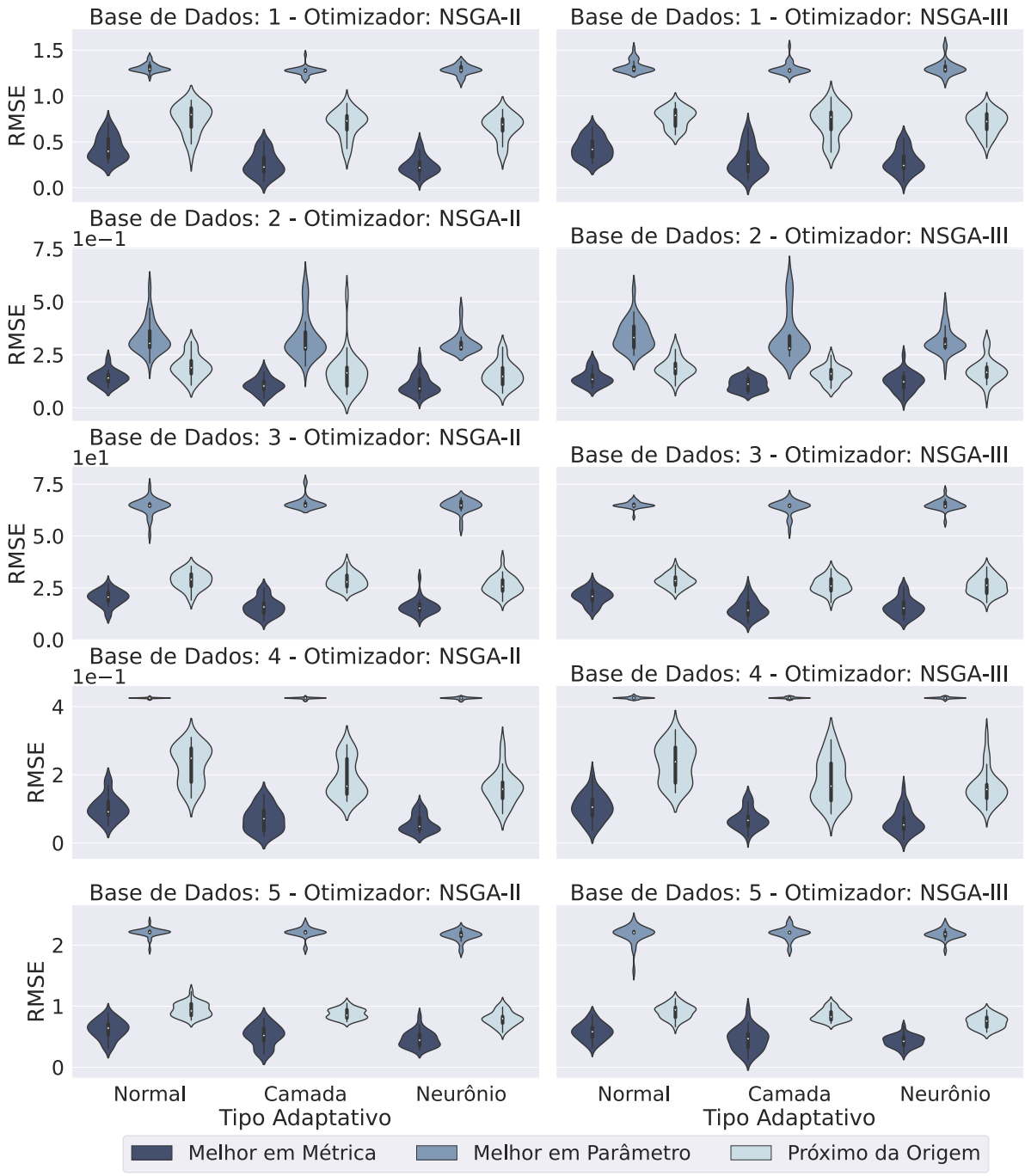
Continua na próxima página

Tabela 43 – Resultados do Experimento 3, para o otimizador NSGA-III, para diferentes soluções da frente de Pareto.

Base de Dados	Pareto	Tipo Adaptativo	RMSE					Qtd Parâmetros				
			$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
		Camada	**1.918**	**2.192**	0.099	2.206	2.370	723	**909**	314	725	**1690**
		Neurônio	1.914	2.181	**0.079**	**2.182**	**2.360**	**962**	**1300**	**280**	**1203**	2172
	Próximo da Origem	Normal	0.664	**0.913**	**0.111**	**0.940**	**1.125**	**8952**	19686	**6845**	**17404**	**35572**
		Camada	**0.719**	0.853	**0.095**	0.827	1.056	**12052**	**20745**	5377	**21407**	29404
		Neurônio	**0.574**	**0.744**	0.102	**0.759**	**0.938**	11322	**19258**	**4187**	19350	**26415**

