Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ferramenta Web de Apoio ao Diagnóstico da Síndrome do Túnel do Carpo: Classificação da Gravidade com Aprendizado de Máquina a partir de Dados Clínicos

Felipe Fontenele dos Santos
Gabriel Martins de Almeida

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientadora
Profa. Dra. Roberta Barbosa Oliveira

Brasília
2025

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ferramenta Web de Apoio ao Diagnóstico da Síndrome do Túnel do Carpo: Classificação da Gravidade com Aprendizado de Máquina a partir de Dados Clínicos

Felipe Fontenele dos Santos
Gabriel Martins de Almeida

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Profa. Dra. Roberta Barbosa Oliveira (Orientadora)
CIC/UnB

Prof. Dr. Jan Mendonca Correa
CIC/UnB

Prof. Dr. Camilo Chang Dórea
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 22 de Julho de 2025

Dedicatória

Dedico este trabalho à mulher que me deu a vida, que nunca recusou comprar um livro que pedia e sempre me garantiu todo o necessário para ir atrás da maior riqueza que posso conseguir nesta vida: o conhecimento. Sem seu apoio, carinho e cuidado, eu não seria nada. *Felipe Fontenele.*

Dedico este Trabalho de Conclusão de Curso aos meus pais, cujo amor incondicional e apoio constante me ensinaram o verdadeiro valor do estudo. A vocês, que confiaram no meu potencial mesmo nos momentos de dúvida, ofereço minha eterna gratidão.

Ao meu amigo Caio, cujo companheirismo e criatividade foram fonte constante de inspiração, tornando esta jornada mais leve e motivadora. *Gabriel Martins.*

Agradecimentos

Eu, Felipe, agradeço à minha família e ao ambiente acadêmico que me permitiu vislumbrar pessoas, ideias, projetos e conhecimentos de várias formas diferentes. Chego ao final dessa graduação com um enorme sentimento de gratidão por ter convivido com pessoas tão especiais e ter conhecido outras tão marcantes como a minha noiva, meu colega com o qual pude compartilhar minha primeira viagem e show, ao meu querido amigo Pedro, que me fez não desistir da graduação no momento mais difícil. Agradeço também aos reis, os quais tenho um enorme carinho e amizade há tantos anos.

Eu, Gabriel, gostaria de agradecer, primeiramente, aos meus pais, à minha companheira e aos amigos que estiveram ao meu lado nesta jornada. Em especial, um agradecimento aos "Cavaleiros do Migué", um pilar fundamental nessa jornada, e ao grupo "T7", com menção honrosa a dois grandes amigos que, além de inspiração, foram essenciais no apoio financeiro desta caminhada. Por fim, registro meu carinho eterno por minhas cachorras Brisa e Neguinha.

Agradecemos em conjunto e de maneira muito especial, o apoio, paciência e o conhecimento que a Profa. Dra. Roberta dedicou e compartilhou conosco durante esse árduo trabalho.

Resumo

A síndrome do túnel do carpo (STC) é a neuropatia compressiva mais prevalente dos membros superiores e, quando não tratada, pode causar perda funcional permanente. Embora a eletrodiagnose (estudos de condução nervosa) seja amplamente utilizada, trata-se de um exame invasivo, dependente de equipamento e que pode falhar em identificar alterações nos estágios iniciais da STC. A ultrassonografia de alta resolução surge como alternativa morfológica rápida e indolor, e os avanços em *Machine Learning* (ML) possibilitam integrar variáveis clínicas, ultrassonográficas e eletrofisiológicas obtidas por eletroneuromiografia (ENMG). Nesse cenário, sistemas de diagnóstico auxiliado por computador destacam-se como ferramentas promissoras na detecção precoce e apoio à decisão clínica. Este trabalho visa desenvolver um modelo de ML para classificar a gravidade da STC com base em dados clínicos e encapsulá-lo em uma ferramenta de diagnóstico assistido por computador baseada na *web*. A base de dados contou com 1037 exames clínicos de mãos, fortemente desbalanceados entre os níveis de gravidade. Foram testados cinco cenários para lidar com o desbalanceamento e a escassez de dados: (1) dados originais; (2) balanceamento de classes com SMOTE (*Synthetic Minority Over-sampling Technique*); (3) aumento de dados com CTGAN (*Conditional Tabular Generative Adversarial Network*); (4) aumento com CTGAN e balanceamento com SMOTE; e (5) balanceamento de classes com CTGAN, considerando algoritmos de ML tradicionais e de *deep learning*. Os modelos baseados em *ensemble* de árvores apresentaram melhor desempenho, com F1-score de 0.77 obtido pelo Gradient Boosting no cenário com CTGAN e SMOTE, embora a separação entre classes ainda represente um desafio. Os modelos foram implantados na aplicação *web* desenvolvida, compatível com práticas de *Machine Learning Operations* (MLOps), visando auxiliar especialistas no acompanhamento de pacientes por meio da análise automática da severidade da STC.

Palavras-chave: aprendizado de máquina, redes neurais profundas, diagnóstico auxiliado por computador, aplicativo web, síndrome do túnel do carpo

Abstract

Carpal Tunnel Syndrome (CTS) is the most prevalent compressive neuropathy of the upper limbs; if left untreated, it can lead to permanent functional loss. Although electrodiagnosis (nerve conduction studies) is commonly employed, it is an invasive procedure that may not detect abnormalities during the early stages of CTS. High-resolution ultrasonography provides a fast and non-invasive morphological alternative, and recent advances in Machine Learning (ML) have enabled the integration of clinical, ultrasonographic, and electrophysiological variables obtained through electroneuromyography (ENMG). In this context, computer-aided diagnosis (CAD) systems have emerged as promising tools to support early detection and clinical decision-making. This study aimed to develop an ML model capable of classifying the severity of CTS from clinical data and to encapsulate the model into a web-based computer-aided diagnosis tool. The dataset consisted of 1 037 clinical hand examinations, heavily imbalanced across severity levels. Five experimental scenarios were assessed to address data imbalance and scarcity: (1) baseline (no augmentation or balancing); (2) balancing using SMOTE (Synthetic Minority Over-sampling Technique); (3) augmentation using CTGAN (Conditional Tabular Generative Adversarial Network); (4) augmentation with CTGAN combined with balancing using SMOTE; and (5) augmentation using CTGAN, applied to both traditional ML algorithms and deep learning models. Ensemble tree-based models demonstrated superior performance, with the highest F_1-score (0.77) achieved by Gradient Boosting under the scenario involving augmentation with CTGAN and balancing with SMOTE, although class discrimination remained challenging. The ensemble models were deployed in the developed web application, which adheres to machine learning operations (MLOps) practices, to assist specialists in monitoring patients through automated analysis of CTS severity.

Keywords: machine learning, deep neural networks, computer-aided diagnosis, web application, carpal tunnel syndrome

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

Adam *Adaptive Moment Estimation.*

ADx-STC Assistente de Diagnóstico da Síndrome do Túnel do Carpo.

AKS *Azure Kubernetes Service.*

API *Application Programming Interface.*

AUC *Area Under the Curve.*

AWS *Amazon Web Service.*

BCTQ *Boston Carpal Tunnel Questionnaire.*

CAD *Computer-Aided Diagnosis.*

CI/CD *Continuous Integration and Continuous Delivery.*

CLR *Cyclical Learning Rate.*

CNN *Convolutional Neural Network.*

CSA *Cross Sectional Area.*

CTD *Carpal Tunnel Decompression.*

CTGAN *Conditional Tabular GAN.*

CV Coeficiente de Variação.

DL *Deep Learning.*

DNN *Deep Neural Network.*

DT *Decision Tree.*

ELU *Exponential Linear Unit.*

ENMG Eletroneuromiografia.

GAN *Generative adversarial networks.*

GD *Gradient Descent.*

HTTP *Hypertext Transfer Protocol.*

HyVaN-AMIS *Hybrid Variable Neighborhood Adaptive Search with Artificial Multiple Intelligence System.*

IA Inteligência Artificial.

IC Intervalo de Confiança.

JSON *JavaScript Object Notation.*

KNN *K-Nearest Neighbors.*

MAE *Mean Absolute Error.*

ML *Machine Learning.*

MLOps *Machine Learning Operations.*

MLP *Multilayer Perceptron.*

MSE *Mean Square Error.*

MVC *Model-View-Controller.*

NCS *Nerve Conduction Studies.*

OSA-CTSD *One-Stop Automated Diagnostic System for CTS Diagnosis.*

PB *Palmar Bowing.*

RBF *Radial Basis Function.*

ReLU *Rectified Linear Unit.*

REST *Representational State Transfer.*

RF *Random Forest.*

RNA Rede Neural Artificial.

ROC *Receiver Operating Characteristic.*

SGD *Stochastic Gradient Descent.*

SGDR *Stochastic Gradient Descent with Warm Restarts.*

SMOTE *Synthetic Minority Over-sampling Technique.*

STC Sindrome do Túnel do Carpo.

SVM *Support Vector Machine.*

XGBoost *Extreme Gradient Boosting.*

XML *Extensible Markup Language.*

Capítulo 1

Introdução

A Síndrome do Túnel do Carpo (STC) é a neuropatia compressiva mais frequente dos membros superiores, respondendo por até 90% dos casos de parestesias da mão e afetando aproximadamente 3% a 5% da população adulta mundial [3, 4]. Caracteriza-se pela compressão crônica do nervo mediano no canal carpiano, o que desencadeia dor, dormência, perda de força e comprometimento funcional progressivo. Quando não diagnosticada precocemente, a STC pode evoluir para atrofia tenar irreversível, impactando significativamente a qualidade de vida e a produtividade laboral [5].

A eletroneuromiografia (ENMG) é um método amplamente utilizado para confirmação da STC. Entretanto, o exame é invasivo, exige equipamento especializado e apresenta sensibilidade variável nos estágios iniciais [6]. A ultrassonografia de alta resolução desponta como alternativa por ser rápida, indolor e capaz de quantificar alterações morfológicas do nervo, como a *Cross Sectional Area* (CSA). Apesar disso, estudos indicam que critérios isolados baseados na CSA apresentam sobreposição entre os estágios leve (*mild*) e moderado (*moderate*), dificultando a estratificação adequada da gravidade [3]. Essa limitação retarda intervenções precoces e contribui para a alta incidência de casos graves (*severe*) no momento da avaliação cirúrgica.

Em termos computacionais, o problema clínico descrito configura-se como uma tarefa de classificação multiclasse, na qual cada exame precisa ser atribuído a uma, e somente uma, entre três diferentes classes de gravidade (leve, moderada e grave). Na literatura de Aprendizado de Máquina (ML), classificação é um subtipo de problema supervisionado em que um algoritmo infere uma função a partir de pares rotulados, visando minimizar a taxa de erro em novos dados [7]. Os classificadores tradicionais, como *Random Forests* e o *Support Vector Machines* (SVM), caracterizam-se por estruturas relativamente simples, interpretabilidade razoável e bom desempenho em bases de tamanho moderado. Por outro lado, a ascensão do *Deep Learning* (DL) tem popularizado as redes neurais profundas, compostas por múltiplas camadas não lineares capazes de extrair representações hierár-

quicas dos dados [8]. Essas arquiteturas, quando devidamente regularizadas, alcançam desempenho de estado da arte, mas exigem conjuntos de dados extensos e balanceados.

Avanços recentes em ML têm permitido a construção de modelos capazes de integrar variáveis clínicas, ultrassonográficas e eletrofisiológicas, oferecendo suporte objetivo à decisão médica [9]. Park *et al.* [1] demonstraram que o *Extreme Gradient Boosting* (XGBoost) atinge uma boa acurácia na classificação multiclasse da STC, enquanto Elseddik *et al.* [6] relataram ótimos resultados realizando a classificação binária ao combinar dados clínicos, ultrassonográficos e *Nerve Conduction Studies* (NCS). Apesar dos resultados promissores reportados na literatura, persiste uma lacuna quanto à comparação sistemática de algoritmos de aprendizado de máquina e redes neurais profundas em cenários de classificação multiclasse em que é necessário determinar a qual dentre várias classes uma amostra pertence, ao contrário de um problema de classificação binária.

Um dos grandes problemas enfrentados neste trabalho foi o desbalanceamento entre as classes da STC disponíveis na base de dados utilizada. Quando algumas categorias estão sub-representadas em relação às demais, os modelos tendem a enviesar suas previsões em favor das classes majoritárias, o que pode mascarar o desempenho real em cenários multiclasse. Essa limitação compromete particularmente as arquiteturas de aprendizado profundo, caracterizadas pela alta capacidade de representação e pela dependência de grandes volumes de dados rotulados para evitar *overfitting* e garantir boa generalização [10]. Para reduzir tal viés, empregam-se técnicas de aumento de dados (*data augmentation*), que são capazes de sintetizar amostras verossímeis e enriquecer a variabilidade do conjunto, e técnicas de balanceamento, que ajustam as proporções entre classes. O *Synthetic Minority Over-sampling Technique* (SMOTE) [11] e geradores adversariais condicionais, como o *Conditional Tabular GAN* (CTGAN) [12], constituem duas estratégias complementares amplamente exploradas neste trabalho, com o objetivo de mitigar o impacto do desbalanceamento, viabilizar a utilização de modelos mais complexos e aprimorar também a eficácia dos algoritmos tradicionais de aprendizado de máquina.

Quando integrados a Sistemas de Diagnóstico Assistido por Computador (CAD), modelos de ML ampliam a acurácia e a rapidez no suporte à decisão clínica, reduzindo a variabilidade interobservador e custos operacionais [13, 14]. Finalmente, a disponibilização dos algoritmos por meio de uma interface *web* permite que clínicas e hospitais acessem o classificador automático em tempo real, sem necessidade de infraestrutura local robusta, facilitando a adoção da tecnologia na prática diária. Isso pode ser viabilizado pelas boas práticas de *Machine Learning Operations* (MLOps) que integram ML e DevOps para padronizar, automatizar e monitorar o ciclo de vida dos modelos, do treinamento à implantação.

1.1 Objetivos

Tendo em vista a contextualização da literatura e as problemáticas expostas, este trabalho visa abordar duas tangentes do problema de classificação de gravidade da STC através de uma abordagem voltada para o treinamento de modelos tradicionais e de aprendizado profundo e o desenvolvimento de uma ferramenta CAD no contexto *web* para o auxílio do diagnóstico da gravidade da STC. Portanto, é previsto realizar a classificação multiclasse da gravidade da STC a partir de dados clínicos, tendo como objetivos específicos:

1. Investigar o efeito da otimização de hiperparâmetros sobre o desempenho dos modelos de classificação;
2. Avaliar o desempenho de diferentes classificadores, considerando algoritmos de aprendizado de máquina tradicional e aprendizado profundo;
3. Analisar o impacto de diferentes técnicas de reamostragem, incluindo algoritmos tradicionais e contemporâneos, na qualidade da classificação;
4. Avaliar como as técnicas de reamostragem afetam o desempenho dos classificadores em cada classe.

Ademais, é objetivado desenvolver uma ferramenta *web* para CAD, integrando um modelo de aprendizado de máquina treinado para a classificação da STC com os objetivos específicos a seguir:

1. Projetar e implementar uma interface *web* baseada na arquitetura MVC, visando usabilidade e compatibilidade com fluxos clínicos reais;
2. Garantir conformidade com boas práticas de MLOps, incluindo versionamento de modelos, *logging* de inferências e reprodutibilidade.

As contribuições decorrentes da execução dos objetivos propostos visam avançar a aplicação de ML no diagnóstico auxiliado por computador da STC, promovendo triagens mais precoces, reduzindo custos assistenciais e fortalecendo a prática clínica baseada em evidências.

1.2 Estrutura do Trabalho

Este trabalho está organizado em sete capítulos, cada qual com um objetivo específico e complementar:

- **Capítulo 2 Fundamentação Teórica**: apresenta os conceitos essenciais de anatomia do túnel do carpo, princípios físicos da ultrassonografia de alta resolução, fundamentos de ML e Aprendizado Profundo, além de técnicas de reamostragem de dados (SMOTE e CTGAN) que embasam os experimentos;
- **Capítulo 3 Revisão de Literatura**: sintetiza pesquisas recentes sobre sistemas de apoio ao diagnóstico da STC, comparando abordagens que utilizam sinais eletrofisiológicos, dados ultrassonográficos ou variáveis clínicas; identifica lacunas e motiva a proposta deste estudo;
- **Capítulo 4 Metodologia**: apresenta uma descrição detalhada dos dados utilizados, dos experimentos propostos, da estrutura do *pipeline* desenvolvido para a realização dos objetivos previamente apresentados e da ferramenta *web* construída;
- **Capítulo 5 Resultados**: apresenta os resultados obtidos a partir dos experimentos;
- **Capítulo 6 Discussão de Resultados**: analisa os resultados e discute limitações metodológicas;
- **Capítulo 7 Conclusão**: resume as contribuições do trabalho, destaca o impacto potencial do modelo de classificação na triagem da STC e indica direções futuras.

Capítulo 2

Fundamentação Teórica

O presente capítulo objetiva apresentar os principais conceitos teóricos que serão necessários para proporcionar o entendimento integral deste trabalho. Inicialmente, será apresentada a principal técnica de aquisição de dados dos sinais via ultrassonografia; em seguida, serão tratados alguns conceitos essenciais para o desenvolvimento da ferramenta proposta e, por fim, será abordada a temática de aprendizado de máquina.

2.1 Ultrassonografia

O princípio físico fundamental da ultrassonografia diagnóstica é o mesmo para qualquer aplicação. A sonda (transdutor) emite pulsos sonoros de alta frequência (na ordem de milhões de ciclos por segundo, ou seja, tipicamente entre 2 e 15 MHz) [15, 16]. Os ecos gerados pela reflexão desses pulsos nas interfaces dos tecidos são captados de volta pelo transdutor. Medindo-se o tempo de trânsito (intervalo entre a emissão e o retorno) de cada eco e assumindo-se a velocidade do som de aproximadamente 1540 m/s nos tecidos moles, o equipamento calcula a distância do refletor (profundidade da estrutura) [3, 15].

Esse princípio é a base da formação da imagem ultrassonográfica, pois cada tipo de tecido reflete os pulsos de modo distinto em razão de suas propriedades físicas (como densidade e impedância acústica). Diferenças de impedância acústica entre dois meios determinam a fração de energia sonora refletida na interface. Essas interfaces com grande descontinuidade de impedância geram ecos de maior amplitude (mais intensos) [15]. Com o processamento digital adequado dos sinais recebidos, a máquina converte as informações dos ecos em imagens bidimensionais de alta resolução, permitindo a visualização detalhada dos tecidos e estruturas internas [16].

Em comparação à eletroneuromiografia (ENMG), tradicionalmente considerada para a confirmação e estratificação da STC, a ultrassonografia de alta resolução demonstra vantagens significativas: maior conforto e aceitabilidade, pois dispensa estimulação elé-

trica e punção; ampla acessibilidade e menor custo, favorecendo sua adoção em diversos níveis de atenção; elevada reprodutibilidade quando se aplicam protocolos padronizados para medir a área de secção transversal do nervo mediano; e utilidade para monitoramento evolutivo e acompanhamento pós-operatório, permitindo avaliações seriadas sem desconforto adicional [3, 17, 18].

A incorporação de técnicas de Inteligência Artificial (IA), especialmente *machine learning* e *deep learning* com redes neurais, tem potencializado ainda mais esse método de imagem [4]. A combinação de ultrassonografia de alta resolução com algoritmos de IA permite segmentar, mensurar e classificar automaticamente padrões ecográficos do nervo mediano, reduzindo a subjetividade do exame e acelerando o fluxo de trabalho clínico [4, 9]. Modelos supervisionados treinados com dados clínicos e sonográficos já demonstraram alto desempenho para diagnosticar e estratificar a gravidade da STC [5, 19].

Entre os parâmetros quantificados, destacam-se: a *Cross-Sectional Area* (CSA), que corresponde à área da secção transversal do nervo mediano (valores elevados estão associados a maior probabilidade de presença e gravidade da síndrome); e o *Palmar Bowing* (PB), que expressa a deformação do retináculo flexor (teto do túnel do carpo) em direção à palma da mão quando a pressão intratúnel se eleva. Ambos os indicadores apresentam correlação significativa com a gravidade clínica e eletrofisiológica da doença [3, 20]. A base de dados disponibilizada por Park *et al.* [1], contém as variáveis idade, sexo, presença de diabetes, escala numérica de dor, lado afetado, fraqueza e outras, como as medidas de CSA e PB, que foram extraídas a partir de imagens ultrassonográficas e, todas, foram empregadas neste trabalho como variáveis preditoras para estimar a gravidade da STC em cada indivíduo avaliado. É possível observar na imagem 2.1, como é visto o nervo mediano (*Median nerve*) e os nervos flexores (*Flexor Tendons*) em uma imagem ultrassonográfica.

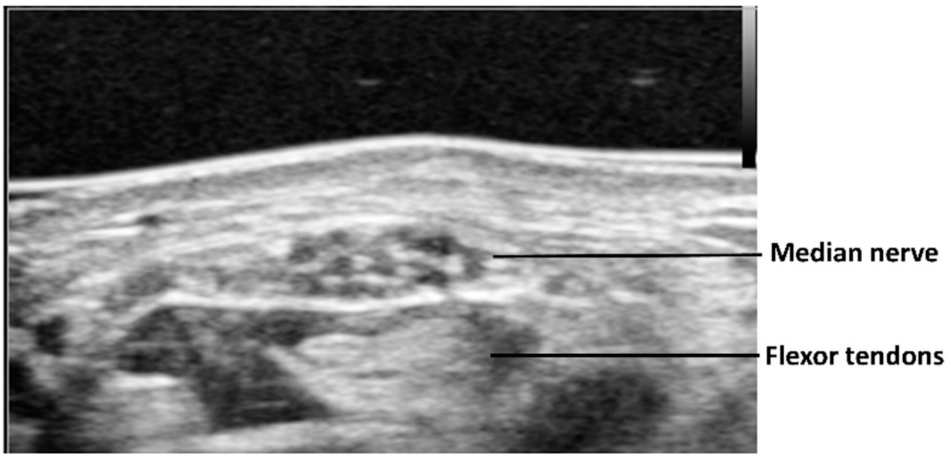


Figura 2.1: Imagem ultrassonográfica do nervo mediano. Fonte: Park *et al.* [1].

2.2 Sistemas de Diagnóstico Auxiliado por Computador

Os sistemas de Diagnóstico Auxiliado por Computador (*Computer-Aided Diagnosis*, CAD) consistem em plataformas de *software* capazes de processar automaticamente imagens médicas, extrair características quantitativas e fornecer ao clínico indicadores objetivos que auxiliam na detecção e classificação de patologias [4, 19]. Em ultrassonografia, esses sistemas combinam técnicas de pré-processamento de sinal, segmentação semântica e algoritmos de ML para destacar regiões de interesse (como o nervo mediano) e quantificar parâmetros relevantes, reduzindo a subjetividade inerente à interpretação visual [5].

A importância do CAD na prática diagnóstica está além de sua capacidade de padronizar medições e minimizar a variabilidade interobservador. Estudos demonstraram que a incorporação de *ultrasomics features* (atributos extraídos de texturas e formas do nervo mediano em imagens de alta resolução) pode elevar a acurácia diagnóstica em até 90,1% [19], conforme validado por classificadores de máquinas de vetores de suporte (*Support Vector Machines*, SVM) treinados em conjuntos de dados representativos [5, 19]. Além disso, sistemas CAD permitem rastrear automaticamente mudanças na área de secção transversal do nervo ao longo do tratamento, otimizando o monitoramento longitudinal do paciente.

No contexto da STC, aplicações de CAD já se mostraram promissoras, como por exemplo, o *One-Stop Automated Diagnostic System for CTS Diagnosis* (OSA-CTSD) integra segmentação em tempo real do nervo mediano, mensuração biométrica e diagnóstico explicável em um único fluxo, alcançando *Dice scores* superiores a 85,7% e ganhos de até 5,8% no *F1-score* em comparação a radiologistas menos experientes [21]. Dessa forma, a variabilidade, o interobservador e a dependência da especialização do examinador não apenas são reduzidas, mas também abrem caminho para fluxos de trabalho mais rápidos, reprodutíveis e escaláveis, favorecendo a detecção precoce e o monitoramento contínuo da STC. Outros trabalhos empregaram abordagens de SVM e redes neurais convolucionais para classificação binária, reportando acurácias próximas a 90% e demonstrando que o CAD pode dispensar parte da intervenção manual na medição da área do nervo [19].