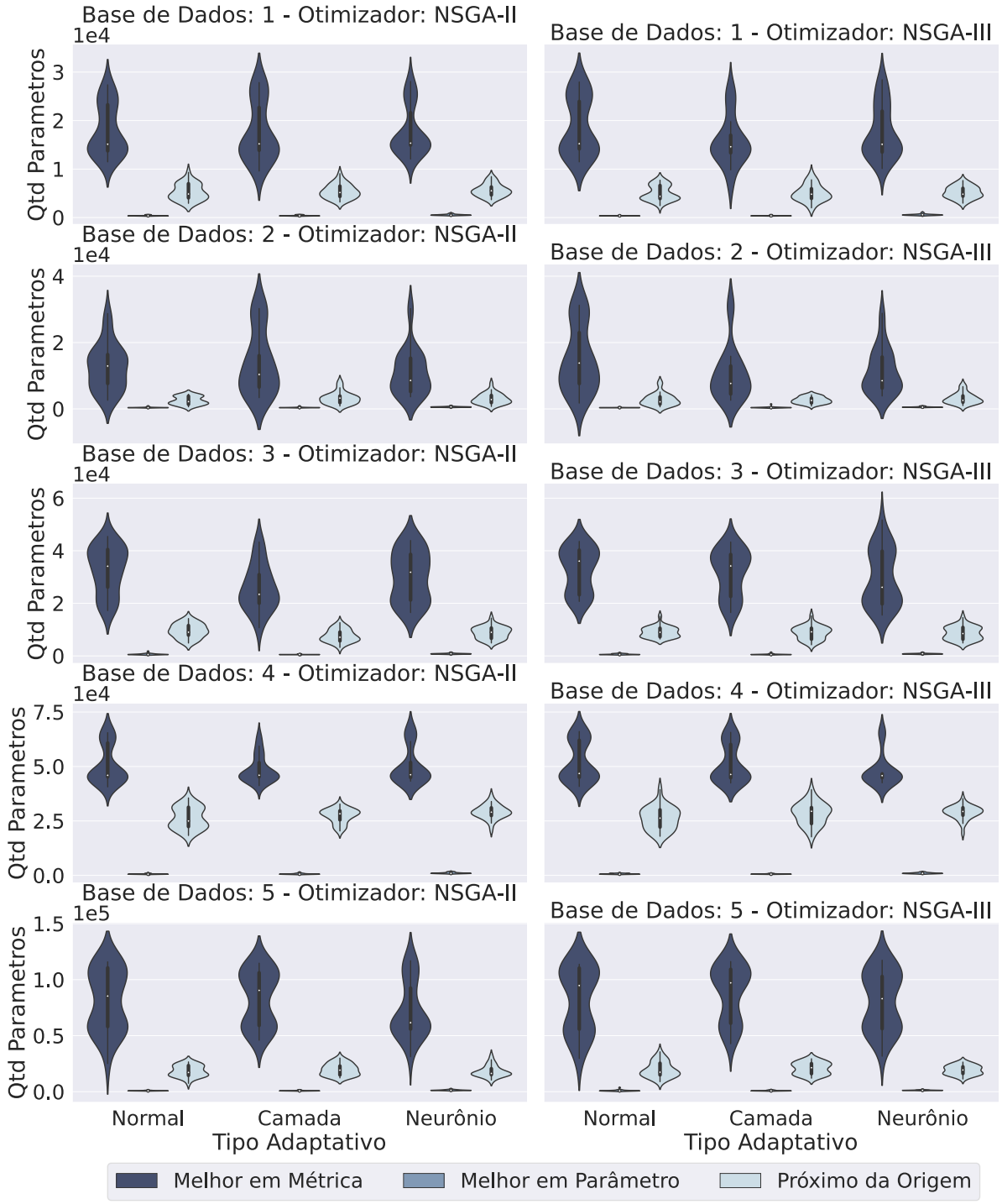
Fonte: Elaborada pelo autor (2023).

Figura 106 – *Violin Plots* do Experimento 3 para a métrica RMSE.



Fonte: Elaborada pelo autor (2023).

Figura 107 – *Violin Plots* do Experimento 3 para a Quantidade de Parâmetros.



Fonte: Elaborada pelo autor (2023).

C.3 Pegada de Carbono

A Tabela 44 exibe as estimativas médias de emissão de CO_2 em quilogramas associadas ao treinamento das três soluções retiradas da frente de Pareto encontrada, agregando todas as 30 execuções independentes do terceiro experimento. Essas soluções incluem aquela que apresentou o melhor desempenho em métricas, a que apresentou o menor número de parâmetros e, por fim, a solução intermediária que está mais próxima da origem de acordo com a distância euclidiana. Os valores apresentados com a cor **"Azul Escuro"** representam os melhores resultados encontrados, enquanto os valores apresentados com a cor **"Azul Claro"** representam os piores resultados encontrados.

Tabela 44 – Estimativas de Emissões de CO_2, em quilograma, do Experimento 3.

Base de Dados	Otimizador	Emissões de CO_2 (kg)								
		Parâmetro			Origem			Métrica		
		Normal	Camada	Neurônio	Normal	Camada	Neurônio	Normal	Camada	Neurônio
1	NSGA-II	**4.11e-07**	8.58e-07	**1.17e-06**	5.87e-07	**3.80e-07**	**9.59e-07**	4.35e-07	**4.24e-07**	**7.32e-07**
	NSGA-III	**6.38e-07**	1.15e-06	**1.27e-06**	4.60e-07	**2.61e-07**	**6.35e-07**	**3.27e-07**	7.35e-07	**1.56e-06**
2	NSGA-II	4.68e-07	**4.32e-07**	**5.25e-07**	**2.17e-07**	**9.12e-07**	2.99e-07	**2.90e-07**	4.11e-07	**1.03e-06**
	NSGA-III	6.51e-07	**3.01e-06**	**4.88e-07**	**2.97e-07**	3.70e-07	**1.13e-06**	**2.37e-07**	7.95e-07	**6.10e-06**
3	NSGA-II	**4.76e-07**	**2.97e-06**	8.98e-07	4.71e-07	**4.08e-07**	**1.17e-06**	**1.00e-06**	**1.60e-06**	1.29e-06
	NSGA-III	**4.56e-07**	**4.79e-07**	4.69e-07	5.60e-07	**7.63e-07**	**5.35e-07**	**4.16e-07**	**2.52e-07**	3.64e-07
4	NSGA-II	**2.33e-06**	**1.33e-06**	1.47e-06	**1.06e-06**	**2.39e-06**	1.49e-06	2.42e-06	**5.84e-07**	**3.05e-06**
	NSGA-III	8.88e-07	**8.48e-07**	**2.04e-06**	**2.33e-06**	**1.16e-06**	2.23e-06	**3.44e-06**	1.38e-06	**9.29e-07**
5	NSGA-II	8.11e-07	**3.29e-07**	**1.33e-06**	1.13e-06	**1.31e-06**	**9.23e-07**	**1.45e-06**	3.81e-06	**4.87e-06**
	NSGA-III	**9.39e-07**	**1.31e-06**	1.25e-06	1.22e-06	**9.59e-07**	**2.48e-06**	**5.90e-07**	**2.58e-06**	8.94e-07

Fonte: Elaborada pelo autor (2023).

APÊNDICE D – COMPLEMENTO DO EXPERIMENTO IV

Todas as tabelas e figuras apresentadas aqui utilizam um mapa de cores com brilho linearmente decrescente (ou crescente). Isso implica que as informações serão mantidas caso sejam impressas em preto e branco ou visualizadas por pessoas com deficiência de daltonismo.

As informações estão separadas em duas seções: uma referente aos resultados das frentes de Pareto e a outra às soluções extraídas das frentes de Pareto.

D.1 Análise da Frente de Pareto

As Tabelas 45 e 46 apresentam os resultados sumarizados do quarto experimento numérico para a análise da frente de Pareto. Para cada algoritmo de otimização e tipo adaptativo da ANN, exibimos, para as métricas Hipervolume e Espaçamento, o valor mínimo, médio, mediano e máximo, além do desvio padrão para as 30 execuções independentes. Os valores apresentados com a cor **"Azul Escuro"** representam os melhores valores encontrados, enquanto os valores apresentados com a cor **"Azul Claro"** representam os piores valores encontrados.

As Figuras 108 e 109 representam os *Violin plots* das métricas Hipervolume e Espaçamento, respectivamente, para o problema do Experimento 4 para as 30 execuções independentes. Optamos por utilizar os *Violin plots* devido à sua capacidade de combinar as informações presentes em um *boxplot* com um gráfico de densidade.

Por fim, a Figura 110 apresenta as frentes de Pareto encontradas, sumarizando as soluções obtidas em 10 execuções independentes para os tipos de estratégias (adaptativas ou não) e algoritmo de otimização.

Tabela 45 – Resultados do Experimento 4, para o otimizador NSGA-II, para a frente de Pareto encontrada.

	Hipervolume					Espaçamento				
Tipo Adaptativo	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
Normal	**0.841**	**0.866**	**0.013**	**0.866**	**0.887**	**46.658**	**115.609**	**39.525**	**114.368**	**179.510**
Camada	0.896	0.922	0.012	0.926	0.935	31.169	**89.037**	40.692	**81.538**	**162.427**
Neurônio	**0.915**	**0.930**	**0.009**	**0.927**	**0.945**	**27.725**	93.720	**45.899**	83.735	178.320