A fusão de CAD com técnicas avançadas de IA, como aprendizado profundo explicável e fusão multimodal, para enriquecer o diagnóstico com informações de rigidez tecidual e padrões dinâmicos de deslizamento nervoso [3]. Tais inovações prometem não apenas aprimorar a precisão diagnóstica, mas também favorecer fluxos de trabalho mais ágeis e reprodutíveis, consolidando o CAD como ferramenta indispensável na abordagem da STC.

Perspectivas futuras para os sistemas CAD envolvem a convergência para modelos

multimodais, capazes de integrar imagens médicas, sinais clínicos e metadados eletrônicos em arquiteturas únicas, como *transformers* e *graph neural networks*, oferecendo maior robustez diagnóstica, explicabilidade e potencial para aplicações em tempo real no fluxo clínico [22]. Paralelamente, estudos focados em tarefas específicas, como a segmentação automática do nervo mediano na síndrome do túnel do carpo, já demonstram que redes U-Net treinadas em ecografia alcançam desempenho próximo ao de especialistas humanos, indicando que módulos de IA hiper-especializados serão componentes fundamentais de plataformas CAD modulares [23]. Novo *et al.* [14] desenvolveram um *software web* CAD para o apoio ao diagnóstico e à colaboração interdisciplinar em pesquisas sobre doenças vasculares, sendo validado por mais de 800 pacientes e demonstrando ótimos resultados quanto à precisão das ferramentas propostas.

2.3 Aplicação *web*

Esta seção apresenta um panorama conciso de três paradigmas complementares que fundamentam os sistemas de software contemporâneos. Primeiramente, é abordado o padrão arquitetural *Model–View–Controller* (MVC), cuja separação entre dados, apresentação e lógica de controle aprimora a manutenibilidade e a escalabilidade de aplicações *web*. Em seguida, examinam-se os princípios da Transferência de Estado Representacional (*Representational State Transfer*, REST), evidenciando como suas restrições de interface uniforme, comunicação sem estado (*stateless*) e cacheabilidade tornam as *Application Programming Interfaces* (APIs) leves e interoperáveis, por melhorar a performance do servidor, evitando sempre processar requisições idênticas, conectando serviços heterogêneos em ambientes distribuídos. Por fim, introduzem-se as Operações de Aprendizado de Máquina (*Machine Learning Operations*, MLOps), disciplina que estende as práticas de DevOps a todo o ciclo de vida de modelos de ML, destacando como *pipelines* de *Continuous Integration And Continuous Delivery* (CI/CD) automatizados (conjunto de práticas que têm por objetivo automatizar e otimizar o processo de desenvolvimento, teste e implantação de alterações), ambientes reproduzíveis e monitoramento contínuo reduzem a distância entre experimentação e produção.

2.3.1 Arquitetura MVC para Aplicações *Web*

A arquitetura MVC representa um dos paradigmas mais consolidados no desenvolvimento de aplicações *web* modernas [24]. Fundamentada na separação de responsabilidades, como é possível observar na Figura 2.2, essa abordagem organiza o sistema em três camadas funcionais: o modelo (*model*), responsável pela lógica de negócios e manipulação de dados; a visão (*view*), encarregada da apresentação e interface com o usuário; e o controlador

(*controller*), que atua como intermediador, recebendo entradas do usuário, processando eventos e coordenando a atualização do modelo e da visualização [25]. Essa separação modular promove maior manutenibilidade, escalabilidade e reutilização de código, aspectos essenciais em ambientes colaborativos e de rápida evolução tecnológica, como sistemas *web* baseados em microsserviços ou plataformas de apoio ao diagnóstico médico, como é possível observar em Novo *et al.* [14].

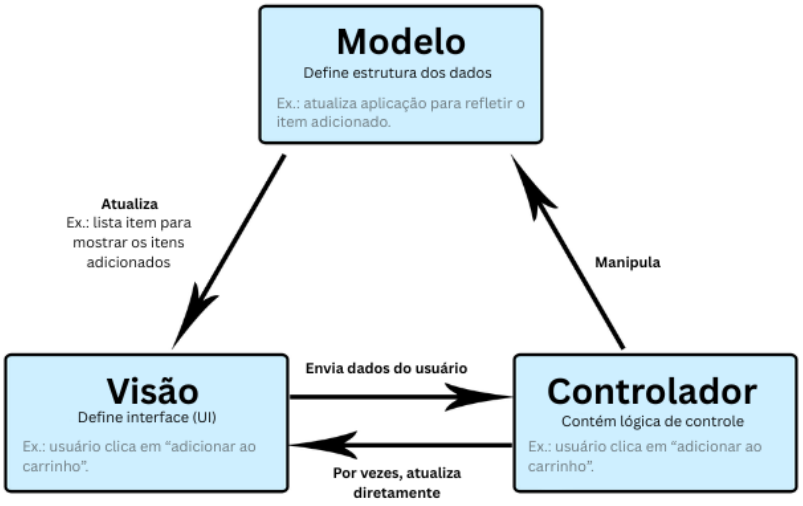


Figura 2.2: Diagrama *Model-View-Controller*. Fonte: adaptado de Mozilla Developer Network [2].

2.3.2 APIs REST

Proposto originalmente por Fielding [26], o estilo arquitetural REST define um conjunto de restrições, como interface uniforme, comunicação sem estado (*stateless*), cacheabilidade e arquitetura em camadas, que visam promover escalabilidade, simplicidade e desempenho na troca de dados em ambientes distribuídos.

REST utiliza predominantemente o Protocolo de Transferência de Hipertexto (*Hypertext Transfer Protocol*, HTTP) e operações padronizadas para manipulação de recursos, como: *GET*, que realiza a busca de dados; *POST*, que insere dados na aplicação; *PUT*, para alterar dados; e *DELETE*, para remover dados. Esses dados enviados junto aos verbos HTTP são representados usualmente em formatos como *JavaScript Object Notation* (JSON) ou *Extensible Markup Language* (XML). Essa padronização permite que sistemas heterogêneos interajam de forma transparente, favorecendo a integração de microsservi-

ços, dispositivos móveis, aplicações *web* e até sistemas legados com *backends* modernos [27]. Na Figura 2.3 é possível observar simplificadamente a estrutura de APIs REST.

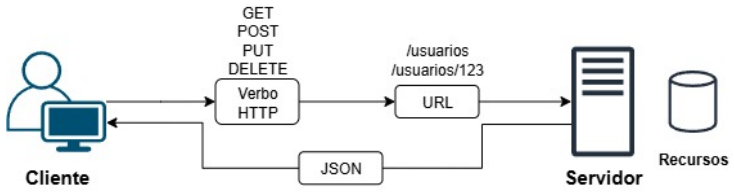


Figura 2.3: Diagrama descritivo e simplificado da estrutura de APIs REST.

2.3.3 MLOps

MLOps refere-se a um conjunto de práticas que integram o aprendizado de máquina aos princípios de DevOps, que é um conjunto de princípios que combina desenvolvimento (*development*) e operações (*operations*) para melhorar o ciclo de vida de desenvolvimento de *software*, permitindo uma maior velocidade na entrega e automatização de monitoramento, testes e *deployments*. O objetivo do MLOps é automatizar e gerenciar todo o ciclo de vida de modelos de ML [28]. Essa abordagem visa facilitar a colaboração entre cientistas de dados, engenheiros e equipes de tecnologia, promovendo maior eficiência na implantação, monitoramento e manutenção dos modelos.

O conceito surgiu em resposta aos desafios do desenvolvimento tradicional de ML, como a complexidade de ambientes, alto custo computacional e ausência de padronização [29]. Ao unir CI/CD e automação, o MLOps reduz o tempo de entrega, melhora o desempenho dos modelos e garante escalabilidade com governança.

2.4 Aprendizado de Máquina

O aprendizado de máquina (*Machine Learning*, ML) é um subcampo da IA que se dedica ao desenvolvimento de algoritmos capazes de aprender padrões a partir de dados e realizar predições ou decisões com base em exemplos anteriores [30]. Essa abordagem rompe com a lógica determinística tradicional, promovendo modelos que se adaptam conforme o volume e a complexidade dos dados disponíveis. Em contraste com algoritmos baseados em regras fixas, os métodos de ML buscam generalizar o comportamento observado em amostras de treino para novos dados não vistos, o que é particularmente útil em domínios com alta variabilidade e incerteza [31].

O ciclo de vida de um projeto de ML pode envolver diversas etapas iniciais: pré-processamento de dados, extração e seleção de atributos e aumento e balanceamento de dados [31]. A partir dessas etapas, os algoritmos de aprendizado de máquina são treinados

para uma tarefa de classificação. A avaliação dos resultados de classificação vai além da acurácia, envolvendo métricas como precisão, revocação, *F1-score* e área sob a curva (*Area Under the Curve*, AUC) [32]. Esses dados são especialmente importantes em domínios biomédicos, em que o desequilíbrio entre classes ou a presença de falsos negativos pode comprometer a tomada de decisão clínica. Além disso, a interpretabilidade e a robustez do modelo são critérios relevantes quando se busca sua aplicação em contextos sensíveis como a saúde pública e diagnósticos individuais.

2.4.1 Aprendizado Supervisionado

O aprendizado supervisionado é um dos principais paradigmas de ML e consiste em treinar algoritmos a partir de um conjunto de dados rotulados, ou seja, em que cada entrada está associada a uma saída desejada [30]. O objetivo do aprendizado supervisionado é estimar uma função de mapeamento

$$f : \mathcal{X} \longrightarrow \mathcal{Y},$$

capaz de associar cada exemplo de entrada $x \in \mathcal{X}$ ao respectivo rótulo ou valor de saída $y \in \mathcal{Y}$. Durante o treinamento, a função f é ajustada a partir de um conjunto de pares rotulados $\{(x_i, y_i)\}_{i=1}^{n}$; após esse processo, espera-se que ela generalize, isto é, produza previsões corretas para dados inéditos. Esse paradigma fundamenta tarefas como classificação (quando y é uma categoria), regressão (valor contínuo), reconhecimento de padrões e detecção de anomalias.

2.4.2 Tarefa de Classificação

No contexto do aprendizado supervisionado, classificação refere-se ao processo de atribuir uma ou mais categorias predefinidas a uma entrada com base em um conjunto de exemplos rotulados. Tais modelos são treinados para aprender uma função de mapeamento entre variáveis de entrada (*features*) e seus respectivos rótulos (*classes*), com o objetivo de generalizar corretamente sobre novos dados ainda não observados [30].

A classificação binária é um caso particular de classificação no qual o conjunto de rótulos contém exatamente duas classes mutuamente exclusivas, por exemplo, a identificação da presença ou ausência de uma doença (positivo ou negativo). Nestes problemas, o objetivo do classificador é aprender a distinguir entre a classe positiva e a classe negativa. Por outro lado, a classificação multiclasse envolve problemas cujo conjunto de rótulos contém três ou mais categorias (por exemplo, estratificar gravidade em leve, moderada e grave). Além disso, há estratégias formais para estender classificadores binários a cenários multiclasse, como *One-vs-Rest* (treinar um classificador binário por classe, distinguindo-a das

demais) e *One-vs-One* (treinar um classificador para cada par de classes); cada abordagem traz *trade-offs* em termos de custo computacional e consistência das decisões.

2.4.3 Algoritmos de Classificação

No contexto biomédico, o aprendizado de máquina tem demonstrado grande potencial para suporte ao diagnóstico [33], análise preditiva [34] e personalização do tratamento [35]. Modelos supervisionados tradicionais (como árvore de decisão, máquina de vetores de suporte, k-vizinhos mais próximos), incluindo redes neurais artificiais e redes neurais profundas, além de métodos *ensemble* (por exemplo, *Random Forest* e *Gradient Boosting*), têm sido amplamente utilizados em tarefas como detecção de doenças, classificação de imagens médicas e análise de séries temporais fisiológicas [9], principalmente por ganho de desempenho e facilidade de auditoria. Esse tipo de abordagem supervisionada fornece a base para diversos algoritmos de classificação, cujas especificidades, como mecanismo de decisão, capacidade de generalização e complexidade computacional, são mais relevantes.

Árvore de Decisão

A árvore de decisão (*Decision Tree*, DT) é uma técnica baseada em uma estrutura hierárquica de decisões. O modelo funciona dividindo recursivamente o espaço de atributos por meio de testes binários ou multi-ramo, formando uma árvore na qual cada nó interno representa uma condição de decisão e cada nó folha corresponde a uma predição de classe ou valor [36]. A construção da árvore envolve a escolha dos atributos mais informativos em cada divisão, com base em medidas como entropia. O objetivo é maximizar a pureza dos subconjuntos gerados, separando progressivamente os dados em grupos mais homogêneos em relação ao alvo [37].

As DTs são populares por sua interpretação intuitiva e visual, além de não exigirem normalização dos dados nem suposições sobre sua distribuição. Contudo, são suscetíveis ao *overfitting*, especialmente quando crescem sem restrições. Estratégias como poda (*pruning*), definição de profundidade máxima e limiares de divisão são utilizadas para mitigar esse problema [38]. Alguns de seus algoritmos incluem o *Random Forest* [39], *Gradient Boosting* [40] e XGBoost [41].

Floresta Aleatória

A RF é um algoritmo de aprendizado supervisionado baseado no conceito de *ensembles* de árvores de decisão, que combina o resultado de várias DTs para produzir uma predição única mais robusta do que a de qualquer árvore individual. O modelo constrói múltiplas DTs independentes durante o treinamento e combina suas previsões, geralmente por

votação majoritária (para classificação) ou média (para regressão), resultando em uma predição mais robusta e precisa [39]. A diversidade entre as árvores é promovida por meio de dois mecanismos principais: o *bootstrap* (amostragem com reposição dos dados) e a seleção aleatória de atributos em cada divisão da árvore. Essa combinação reduz a variância do modelo e mitiga o risco de *overfitting* característico de árvores individuais, mantendo ao mesmo tempo boa capacidade de generalização [37].

Gradient Boosting Model

O *gradient boosting model* é baseado em *ensembles* de modelos individuais, geralmente DT, treinados de forma sequencial para minimizar uma função de perda por meio de *Stochastic Gradient Descent* (SGD) [40]. A ideia central é que cada novo modelo corrija os erros cometidos pelos anteriores, ajustando-se aos resíduos da predição acumulada. A cada iteração, uma nova árvore é ajustada para prever o gradiente negativo da função de perda em relação às predições atuais. O modelo final é a soma ponderada de todas as árvores anteriores, o que permite a construção de modelos altamente flexíveis e não lineares, capazes de capturar interações complexas entre variáveis [37]. O *gradient boosting model* é sensível a *overfitting*, especialmente quando o número de árvores ou a profundidade dos modelos é grande. Por isso, são utilizados hiperparâmetros como a taxa de aprendizado (*learning rate*), número de iterações, subamostragem e regularização para controlar a complexidade e melhorar a generalização [42].

Extreme Gradient Boosting

O *Extreme Gradient Boosting* (XGBoost) é uma implementação otimizada do algoritmo de *gradient boosting*, desenvolvida para oferecer alta eficiência computacional, escalabilidade e desempenho preditivo [41]. Entre suas principais inovações estão o uso de regularização L1 (penalização baseada na norma de Manhattan) e L2 (penalização baseada na norma Euclidiana), poda baseada em profundidade máxima, paralelismo em nível de árvore e estratégias avançadas de tratamento de valores ausentes. Essas melhorias tornam o XGBoost menos propenso ao *overfitting* e altamente eficiente mesmo em conjuntos de dados grandes e complexos [42]. Sua capacidade de lidar com variáveis heterogêneas e capturar interações não lineares torna-o uma ferramenta robusta para aplicações em saúde [43].

Máquina de Vetores de Suporte

Notabilizando-se por sua robustez em problemas de alta dimensionalidade e separabilidade não linear, o SVM tem como princípio central encontrar o hiperplano ótimo que melhor

separa as classes no espaço das *features*, maximizando a margem entre os pontos de diferentes classes mais próximos a esse hiperplano, denominados vetores de suporte [44]. Em sua forma mais simples, o SVM realiza uma separação linear. Contudo, para lidar com casos não linearmente separáveis, o modelo utiliza a técnica do *kernel trick*, que projeta os dados para espaços de maior dimensionalidade, permitindo a separação das classes por meio de superfícies complexas. *Kernels* comuns incluem a função de base radial (*Radial Basis Function*, RBF), polinomial e sigmoide, sendo determinante a escolha do *kernel* e seus parâmetros para o desempenho final do classificador [37].

k-Vizinhos Mais Próximos

O k-vizinhos mais próximos (*k-Nearest Neighbors*, KNN) se baseia na ideia de que exemplos semelhantes tendem a compartilhar a mesma classe, o que permite classificar uma nova instância observando as classes mais frequentes entre seus k-vizinhos mais próximos no espaço das variáveis explicativas [37]. Para medir a similaridade entre instâncias, o KNN geralmente utiliza métricas de distância, como a Euclidiana ou de Manhattan [37]. O valor de k é um hiperparâmetro sensível que influencia diretamente a capacidade de generalização do modelo: valores muito baixos tendem a gerar modelos altamente sensíveis ao ruído (*overfitting*), enquanto valores muito altos podem suavizar excessivamente as fronteiras de decisão, levando ao *underfitting* [38]. Por não possuir uma fase explícita de treinamento, o KNN é considerado um método de aprendizado preguiçoso (*lazy learning*). Todas as operações de cálculo são realizadas apenas no momento da inferência, o que, embora reduza a complexidade do treinamento, pode resultar em elevado custo computacional na fase de predição, especialmente em conjuntos de dados volumosos ou de alta dimensionalidade [45].

Rede Neural Artificial

A Rede Neural Artificial (RNA) é um modelo computacional inspirado na forma como os neurônios biológicos processam informação. Ela pode ser implementada por meio de arquiteturas *feedforward*, tais como as *Multi-Layer Perceptron* (MLP). A MLP é composta por camadas densamente conectadas: uma camada de entrada, geralmente uma ou duas camadas ocultas e uma camada de saída. Cada neurônio realiza uma combinação linear dos dados de entrada, seguida por uma função de ativação não linear, como *Rectified Linear Unit* (ReLU) ou *softmax* (que são abordadas na Seção 2.4.4), permitindo que o modelo aprenda relações complexas e não lineares [46]. Durante o treinamento, o MLP ajusta seus pesos por meio do algoritmo de *backpropagation*, que propaga o erro da saída para as camadas anteriores e utiliza métodos de otimização baseados em gradiente, SGD ou variantes como *Adaptive Moment Estimation* (Adam) [47].

Rede Neural Profunda

As redes neurais profundas (*Deep Neural Network*, DNN) são RNAs compostas por múltiplas camadas ocultas entre a entrada e a saída. Esse aumento na profundidade permite que o modelo aprenda representações hierárquicas dos dados, capturando padrões complexos e não lineares que modelos rasos (*shallow networks*) não conseguem representar adequadamente [46]. Uma DNN é geralmente uma extensão do MLP, em que o número de camadas e neurônios é maior. Cada camada realiza transformações sucessivas nos dados por meio de combinações lineares seguidas de funções de ativação não lineares.

2.4.4 Funções de Ativação

As funções de ativação desempenham um papel essencial nas RNA ao introduzirem comportamento não linear entre as camadas. Essa propriedade é fundamental para que a rede consiga modelar relações complexas nos dados e resolver problemas que não podem ser representados por combinações lineares simples [46].

Durante o processo de propagação, cada neurônio aplica uma função de ativação à sua entrada ponderada, permitindo que o modelo aprenda limiares de ativação, regiões de decisão e padrões hierárquicos. Sem essas funções, mesmo redes com várias camadas se comportariam como um modelo linear equivalente [7].

Diversas funções têm sido propostas, cada uma com características próprias em termos de curvatura, derivabilidade, saturação e desempenho computacional. Entre as mais utilizadas estão a função ReLU e a *softmax* [48].

ReLU

A ReLU é uma das funções de ativação mais utilizadas em DNNs, especialmente em tarefas de visão computacional e aprendizado supervisionado [49]. Sua popularidade se deve à simplicidade computacional, ao bom desempenho em redes profundas e à sua eficácia em evitar o problema do gradiente desvanecido, comum em funções como a sigmoide ou a tangente hiperbólica [46, 50].

A função ReLU é definida por:

$$f(x) = \max(0, x), \tag{2.1}$$

em que x representa o valor de entrada recebido por um neurônio. Em outras palavras, a função ReLU transforma a entrada de modo que valores negativos são mapeados para zero, enquanto valores positivos são mantidos inalterados. Essa característica torna a ReLU uma função não linear, mas ainda assim muito eficiente do ponto de vista computacional.

Apesar de suas vantagens, a ReLU apresenta uma limitação conhecida como *dying ReLU*, na qual neurônios podem ficar permanentemente inativos ao receberem apenas entradas negativas, impedindo o aprendizado. Para mitigar esse efeito, variantes como *Leaky ReLU* [51], *Exponential Linear Unit* (ELU) [52] e *Parametric ReLU* [53] foram propostas. Ainda assim, a ReLU permanece como a função padrão em diversas arquiteturas modernas, por oferecer uma combinação equilibrada entre desempenho, estabilidade e custo computacional [46].

Softmax

A função *softmax* é amplamente utilizada em camadas de saída de RNA para problemas de classificação multiclasse [7]. Sua principal característica é transformar um vetor de valores reais em uma distribuição de probabilidade, em que cada saída representa a probabilidade relativa de pertencimento a uma das classes possíveis [7]. Matematicamente, a função é definida como:

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{2.2}$$

na qual z_i é o valor da i-ésima classe e K é o número total de classes. O resultado é um vetor com K elementos, todos no intervalo $(0, 1)$, cuja soma é igual a 1.

A *Softmax* facilita a interpretação probabilística da saída do modelo, sendo especialmente adequada para tarefas em que uma única classe deve ser atribuída por amostra (por exemplo, diagnóstico exclusivo).

2.4.5 Função de Perda

Em modelos de aprendizado supervisionado, a função de perda (ou função de custo) é um componente essencial responsável por quantificar o erro entre a predição realizada pelo modelo e o valor real esperado. Ela fornece uma medida escalar que é minimizada durante o treinamento, orientando o ajuste dos pesos da rede por meio de algoritmos de otimização, como o gradiente descendente [46].

A escolha da função de perda depende do tipo de tarefa. Para problemas de regressão, funções como o erro quadrático médio (*Mean Square Error* , MSE) ou o erro absoluto médio (*Mean Absolute Error*, MAE) são frequentemente utilizadas [54]. Já em tarefas de classificação, especialmente com múltiplas classes, funções baseadas na entropia cruzada são preferidas por sua interpretação probabilística e propriedades numéricas desejáveis [7]. Essas funções guiam a aprendizagem ao penalizar predições incorretas de forma proporcional à sua distância dos rótulos reais e, no contexto de classificações multiclasse

com rótulos inteiros e mutuamente exclusivos, uma função amplamente utilizada é a *Sparse Categorical Crossentropy* [54].

Sparse Categorical Crossentropy

A versão tradicional da entropia cruzada multiclasse exige que os rótulos estejam codificados como vetores *one-hot*, ou seja, vetores binários em que apenas a posição correspondente à classe correta é 1 (um). A *sparse categorical crossentropy*, por outro lado, permite que o rótulo da amostra seja um único número inteiro, representando diretamente a classe correta, evitando a necessidade de conversão.

Matematicamente, para uma amostra com C classes e saída de rede $\mathbf{y} = (y_1, y_2, ..., y_C)$ após a função *softmax*, e um rótulo verdadeiro $t \in \{0, 1, ..., C-1\}$, a função é definida como:

$$\mathcal{L} = -\log(y_t). \tag{2.3}$$

Essa forma penaliza a rede com base na log-probabilidade atribuída à classe correta. Quanto menor a probabilidade prevista para a classe verdadeira, maior será a perda. Como resultado, a rede é incentivada a aumentar a confiança nas predições corretas durante o treinamento.

2.4.6 Otimizadores

Os otimizadores são algoritmos responsáveis por atualizar os pesos do modelo com base nos gradientes da função de perda em relação aos parâmetros. O algoritmo de base mais conhecido é o *Gradient Descent* (GD), que ajusta os pesos na direção do gradiente negativo da função de perda. Contudo, variantes mais sofisticadas foram desenvolvidas para superar limitações como lentidão em regiões rasas, sensibilidade à escala das variáveis e dificuldades em ajustar o *learning rate* [47].

Entre os otimizadores modernos mais utilizados, destacam-se o Adam [55], que combina os benefícios do *momentum* e da adaptação individual da taxa de aprendizado, e algoritmos com agendamento dinâmico da taxa de aprendizado, como o *cosine decay with restarts*, que contribuem para uma melhor exploração do espaço de parâmetros durante o treinamento. As próximas seções abordam esses dois otimizadores amplamente adotados na prática: o *Adam* e o *cosine decay restarts*, detalhando seus funcionamentos e vantagens em relação às estratégias tradicionais.