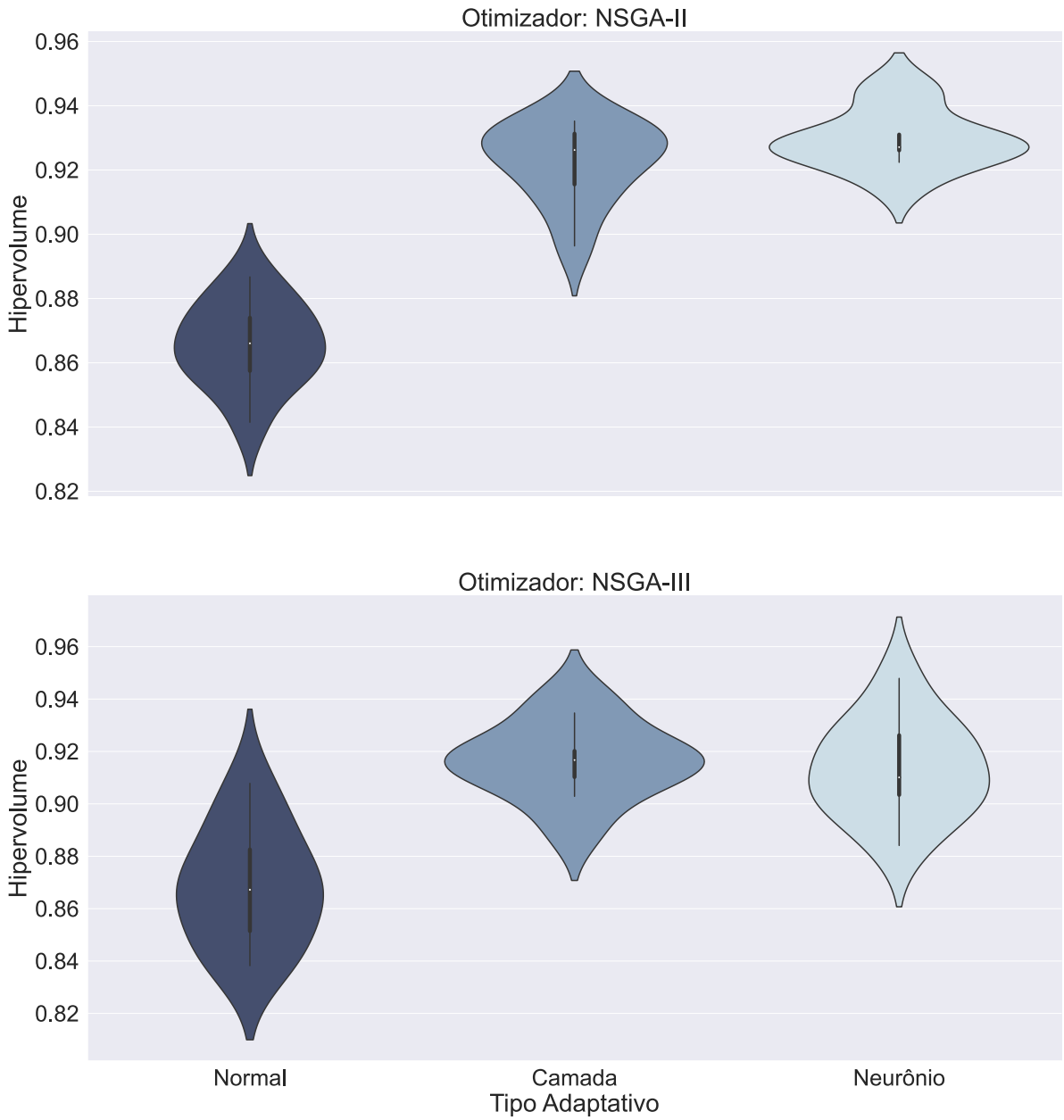
Fonte: Elaborada pelo autor (2023).

Tabela 46 – Resultados do Experimento 4, para o otimizador NSGA-III, para a frente de Pareto encontrada.

	Hipervolume					Espaçamento				
Tipo Adaptativo	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
Normal	**0.838**	**0.868**	**0.022**	**0.867**	**0.908**	19.247	**137.302**	**89.251**	**151.150**	**237.232**
Camada	**0.889**	**0.916**	**0.015**	**0.917**	0.940	**14.032**	126.591	**68.332**	**118.543**	**225.258**
Neurônio	0.884	0.913	0.019	0.910	**0.948**	**25.687**	**113.269**	71.286	124.415	235.450