Adam

O Adam combina as ideias do *momentum*, que acumula gradientes passados para suavizar as atualizações, com a adaptação de taxas de aprendizado por parâmetro, semelhante ao algoritmo *adaptive gradient* [56].

O Adam mantém estimativas exponencialmente decrescentes da média (m_t) e da variância (v_t) dos gradientes ao longo do tempo. A atualização dos pesos do modelo θ é realizada segundo a seguinte equação:

$$\theta_{t+1} = \theta_t - \alpha \times \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{2.4}$$

em que $\hat{m}_t$ e $\hat{v}_t$ são as estimativas corrigidas do primeiro e segundo momentos, α é a taxa de aprendizado e ϵ é um termo de estabilidade numérica. Ele é robusto em termos de hiperparâmetros e funciona bem mesmo com valores padrões, como $\beta_1 = 0.9$, $\beta_2 = 0.999$ e $\epsilon = 10^{-8}$ [56].

Cosine Decay with Restarts

O *Cosine Decay with Restarts* é uma técnica de agendamento de taxa de aprendizado (*learning rate schedule*) que visa melhorar o processo de otimização durante o treinamento de DNNs. Ao invés de manter uma taxa de aprendizado constante ou decrescer linearmente, esse método emprega uma curva cosseno para reduzir suavemente o valor de α ao longo das épocas, permitindo reinicializações periódicas (*restarts*) para escapar de mínimos locais e melhorar a generalização [57].

A taxa de aprendizado ao longo do tempo t é ajustada pela seguinte equação:

$$\alpha_t = \alpha_{\min} + \frac{1}{2}(\alpha_{\max} - \alpha_{\min}) \left(1 + \cos\left(\frac{T_{cur}}{T_i}\pi\right)\right), \tag{2.5}$$

na qual:

- $\alpha_{\max}$ é a taxa de aprendizado inicial do ciclo;
- $\alpha_{\min}$ é a menor taxa de aprendizado permitida;
- T_{cur} é o número de épocas desde o início do ciclo atual;
- T_i é a duração do ciclo atual.

Após cada ciclo de T_i épocas, a taxa de aprendizado é reiniciada para $\alpha_{\max}$, e a duração do ciclo pode ser aumentada progressivamente, por exemplo, dobrando seu valor. Essa abordagem, conhecida como *Stochastic Gradient Descent with Warm Restarts* (SGDR), melhora a exploração do espaço de otimização e favorece uma convergência mais suave [57].

2.5 Balanceamento de Dados

Em tarefas de aprendizado supervisionado, especialmente em problemas de classificação, é comum que os conjuntos de dados apresentem desbalanceamento entre classes, ou seja, uma distribuição desigual do número de amostras entre as categorias pelo fato de que eventos de interesse costumam ser raros (como fraudes e doenças graves, por exemplo) e coletar esses exemplos minoritários pode ser caro, invasivo ou até eticamente limitado [58, 59, 60]. Esse desequilíbrio pode comprometer significativamente o desempenho dos modelos, que tendem a favorecer a classe majoritária, resultando em alta acurácia aparente, mas baixa capacidade de detecção da classe minoritária [61].

O balanceamento de dados consiste em aplicar técnicas que ajustam a distribuição das classes no conjunto de treinamento, com o objetivo de promover uma aprendizagem mais equitativa e melhorar métricas como revocação, precisão e *F1-score*, que são abordadas na Seção 2.6. Entre as abordagens mais utilizadas na literatura, Chen *et al.* [62] destaca os métodos clássicos como *oversampling*, *undersampling*, algoritmos baseados em síntese como o *Synthetic Minority Over-sampling Technique* (SMOTE), e técnicas mais recentes de geração de dados sintéticos via GANs, como o *Conditional Tabular GAN* (CTGAN). Além disso, o balanceamento pode ser complementado por outras estratégias de pré-processamento, como a codificação de variáveis categóricas com *label encoder* e a seleção de atributos com baixa variância por meio do *variance threshold*, visando melhorar a qualidade e a representatividade dos dados de entrada [63].

Uma estratégia complementar amplamente explorada é o aumento de dados (*data augmentation*) [64], que consiste em gerar novas instâncias a partir de transformações controladas dos dados existentes. Embora tradicionalmente aplicada em imagens, essa abordagem tem sido adaptada com sucesso para dados tabulares por meio de técnicas como o CTGAN, citado anteriormente, que aprende a distribuição dos dados e gera amostras sintéticas realistas [65]. O uso de dados aumentados pode não apenas mitigar o desbalanceamento, mas também enriquecer a variabilidade da base, contribuindo para a robustez e generalização dos modelos preditivos. As seções a seguir exploram essas abordagens em detalhe, apresentando suas vantagens, limitações e aplicações práticas no contexto de modelos de ML.

2.5.1 *Oversample*

A técnica de *oversampling* consiste em aumentar a representatividade da classe minoritária no conjunto de treinamento, por meio da duplicação de exemplos existentes [62]. Embora seja simples de implementar, o *oversampling* pode levar a um problema de *overfitting*, uma vez que os modelos podem memorizar as instâncias repetidas em vez de aprender

generalizações. Por isso, essa técnica é frequentemente combinada com algoritmos de geração de amostras sintéticas, como o SMOTE, que visam mitigar essa limitação.

2.5.2 SMOTE

O SMOTE é uma técnica de balanceamento baseada na geração de novas instâncias sintéticas da classe minoritária. Ao contrário do *oversampling* tradicional, que apenas replica exemplos existentes, o SMOTE cria amostras artificiais interpolando entre exemplos reais próximos no espaço das variáveis explicativas [11]. Esse processo consiste em selecionar uma instância da classe minoritária e um de seus vizinhos mais próximos e gerar um novo ponto ao longo da linha que os conecta. Essa abordagem contribui para a densificação do espaço ocupado pela classe minoritária, tornando as fronteiras de decisão mais regulares e mitigando o risco de *overfitting* associado à duplicação direta.

2.5.3 CTGAN

O CTGAN é uma coleção de geradores de dados sintéticos baseados em aprendizado profundo, desenvolvida especificamente para a geração de dados tabulares sintéticos, incluindo atributos mistos (numéricos e categóricos) e distribuições desbalanceadas [12].

Diferentemente de técnicas tradicionais como o SMOTE, que se baseiam em interpolação linear, o CTGAN utiliza uma arquitetura condicional para aprender a distribuição conjunta dos dados e gerar novas instâncias que preservam a complexidade estatística do conjunto original. Essa abordagem é especialmente útil em cenários com múltiplas variáveis categóricas e interdependências não triviais.

O CTGAN é composto por dois componentes principais: um gerador, responsável por produzir amostras sintéticas; e um discriminador, que tenta distinguir entre amostras reais e sintéticas. O processo de treinamento adversarial entre essas duas redes permite que o gerador aprenda a gerar dados altamente realistas, contribuindo para o balanceamento do conjunto de dados sem introduzir ruído estrutural significativo.

2.6 Métricas de Avaliação de Desempenho

A matriz de confusão é uma ferramenta fundamental na avaliação de desempenho de classificadores, especialmente em problemas de classificação. Ela organiza os resultados das predições do modelo em uma matriz que permite comparar os rótulos previstos com os rótulos reais do conjunto de testes, proporcionando uma visão mais detalhada do comportamento do classificador, além da acurácia global [32].

Para problemas binários, ela contém quatro elementos:

- **Verdadeiro Positivo (VP)**: instâncias da classe positiva corretamente classificadas;
- **Falso Positivo (FP)**: instâncias da classe negativa incorretamente classificadas como positivas;
- **Verdadeiro Negativo (VN)**: instâncias da classe negativa corretamente classificadas;
- **Falso Negativo (FN)**: instâncias da classe positiva classificadas erroneamente como negativas.

Em cenários multiclasse com K categorias, a matriz torna-se quadrada ($K \times K$). Cada linha i corresponde à classe real C_i e cada coluna j à classe prevista C_j. A célula M_{ij} indica quantas amostras de C_i foram atribuídas a C_j. O exemplo abaixo ilustra o caso de três classes (C_1, C_2, C_3):

$$\mathbf{M} = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix},$$

no qual os elementos da diagonal M_{11}, M_{22}, M_{33} representam os verdadeiros positivos de cada classe, e os termos fora da diagonal correspondem às diferentes combinações de falsos positivos e falsos negativos.

Para calcular métricas por classe, adota-se a abordagem *um-contra-todos*:

$$\mathrm{VP}_k = M_{kk}, \quad \mathrm{FN}_k = \sum_{j \neq k} M_{kj}, \quad \mathrm{FP}_k = \sum_{i \neq k} M_{ik}, \quad \mathrm{VN}_k = \sum_{i \neq k} \sum_{j \neq k} M_{ij},$$

permitindo derivar precisão, revocação e *F1-score* para cada classe C_k. As versões *macro-average* (média aritmética sobre k) e *micro-average* (agregação global de VP, FP, FN) fornecem perspectivas complementares sobre o desempenho multiclasse.

Acurácia

A acurácia quantifica a proporção de predições corretas em relação ao total de instâncias avaliadas. Em um cenário binário, a métrica é dada pela equação:

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN}. \tag{2.6}$$

Para problemas com K classes, a matriz de confusão torna-se $K \times K$. Se M_{ij} indica o número de amostras cuja classe real é C_i e a predição é C_j, a acurácia global é calculada

pela soma da diagonal principal dividida pelo total de amostras, como mostra a Equação 2.7:

$$\text{Acurácia} = \frac{\sum_{k=1}^{K} M_{kk}}{\sum_{i=1}^{K}\sum_{j=1}^{K} M_{ij}}. \tag{2.7}$$

Essa definição equivale à chamada *micro-average accuracy*, pois considera todos os exemplos simultaneamente; em bancos de dados balanceados, ela coincide com a média simples da taxa de acerto por classe.

Apesar de intuitiva, a acurácia pode ser enganosa quando a distribuição de classes é altamente assimétrica, permitindo que um classificador obtenha valores elevados simplesmente ao favorecer a classe majoritária [32]. Nesses casos, métricas como *balanced accuracy* (média das revocações por classe), *macro-F1* ou a análise detalhada da matriz de confusão fornecem uma visão mais fiel do desempenho do modelo.

Revocação

A revocação (também chamada de sensibilidade ou *recall*) é uma métrica que avalia a capacidade do modelo em identificar corretamente as instâncias positivas. Ela é definida como a razão entre o número de verdadeiros positivos e o total de instâncias reais da classe positiva:

$$\text{Revocação} = \frac{VP}{VP + FN}. \tag{2.8}$$

O valor da métrica varia em $[0, 1]$; quanto mais próximo de 1, menor a probabilidade de o classificador "perder" exemplos positivos, característica desejável em contextos críticos como diagnóstico médico [66].

No contexto de multiclasses, utiliza-se a estratégia *um-contra-todos*. Seja $\mathbf{M}$ a matriz de confusão $K \times K$ em que M_{ij} representa o número de amostras da classe real C_i preditas como C_j. A revocação específica da classe C_k é:

$$\text{Revocação}_k = \frac{M_{kk}}{\sum_{j=1}^{K} M_{kj}}, \quad k = 1, \ldots, K. \tag{2.9}$$

Para reportar um único valor global, costuma-se empregar a *macro-revocação*, média aritmética das revocações por classe, e a *micro-revocação*, que resulta da razão entre a soma de todos os verdadeiros positivos e a soma de todos os verdadeiros positivos mais falsos negativos [67]. A macro-revocação atribui peso igual a cada classe, evidenciando

eventuais falhas em classes minoritárias, enquanto a micro-revocação pondera as contribuições de acordo com a frequência das classes.

Uma revocação elevada pode ocorrer às custas de um maior número de falsos positivos, reduzindo a precisão. Para balancear essa troca, recorre-se a métricas harmônicas, como o *F1-score*, que concilia ambas as dimensões [68].

Precisão

A precisão (*precision*) quantifica a proporção de predições positivas que, de fato, pertencem à classe positiva. No caso binário, sua equação é definida como:

$$\text{Precisão} = \frac{VP}{VP + FP}. \tag{2.10}$$

Para o contexto de multiclasse, emprega-se a estratégia *um-contra-todos*. Seja $\mathbf{M}$ a matriz de confusão $K \times K$ em que M_{ij} denota o número de amostras da classe real C_i preditas como C_j. A precisão da classe C_k é:

$$\text{Precisão}_k = \frac{M_{kk}}{\sum_{i=1}^{K} M_{ik}}, \quad k = 1, \ldots, K. \tag{2.11}$$

A precisão não avalia quantos positivos reais são perdidos (falsos negativos). Portanto, é interessante ser interpretada em conjunto com a *revocação* [68].

F1-score

O *F1-score* é a média harmônica entre precisão e revocação, proporcionando uma avaliação equilibrada do desempenho do classificador, principalmente em situações de classes desbalanceadas ou quando falsos positivos e falsos negativos têm custos assimétricos [66, 68]. No caso binário, sua expressão é dada pela equação:

$$F_1 = 2 \times \frac{\text{Precisão} \times \text{Revocação}}{\text{Precisão} + \text{Revocação}}. \tag{2.12}$$

Quanto mais próximo de 1, melhor o equilíbrio entre alta cobertura de positivos (revocação) e baixa taxa de alarmes falsos (precisão). A métrica penaliza modelos que apresentam grande discrepância entre essas duas dimensões [69].

Nos problemas de multiclasse, dado uma classificação com K classes e matriz de confusão $\mathbf{M}$ ($K \times K$), calcula-se primeiro, para cada classe C_k, a precisão Prec_k e a revocação Rec_k. O *F1-score* por classe é:

$$F_{1,k} = 2 \times \frac{\mathrm{Prec}_k \times \mathrm{Rec}_k}{\mathrm{Prec}_k + \mathrm{Rec}_k}, \quad k = 1, \ldots, K.$$

Três agregações são usuais [67]:

$$\text{Macro-}F_1 = \frac{1}{K} \sum_{k=1}^{K} F_{1,k}, \tag{2.13}$$

$$\text{Micro-}F_1 = 2 \times \frac{\sum_{k=1}^{K} VP_k}{2\sum_{k=1}^{K} VP_k + \sum_{k=1}^{K} FP_k + \sum_{k=1}^{K} FN_k}, \tag{2.14}$$

$$\text{Weighted-}F_1 = \frac{1}{N} \sum_{k=1}^{K} n_k \, F_{1,k}, \tag{2.15}$$

em que VP_k, FP_k e FN_k são, respectivamente, verdadeiros positivos, falsos positivos e falsos negativos da classe C_k; n_k é o número de amostras em C_k e $N = \sum_k n_k$.

Macro-F_1 dá peso igual a todas as classes, enfatizando desempenho em minorias; *micro*-F_1 agrega os erros globalmente, refletindo a distribuição original; *weighted*-F_1 pondera cada $F_{1,k}$ pela frequência da classe, servindo de compromisso quando a distribuição de classes é muito assimétrica.

Curva ROC

A *Receiver Operating Characteristic* (ROC) é um gráfico que relaciona a taxa de verdadeiros positivos à taxa de falsos positivos para todos os limiares de decisão possíveis de um classificador binário [32]. Para cada ponto da curva, calculam-se:

- Eixo y: **Taxa de Verdadeiros Positivos (TPR)** = $\frac{VP}{VP+FN}$;
- Eixo x: **Taxa de Falsos Positivos (FPR)** = $\frac{FP}{FP+VN}$.

Quanto mais a curva se aproxima do canto superior esquerdo (alta TPR e baixa FPR), melhor o poder discriminativo do modelo. A área sob essa curva, *Area Under the Curve* (AUC), varia de 0 a 1; valores próximos a 0,5 indicam desempenho aleatório, enquanto AUC $\geq$ 0,9 denotam desempenho excelente.

Para K classes em um contexto de multiclasse, a estratégia habitual é um-contra-todos: calcula-se uma curva ROC (e sua AUC) para cada classe C_k considerando-a "positiva" e agregam-se os resultados de duas formas [70]:

$$\text{Macro-AUC} = \frac{1}{K}\sum_{k=1}^{K} \text{AUC}_k,$$

$$\text{Micro-AUC} = \text{AUC calculada sobre a concatenação de todas as probabilidades.}$$

A *macro-AUC* pondera igualmente cada classe, evidenciando falhas em minorias; a *micro-AUC* reflete a distribuição original de exemplos.

Intervalo de Confiança

O Intervalo de Confiança (IC) fornece uma faixa de valores plausíveis para um parâmetro populacional, construída de modo que, em repetições independentes do experimento, cubra o verdadeiro parâmetro em $100(1-\alpha)\%$ dos casos [71]. Para a média populacional μ com variância desconhecida, o IC bilateral de $100 \times (1-\alpha)\%$ é:

$$\Big[\, \bar{x} \pm t_{\alpha/2,\, n-1}\, \tfrac{s}{\sqrt{n}} \Big],$$

em que $\bar{x}$ é a média amostral, s o desvio-padrão amostral e $t_{\alpha/2,\, n-1}$ o quantil da distribuição t com $n-1$ graus de liberdade. Valores de $\alpha = 0{,}05$ geram o popular IC de 95%. ICs concentram-se na magnitude do efeito com incerteza explícita, superando a interpretação dicotômica dos p-valores [72].

Coeficiente de Variação

O Coeficiente de Variação (CV) é uma medida adimensional de dispersão relativa, definida como:

$$\text{CV} = \frac{\sigma}{\mu} \times 100\%,$$

em que σ é o desvio-padrão e μ a média da distribuição. Ao normalizar pela média, o CV permite comparar a variabilidade de conjuntos com unidades ou magnitudes distintas, como por exemplo, avaliar o risco ajustado ao retorno em finanças [73]. Contudo, se μ estiver próximo de zero, o CV torna-se instável e pouco interpretável.

Ganho em Pontos Percentuais

O ganho em pontos percentuais (p.p.) expressa a diferença aritmética direta entre duas proporções. Se uma métrica passa de $P_0\%$ para $P_1\%$, o ganho absoluto é

$$\Delta_{\text{p.p.}} = P_1 - P_0 \ \text{ pontos percentuais.}$$

Diferencia-se de *variação percentual relativa*, calculada por $(P_1 - P_0)/P_0 \times 100\%$. Por exemplo, subir de 40% para 44% representa um aumento de 4 p.p., mas um acréscimo relativo de 10% . O uso de pontos percentuais evita ambiguidade em relatórios de desempenho ao comunicar melhorias absolutas de forma clara [74].

Capítulo 3

Revisão de Literatura

De acordo com a literatura pesquisada, este capítulo será utilizado para apresentar uma revisão de diferentes métodos de aprendizado de máquina relacionados à tarefa de detectar a STC, tendo em vista que todas as obras mencionadas tiveram relevância e influência na proposta e elaboração deste trabalho.

3.1 Modelos de Aprendizado de Máquina para o Diagnóstico da Síndrome do Túnel do Carpo

Park *et al.* [1] implementaram e avaliaram duas estratégias de classificação para determinar a gravidade eletrodiagnóstica da STC utilizando vários algoritmos de ML. Com 1037 dados extraídos de mãos de pacientes, todas com a STC em diferentes intensidades, os autores implementaram a estratégia de classificação *one-versus-rest* e multiclasse, utilizando os dados separados nas classes Leve, Moderada e Grave. Como resultado da classificação multiclasse, os autores apontaram que o algoritmo com melhor desempenho foi o XGBoost, o modelo apresentou uma acurácia de 76.6% na fase de teste. Quando analisadas separadamente, as acurácias foram de 83.6% para a Classe Leve, 71.8% para a Moderada e 83.5% para a Grave. Na classificação *One-Versus-Rest*, foram apresentados melhores resultados nos testes em comparação com a abordagem multiclasse, com acurácia de 83.6% para a Classe Leve, 78.8% para a Moderada e 90.9% para a Grave. A análise da curva ROC indicou que o modelo teve melhor desempenho para casos graves, com 95.0% de acurácia, seguido de 86.0% para casos leves e 81.0% para moderados.

O estudo conduzido por Tamis *et al.* [75] analisou 65 mãos, analisadas a partir de variáveis clínicas e eletrofisiológicas, para propor um sistema de diagnóstico automático da STC baseado em sinais de condução nervosa. As mãos foram classificadas em três grupos: controle, leve/moderada e grave. Foram extraídas 34 características dos sinais, divididas

entre variáveis comuns (já utilizadas na prática clínica) e novas (baseadas em propriedades geométricas e matemáticas dos sinais). Essas variáveis alimentaram cinco classificadores diferentes, com destaque para a SVM que apresentou o melhor desempenho: acurácia de 95.1% para distinguir controles versus pacientes (2 classes) em comparação ao diagnóstico neurofisiológico e 89.1% em relação à classificação clínica; para três classes (controle, leve/moderada, grave), alcançou 94.2% (NCS) e, para quatro classes, 91.2%. A combinação de características comuns e inéditas superou o uso isolado de cada grupo, indicando que descritores geométricos do sinal sensorial complementam as métricas tradicionais, potencialmente reduzindo o viés humano e acelerando a decisão diagnóstica.

Com o objetivo de prever quais pacientes se beneficiariam cirurgicamente da descompressão do túnel do carpo (*Carpal Tunnel Decompression*, CTD) e tendo em vista que até 25% dos pacientes submetidos à CTD não apresentam melhora significativa nos sintomas - o que implica em custos elevados e risco cirúrgico sem retorno clínico -, Harrison *et al.* [76] treinaram modelos de ML para prever melhoras funcionais e sintomáticas, utilizando como métrica a *Minimal Important Change*. A base de dados utilizada envolveu 1916 pacientes submetidos à CTD entre 2010 e 2019, com avaliação clínica e coleta de medidas subjetivas por meio do questionário *QuickDASH*. Os modelos testados incluíram regressão logística regularizada, k-vizinhos mais próximos KNN, SVM, RNA e XGBoost, sendo este último o de melhor desempenho, atingindo 71.8% de acurácia para melhora funcional e 75.9% para melhora sintomática. Além disso, DT baseadas em respostas a apenas duas perguntas foram capazes de prever com boa acurácia os casos com maior chance de sucesso cirúrgico. Os resultados demonstram o valor do uso combinado de dados clínicos e respostas subjetivas na criação de ferramentas preditivas.

Utilizando um algoritmo baseado em CNN, Smerilli *et al.* [77] buscaram automatizar a identificação e medição da CSA do nervo mediano em imagens de ultrassom adquiridas no túnel do carpo. O estudo utilizou 246 imagens de 103 pacientes com doenças reumáticas. Com a arquitetura *Mask R-CNN* utilizando o *backbone* ResNet101 e *Feature Pyramid Network*, o modelo foi treinado para segmentar o nervo mediano e calcular automaticamente sua CSA. O desempenho foi avaliado com base em métricas como precisão (96%), revocação (98%) e *Dice Similarity Coefficient* (88%). Nos testes, o algoritmo identificou corretamente o nervo mediano em 83.7% das imagens; e em 95.3% quando excluídas variantes anatômicas. A concordância entre a medição automática da CSA e a manual feita por sonografistas foi excelente, com *Intraclass Correlation Coefficient* de 97%.

Elseddik *et al.* [19] propuseram uma abordagem baseada em ML para diagnosticar e prever o prognóstico da STC a partir de dados clínicos, históricos, NCS, ultrassonografia e BCTQ. Foram agregados dados de 160 pacientes; 80 diagnosticados com STC (com 40 mãos esquerdas e 40 direitas, classificadas em graus leve, moderado e grave) e 80 pacientes

com sintomas sobrepostos a outras condições, como radiculopatia cervical, tendinopatia de *De Quervain* e neuropatia periférica. Para a classificação, os autores desenvolveram um *ensemble* utilizando *Bootstrap aggregating* com floresta aleatória (*Random Forest*, RF), que distinguiu STC de não-STC com resultados promissores, alcançando acurácia de 95.5%, precisão de 96.3% e revocação de 91.9%. Paralelamente, foi criado um modelo preditivo para estimar a probabilidade de melhora clínica após o processo de hidrodissecção guiada por ultrassonografia, com avaliações realizadas em três intervalos: 1, 3 e 6 meses. O desempenho prognóstico evidenciou acurácias de 87.7% após um mês, 90.1% após três meses e 91.2% após seis meses, demonstrando que os melhores resultados foram obtidos com o acompanhamento de longo prazo.