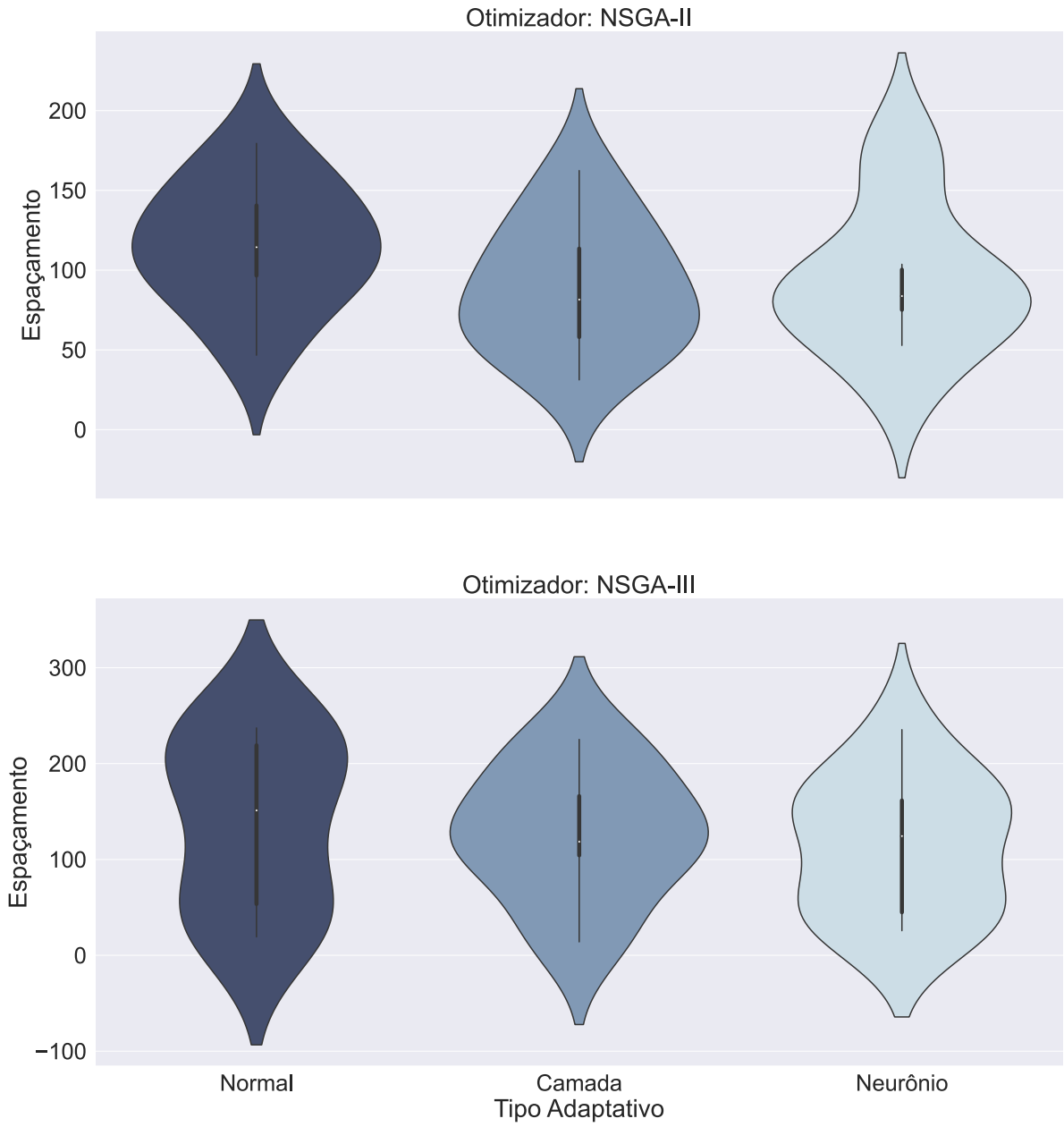
Fonte: Elaborada pelo autor (2023).

Figura 108 – *Violin Plots* do Experimento 4 para a métrica Hipervolume.



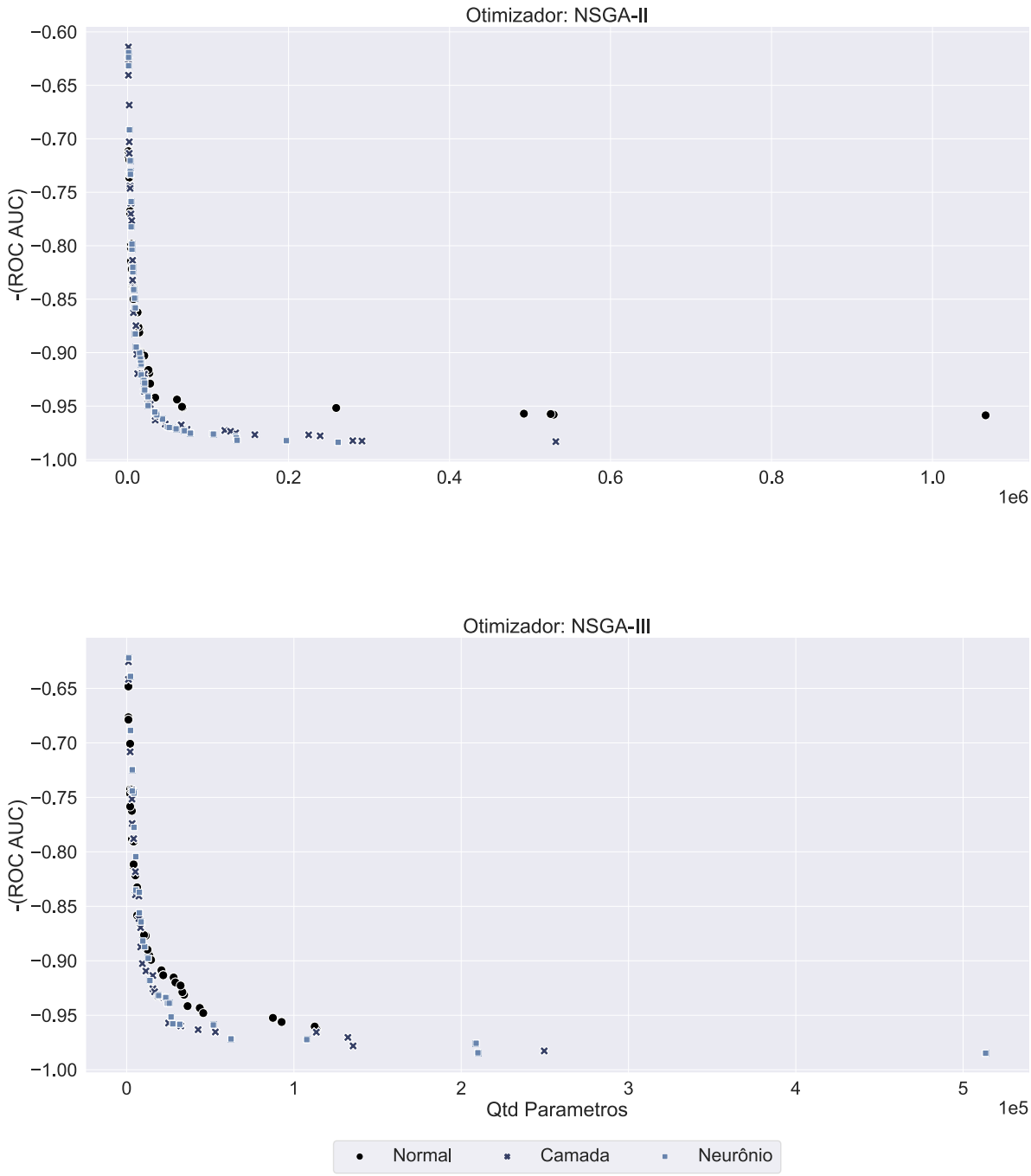
Fonte: Elaborada pelo autor (2023).