Elseddik *et al.* [78] propuseram uma abordagem híbrida para diagnóstico, predição e monitoramento da STC utilizando DL e ML. Com uma amostra de 160 pacientes; 80 com STC e 80 controles, os autores integraram dados clínicos, históricos, escores do *Boston Carpal Tunnel Questionnaire* (BCTQ), resultados de estudos de condução nervosa (*Nerve Conduction Studies*, NCS) e medições da área seccional do nervo mediano (*Cross Sectional Area*, CSA) por ultrassom. No diagnóstico, foi implementada uma DNN com 6 camadas (entrada, 4 ocultas com *ReLU* e saída com *sigmoid*), utilizando regularização L2 e *dropout*. Entre os otimizadores testados (*Gradient descent*, *Adagrad* e Adam), o Adam obteve melhor desempenho, com acurácia de 96.9%, precisão de 98.2%, revocação de 96.3% e AUC de 97.2%. A análise com *Shapley Additive Explanations* destacou a relevância da CSA inicial, dos escores *Functional Status Scale* e dos sinais clínicos. Para a predição da evolução do tratamento, os autores aplicaram modelos de regressão; um Perceptron de Múltiplas Camadas (MLP) e uma Floresta Aleatória (RF); para estimar a variação da CSA após 1, 3 e 6 meses de injeção terapêutica. O MLP apresentou melhor desempenho aos 6 meses, alcançando acurácia de 95.22%, coeficiente de determinação de 0.667, *Median Aboslute Error* de 0.0132 e *Mean Square Error* de 0.0639.

Na Tabela 3.1, apresentamos um resumo dos principais trabalhos de ML aplicados ao diagnóstico e à classificação da gravidade da STC encontrados na literatura. Essa tabela inclui informações sobre o problema abordado, as técnicas utilizadas e os resultados obtidos para cada referência. A contribuição do presente trabalho em relação aos estudos revisados está na utilização do modelo gerador CTGAN combinado ao SMOTE para aumento e balanceamento de dados sintéticos, aliado à avaliação de uma arquitetura de redes neurais profundas, com o objetivo de melhorar o desempenho na classificação da gravidade da STC.

Tabela 3.1: Visão geral dos trabalhos sobre aplicações de ML.

Ano	**Autores**	**Problemas**	**Técnicas**	**Resultados**
2021	Park *et al.* [1]	Classificação da gravidade da STC utilizando a abordagem multiclasse	*Neural Network*	Acurácia: 73.70%
			SVM	Acurácia: 74.50%
			KNN	Acurácia: 75.60%
			Classification and Regression Tree	Acurácia: 70.8%
			Random Forest	Acurácia: 75.60%
			Stochastic Gradient Boosting	Acurácia: 73.40%
			XGBoost	Acurácia: 76.60%
2021	Tamis *et al.* [75]	Diagnóstico da STC (classificação multiclasse)	SVM	Acurácia: 95.1%
2022	Harrison *et al.* [76]	Identificação de pacientes que se beneficiariam de cirurgia de descompressão (classificação binária)	Regressão Logística	Acurácia: 69.80%
			KNN	Acurácia: 67.90%
			SVM	Acurácia: 70.60%
			XGBoost	Acurácia: 71.80%
			RNA	Acurácia: 66.00%
2022	Smerilli *et al.* [77]	Segmentação automática do nervo mediano em ultrassonografia	Mask R-CNN (ResNet101 + *Feature Pyramid Network*)	Precisão: 96.00% Revocação: 98.00%
2023	Elseddik *et al.* [19]	Diagnóstico da STC (classificação binária)	*Ensemble* (RF + *Bagging*)	Acurácia: 95.50% Revocação: 91.90% Precisão: 96.30%
2023	Elseddik *et al.* [78]	Diagnóstico da STC (classificação binária)	DNN	Acurácia: 96.90% Revocação: 96.30% Precisão: 98.20%

3.2 Sistemas CAD e MLOps: Desafios e Aplicações no Contexto Médico

Dominik *et al.* [79] discutem as dificuldades comuns na implantação de projetos de ML em produção, especialmente a dependência de intervenções manuais e a falta de uma metodologia sistemática, e defendem a adoção de práticas de MLOps para superar essas barreiras. A partir de revisão bibliográfica e entrevistas, os autores identificam nove princípios essenciais: CI/CD, orquestração de *workflows*, reprodutibilidade, versionamento, colaboração interdisciplinar, treino/avaliação contínuos, rastreamento de metadados, monitoramento e ciclos de *feedback*. Com base nesses princípios, propõem componentes técnicos críticos (repositórios de código, *feature stores*, infraestruturas de treinamento, registro/versionamento de modelos e módulos de *deploy*/monitoramento) e ressaltam a necessidade de papéis bem definidos entre *stakeholders* e engenheiros.

Novo *et al.* [14] propuseram o desenvolvimento do Wivern, um sistema baseado na *web* voltado para o apoio ao diagnóstico e colaboração interdisciplinar em pesquisas sobre doenças vasculares. A metodologia consistiu na criação de uma plataforma modular que integra ferramentas CAD automatizadas e semiautomatizadas para análise de retinografias, ultrassonografias carotídeas e sinais de pressão arterial, permitindo o cálculo automatizado de índices de risco cardiovascular (como SCORE, *Framingham* e *Metabolic Syndrome*). A interface é acessada via navegador e implementada com tecnologias HTML5, JavaScript e Java EE, proporcionando compatibilidade entre dispositivos e facilidade de uso clínico. O sistema foi validado com mais de 800 pacientes, processando dados clínicos, 800 imagens de retina e 400 imagens de carótidas. Os resultados demonstraram alta precisão das ferramentas de micro e macrocirculação, com destaque para o módulo de análise de vasos retinianos, que obteve 99,15% de precisão na detecção de vasos.

Sethanan *et al.* [13] desenvolveram o TB-DRD-CXR, um CAD com interface *web*, projetado para classificar automaticamente distintos subtipos de tuberculose resistente a medicamentos com base em imagens de radiografia torácica. A pesquisa parte da demanda por soluções diagnósticas ágeis, precisas e de fácil acesso, particularmente relevantes em regiões com infraestrutura médica limitada. Para atender a essa necessidade, os autores propuseram um modelo de DL baseado em *ensemble* heterogêneo, que incorpora métodos de segmentação pulmonar, técnicas de aumento de dados, estratégias otimizadas de ajuste da taxa de aprendizado (*AdaDeltaS* e CLR), além de um mecanismo de fusão de decisões denominado HyVaN-AMIS. O sistema foi treinado com 5039 imagens de radiografia torácica e validado em um conjunto independente de 760 imagens, alcançando uma acurácia média de 96,1%. Quando comparado a arquiteturas convencionais como *DenseNet201*, *EfficientNet* e *ConvNeXtSmall*, o modelo proposto apresentou melhoria relativa de até

93,4% em termos de acurácia, além de demonstrar excelente aceitabilidade, evidenciada por um escore médio de 96,7% no *System Usability Scale* entre os 33 profissionais da saúde que participaram dos testes.

Segundo o artigo de Shethiya [80], o qual aborda a implantação de modelos de ML em aplicações *web* .NET utilizando o *Azure Kubernetes Service* (AKS), são ressaltadas as vantagens de adotar uma arquitetura de microsserviços combinada com práticas de MLOps para aprimorar o desempenho, escalabilidade e segurança. Destaca-se que o AKS possibilita aos desenvolvedores empacotar contêineres de IA de forma similar aos contêineres Docker, facilitando a integração com aplicações .NET e promovendo uma solução mais consistente, portátil e flexível. A incorporação do Kubernetes permite o balanceamento de carga, o escaneamento automático e a autorrecuperação, assegurando alta disponibilidade e desempenho mesmo sob demandas elevadas. Além disso, o AKS oferece suporte a robustas *pipelines* de CI/CD. Ferramentas como Azure DevOps e GitHub Actions podem ser utilizadas para automatizar o re-treinamento, teste e implantação dos modelos, tornando o ciclo de vida do aprendizado de máquina mais eficiente e robusto. Por fim, o artigo evidencia a importância do gerenciamento de recursos por meio do AKS, incluindo controle de acesso, políticas de rede e gerenciamento de segredos, o que garante a segurança dos modelos de IA e dos dados das aplicações, conforme os padrões da indústria. A descentralização da lógica dos modelos de IA por meio de microsserviços também é enfatizada, permitindo que serviços de inferência sejam implantados de forma independente e escalados conforme a demanda, sem afetar o desempenho geral da aplicação .NET.

A Tabela 3.3 apresenta uma síntese dos principais trabalhos da literatura que abordam a operacionalização de modelos de ML na área médica. Os estudos selecionados destacam desafios relacionados ao desenvolvimento, implantação e escalabilidade de sistemas CAD, bem como práticas associadas ao ciclo de vida de modelos de ML em ambientes produtivos. Neste trabalho, busca-se incorporar as melhores práticas de MLOps e estratégias de desenvolvimento de sistemas CAD, com o objetivo de promover maior eficiência, reprodutibilidade e integração dos modelos de ML ao fluxo de trabalho clínico, seguindo as boas práticas e técnicas mencionadas na literatura.

Tabela 3.3: Visão geral dos estudos que abordam o desenvolvimento e aplicação de sistemas CAD.

Ano	**Autores**	**Problemas**	**Técnicas**
2017	Novo *et al.* [14]	Desenvolvimento de uma plataforma *web* para integrar CAD e colaboração médica no estudo de doenças vasculares.	JavaEE, CSS, JavaScript, C++, PostgreSQL, MVC, cliente-servidor
2023	Sethanan *et al.* [13]	Desenvolvimento de uma interface *web* para a classificação automática de subtipos de tuberculose resistente com apoio diagnóstico por imagens de radiografia torácica.	HTML, JavaScript, *U-Net*, *Convolutional Neural Network*
2025	Shethiya [80]	Implantação de modelos de ML escaláveis em aplicações *web* .NET utilizando AKS seguindo as melhores práticas de MLOps.	.NET, AKS, MVC, CI/CD, *Azure Container Registry*

Capítulo 4

Metodologia

Neste trabalho, comparam-se algoritmos de ML para classificar a gravidade da STC (leve, moderada e grave) a partir de dados clínicos. Além disso, é desenvolvida uma ferramenta *web* para auxiliar profissionais de saúde na avaliação da gravidade STC em seus pacientes a partir dos melhores modelos de ML analisados neste trabalho. A Figura 4.1 apresenta a metodologia proposta neste trabalho. A partir da base original, realiza-se a divisão das amostras em treinamento (80%) e teste (20%), preservando a mesma proporção inicial entre as classes. Em seguida, aplica-se validação cruzada com cinco iterações sobre o conjunto de treinamento; em cada iteração, o conjunto é novamente particionado em treino (80% das amostras) e validação (20% das amostras). Em cada iteração, treinam-se sete modelos (seis de ML tradicionais e um de DL) e avalia-se o desempenho multiclasse, registrando, para cada modelo, a melhor acurácia obtida no *fold* correspondente. As únicas variações experimentais referem-se às estratégias de aumento e balanceamento de dados (base original, SMOTE e CTGAN), assegurando que, através da utilização de *seeds*, quaisquer diferenças de desempenho decorram exclusivamente das técnicas de geração e balanceamento aplicadas. Por fim, os modelos com melhores acurácias são avaliados no conjunto de teste a partir da análise do desempenho entre as classes para identificar quais modelos serão considerados para uso na ferramenta web.

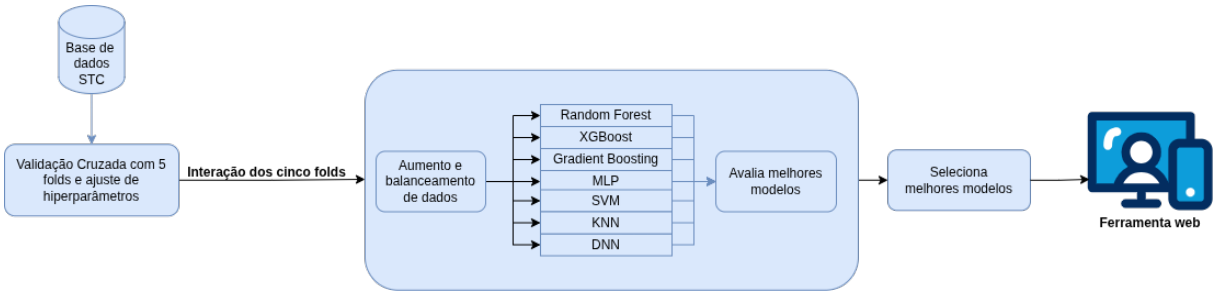


Figura 4.1: Visão geral da metodologia proposta.

As próximas seções apresentam, de forma encadeada, as etapas do fluxo metodológico. Inicialmente, a Seção 4.1 descreve a base de dados. Em seguida, a Seção 4.2 justifica o uso da validação cruzada com cinco *folds* e, na Seção 4.3, detalha-se como o ajuste de hiperparâmetros é conduzido em cada *fold*. Ademais, a Seção 4.4 define os cenários de aumento e balanceamento de dados que alimentam a modelagem. A Seção 4.5 especifica o *pipeline* de algoritmos de ML e as configurações adotadas para cada modelo, enquanto a Seção 4.6 descreve a estratégia de treinamento correspondente. Na sequência, a Seção 4.7 apresenta os procedimentos de avaliação e os experimentos realizados, sintetizando métricas e análises. Por fim, a Seção 4.8 expõe a arquitetura da ferramenta *web* desenvolvida e sua integração ao *pipeline* proposto.

4.1 Base de Dados

Nesta Seção é abordada a origem e as principais características da base de dados utilizada no estudo, além do detalhamento dos onze atributos que compõem a mesma, discutindo criticamente a relevância de cada variável para a estratificação da gravidade da STC, observando, por exemplo, o limitado poder discriminatório de idade e sexo, a correlação positiva do índice de massa corporal com formas mais graves da doença e a elevada capacidade diagnóstica dos parâmetros ultrassonográficos, sobretudo da área de secção transversal.

Origem dos Dados

A base de dados utilizada neste trabalho foi disponibilizada por Park *et al.* [1], a qual foi construída a partir de informações clínicas coletadas no setor de neurologia do Hospital Universitário da Universidade de Kaferelshikh, no Egito. Foram incluídos pacientes diagnosticados com STC por ultrassonografia, bem como indivíduos com condições clínicas semelhantes, como radiculopatia cervical e neuropatias periféricas. No entanto, foram excluídos os seguintes casos: outras lesões concomitantes de nervo periférico; radiculopatia cervical inferior; doença vascular periférica; artrite na mão ou punho; histórico de cirurgias anteriores realizadas na mão ou no punho; e dados faltantes ou inconsistentes [1]. O grupo-controle foi pareado por idade e sexo, originando uma base de dados com 1037 amostras de mãos, cuja distribuição de gravidade da STC revelou-se desbalanceada: 507 casos leves, 276 moderados e 254 graves.

Variáveis e Atributos Coletados

Como mencionado na Seção 4.1, a base de dados final é composta por 1037 amostras de mãos, cada uma descrita por 11 atributos preditores e um atributo-alvo, totalizando doze colunas. A Tabela 4.1 resume a organização desses atributos, já convertidos para os nomes de campo utilizados na planilha analítica.

Tabela 4.1: Atributos clínicos disponíveis na base de dados por Park *et al.* [1].

Categoria (n)	**Atributo (nome do campo)**	**Tipo**
Demográficos (3)	*age* – idade (anos)	Numérico
	sex – sexo (0 = feminino, 1 = masculino)	Binário
	bmi – indíce de massa corporal (kg/m^2)	Numérico
Comorbidade (1)	*diabetes* – presença de diabetes (0=não, 1=sim)	Binário
Características clínicas (4)	*symptom*_duration – duração dos sintomas (meses)	Numérico
	nrs – escore de escala numérica de dor (0–10)	Numérico
	np – dor noturna (0=não, 1=sim)	Binário
	weakness – fraqueza/atrofia tenar (0=não, 1=sim)	Binário
Lateralidade (1)	*side* – lado afetado (0 = *left*, 1 = *right*)	Binário
Ultrassom (2)	*csa* – área transversal do nervo mediano (mm^2)	Numérico
	pb – abaulamento palmar do retináculo flexor (mm)	Numérico
Atributo-alvo (1)	*severity* – grau eletrodiagnóstico (leve | moderada | grave)	Categórico

Idade. Embora idades mais avançadas aumentem a prevalência de STC, em Park *et al.* [1] não foram encontradas diferenças significativas na distribuição etária entre os grupos leve, moderada e severa, sugerindo que a idade, isoladamente, não explica a gravidade. Estudos recentes indicam que a degeneração neural associada ao envelhecimento agrava a STC. Aghda *et al.* [81] mostraram que pacientes mais idosos sofrem maior perda axonal e desmielinização do que pacientes jovens. Do mesmo modo, Grönfors *et al.* [82] verificaram por ultrassonografia que ter mais de 60 anos, combinado a sinais de perda axonal, aumenta a probabilidade de o nervo mediano exibir CSA acima de 18 mm^2.

Sexo. Modelos de ML indicaram baixa importância preditiva do sexo, apesar de o sexo feminino ser fator de risco para desenvolver STC, ele pouco contribui para estratificar sua gravidade [83].

Índice de massa corporal. Diferentemente da idade e do sexo, o índice de massa corporal mostrou correlação direta com a gravidade, visto que, fisiopatologicamente, maior adiposidade eleva a pressão no túnel do carpo [1, 83].

Diabetes. A prevalência de diabetes apresentou baixa relevância nos modelos de ML utilizados em Park *et al.* [1].

Duração dos sintomas. Tempo prolongado de sintomas refletiu compressão crônica, apresentando forte correlação com a gravidade eletrodiagnóstica, coerente com a deterioração funcional progressiva do nervo mediano [1].

Intensidade da dor. A dor reflete tanto a intensidade quanto a duração da compressão, aparecendo como indicador clínico confiável de piora [1].

Dor noturna. A presença de dor noturna se mostrou redundante em relação à intensidade da dor nos modelos de ML utilizados em Park *et al.* [1].

Fraqueza ou atrofia tenar. Sinal clínico de comprometimento motor do nervo mediano que indica denervação crônica, usualmente confirmada por informações eletromiográficas de degeneração axonal. Do ponto de vista clínico, sua presença reforça o diagnóstico de STC e auxilia na estratificação de gravidade, pois se associa às formas moderadas a graves, correlacionando-se com pior prognóstico funcional e maior probabilidade de necessidade de descompressão cirúrgica [84].

Lateralidade. A literatura permanece inconclusiva quanto à influência da dominância manual na gravidade da STC [83].

Área de secção transversal. Revisões clínicas recentes confirmam que a medição ultrassonográfica da CSA do nervo mediano no nível do pisiforme é o parâmetro mais sensível e específico para classificar a STC, além de correlacionar-se de forma consistente com os estágios eletrofisiológicos de gravidade. Lin *et al.* [85] sintetizaram dezesseis revisões e apontaram *thresholds* entre 9 e 10.5 mm^2 como os de melhor desempenho (sensibilidade $\approx$ 87–90%; especificidade $\approx$ 85–88%), observando aumento progressivo da CSA conforme a piora clínica. Corroborando esse achado, Casas *et al.* [86] identificaram a CSA como discriminador principal, superando índices derivados e outros parâmetros morfológicos, reforçando seu valor como biomarcador de gravidade.

Abaulamento Palmar do Retináculo Flexor. Segundo *Braham et al.* [87] o PB maior que 2 mm se associou fortemente a maiores latências motoras distais e menores velocidades sensitivas, elevando a acurácia diagnóstica clínica global para 85%. Resultados complementares de Ng *et al.* [88] mostraram que PB 1 mm na saída do túnel apresenta especificidade de 97% e, quando combinado com CSA maior que 14 mm^2, atinge sensibilidade de 100% e acurácia de 93% [89]. Coletivamente, esses resultados indicam que a inclusão do PB maior que 1 ou 2 mm, em conjunto com a CSA, refina a estratificação de gravidade e aumenta a confiabilidade.

Gravidade Na STC, a gravidade é comumente graduada em leve, moderada e grave a partir da combinação entre quadro clínico e informações eletrofisiológicas. Na forma leve, predominam parestesias noturnas sem déficit motor; nos estudos de condução observa-se uma lentidão sensitiva com latência motora distal preservada [90, 91, 92]. Na forma moderada, os sintomas tornam-se diurnos, com hipoestesia objetiva e possível perda de destreza; eletrofisiologicamente, além das alterações sensitivas, há prolongamento da latência motora distal [90, 91]. Nos casos graves, costuma haver dormência persistente, fraqueza de oposição/abdução do polegar e atrofia tenar; nos exames, é frequente a ausência do potencial sensitivo mediano e marcada redução (ou ausência, em estágios extremos) da resposta motora [90, 91, 93, 94].

4.2 Validação Cruzada

A robustez dos resultados foi assegurada mediante a utilização da técnica de validação cruzada *k-fold* estratificada. Estudos de simulação mostram que, em amostras pequenas, essa estratégia oferece estimativas menos enviesadas do que divisões simples treino–teste e ajuda a conter o *overfitting* [95]. No contexto específico da STC, Yetiş *et al.* [10] empregaram validação cruzada repetida para prever a presença de STC e relataram que o procedimento estabilizou a acurácia dos algoritmos [10]. Inspirado nessas evidências, este trabalho estabeleceu 5 *folds* de tamanho aproximadamente igual, equilibrando o custo computacional e a estabilidade das métricas; dessa forma, cada amostra atuou exatamente uma vez como validação e quatro vezes como treino para o desenvolvimento dos modelos de ML.

Utilizou-se a técnica de embaralhamento dos índices (*shuffle*) antes da divisão, a fim de prevenir viés decorrente da ordem original da base de dados. A semente do gerador de números pseudoaleatórios foi fixada, garantindo total reprodutibilidade das divisões. A divisão estratificada preserva, em cada *fold*, a proporção natural entre as classes leve, moderada e severa, o que é essencial para a avaliação de algoritmos em cenários desbalanceados.

Em cada iteração $i \in \{1, \ldots, 5\}$, quatro *folds* ($\sim 80\%$ dos dados originais) formam o conjunto de treinamento, enquanto o *fold* remanescente ($\sim 20\%$) constitui o conjunto de validação do melhor modelo identificado no procedimento de validação cruzada. Para todos os cinco *folds*, a distribuição original das classes nos conjuntos de treino e de validação manteve-se praticamente constante, em torno de 48% de amostras para a classe leve, 26% para a moderada e 26% para a grave. Esses resultados foram obtidos considerando a média dos cinco *folds* antes de realizar as operações de aumento e balanceamento. A Tabela 4.3 apresenta os valores médios observados da quantidade de amostras separadas

para cada classe no conjunto de treinamento e validação. Nesse estado inicial, antes de qualquer operação de aumento ou balanceamento, observa-se que tanto o conjunto de treinamento quanto o de validação preservam a mesma proporção das três classes, garantindo consistência estatística ao longo dos *folds*.

Tabela 4.3: Distribuição das amostras entre as classes em 80% da base de dados antes dos experimentos (média sobre os 5 *folds*).

Conjunto	Leve	Moderada	Grave	Total
Treinamento	320 (48.1%)	174 (26.2%)	170 (25.6%)	664
Validação	80 (48.2%)	43 (25.9%)	43 (25.9%)	166
Total	400	217	213	**830**

4.3 Otimização de Hiperparâmetros

Com o objetivo de obter o melhor desempenho possível de cada algoritmo, foi realizada uma etapa dedicada à otimização de hiperparâmetros, utilizando a técnica de *Grid Search* [96]. Essa etapa é fundamental para ajustar as configurações internas de cada algoritmo e maximizar a sua capacidade preditiva, especialmente em cenários de dados desbalanceados ou aumentados artificialmente.

O *Grid Search* [96] foi empregado como estratégia sistemática de busca exaustiva, testando combinações de parâmetros pré-definidos para cada algoritmo. Por conta de limitação de *hardware*, não foi utilizado o *Grid Search* para buscar os melhores hiperparâmetros dos algoritmos MLP e DNN. Para estes, foi realizada uma abordagem empírica ajustando os hiperparâmetros de acordo com a observação dos melhores resultados obtidos nos experimentos realizados.

A Tabela 4.4 apresenta o conjunto de hiperparâmetros e as respectivas faixas de valores testados via *Grid Search* para cada algoritmo de ML . Cada conjunto de hiperparâmetros foi avaliado por meio de validação cruzada *k-fold* (Seção 4.2), buscando identificar a configuração que equilibrasse melhor a acurácia e a capacidade de generalização, como é descrito mais detalhadamente na Seção 4.5.

Tabela 4.4: Hiperparâmetros e intervalos de valores avaliados para ajuste dos algoritmos de ML.