Figura 109 – *Violin Plots* do Experimento 4 para a métrica Espaçamento.



Fonte: Elaborada pelo autor (2023).

Figura 110 – Frentes de Pareto do Experimento 4.



Fonte: Elaborada pelo autor (2023).

D.2 Análise de Soluções da Frente de Pareto

As Tabelas 47 e 48 apresentam os resultados sumarizados do quarto experimento numérico para a análise de soluções extraídas da frente de Pareto. Para cada algoritmo de otimização, tipo de solução extraída e tipo adaptativo da ANN, exibimos os valores mínimos, médios, medianos e máximos, além do desvio padrão para as 10 execuções independentes das métricas ROC AUC e Quantidade de Parâmetros. Os valores apresentados com a cor **"Azul Escuro"** representam os melhores resultados encontrados, enquanto os valores apresentados com a cor **"Azul Claro"** representam os piores resultados encontrados.

As Figuras 106 e 107 apresentam os *Violin plots* das métricas ROC AUC e Quantidade de Parâmetros, respectivamente, para os problemas do Experimento 4 para as 10 execuções independentes. Optamos por utilizar os *Violin plots* devido à sua capacidade de combinar as informações presentes em um *boxplot* com um gráfico de densidade.

Tabela 47 – Resultados do Experimento 4, para o otimizador NSGA-II, para diferentes soluções da frente de Pareto.

		ROC AUC					Qtd Parametros				
Pareto	Tipo Adaptativo	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
Melhor em Métrica	Normal	**0.786**	**0.937**	**0.056**	**0.955**	**0.973**	187361	**565685**	**288593**	**526240**	1065999
	Camada	0.969	0.982	0.007	0.982	**0.992**	**198889**	541292	**348616**	**398760**	**1032776**
	Neurônio	**0.976**	**0.985**	**0.006**	**0.988**	**0.992**	**125443**	**500854**	333852	457940	**1088611**
Melhor em Parâmetro	Normal	**0.592**	**0.632**	**0.039**	**0.625**	0.704	**1055**	**1266**	**658**	**1056**	**3138**
	Camada	**0.556**	0.627	**0.048**	0.623	**0.709**	1081	**1824**	**1379**	1095	**5227**
	Neurônio	0.563	**0.610**	0.041	**0.612**	**0.677**	**1263**	1585	708	**1276**	3369
Próximo da Origem	Normal	**0.886**	**0.934**	**0.025**	**0.937**	**0.966**	20816	**63812**	**33650**	**61408**	**131181**
	Camada	**0.916**	0.951	**0.025**	0.955	**0.984**	**26032**	47584	**23304**	**35428**	**94708**
	Neurônio	**0.916**	**0.964**	**0.020**	**0.970**	**0.984**	**20395**	**45720**	26008	41472	105675

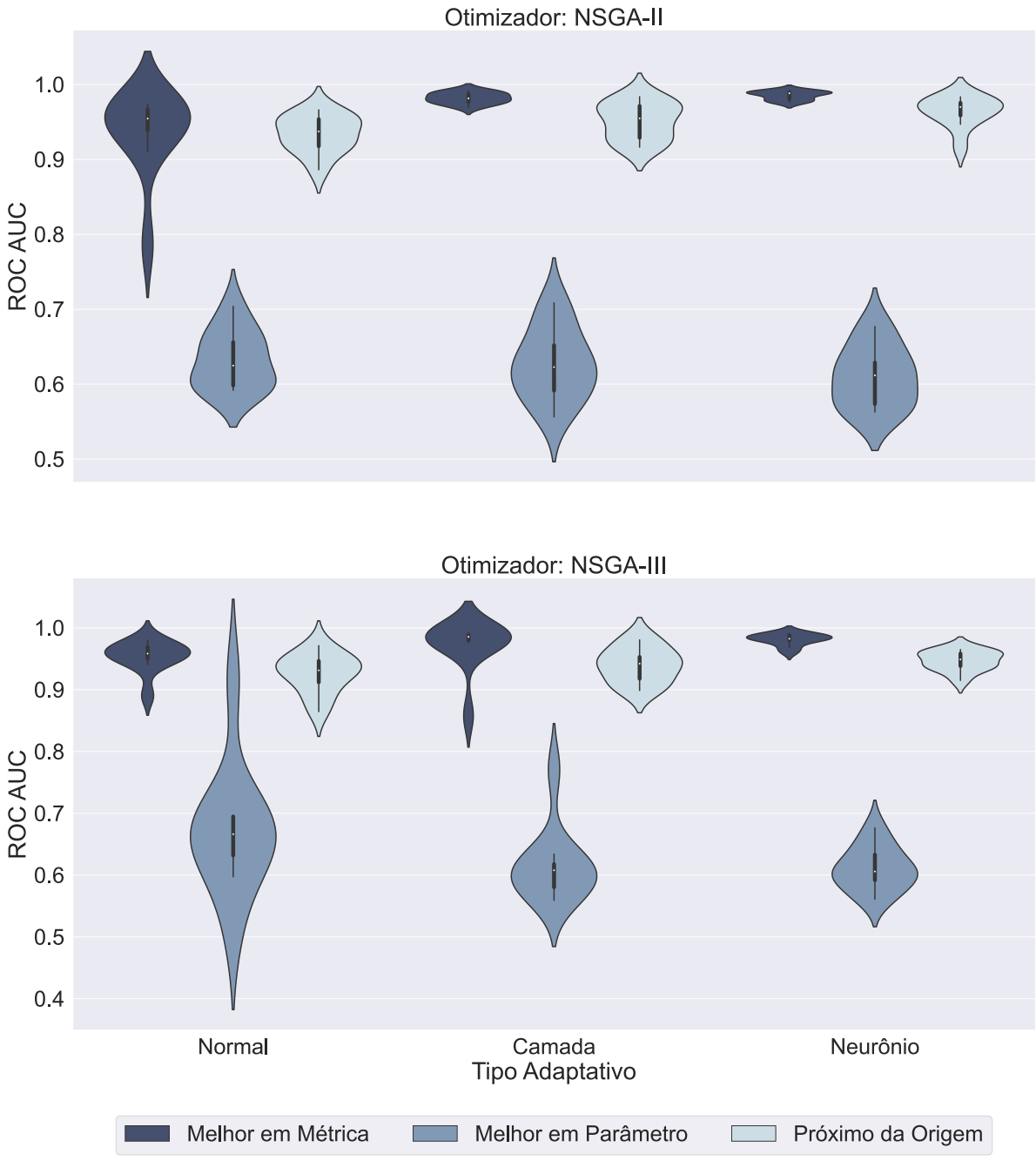
Fonte: Elaborada pelo autor (2023).

Tabela 48 – Resultados do Experimento 4, para o otimizador NSGA-III, para diferentes soluções da frente de Pareto.

		ROC AUC					Qtd Parametros				
Pareto	Tipo Adaptativo	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$	$\min_X$	$\overline{X}$	σ_X	$\tilde{X}$	$\max_X$
Melhor em Métrica	Normal	0.890	**0.954**	0.025	**0.958**	**0.980**	113465	**782636**	**356035**	**1021234**	**1064975**
	Camada	**0.858**	0.972	**0.040**	**0.985**	**0.992**	**63508**	**428672**	309111	**336664**	**1056671**
	Neurônio	**0.961**	**0.981**	**0.010**	0.982	0.991	**209971**	542154	**280037**	520841	1064750
Melhor em Parâmetro	Normal	**0.512**	**0.672**	**0.103**	**0.666**	**0.917**	**1055**	**3863**	**7846**	**1056**	**26021**
	Camada	0.559	0.615	0.059	0.608	0.771	1081	**1401**	**702**	1088	**3164**
	Neurônio	**0.561**	**0.610**	**0.035**	**0.606**	**0.677**	**1263**	1686	722	**1278**	3326
Próximo da Origem	Normal	**0.864**	**0.926**	**0.032**	**0.931**	0.971	21876	**65381**	**35882**	**57772**	**124935**
	Camada	0.899	0.940	0.029	0.942	**0.981**	**15660**	**35213**	**12991**	**32310**	**54154**
	Neurônio	**0.915**	**0.947**	**0.016**	**0.949**	**0.965**	**25446**	41966	23800	33049	105550