Modelo	**Hiperparâmetro**	**Valores testados**
Random Forest	Número de árvores	[50, 100, 150]
	Profundidade máxima	[5, 10, 15]
XGBoost	Número de árvores	[50, 100]
	Profundidade máxima	[3, 5, 7]
	Taxa de aprendizado	[0.01, 0.1, 0.2]
Gradient Boosting	Número de árvores	[100, 200, 300]
	Taxa de aprendizado	[0.05, 0.1, 0.2]
	Profundidade máxima	[2, 3, 4]
	Subamostragem	[0.6, 0.8, 1.0]
SVM	Kernel	[rbf, linear]
	Penalidade *(C)*	[0.1, 1, 3, 10, 30, 100]
	Gamma	[0.01, 0.03, 0.1, 0.3, scale]
KNN	Número de vizinhos	[3, 5, 7, 11]
	Ponderação	[*uniform*, *distance*]
	Distância *(p)*	[1, 2]
	Tamanho da folha	[15, 30, 45]
MLP	Camadas ocultas	[32, 64]
	Taxa de *dropout*	[0.1 – 0.3]
	Regularização L2	[0,0001 – 0,001]
	Taxa de aprendizado	[0.001 – 0.0005]
DNN	Camadas ocultas	[256,128,64], [128,64,32,16], [128,128,64,64]
	Taxa de *dropout*	[0.3 – 0.5]
	Regularização L2	[0.001 – 0.0005]
	Taxa de aprendizado	[0.001 – 0.0005]
	Épocas	[120, 300]
	Tamanho do *batch*	[16, 32]

4.4 Aumento e Balanceamento dos Dados

Um dos principais obstáculos enfrentados neste trabalho foi a limitação da base de dados original, tanto em volume total de amostras quanto na distribuição desigual entre as classes. Esses cenários comprometem a eficácia de modelos de ML e, principalmente, DL, que são caracterizados pela alta capacidade de representação, capazes de aprender características muito complexas dos dados. Para realizar a estimativa de parâmetros de forma confiável e evitar que a rede memorize apenas os exemplos de treinamento (*overfitting*), é preciso dispor de grande quantidade de dados rotulados [97, 98]. Dessa forma, Alzubaidi *et al.* [97] observam que *"grandes conjuntos de dados de treinamento (por exemplo, o ImageNet) asseguram um desempenho adequado do DL, enquanto dados de treinamento insuficientes levam a resultados ruins"*, e enfatizam que o desempenho ótimo do modelo exige um grande montante de dados. Yousef e Allmer [99] também ressaltam que modelos de DL *"requerem um grande conjunto de dados para aprender padrões complexos"* e que dados insuficientes, em relação à capacidade do modelo, levam a sobreajuste e baixa generalização.

Além do aumento de dados previamente justificado, fez-se necessário tratar o desbalanceamento das classes na base original. Distribuições assimétricas tendem a enviesar os classificadores em direção à classe majoritária, degradando sistematicamente o desempenho nas classes minoritárias, efeito amplamente documentado em ML e DL [61, 100]. Embora certos modelos baseados em árvores e *ensembles* apresentem alguma robustez a assimetrias moderadas, a literatura indica que nenhum algoritmo é intrinsecamente imune aos efeitos do desbalanceamento [101].

Diante disso, adotou-se estratégias de aumento e balanceamento dos dados com o objetivo de mitigar o impacto do desbalanceamento, viabilizar o uso de algoritmos mais complexos e garantir maior robustez nas predições. Para esse fim, duas técnicas distintas foram utilizadas: o CTGAN (Seção 2.5.3) e o SMOTE (Seção 2.5.2). Ambas as técnicas foram aplicadas em diferentes experimentos, com o objetivo de avaliar seu impacto no desempenho dos algoritmos de aprendizado de máquina e aprendizado profundo. Os experimentos conduzidos permitiram comparar o comportamento dos algoritmos em cenários com e sem aumento e balanceamento dos dados, sendo os detalhes dessas análises descritos na Seção 4.7.

Para a utilização das técnicas de aumento e balanceamento, adotou-se o seguinte critério: o SMOTE é empregado exclusivamente para balanceamento de classes, ao passo que o CTGAN é utilizado tanto para aumento de dados (sem alterar a proporção entre classes) quanto para balanceamento (igualando as contagens entre classes).

Balancear com SMOTE. O balanceamento supervisionado é realizado por superamostragem sintética das classes minoritárias via SMOTE, tomando como referência a

classe majoritária. Utiliza-se $k = 5$ vizinhos na etapa de interpolação, gerando novas instâncias até que cada classe atinja a contagem da classe com maior número de amostras. Não há modificação na distribuição da classe majoritária, e nenhuma etapa de aumento global é aplicada nesta configuração.

Aumentar com CTGAN (sem balancear). Para aumento global do conjunto, treina-se um CTGAN do zero por 500 épocas (sem parada antecipada), com *batch size* definido como $\min(64, N)$, `pac`=1 e aceleração por GPU quando disponível. Em seguida, são amostradas 700 instâncias sintéticas no total. Como o modelo é treinado sobre a base completa (incluindo a coluna de classe), a quantidade gerada por classe tende a ser *proporcional* à distribuição original; isto é, classes mais frequentes recebem mais amostras sintéticas, preservando o desbalanceamento existente.

Balancear com CTGAN. Para balanceamento com geração sintética, calcula-se a contagem da classe majoritária e, para cada classe c, treina-se um CTGAN exclusivo apenas com as amostras de c (mesma configuração de 500 épocas, sem *early stopping*). Gera-se exatamente o número de instâncias faltantes (*gap* até a classe majoritária) e, por fim, concatena-se o conjunto sintético ao original, igualando as contagens entre as classes sem ultrapassar a referência da classe majoritária.

4.5 *Pipeline* de Algoritmos de ML e DL

Com o objetivo de simular um ambiente de testes para a comparação entre diferentes algoritmos de ML, foi construído um *pipeline* composto por múltiplos algoritmos executados em paralelo. Essa abordagem permite não apenas a comparação direta entre diferentes algoritmos de ML e DL, mas também a análise do impacto de diferentes estratégias de aumento e balanceamento de dados sobre cada algoritmo.

O *pipeline* foi configurado para executar sete algoritmos distintos: *Random Forest* [37], XGBoost [41], *Gradient Boosting* [37], SVM [37], KNN [37], MLP [46] e DNN [46], apresentados na Seção 2.4.3. A seguir, são descritas brevemente as configurações utilizadas para cada um dos algoritmos de acordo com os hiperparâmetros ajustados, como descrito na Seção 4.3.

O *Random Forest* considera uma quantidade específica de árvores de decisão, cada uma com profundidade máxima limitada, buscando equilibrar desempenho e evitar o *overfitting*. Para a otimização, foram exploradas variações na profundidade e na quantidade de árvores.

Utilizou-se o XGBoost como classificador multiclasse (multi:softmax, três classes), retornando diretamente o rótulo previsto. Para reprodutibilidade, fixou-se a semente 42.

A otimização considerou combinações simples de número de árvores, profundidade das árvores e taxa de aprendizado, buscando equilíbrio entre desempenho e estabilidade.

O *Gradient Boosting* foi configurado com um conjunto específico de árvores, utilizando uma taxa de aprendizado e profundidade máxima. Para promover diversidade entre as árvores e melhorar a capacidade de generalização do algoritmo, foi aplicada uma taxa de amostragem de 80%.

A SVM foi implementada adotando por padrão o *kernel* RBF (C = 1,0; *gamma* = *scale*; semente = 42). Também foram avaliados o *kernel* linear e variações de C e *gamma* para controlar a margem e a complexidade do classificador. Quando pertinente, a implementação permite ativar ponderação de classes para mitigar desbalanceamento. Para métricas baseadas em probabilidade (por exemplo, ROC/AUC), é possível habilitar a estimação probabilística e utilizar *predict_proba*, mantendo o restante do fluxo inalterado.

O KNN foi implementado com o número de vizinhos mais próximos definidos, ponderando a influência de cada um deles com base na distância até o ponto avaliado. Foi utilizada a métrica de distância euclidiana (Minkowski com hiperparâmetro p igual a 2). A otimização considerou variações no número de vizinhos, nos métodos de ponderação e no tamanho da folha para navegação mais eficiente na estrutura de vizinhança. Foi utilizado também o hiperparâmetro *gamma*, que controla quanto os vizinhos muito próximos pesam em relação aos mais distantes. Com *gamma* menor (0,01; 0,03), mais vizinhos influenciam a decisão, deixando o modelo mais suave e resistente a ruído. Com *gamma* intermediário/maior (0,1; 0,3), quase só os vizinhos bem próximos contam, tornando o classificador mais sensível a detalhes locais, o que pode melhorar a separação entre classes, mas também aumenta o risco de *overfitting*. A opção *scale* calcula automaticamente um valor baseado nos próprios dados (funciona melhor quando as variáveis estão normalizadas) e serve como ponto de partida estável.

A MLP consistiu em uma estrutura rasa, com onze neurônios na camada de entrada, representando as onze dimensões das variáveis de entrada, três neurônios na camada de saída, correspondentes às três classes-alvo, duas camadas ocultas, 200 épocas, 67 neurônios e *batch size* no valor de 32, com uma condição de parada medida a partir do decaimento da acurácia em 20 épocas. Além disso, os hiperparâmetros utilizados foram testados de maneira empírica e, diferente dos modelos de ML, foi utilizada uma abordagem manual para a escolha dos melhores hiperparâmetros, tornando-os fixos para os cinco experimentos, como é possível observar na Tabela 4.5.

Por fim, foi utilizado um algoritmo de DNN, baseado na arquitetura do MLP, porém com maior número de camadas e maior capacidade de representação. A rede também operou com onze variáveis de entrada e três de saída, sendo indicada para testar o potencial do aprendizado profundo, especialmente em cenários com dados aumentados. Assim como

a MLP, os melhores hiperparâmetros foram escolhidos de maneira manual e os melhores resultados obtidos, de maneira geral, podem ser observados na Tabela 4.5.

Esse *pipeline* proporciona um cenário controlado e abrangente para a avaliação dos algoritmos, permitindo investigar quais modelos apresentam melhor desempenho no contexto específico da base analisada, tanto em termos de acurácia quanto de equilíbrio entre as classes.

Tabela 4.5: Melhores hiperparâmetros selecionados para as redes neurais.

Algoritmo	**Hiperparâmetro**	**Valor**
MLP	Camadas ocultas	2
	Taxa de *dropout*	0.1
	Regularização L2	0.0001
DNN	Camadas ocultas	[128, 128, 64, 64]
	Taxa de *dropout*	0.4
	Regularização L2	0.001
	Épocas	120
	Tamanho do *batch*	16

4.6 Treinamento dos Modelos

Na etapa de treinamento, 830 amostras (80%) dos dados são destinadas ao ajuste do modelo e enquanto os 207 restantes (20%) compõem o conjunto de teste. Dentro desses 80%, aplica-se a validação cruzada, que divide os dados em diferentes *folds*, alternando-os entre conjuntos de treino e validação, assim como mencionado na Seção 4.2. Esse procedimento garante que todos os dados sejam utilizados durante os experimentos definidos na Seção 4.7.

Em cada divisão da validação cruzada, os algoritmos de ML e suas configurações (descritos na Seção 4.5) foram treinados em paralelo com as configurações de hiperparâmetros definidas na Seção 4.3. Na etapa de validação, dentre todas as execuções, selecionaram-se os algoritmos e as combinações de hiperparâmetros que obtiveram os melhores resultados segundo o F1-*score*, métrica estabelecida na Seção 4.7.

Para cada algoritmo avaliado, foi calculada a média e o intervalo de confiança dos resultados obtidos nos 5 *folds* da validação cruzada, o que permite uma análise mais

robusta e menos sensível a variações específicas de partição dos dados. A partir dessa média, foi identificado o melhor desempenho global para cada algoritmo.

Adicionalmente, o melhor algoritmo identificado foi aquele que apresentou as melhores métricas a partir do conjunto de teste. Esses resultados foram salvos para permitir análises posteriores mais detalhadas, bem como para eventual reuso ou validação externa do algoritmo. Essa estratégia visa garantir tanto a confiabilidade estatística da avaliação quanto a rastreabilidade das melhores execuções, além de utilizar os melhores resultados na ferramenta *web* proposta.

4.7 Avaliação dos Modelos

A principal métrica adotada para comparar os algoritmos foi o *F1-score*, pois ela combina de forma equilibrada as informações de precisão e de revocação. Em um contexto multiclasse, em que cada paciente deve ser corretamente classificado como Leve, Moderada e Grave STC, é fundamental controlar simultaneamente ambos os tipos de erro: falsos negativos podem atrasar o tratamento de casos graves, enquanto falsos positivos podem levar a intervenções desnecessárias. O *F1-score*, especialmente em sua versão macro, garante que o desempenho seja avaliado de maneira uniforme entre as classes e que nenhum padrão seja injustamente privilegiado ou penalizado, fornecendo assim uma medida robusta para a seleção do algoritmo com melhor desempenho.

Embora o *F1-score* macro tenha sido adotado como a métrica principal de avaliação neste estudo, a revocação foi monitorada como métrica secundária. Isso se justifica pelo caráter crítico do problema médico: em um cenário multiclasse, como o da STC, um valor reduzido de revocação em determinadas classes pode indicar que o modelo está deixando de identificar pacientes relevantes. Na prática, ao analisar a revocação em configurações *one-vs-rest*, considera-se cada classe como positiva em relação às demais. Nesse contexto, falsos negativos representam pacientes que pertencem a essas categorias críticas, mas foram classificados como leves, o que pode atrasar intervenções clínicas necessárias. Assim, acompanhar a revocação por classe permite verificar se o algoritmo está subestimando alguma categoria de maior risco, complementando a visão global fornecida pelo *F1-score*.

Para avaliar a classificação multiclasse da gravidade da STC, foram definidos cinco experimentos baseados em estratégias para aumentar os dados e balancear as classes. Todos os algoritmos foram avaliados por meio de validação cruzada estratificada com 5-*folds*, como apresentado na Seção 4.2, garantindo que a proporção entre as classes fosse preservada em cada partição dos dados. Para assegurar a reprodutibilidade dos resultados, foi adotado um critério fixo de aleatoriedade durante o particionamento. Para cada experimento, são avaliados os melhores hiperparâmetros para ajuste dos modelos a

partir da validação cruzada e considerando a acurácia média dos *folds*. Após a etapa de treinamento, os algoritmos selecionados a partir do melhor *fold* são avaliados a partir do conjunto de teste, considerando as métricas de desempenho apresentadas na Seção 2.6, incluindo a acurácia, F1-*score*, precisão, revocação e AUC por classe.

Base de Dados Desbalanceada e Sem Aumento. Nesta configuração, nenhum método de aumento ou balanceamento é aplicado: a base mantém exatamente a distribuição com que foi disponibilizada. Desse modo, ela cumpre o papel de grupo-controle, fornecendo um ponto de referência para quantificar o impacto das técnicas de reamostragem que serão avaliadas nos outros experimentos. Os resultados obtidos neste experimento refletem, portanto, o desempenho dos algoritmos sobre a distribuição real dos dados, sem qualquer intervenção que altere a proporção entre as classes.

Balanceamento de Classes utilizando SMOTE. Neste experimento, aplica-se o SMOTE para atenuar o desbalanceamento entre as classes. O objetivo é aproximar as frequências das classes, reduzindo o viés do algoritmo em favor da maioria.

Aumento de Dados utilizando CTGAN. Nesta configuração, é observado o CTGAN como técnica de aumento artificial de dados, que é configurada de acordo com as configurações descritas na Seção 4.4. Foi tido como objetivo observar como os algoritmos iriam se comportar diante do aumento das amostras, observando se a implementação geraria algum benefício.

Aumento de Dados utilizando CTGAN e Balanceamento de Classes utilizando SMOTE. Neste experimento, foram utilizados o SMOTE e o CTGAN de forma sequencial para explorar, simultaneamente, a geração de amostras diversificadas e o balanceamento preciso entre as classes. Esse procedimento em dois estágios combina as virtudes de ambos os métodos: a capacidade do CTGAN de reproduzir padrões complexos e, depois, a ação dirigida do SMOTE para eliminar qualquer desbalanceamento residual.

Balanceamento de Classes utilizando CTGAN. O CTGAN é utilizado para o balanceamento dos dados, com o objetivo de observar a maneira como ele se comportaria.

4.8 Aplicacão *Web*

Uma ferramenta *web* foi desenvolvida com o objetivo de auxiliar os especialistas no diagnóstico de STC. Nas próximas seções, são apresentados os detalhes referentes ao desenvolvimento da ferramenta proposta.

4.8.1 Modelagem da Aplicacão

Os requisitos funcionais do sistema incluem o cadastro de usuários, a submissão de dados clínicos, o processamento por algoritmos de ML e a geração de diagnósticos prováveis da STC, classificados em leve, moderada ou severa. Esses requisitos foram inspirados em trabalhos como o de Sethanan *et al.* (2023) [13], que desenvolveram uma aplicação *web* para apoio diagnóstico baseada em classificação automática de imagens médicas.

Quanto aos requisitos não funcionais, destacam-se a usabilidade, a segurança dos dados clínicos, a portabilidade (uso via navegador) e a manutenibilidade da aplicação. Esses aspectos seguem boas práticas descritas em Shethiya (2025) [80], que discute a implantação escalável de modelos de ML em ambientes *web* utilizando princípios de MLOps.

Modelagem dos requisitos

O sistema foi estruturado em diferentes perfis de usuários, cada um com responsabilidades e níveis de acesso específicos. Essa definição garante a separação de funções, a segurança no manuseio dos dados clínicos e o controle adequado sobre as operações realizadas na aplicação. A seguir, apresentam-se as permissões atribuídas a cada perfil:

- **Paciente:** cadastra e-mail, nome completo e dados solicitados pelo médico para a análise clínica (descrito na Seção 4.1) e visualiza seu próprio histórico de predições contendo os dados clínicos, data de criação do histórico, resultado da predição, nome do médico e paciente.
- **Médico:** acessa os dados do paciente, submete-os ao modelo de ML, visualiza o resultado e pode consultar outros médicos e pacientes, porém só tem permissão para criar novos usuários com perfil de paciente.
- **Administrador:** gerencia todo o fluxo de usuários, podendo cadastrar, inativar ou reativar administradores, médicos e pacientes, além de acessar estatísticas e relatórios globais.

Como ilustrado na Figura 4.2, o módulo de predição organiza o fluxo entre os perfis do sistema. O paciente cadastra seus dados pessoais, como e-mail, nome completo e data de nascimento, que servem de entrada para a solicitação de análise realizada por um médico. Nessa etapa, o médico seleciona um dos modelos ML treinados, disponíveis na ferramenta, e submete os dados, retornando como resultado a classificação da gravidade da STC. Todos os resultados são então armazenados em ordem cronológica de acordo com a data de criação do histórico, com possibilidade de filtragem e exportação para Excel

por todos os usuários, respeitando as restrições de visualização de cada perfil e os filtros definidos na consulta.

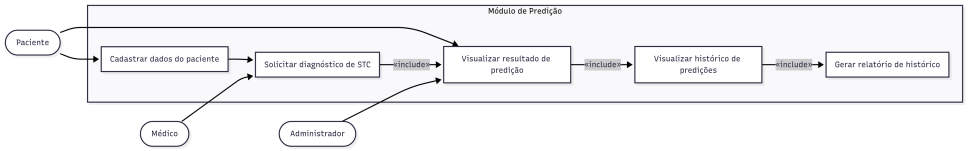


Figura 4.2: Diagrama de caso de uso do Módulo de Predição.

Como ilustrado na Figura 4.3, o **Módulo de Gestão de Usuários** adota permissões estratificadas. O **paciente** não possui privilégios de administração: pode apenas consultar e editar o próprio cadastro. O **médico** pode editar o próprio perfil e *cadastrar e editar novos pacientes*, não sendo permitido cadastrar/editar perfis de médico ou administrador, nem ativar/desativar contas. Já o **administrador** detém a gestão completa: pode cadastrar pacientes, médicos e administradores, editar os dados de qualquer usuário e ativar ou desativar contas. Esses requisitos seguem o princípio do mínimo privilégio e reduzem o risco de alterações indevidas.

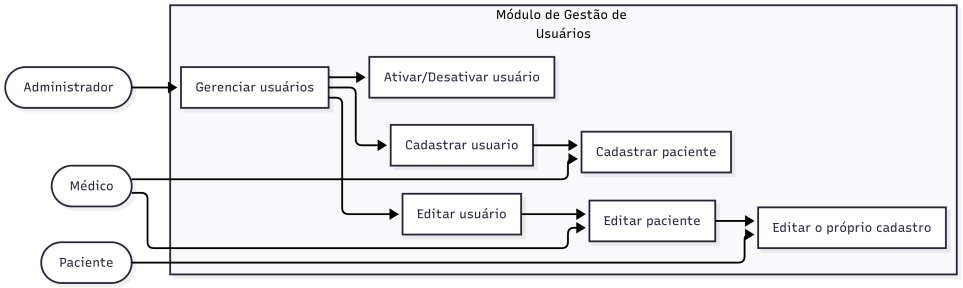


Figura 4.3: Diagrama de caso de uso Módulo de Gestão de Usuários.

Modelagem de Comportamento

Como descrito no diagrama de sequência da ferramenta presente na Figura 4.4, o médico é responsável por iniciar o fluxo de predição da STC. Ele submete os dados clínicos do paciente, conforme descrito na Seção 4.1, ao modelo de ML, que então recupera as informações do paciente, conforme descrito na Seção 4.8.1, e aciona o modelo selecionado para estimar a gravidade da síndrome. Por fim, o resultado é gravado no banco de dados, contendo o modelo selecionado, o resultado da predição e os dados clínicos do paciente, como descrito na Seção 4.8.1, assegurando o registro cronológico das avaliações para consultas e análises futuras.

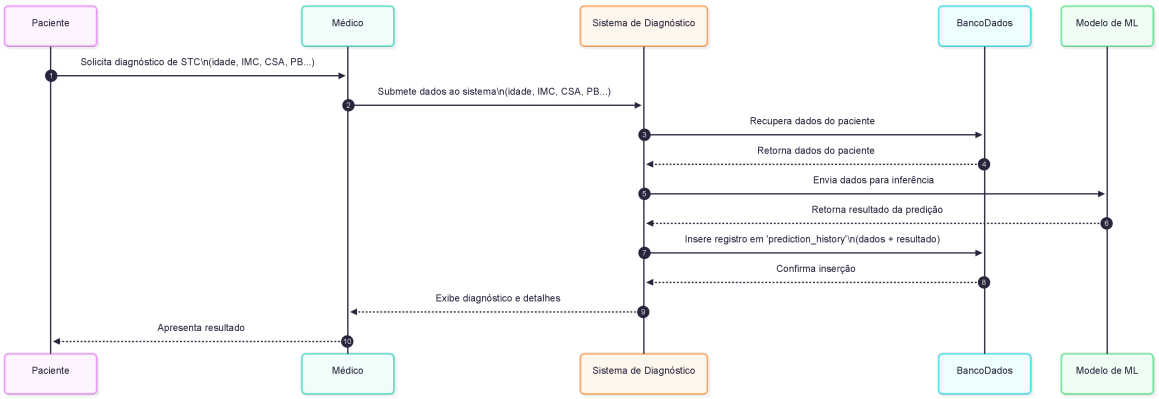


Figura 4.4: Diagrama de sequência para o Modulo de Predição.

Modelagem de Dados

Conforme ilustrado na Figura 4.5, o diagrama de Entidade-Relacionamento evidencia a entidade "USUARIO", que reúne os atributos cadastrais (*id*, *email*, *senha*, *nome_de_usuario*, *nome_completo*, *cpf*, *endereco*) e atributos de controle (*ativo*, *criado_em*, *papel*). A *data_de_nascimento* é utilizada tanto para gestão do usuário quanto para obter a *idade* enviada aos modelos de ML. Por fim, o atributo "papel", indica se o usuário possui o perfil de Administrador, Médico ou Paciente. Os atributos específicos de um Paciente são representados como opcionais na própria entidade, e o controlador do sistema garante que nenhum registro seja criado com campos críticos em branco. Além disso, a entidade "HISTORICO_DE_PREDICOES" persiste o modelo selecionado, o resultado da predição (Leve, Moderada e Grave) e os dados clínicos, descritos na Seção 4.1, que serão submetidos ao modelo de ML. Por fim, o diagrama reforça que a entidade Usuário está relacionada tanto ao usuário que realiza o diagnóstico (médico) quanto ao usuário que é diagnosticado (paciente).

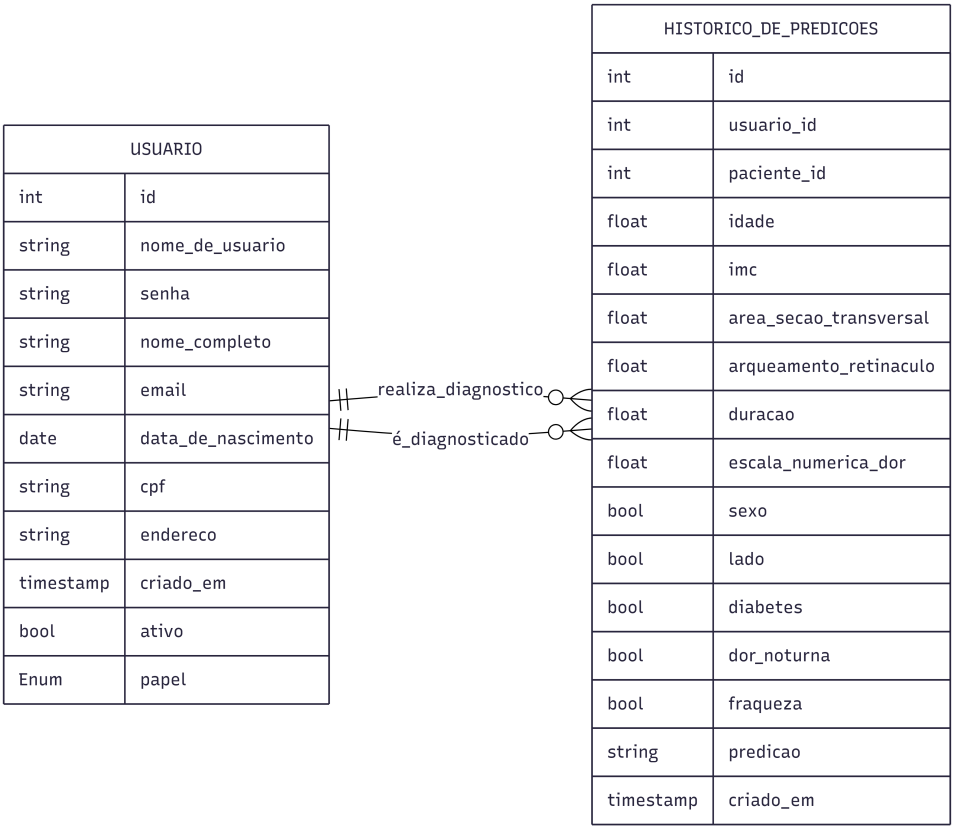


Figura 4.5: Diagrama de Entidade-Relacionamento histórico de predição e usuário.

4.8.2 Arquitetura da Aplicação

Para disponibilizar o algoritmo de ML de forma acessível e interativa, desenvolveu-se a ferramenta *web* Assistente de Diagnóstico da Síndrome do Túnel do Carpo (ADx-STC) seguindo boas práticas de engenharia de software [102], incluindo arquitetura MVC, APIs RESTful e a prática de componentização, garantindo coesão, reutilização e manutenibilidade. Com o objetivo de disponibilizar recursos de apoio ao diagnóstico, facilitar o gerenciamento de pacientes e agilizar a recuperação de dados clínicos.

RESTFul

A comunicação entre cliente e servidor foi implementada seguindo o estilo arquitetural *RESTful*, que favorece simplicidade, escalabilidade e independência entre as camadas da aplicação. Foram adotadas as boas práticas recomendadas, como a utilização adequada dos métodos *HTTP* (GET, POST, PUT, DELETE), o retorno de códigos de status padronizados, a definição de rotas claras e orientadas a recursos, bem como o versionamento explícito dos *endpoints*.

MVC

A arquitetura da ferramenta adota o padrão *Model View Controller* (MVC), conforme apresentado na Seção 2.3.1 da Fundamentação Teórica, garantindo a separação clara das responsabilidades.

- ***Model***: contém as classes e funções responsáveis pelo acesso a dados, incluindo a inicialização do banco de dados, a definição das tabelas de usuários e de histórico de predições, bem como as operações de criação, leitura, atualização e remoção. Esse módulo também encapsula o método que invoca o algoritmo de IA, garantindo que a lógica de inferência permaneça isolada e facilmente testável;
- ***View***: implementada com *templates* genéricos, responsável pela camada de apresentação. Todos os formulários (*login*, cadastro, consulta de histórico, predição) e componentes de interface (tabelas responsivas, formulários e *feedback* de resultados) residem em arquivos de *template*, permitindo a estilização e o uso de bibliotecas de interface sem misturar lógica de aplicação;
- ***Controller***: define os *endpoints* e a lógica de controle. Cada rota relacionada a autenticação, cadastro de dados, predição e consulta de histórico é organizada de forma modular, incluindo validação de sessão, tratamento de erros e conversão de dados de entrada e saída em formatos adequados.

Autenticação e Segurança

A aplicação adota as melhores práticas de segurança para proteção dos dados e controle de acesso. O processo de autenticação de usuários é realizado por meio de credenciais criptografadas, assegurando a confidencialidade das informações de login. Além disso, são utilizados *tokens* de autenticação de sessão, que garantem a integridade das interações durante o uso do sistema e reduzem riscos de sequestro de sessão. Complementarmente, foram seguidas boas práticas de segurança em nível de aplicação e banco de dados, contemplando o princípio de mínimo privilégio, prevenção contra injeções de código e uso de conexões seguras, de modo a garantir a confiabilidade e a proteção dos dados sensíveis dos pacientes.

4.8.3 Integração com Aprendizado de Máquina

A implantação foi planejada segundo dois cenários complementares, sendo: (1) execução direta em ambiente interpretado, permitindo instalação rápida de dependências e inicialização imediata, ideal para desenvolvimento local e validação de alterações; e (2) empaco-

tamento em contêiner, assegurando reprodutibilidade total, isolamento de configurações e facilidade na distribuição e escalonamento em ambientes de nuvem.

Além disso, a aplicação *web* foi integrada a um arquivo *joblib*, responsável por armazenar o estado dos modelos de *IA* previamente treinados. Esse mecanismo permite que o sistema carregue de forma eficiente os melhores modelos selecionados durante a fase experimental, disponibilizando-os para uso em tempo real. Dessa forma, o médico pode solicitar diagnósticos baseados nesses modelos já otimizados, garantindo maior confiabilidade e consistência nos resultados obtidos.

Capítulo 5

Resultados

Neste capítulo, os resultados da otimização dos hiperparâmetros (validação cruzada) e avaliação final (conjunto de teste) serão apresentados conforme cada experimento proposto (Seção 4.7), objetivando observar o comportamento dos algoritmos utilizados e seus desempenhos, indicando quais tiveram os melhores e piores resultados para o contexto apresentado neste trabalho. Ademais, serão descritas quais foram as tecnologias utilizadas e as configurações de ambiente para a realização dos experimentos relacionados aos modelos de ML e para o desenvolvimento da ferramenta *web* para a classificação automática da STC.

5.1 Tecnologias Utilizadas e Configurações de Ambiente

Esta seção apresenta as tecnologias, ferramentas e configurações de ambiente adotadas para o desenvolvimento da aplicação e na condução dos experimentos.

Tecnologias para desenvolvimento

A aplicação *web* foi construída em *Python* 3.8 com o framework *Flask*, estruturada em arquitetura baseada em MVC. No *back-end*, o *Flask* foi responsável pela definição das rotas e pela lógica de negócio. O *front-end* utilizou templates Jinja2 renderizados no servidor, combinando HTML5, CSS3 e JavaScript para a camada de apresentação e interação com o usuário.

Tecnologias de *Machine Learning*

Para preparação e manipulação de dados tabulares foram empregadas *NumPy* e *pandas*. Os modelos clássicos de aprendizado supervisionado foram implementados com

scikit-learn, enquanto *Keras* e *TensorFlow* foram utilizados no desenvolvimento e treinamento de redes neurais. A geração de dados sintéticos realistas foi realizada com a biblioteca *SDV*, por meio do CTGAN.

Ambiente, versionamento e implantação

Os experimentos foram executados em ambiente de nuvem (*Google Colab Pro*), utilizando CPU e memória RAM ampliadas. O código-fonte foi versionado no GitHub, e a aplicação foi implantada (*deploy*) em uma instância Amazon EC2 (AWS), executando Ubuntu 24.04.

5.2 Experimento 1: Base de Dados Desbalanceada e Sem Aumento

O primeiro experimento foi realizado com a base de dados original, sem nenhum aumento de dados ou balanceamento de amostras entre as classes durante o treinamento, conforme definido na Seção 4.7.

Resultados da Validação Cruzada

Adotou-se o *F1-score* (macro) como métrica principal para comparar os modelos; de acordo com essa métrica, obtida na validação cruzada com cinco *folds* observado na Tabela 5.1, os melhores algoritmos foram *Random Forest*, XGBoost e *Gradient Boosting*, com médias semelhantes nas cinco métricas observadas. No entanto, considerando os intervalos de confiança do *F1-score*, verifica-se que o XGBoost se destaca com uma diferença não tão significativa. Além disso, o XGBoost lidera nas métricas de acurácia e revocação, enquanto o *Random Forest* sobressai na precisão.

Tabela 5.1: Média ± 95% IC das métricas obtidas na etapa de treinamento para cada algoritmo do Experimento 1.

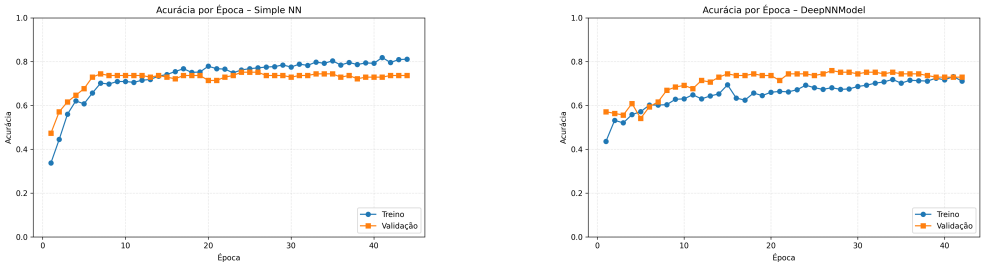
Algoritmo	**Acurácia**	**Precisão**	**Revocação**	***F1-score***
Random Forest	0.754 ± 0.021	**0.745 ± 0.045**	0.714 ± 0.025	**0.724 ± 0.032**
XGBoost	**0.762 ± 0.026**	0.745 ± 0.042	**0.725 ± 0.030**	**0.730 ± 0.035**
Gradient Boosting	0.751 ± 0.034	0.741 ± 0.047	0.721 ± 0.036	0.728 ± 0.038
SVM	0.712 ± 0.046	0.707 ± 0.051	0.698 ± 0.042	0.699 ± 0.046
KNN	0.707 ± 0.037	0.685 ± 0.052	0.662 ± 0.038	0.667 ± 0.040
MLP	0.710 ± 0.027	0.694 ± 0.032	0.675 ± 0.028	0.679 ± 0.026
DNN	0.729 ± 0.055	0.726 ± 0.042	0.706 ± 0.053	0.710 ± 0.051

Conforme a Tabela 5.2, é evidente a desproporção de amostras entre as classes: casos leves são quase metade dos registros aproximadamente (48%), ao passo que moderadas e severas dividem o restante de modo semelhante (26% cada). Essa assimetria persiste em todos os *k-folds* e reflete a distribuição original da base de dados.

Tabela 5.2: Distribuição das amostras nos conjuntos de treino e validação para cada classe nos *k-folds* (Experimento 1).

Fold	Conjunto	Leve	Moderada	Grave	Total
1 ao 4	Treino	319 (48.11%)	174 (26.24%)	170 (25.64%)	663
	Validação	80 (48.19%)	43 (25.90%)	43 (25.90%)	166
5	Treino	320 (48.19%)	174 (26.20%)	170 (25.60%)	664
	Validação	79 (47.88%)	43 (26.06%)	43 (26.06%)	165

O DNN eleva a acurácia de treino até 0.72 para a melhor época e mantém a validação sempre ligeiramente superior (0.70), estabilizando sem sinais de *overfitting* (Tabela 5.1 e 5.3). Já o MLP atinge 0.71 em treino, mas a validação atinge 0.72 após a época 10, (Figura 5.1), podendo indicar sobreajuste; portanto, o DNN generaliza melhor.



(a) MLP (b) DNN

Figura 5.1: Curva de aprendizado das redes neurais construída a partir do conjunto de treino e validação no Experimento 1.

Resultados do Teste

Os resultados reportados no conjunto de teste foram obtidos com os melhores hiperparâmetros definidos na etapa de otimização (Tabela 5.3). Foi possível observar que o *Random Forest* utilizou um número menor de árvores e profundidade mais restrita em relação a experimentos posteriores, enquanto o DNN manteve-se estável sem alterações relevantes.

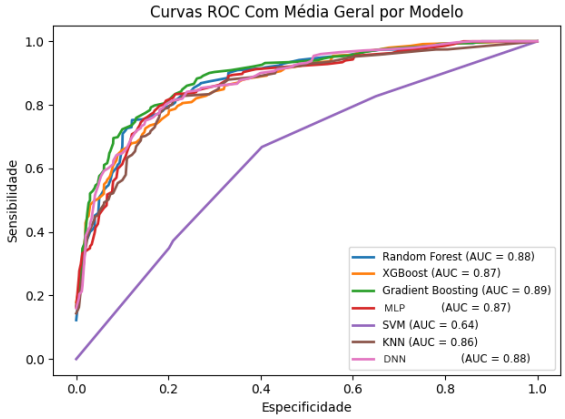


Figura 5.2: Curva ROC do Experimento 1.

Tabela 5.3: Melhores hiperparâmetros selecionados para o Experimento 1.

Algoritmo	**Hiperparâmetro**	**Valor**
Random Forest	Número de árvores	100
	Profundidade máxima	10
XGBoost	Número de árvores	100
	Profundidade máxima	5
	Taxa de aprendizado	0.1
Gradient Boosting	Número de árvores	200
	Taxa de aprendizado	0.05
	Profundidade máxima	4
	Subamostragem	0.6
SVM	Kernel	rbf
	Penalidade (C)	30
	Gamma	0.03
KNN	Número de vizinhos	11
	Ponderação	*distance*
	Distância (p)	2
	Tamanho da folha	15

A Figura 5.3 mostra o *F1-score* dos algoritmos treinados a partir das melhores configurações de hiperparâmetros, permitindo comparar seu desempenho no conjunto de teste, como mostrado na Seção Metodologia 4, que contém 207 amostras (101 leves, 55 modera-

das e 51 graves), correspondendo a 20% do total, preservando a proporção original entre as classes da base de dados.

O *Gradient Boosting* apresentou resultados superiores aos observados na etapa de treinamento em todas as métricas, enquanto o XGBoost mostrou desempenho inferior ao reportado no treinamento. Isso demonstra que ele conseguiu aprender bem apenas um conjunto limitado de amostras, não lidando bem com outras diferentes, como as do conjunto de teste. Em contrapartida, o KNN obteve os piores resultados em todas as métricas, tanto na etapa de treinamento quanto no teste. A DNN apresentou desempenho competitivo no conjunto de teste: obteve o segundo melhor resultado segundo o *F1-score* e superou o desempenho observado no treinamento.

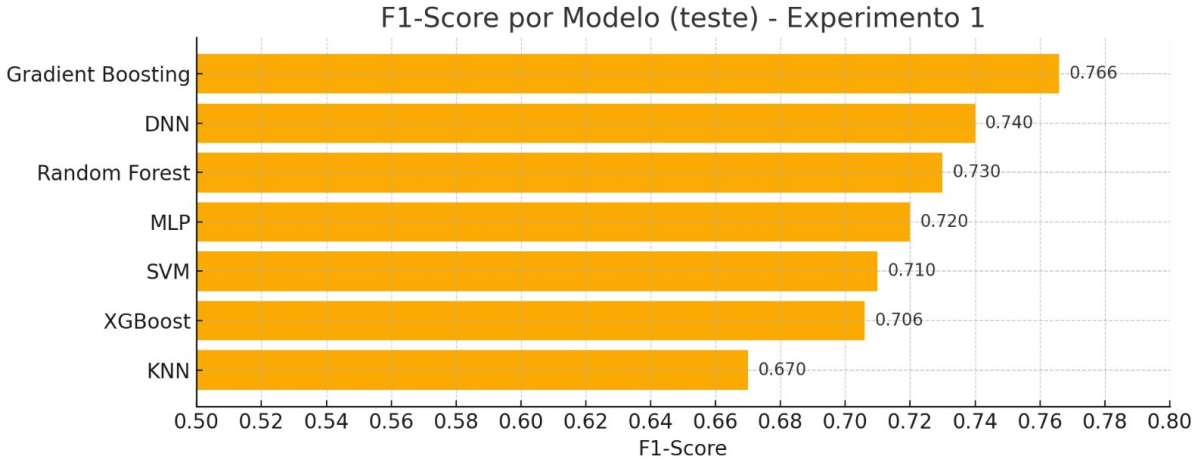
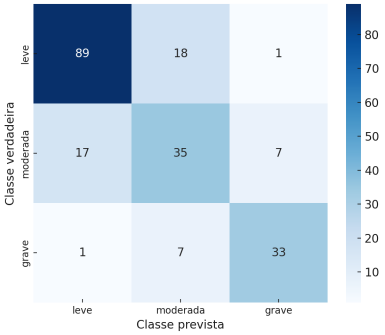
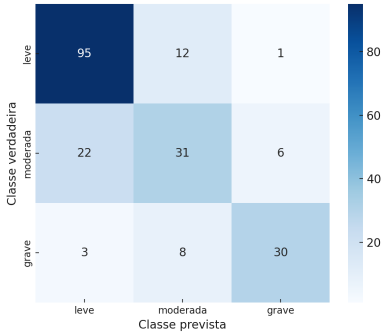


Figura 5.3: Resultados a partir do conjunto de teste obtidos para o Experimento 1.

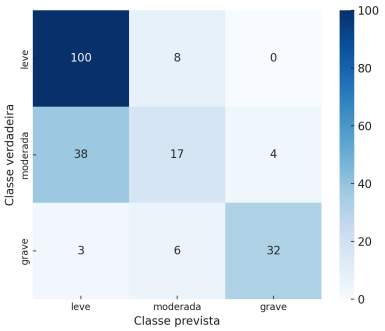
Nas matrizes de confusão observada na Figura 5.4, percebe-se que as classes leve e severa tiveram alta taxa de acerto em todos os algoritmos, especialmente a grave. Já a classe moderada concentrou a maioria dos erros, sendo frequentemente confundida com a leve.



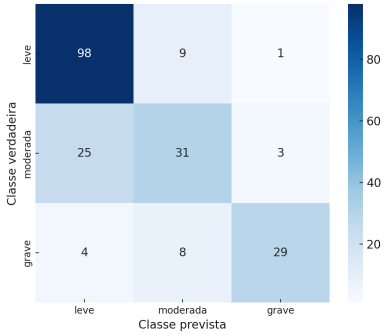
(a) DNN



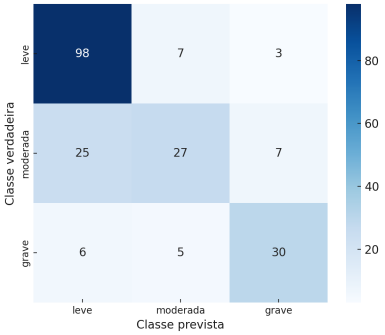
(b) MLP



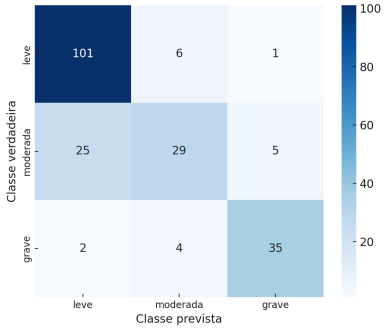
(c) KNN



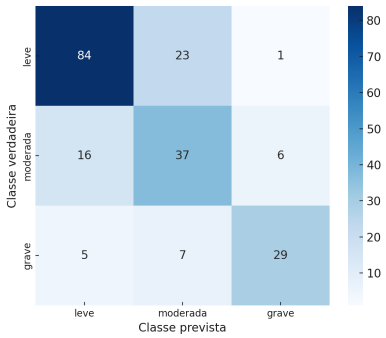
(d) *Random Forest*



(e) XGBoost



(f) *Gradient Boosting*



(g) SVM

Figura 5.4: Matriz de confusão dos algoritmos para o Experimento 1.

No gráfico da Figura 5.5, foi possível observar que houve uma constante dificuldade dos

algoritmos para prever a classificação moderada, sempre sendo confundida com a leve. É possível observar também que os algoritmos SVM e DNN tiveram uma maior quantidade de erros ao tentar prever a classe leve, classificando-a como moderada. O SVM, XGBoost e o *Random Forest* tiveram a maior quantidade de erros para o caso mais sério de erro, prevendo um caso grave como leve.

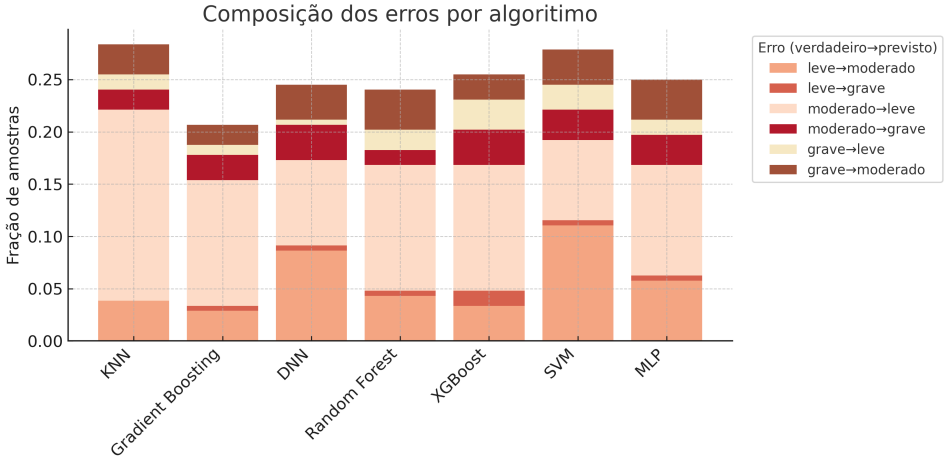


Figura 5.5: Gráfico de composição de erros no conjunto de teste do Experimento 1.

Durante a avaliação no conjunto de teste, a curva ROC, observada no gráfico da Figura 5.2, apontou o *Gradient Boosting* como o de melhor desempenho, seguido por *Random Forest* e pelo algoritmo profundo (DNN), resultando em um *ranking* coerente com as demais métricas deste experimento.

5.3 Experimento 2: Balanceamento de Classes utilizando SMOTE

Resultados da Validação Cruzada

Conforme definido na Seção 4.7, para este segundo experimento, foi realizado um balanceamento das classes no conjunto de treino a cada iteração da validação cruzada, utilizando o SMOTE. Observando a Tabela 5.4, o melhor algoritmo em todas as métricas foi o XGBoost e o pior, em todas as métricas também, foi o DNN por conta do ruído gerado pelo aumento sintético do SMOTE.

Tabela 5.4: Média ± 95% IC das métricas obtidas na etapa de treinamento para cada algoritmo do Experimento 2.

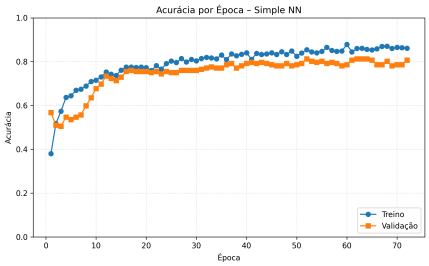
Algoritmo	**Acurácia**	**Precisão**	**Revocação**	***F1-score***
Random Forest	0.752 ± 0.030	0.738 ± 0.042	0.721 ± 0.032	0.727 ± 0.035
XGBoost	**0.761 ± 0.016**	**0.746 ± 0.027**	**0.734 ± 0.021**	**0.738 ± 0.022**
Gradient Boosting	0.738 ± 0.031	0.721 ± 0.042	0.716 ± 0.027	0.717 ± 0.033
SVM	0.680 ± 0.034	0.659 ± 0.034	0.651 ± 0.027	0.653 ± 0.028
KNN	0.684 ± 0.038	0.670 ± 0.046	0.667 ± 0.044	0.667 ± 0.045
MLP	0.695 ± 0.045	0.695 ± 0.056	0.676 ± 0.047	0.680 ± 0.048
DNN	0.635 ± 0.192	0.638 ± 0.268	0.658 ± 0.130	0.627 ± 0.209

A aplicação do SMOTE iguala as frequências no conjunto de treino aproximadamente (33.33% por classe) como mostra a Tabela 5.5, triplicando o número total de amostras aproximadamente (+44.3%). O conjunto de teste não é submetido ao balanceamento, permitindo avaliar a capacidade do algoritmo de generalizar para dados naturalmente desbalanceados.