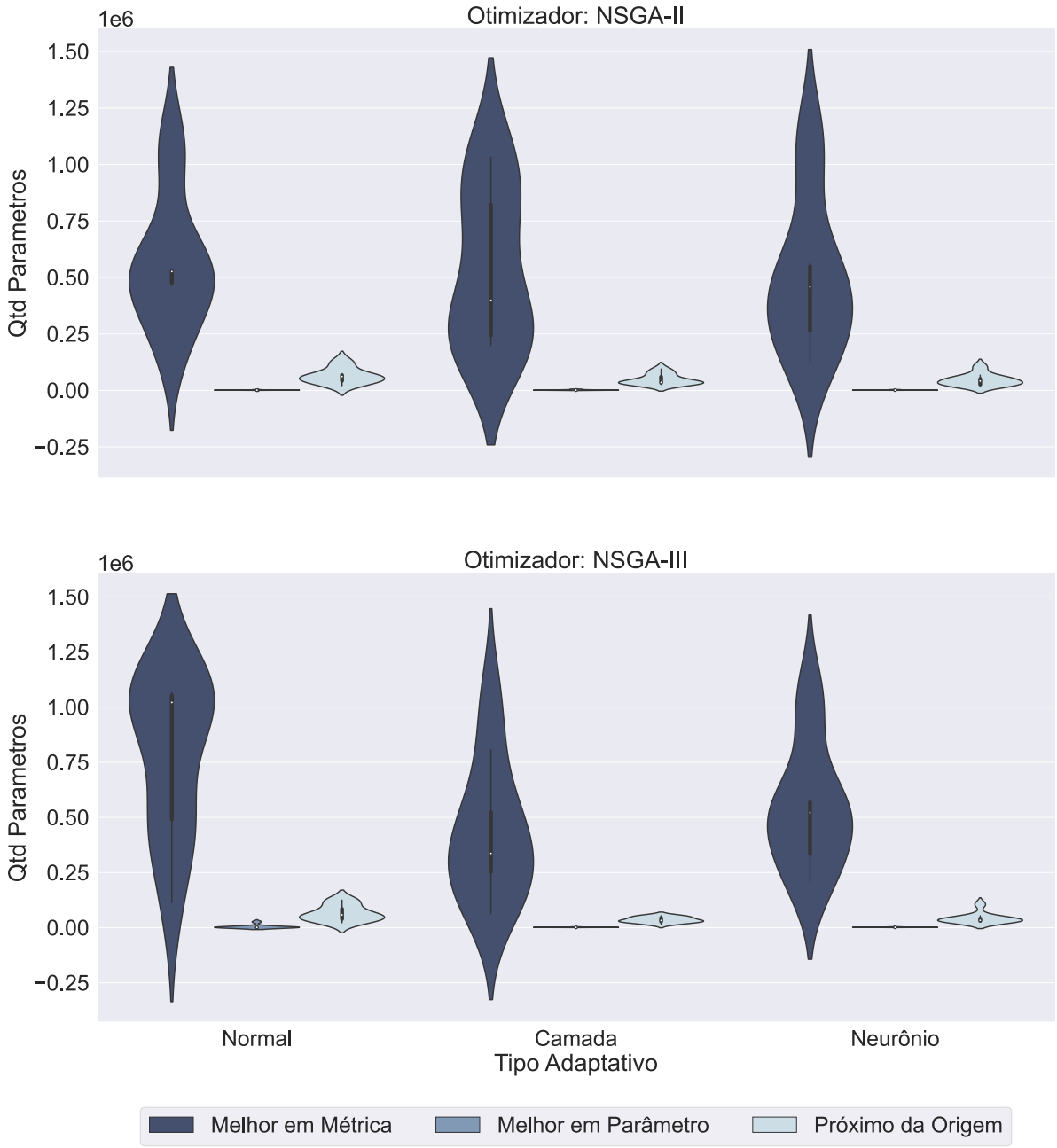
Fonte: Elaborada pelo autor (2023).

Figura 111 – *Violin Plots* do Experimento 4 para a métrica ROC AUC.



Fonte: Elaborada pelo autor (2023).

Figura 112 – *Violin Plots* do Experimento 4 para a Quantidade de Parâmetros.



Fonte: Elaborada pelo autor (2023).

D.3 Pegada de Carbono

A Tabela 49 exibe as estimativas médias de emissão de CO_2 em quilogramas associadas ao treinamento das três soluções retiradas da frente de Pareto encontrada, agregando todas as 10 execuções independentes do quarto experimento. Essas soluções incluem aquela que apresentou o melhor desempenho em métricas, a que apresentou o menor número de parâmetros e, por fim, a solução intermediária que está mais próxima

da origem de acordo com a distância euclidiana. Os valores apresentados com a cor **"Azul Escuro"** representam os melhores resultados encontrados, enquanto os valores apresentados com a cor **"Azul Claro"** representam os piores resultados encontrados.

Tabela 49 – Estimativas de Emissões de CO_2, em quilograma, do Experimento 4.

Otimizador	Tipo Adaptativo	Emissões de CO_2 (kg)		
		Métrica	Origem	Parâmetro
NSGA-II	Normal	1.34e-05	**2.27e-06**	**4.76e-06**
	Camada	**1.77e-05**	8.53e-06	**8.22e-06**
	Neurônio	**8.57e-06**	**9.46e-06**	7.31e-06
NSGA-III	Normal	**3.85e-06**	**4.81e-06**	7.56e-06
	Camada	8.58e-06	**1.47e-05**	**1.21e-05**
	Neurônio	**1.62e-05**	1.06e-05	**6.78e-06**

Fonte: Elaborada pelo autor (2023).