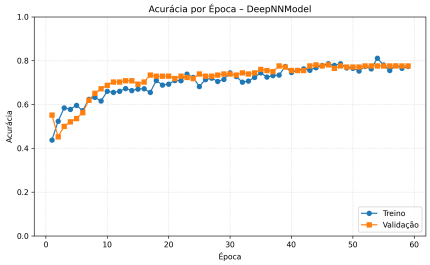
Tabela 5.5: Distribuição das amostras nos conjuntos de treino e validação para cada classe nos k-folds antes e depois do SMOTE (Experimento 2).

Fold	Conjunto	Leve	Moderada	Grave	Total
1 ao 2	Treino (Antes)	319 (48.11%)	174 (26.24%)	170 (25.64%)	663
	Treino (Depois SMOTE)	319 (33.33%)	319 (33.33%)	319 (33.33%)	957
	Teste	80 (48.19%)	43 (25.90%)	43 (25.90%)	166
3 ao 4	Treino (Antes)	319 (48.11%)	173 (26.09%)	171 (25.79%)	663
	Treino (Depois SMOTE)	319 (33.33%)	319 (33.33%)	319 (33.33%)	957
	Teste	80 (48.19%)	44 (26.51%)	42 (25.30%)	166
5	Treino (Antes)	320 (48.19%)	174 (26.20%)	170 (25.60%)	664
	Treino (Depois SMOTE)	320 (33.33%)	320 (33.33%)	320 (33.33%)	960
	Teste	79 (47.88%)	43 (26.06%)	43 (26.06%)	165

A DNN eleva a acurácia de treino de 0.45 para 0.77, como ilustram os gráficos da Figura 5.6, e mantém a validação muito próxima (0.79), sem sinais de *overfitting*, assim como no experimento anterior 5.3. Já a MLP atinge 0.85 em treino, enquanto a validação estabiliza perto de 0.80, com um hiato de 0.05 após a época 25, indicando leve sobreajuste.



(a) MLP



(b) DNN

Figura 5.6: Curva de aprendizado das redes neurais construída a partir do conjunto de treinamento no Experimento 2.

Resultados do Teste

Os resultados reportados no conjunto de teste foram obtidos com os melhores hiperparâmetros definidos na etapa de otimização (Tabela 5.6). Neste caso, houve um ajuste no Random Forest, que aumentou o número de árvores e profundidade máxima, e também variações nos parâmetros de XGBoost e Gradient Boosting, principalmente na taxa de aprendizado. O DNN manteve-se inalterado.

Tabela 5.6: Melhores hiperparâmetros selecionados para o Experimento 2.

Algoritmo	**Hiperparâmetro**	**Valor**
Random Forest	Número de árvores	150
	Profundidade máxima	15
XGBoost	Número de árvores	100
	Profundidade máxima	5
	Taxa de aprendizado	0.2
Gradient Boosting	Número de árvores	300
	Taxa de aprendizado	0.1
	Profundidade máxima	4
	Subamostragem	0.6
SVM	Kernel	rbf
	Penalidade (C)	10
	Gamma	0.3
	Balanceamento de amostras entre as classes	*balanced*

Continuação da Tabela 5.6 da página anterior

Algoritmo	**Hiperparâmetro**	**Valor**
KNN	Número de vizinhos	3
	Ponderação	*distance*
	Distância (p)	1
	Tamanho da folha	15

Em contraste, é ilustrado na Figura 5.7 que o *Random Forest* apresentou resultados superiores aos observados na etapa de treinamento em todas as métricas, enquanto o XGBoost mostrou desempenho inferior ao reportado no treinamento. E o KNN novamente foi o algoritmo com o pior desempenho em todas as métricas no momento do teste.

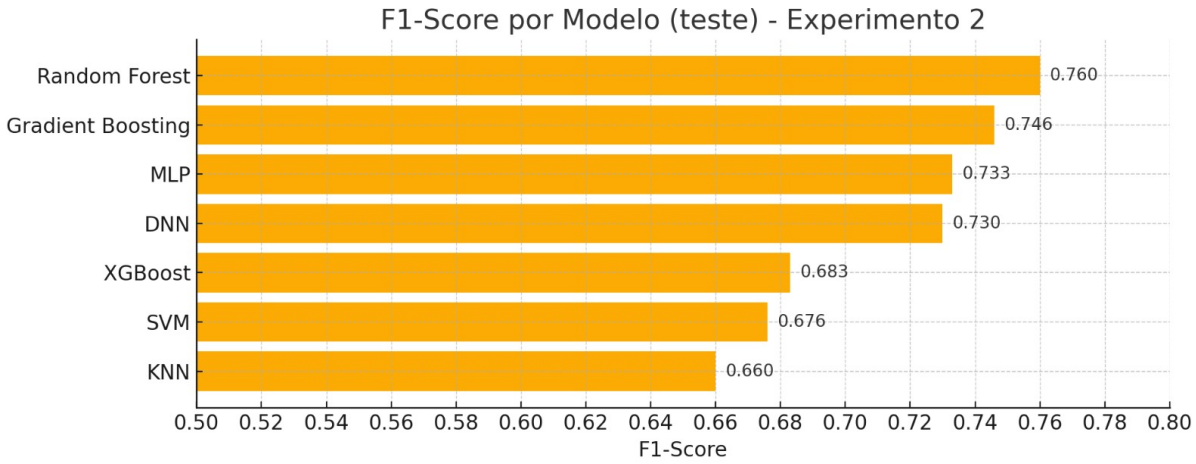
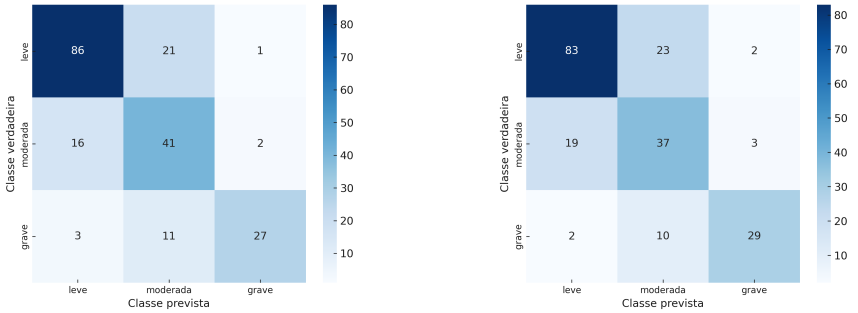


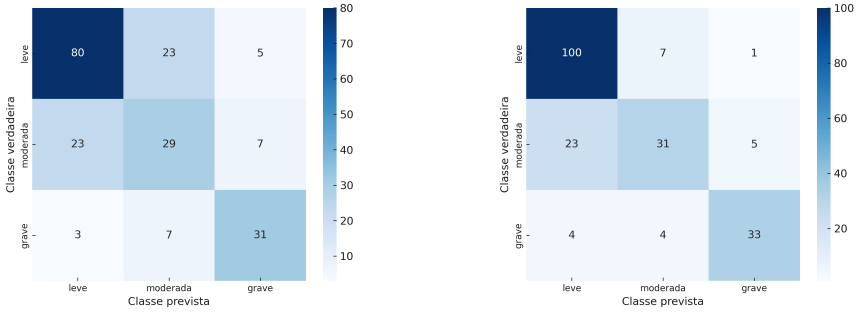
Figura 5.7: Métricas de resultado dos testes realizados para o Experimento 2.

Nas matrizes de confusão da Figura 5.8, é possível observar que as classes leve e severa tiveram uma alta taxa de acerto em todos os algoritmos e a classe moderada concentrou a maioria dos erros, sendo confundida com leve.



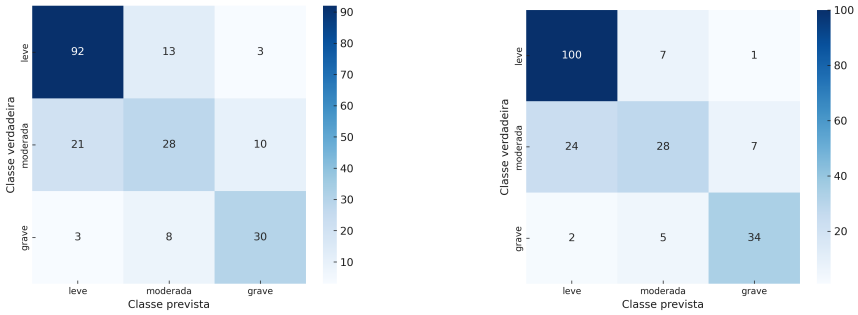
(a) DNN

(b) MLP



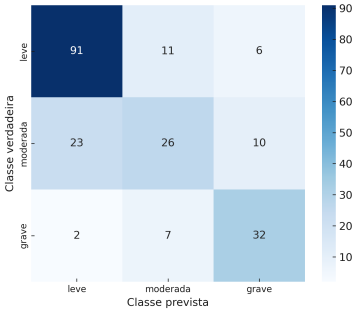
(c) KNN

(d) *Random Forest*



(e) XGBoost

(f) *Gradient Boosting*



(g) SVM

Figura 5.8: Matriz de confusão dos algoritmos com balanceamento de dados para o Experimento 2.

Podemos observar no Gráfico da figura 5.9 a dificuldade que os algoritmos apresentam para prever a classe moderada, confundindo-a, principalmente, com a classe leve. É

possível observar também uma crescente na confusão na previsão da classe leve, sendo confundida com a moderada. O KNN apresentou uma maior taxa de erro, quando comparado ao Experimento 1 (Seção 5.2).

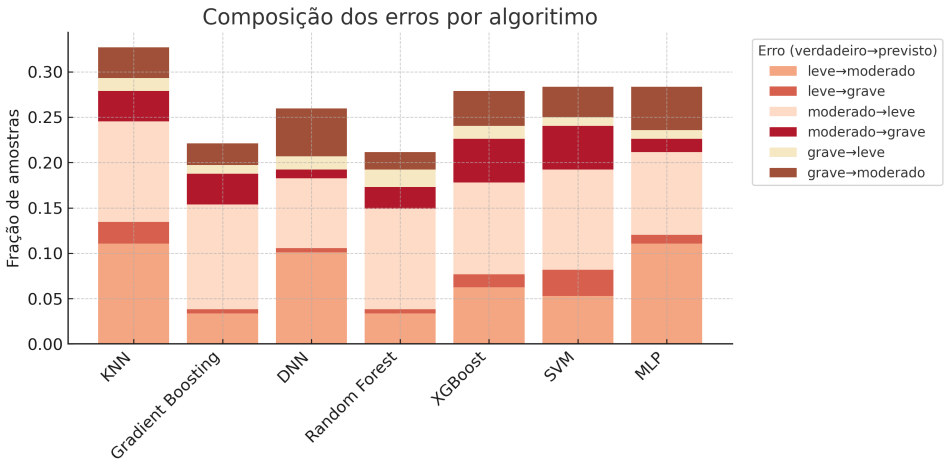


Figura 5.9: Gráfico de composição de erros no conjunto de teste do Experimento 2.

Após a aplicação do SMOTE, as curvas ROC do Gráfico 5.10 indicam que *Random Forest* e *Gradient Boosting* mantêm desempenho elevado, ambos com AUC = 0.89, superando ligeiramente os valores do experimento original. O XGBoost sofreu pequena queda para AUC = 0.86, enquanto a DNN recuou para 0.87 e a MLP para 0.85. O KNN exibiu redução mais acentuada para 0.81, e o SVM permaneceu com desempenho modesto (0.63). Em comparação com a base original, o balanceamento sintético favoreceu os *ensembles* de árvores, mas não beneficiou, e até prejudicou, algoritmos mais sensíveis à distribuição espacial dos dados, possivelmente em razão do ruído introduzido pelas instâncias geradas.

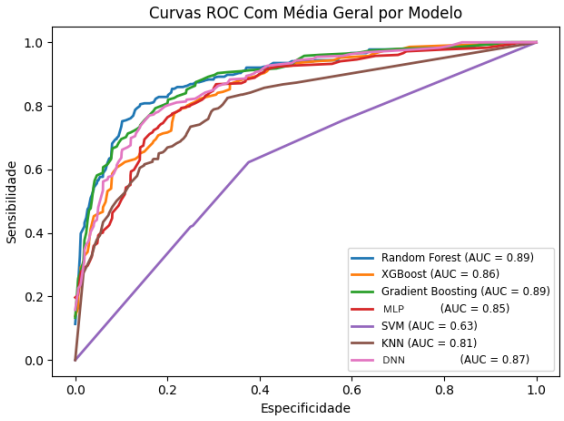


Figura 5.10: Curva ROC do Experimento 2.

5.4 Experimento 3: Aumento de Dados utilizando CTGAN

Resultados da Validação Cruzada

Conforme definido na Seção 4.7, este experimento foi conduzido utilizando o conjunto de treino aumentado via CTGAN, sem aplicação de técnicas explícitas de balanceamento. Com base nos resultados da validação cruzada de cinco *folds* (Tabela 5.7), os algoritmos *Random Forest* e *Gradient Boosting* apresentaram os melhores desempenhos em termos de *F1-score*, com valores muito próximos, mas o *Random Forest* apresentou o melhor desempenho, ainda que com uma margem pequena, quando observado o intervalo de confiança. Ademais, *Random Forest* destacou-se pela maior acurácia e precisão, enquanto o *Gradient Boosting* obteve a melhor revocação.

Por outro lado, os algoritmos com os piores desempenhos na etapa de treinamento foram o KNN, com a menor *F1-score* e revocação, e o MLP, com a menor precisão.

Tabela 5.7: Média ± 95% IC das métricas obtidas na etapa de treinamento para cada algoritmo do Experimento 3.

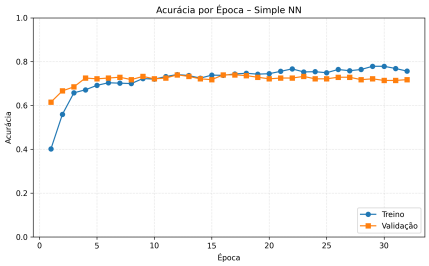
Algoritmo	**Acurácia**	**Precisão**	**Revocação**	***F1-score***
Random Forest	**0.747 ± 0.036**	**0.744 ± 0.056**	0.705 ± 0.034	**0.717 ± 0.040**
XGBoost	0.738 ± 0.033	0.728 ± 0.052	0.699 ± 0.030	0.709 ± 0.036
Gradient Boosting	0.744 ± 0.044	0.738 ± 0.059	**0.707 ± 0.040**	0.717 ± 0.044
SVM	0.709 ± 0.036	0.721 ± 0.035	0.688 ± 0.029	0.697 ± 0.029
KNN	0.705 ± 0.046	0.695 ± 0.063	0.654 ± 0.047	0.664 ± 0.051
MLP	0.710 ± 0.022	0.691 ± 0.023	0.674 ± 0.029	0.677 ± 0.027
DNN	0.725 ± 0.029	0.716 ± 0.039	0.691 ± 0.026	0.697 ± 0.029

A estratégia de aumento puro com CTGAN (Tabela 5.8) expande a base de dados de treino em aproximadamente 105%, mas apenas atenua o desequilíbrio. Embora a classe leve ainda predomine, há ganho relativo nas classes minoritárias, sobretudo na severa (até 30%). Isso evidencia que o gerador aprendeu a maioria das características, mas não garante proporcionalidade.

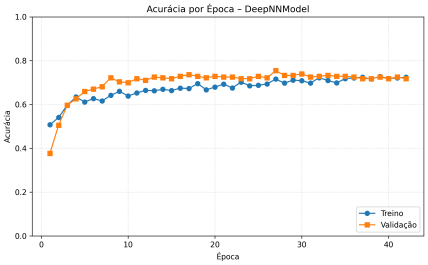
Tabela 5.8: Distribuição das amostras nos conjuntos de treino e validação para cada classe nos k-folds antes e depois do CTGAN (Experimento 3).

Fold	Conjunto	Leve	Moderada	Grave	Total
1	Treino (Antes)	319 (48.11%)	174 (26.24%)	170 (25.64%)	663
	Treino (Depois CTGAN)	621 (45.56%)	339 (24.87%)	403 (29.57%)	1 363
	Teste	80 (48.19%)	43 (25.90%)	43 (25.90%)	166
2	Treino (Antes)	319 (48.11%)	174 (26.24%)	170 (25.64%)	663
	Treino (Depois CTGAN)	586 (42.99%)	359 (26.34%)	418 (30.67%)	1 363
	Teste	80 (48.19%)	43 (25.90%)	43 (25.90%)	166
3	Treino (Antes)	319 (48.11%)	173 (26.09%)	171 (25.79%)	663
	Treino (Depois CTGAN)	575 (42.19%)	393 (28.83%)	395 (28.98%)	1 363
	Teste	80 (48.19%)	44 (26.51%)	42 (25.30%)	166
4	Treino (Antes)	319 (48.11%)	173 (26.09%)	171 (25.79%)	663
	Treino (Depois CTGAN)	624 (45.78%)	350 (25.68%)	389 (28.54%)	1 363
	Teste	80 (48.19%)	44 (26.51%)	42 (25.30%)	166
5	Treino (Antes)	320 (48.19%)	174 (26.20%)	170 (25.60%)	664
	Treino (Depois CTGAN)	611 (44.79%)	375 (27.49%)	378 (27.71%)	1 364
	Teste	79 (47.88%)	43 (26.06%)	43 (26.06%)	165

Tal como no conjunto anterior, a DNN eleva a acurácia de treino de 0.45 para 0.71 e mantém a validação ligeiramente superior (0.74), sinalizando boa generalização sem *overfitting*; a trajetória quase sobreposta reforça a consistência observada anteriormente. Já a MLP volta a aprender mais rápido, encerrando com 0.78 em treino e 0.72 em validação, perda absoluta de 0.06 a partir da época 20, um sobreajuste leve, mas um pouco menor que o visto na rodada anterior (0.05 após a época 25). Portanto, as tendências se mantêm: a DNN oferece desempenho equilibrado e estável, enquanto a MLP, embora alcance maior acurácia de treino, seria interessante ter uma regularização e um *early stopping* mais severa para conter a perda de generalização, como podemos observar na Figura 5.11.



(a) MLP



(b) DNN

Figura 5.11: Curva de aprendizado das redes neurais construída a partir do conjunto de treinamento (Experimento 3.

Resultados do Teste

Os resultados reportados no conjunto de teste foram obtidos com os melhores hiperparâmetros definidos na etapa de otimização (Tabela 5.9). Nota-se que o Random Forest e o XGBoost apresentaram valores diferentes para profundidade e taxa de aprendizado em comparação aos outros experimentos. Já o Gradient Boosting reduziu a taxa de aprendizado, refletindo uma busca por maior estabilidade. O DNN permaneceu padrão.

Tabela 5.9: Melhores hiperparâmetros selecionados para o Experimento 3.

Algoritmo	**Hiperparâmetro**	**Valor**
Random Forest	Número de árvores	150
	Profundidade máxima	15
XGBoost	Número de árvores	50
	Profundidade máxima	3
	Taxa de aprendizado	0.2
Gradient Boosting	Número de árvores	200
	Taxa de aprendizado	0.05
	Profundidade máxima	4
	Subamostragem	0.8
SVM	Kernel	rbf
	Penalidade (C)	10
	Gamma	0.03
	Balanceamento de amostras entre as classes	*balanced*

Continuação da Tabela 5.9 da página anterior

Algoritmo	**Hiperparâmetro**	**Valor**
KNN	Número de vizinhos	11
	Ponderação	*distance*
	Distância (p)	1
	Tamanho da folha	15

No conjunto de teste ilustrado na Figura 5.12, os algoritmos *Gradient Boosting* e XGBoost alcançaram as maiores pontuações de *F1-score* e acurácia, indicando maior generalização. O *Random Forest* obteve a maior precisão, enquanto o *Gradient Boosting* teve o melhor valor de revocação. Os demais algoritmos, especialmente SVM, MLP e DNN, mantiveram desempenhos estáveis, porém inferiores aos métodos de ensemble.

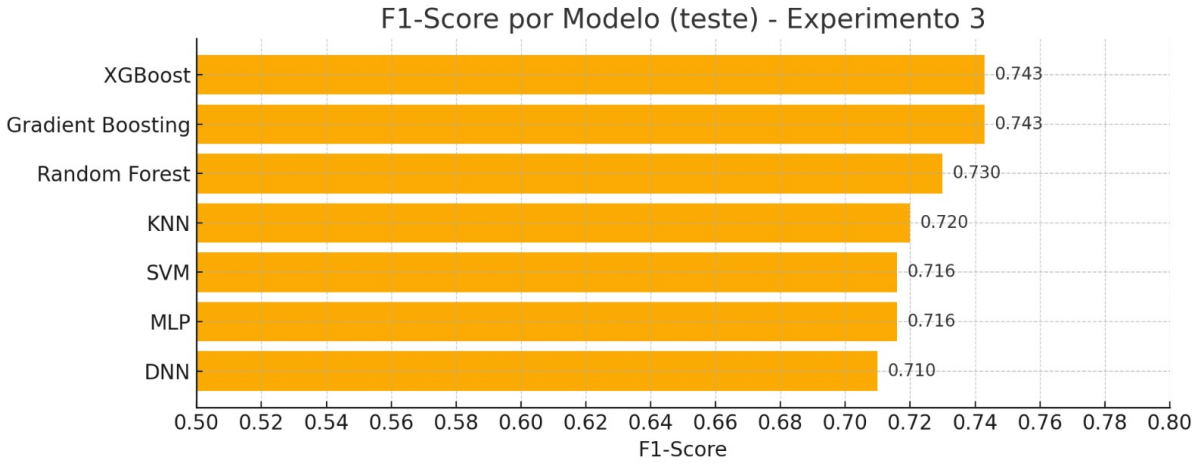
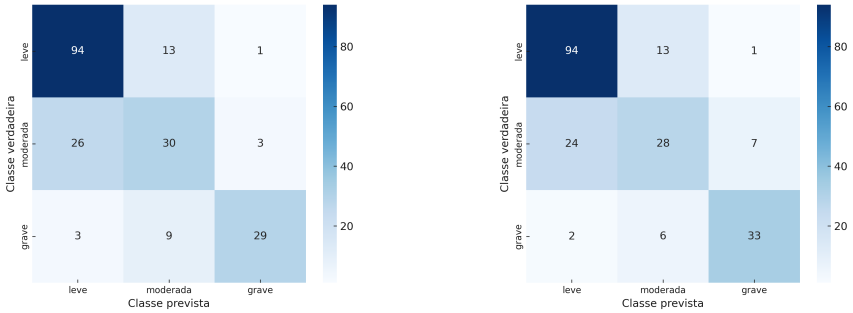


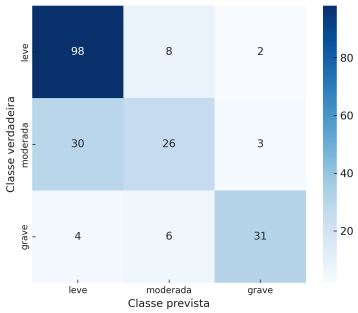
Figura 5.12: Métricas de resultado dos testes realizados para o Experimento 3.

Nas matrizes de confusão observadas na Figura 5.13, percebe-se que as classes leve e severa mantiveram alta taxa de acerto em todos os algoritmos. Já a classe moderada apresentou baixa acurácia, sendo frequentemente confundida com leve. Em geral, houve muitos erros na classificação de moderada, chutando para leve, mas o SVM destacou-se por errar mais as classes leve e severa do que os demais algoritmos neste experimento.

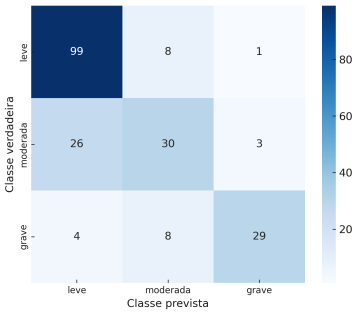


(a) DNN

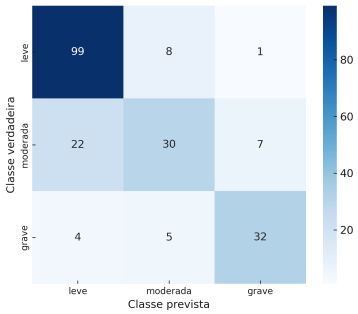
(b) MLP



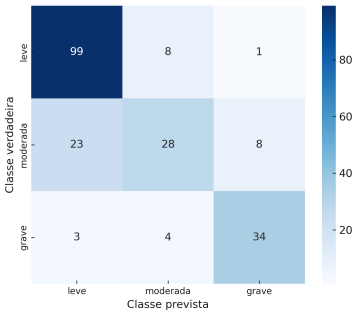
(c) KNN



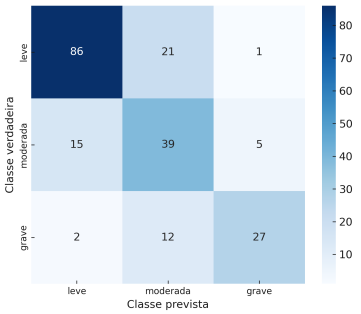
(d) *Random Forest*



(e) XGBoost



(f) *Gradient Boosting*



(g) SVM

Figura 5.13: Matriz de confusão dos algoritmos com aumento de dados para o Experimento 3.

Sobre a composição de erros deste experimento, a dificuldade de prever a classe moderada permanece, sendo confundida, principalmente, com a classe leve. Observamos

também um aumento na quantidade de erros na classificação da classe grave, sendo confundida como moderada. Observamos também que houve uma proximidade na quantidade de erros entre os algoritmos e que ocorreu uma pequena quantidade de erro ao prever a classe grave como leve, como é possível observar no gráfico da figura 5.14.

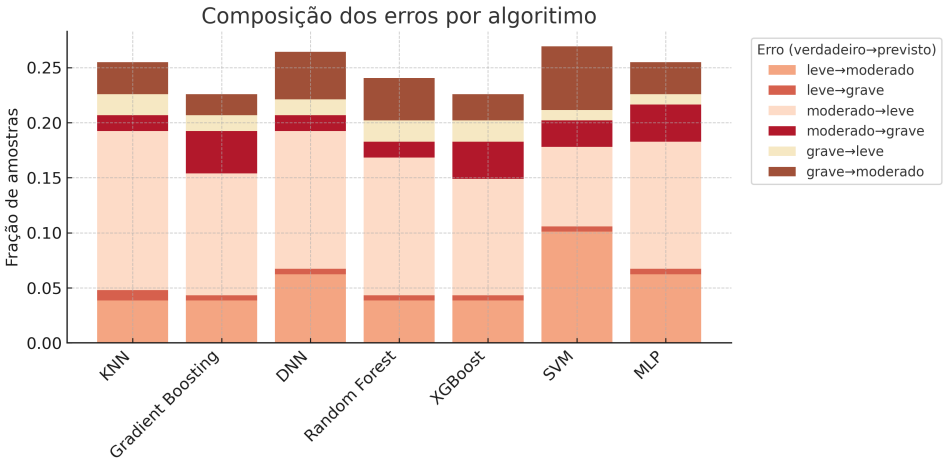


Figura 5.14: Gráfico de composição de erros no conjunto de teste do Experimento 3.

Com a ampliação da base via CTGAN, o *Gradient Boosting* manteve a melhor discriminação (AUC = 0.89) como é ilustrado na Figura 5.15. *Random Forest*, XGBoost e a DNN apresentaram desempenho equivalente (0.88), sendo que o XGBoost recuperou a leve queda registrada após o SMOTE. A MLP permaneceu estável em 0.87, enquanto o KNN retornou a 0.86, revertendo a perda observada no experimento anterior. O SVM elevou-se discretamente para 0.65, mas continua com a menor capacidade discriminativa.

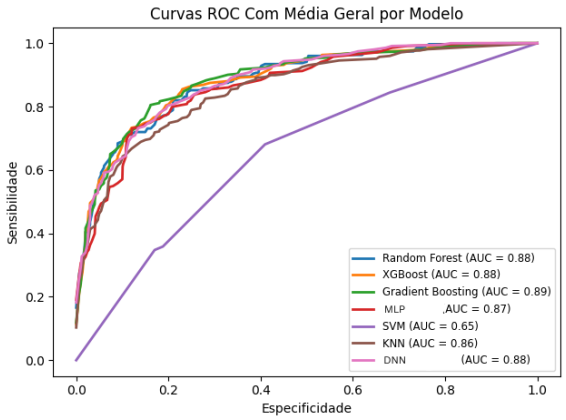


Figura 5.15: Curva ROC do Experimento 3.

5.5 Experimento 4: Aumento de Dados utilizando CTGAN e Balanceamento de amostras entre as Classes utilizando SMOTE

Resultados da Validação Cruzada

Conforme definido na Seção 4.7, neste experimento foi feito tanto um aumento quanto o balanceamento dos dados em cada iteração da validação cruzada, utilizando o CTGAN e o SMOTE, respectivamente. Na Tabela 5.10, tivemos o *Random Forest*, XGBoost e o *Gradient Boosting* com um *F1-score* muito próximo um do outro, tendo o XGBoost com o menor intervalo de confiança entre os três e se sobressaindo, não apenas no *F1-score*, mas na acurácia e na revocação também.

Novamente, aqui podemos observar o KNN apresentando o pior resultado em todas as métricas durante o treinamento.

Tabela 5.10: Média ± 95% IC das métricas obtidas na etapa de treinamento para cada algoritmo do Experimento 4.

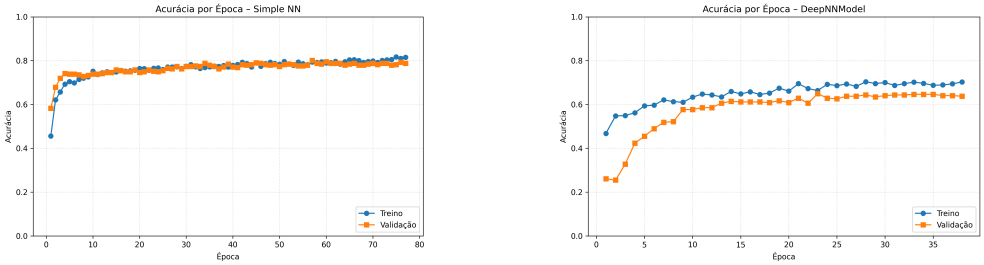
Algoritmo	**Acurácia**	**Precisão**	**Revocação**	***F1-score***
Random Forest	0.742 ± 0.023	**0.731 ± 0.032**	0.711 ± 0.020	0.719 ± 0.024
XGBoost	**0.743 ± 0.020**	0.725 ± 0.026	**0.716 ± 0.019**	**0.719 ± 0.021**
Gradient Boosting	0.741 ± 0.020	0.726 ± 0.024	0.715 ± 0.021	0.719 ± 0.022
SVM	0.691 ± 0.036	0.672 ± 0.038	0.668 ± 0.040	0.668 ± 0.039
KNN	0.659 ± 0.030	0.645 ± 0.034	0.640 ± 0.030	0.641 ± 0.030
MLP	0.702 ± 0.037	0.690 ± 0.035	0.674 ± 0.035	0.680 ± 0.035
DNN	0.702 ± 0.026	0.710 ± 0.027	0.690 ± 0.012	0.691 ± 0.018

Unindo CTGAN e SMOTE (Tabela 5.11), obtém-se a maior base de dados e equilíbrio perfeito de 33.33% por classe nos treinos. Tal combinação busca mitigar viéses de ambas as técnicas: o CTGAN acrescenta variabilidade sintética, enquanto o SMOTE garante equilíbrio numérico.

Tabela 5.11: Distribuição das amostras nos conjuntos de treino e validação para cada classe nos k-folds antes e depois do CTGAN e do SMOTE (Experimento 4).

Fold	Conjunto	Leve	Moderada	Grave	Total
1	Treino (Antes)	319 (48.11%)	174 (26.24%)	170 (25.64%)	663
	Treino (Depois CTGAN)	621 (45.56%)	339 (24.87%)	403 (29.57%)	1 363
	Treino (Depois SMOTE)	621 (33.33%)	621 (33.33%)	621 (33.33%)	1 863
	Teste	80 (48.19%)	43 (25.90%)	43 (25.90%)	166
2	Treino (Antes)	319 (48.11%)	174 (26.24%)	170 (25.64%)	663
	Treino (Depois CTGAN)	586 (42.99%)	359 (26.34%)	418 (30.67%)	1 363
	Treino (Depois SMOTE)	586 (33.33%)	586 (33.33%)	586 (33.33%)	1 758
	Teste	80 (48.19%)	43 (25.90%)	43 (25.90%)	166
3	Treino (Antes)	319 (48.11%)	173 (26.09%)	171 (25.79%)	663
	Treino (Depois CTGAN)	575 (42.19%)	393 (28.83%)	395 (28.98%)	1 363
	Treino (Depois SMOTE)	575 (33.33%)	575 (33.33%)	575 (33.33%)	1 725
	Teste	80 (48.19%)	44 (26.51%)	42 (25.30%)	166
4	Treino (Antes)	319 (48.11%)	173 (26.09%)	171 (25.79%)	663
	Treino (Depois CTGAN)	624 (45.78%)	350 (25.68%)	389 (28.54%)	1 363
	Treino (Depois SMOTE)	624 (33.33%)	624 (33.33%)	624 (33.33%)	1 872
	Teste	80 (48.19%)	44 (26.51%)	42 (25.30%)	166
5	Treino (Antes)	320 (48.19%)	174 (26.20%)	170 (25.60%)	664
	Treino (Depois CTGAN)	611 (44.79%)	375 (27.49%)	378 (27.71%)	1 364
	Treino (Depois SMOTE)	611 (33.33%)	611 (33.33%)	611 (33.33%)	1 833
	Teste	79 (47.88%)	43 (26.06%)	43 (26.06%)	165

Neste experimento, como mostrado na Figura 5.16, o comportamento se inverte em relação ao observado anteriormente, a MLP atinge 0.82 em treino e 0.79 em validação, mantendo um hiato máximo de apenas 0.03 após a época 50 e, portanto, exibe a melhor generalização entre todas as execuções analisadas (antes a perda era 0.06). Já a DNN cresce até 0.67 em treino, mas a validação estabiliza em torno de 0.63, gerando distância de 0.04 e indicando sobreajuste moderado (desempenho inferior à sessão passada, quando sua validação chegara a 0.74). Assim, enquanto a MLP mostra evolução consistente, a DNN revela um alto desempenho de predição, como mostram os gráficos da Figura 5.16.



(a) MLP (b) DNN

Figura 5.16: Curva de aprendizado das redes neurais construída a partir do conjunto de treinamento (Experimento 4).

Resultados do Teste

Os resultados reportados no conjunto de teste foram obtidos com os melhores hiperparâmetros definidos na etapa de otimização (Tabela 5.12). O Random Forest retornou a um número de árvores menor, enquanto o XGBoost ajustou a profundidade máxima. O Gradient Boosting aumentou consideravelmente o número de árvores e apresentou taxa de aprendizado maior em relação ao experimento anterior. O DNN permaneceu estável.

Tabela 5.12: Melhores hiperparâmetros selecionados para o Experimento 4.

Algoritmo	**Hiperparâmetro**	**Valor**
Random Forest	Número de árvores	100
	Profundidade máxima	15
XGBoost	Número de árvores	100
	Profundidade máxima	7
	Taxa de aprendizado	0.2
Gradient Boosting	Número de árvores	300
	Taxa de aprendizado	0.1
	Profundidade máxima	4
	Subamostragem	0.8
SVM	Kernel	rbf
	Penalidade (C)	30
	Gamma	0.3
	Balanceamento de amostras entre as classes	*balanced*

Continuação da Tabela 5.12 da página anterior

Algoritmo	**Hiperparâmetro**	**Valor**
KNN	Número de vizinhos	3
	Ponderação	*distance*
	Distância (p)	1
	Tamanho da folha	15

Nos testes, o *Gradient Boosting* teve as melhores métricas em relação a todos os outros algoritmos, como ilustra a Figura 5.17. As amostras sintéticas geradas pelo CTGAN com o balanceamento inteligente do SMOTE, apesar de gerarem ruído, contribuem de maneira muito significativa para o treinamento do *Gradient Boosting*. Os algoritmos como KNN e SVM acabam se confundindo pela forma como os dados sintéticos são gerados e por conta da sobreposição das características das amostras. O algoritmo DNN não consegue performar tão bem por conta do ruído mencionado anteriormente.

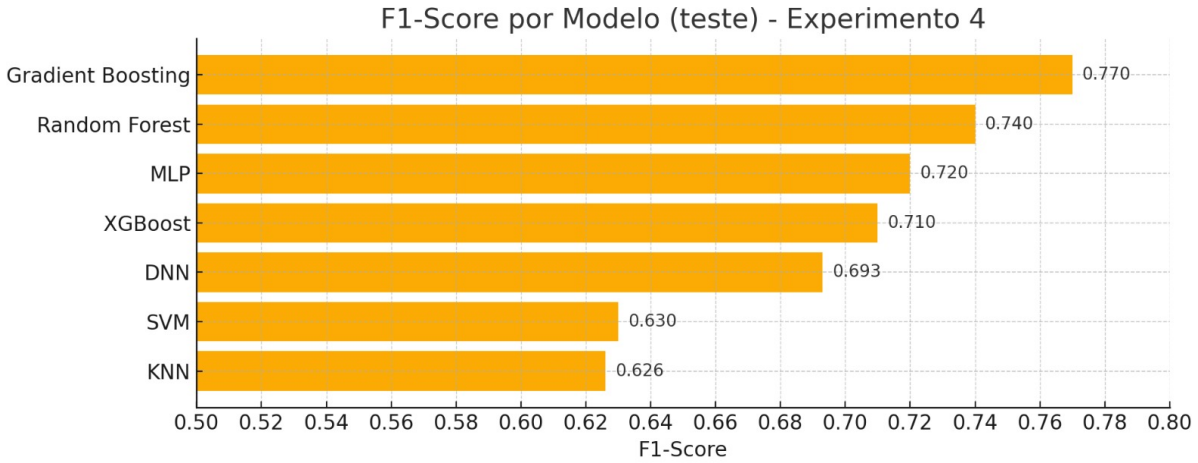
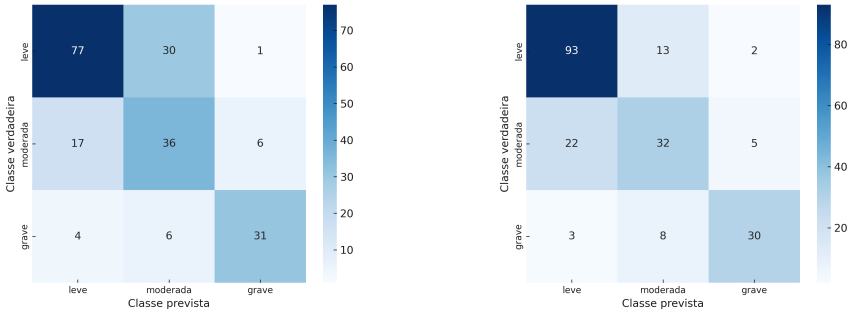


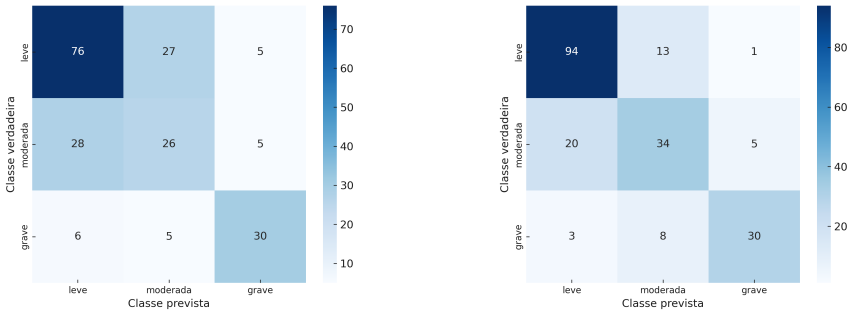
Figura 5.17: Métricas de resultado dos testes realizados para o Experimento 4.

Em 5.18 conseguimos observar o mesmo problema observado nos experimentos anteriores, onde a maior parte dos erros ocorre na previsão da classe leve, principalmente nos algoritmos KNN e SVM. No DNN, podemos observar que ocorreu uma maior dificuldade com a classe leve do que com a classe moderada.



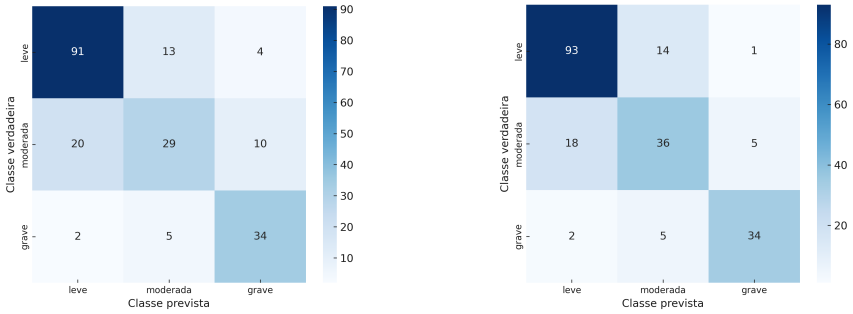
(a) DNN

(b) MLP



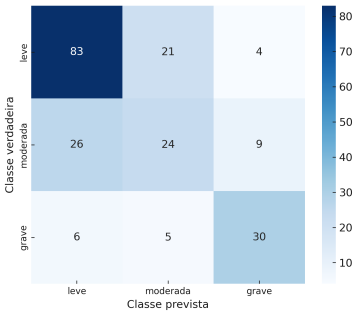
(c) KNN

(d) *Random Forest*



(e) XGBoost

(f) *Gradient Boosting*



(g) SVM

Figura 5.18: Matriz de confusão dos algoritmos com balanceamento e aumento de dados para o Experimento 4.

Sobre a composição de erros deste experimento, podemos enxergar na Figura 5.19 a mesma dificuldade apresentada em experimentos anteriores: classificar a classe moderada

como leve. Contudo, de maneira geral, a taxa de erro foi pequena e próxima entre todos os algoritmos.

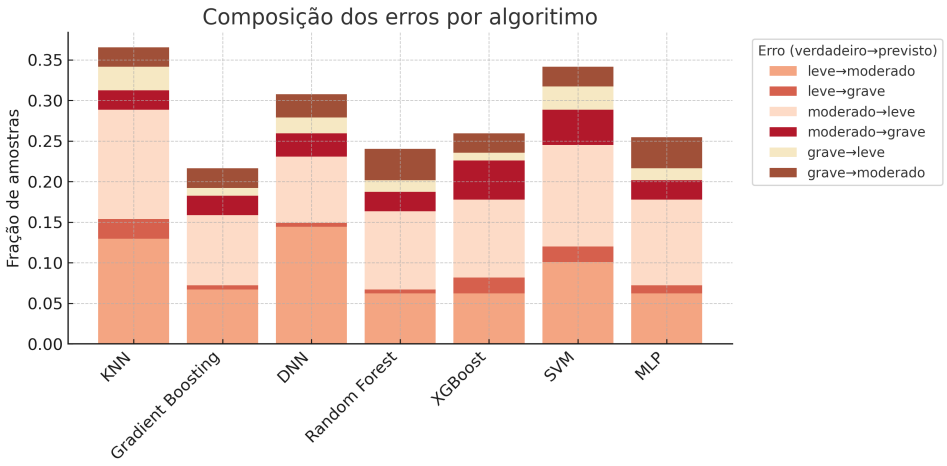


Figura 5.19: Gráfico de composição de erros no conjunto de teste do Experimento 4.

Utilizando o CTGAN e o SMOTE, podemos ver que os algoritmos *Random Forest*, XGBoost e *Gradient Boosting* convergem para desempenho idêntico, todos com AUC = 0.88. O resultado indica um teto de ganho para métodos baseados em árvores depois das fases anteriores: em relação ao cenário original e às versões com balanceamento isolado, não há avanço adicional significativo. A MLP mantém AUC = 0.87, enquanto a DNN recua levemente para 0.86, sugerindo que a mescla de instâncias sintéticas pode introduzir ruído sutil às redes neurais. O KNN foi o mais afetado, caindo para 0.78, valor inferior ao obtido com SMOTE (0.81) e CTGAN (0.86). Isso se dá provavelmente porque a duplicação e a geração de amostras distorcem as relações de vizinhança. Por fim, o SVM tem um aumento de apenas a 0.60, continuando como o algoritmo de menor capacidade discriminativa.

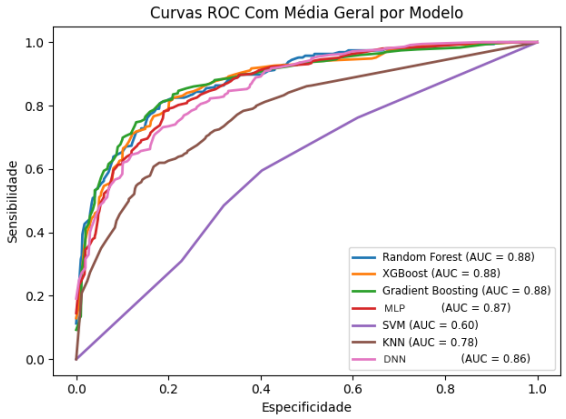


Figura 5.20: Curva ROC do Experimento 4.

5.6 Experimento 5: Balanceamento de amostras entre as Classes utilizando CTGAN

Resultados da Validação Cruzada

Conforme definido na Seção 4.7, neste quinto experimento foi realizado com a base de dados balanceada utilizando CTGAN. É possível observar nos resultados (Tabela 5.13) que o melhor algoritmo foi o *Gradient Boosting* em todas as métricas consideradas. Em contrapartida, os algoritmos MLP e DNN apresentaram os piores resultados, com o MLP registrando a menor acurácia, precisão, revocação e *F1-score*.

Tabela 5.13: Média ± 95% IC das métricas obtidas na etapa de treinamento para cada algoritmo do Experimento 5.

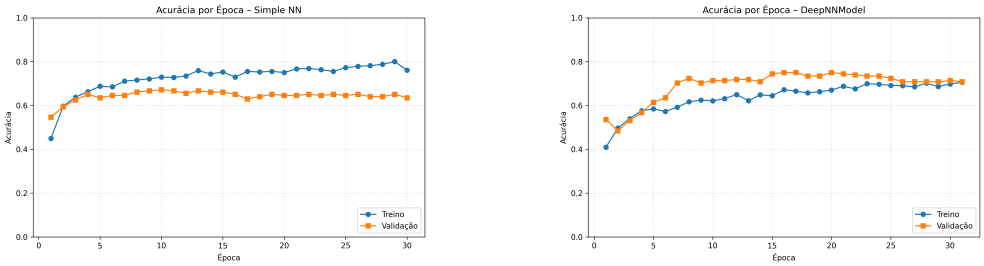
Algoritmo	**Acurácia**	**Precisão**	**Revocação**	***F1-score***
Random Forest	0.730 ± 0.031	0.708 ± 0.050	0.701 ± 0.035	0.703 ± 0.041
XGBoost	0.742 ± 0.017	0.722 ± 0.025	0.714 ± 0.011	0.716 ± 0.014
Gradient Boosting	**0.750 ± 0.027**	**0.732 ± 0.030**	**0.728 ± 0.024**	**0.729 ± 0.025**
SVM	0.688 ± 0.009	0.669 ± 0.011	0.659 ± 0.014	0.663 ± 0.011
KNN	0.708 ± 0.021	0.690 ± 0.018	0.675 ± 0.022	0.679 ± 0.019
MLP	0.657 ± 0.070	0.657 ± 0.117	0.638 ± 0.095	0.638 ± 0.102
DNN	0.666 ± 0.104	0.686 ± 0.059	0.651 ± 0.060	0.646 ± 0.094

Na Tabela 5.14 é apresentada uma abordagem em que o CTGAN é explicitamente condicionado a produzir classes equiproporcionais. O tamanho final do treino fica entre os cenários de SMOTE isolado e a utilização do CTGAN junto ao SMOTE no Experimento 4 (Seção 5.5), mantendo a distribuição 1:1:1.

Tabela 5.14: Distribuição das amostras nos conjuntos de treino e validação para cada classe nos k-folds antes e depois do CTGAN aumentando e balanceando (Experimento 5.

Fold	Conjunto	Leve	Moderada	Grave	Total
1	Treino (Antes)	319 (48.11%)	174 (26.24%)	170 (25.64%)	663
	Treino (Depois CTGAN)	319 (33.33%)	319 (33.33%)	319 (33.33%)	957
	Teste	80 (48.19%)	43 (25.90%)	43 (25.90%)	166
2	Treino (Antes)	319 (48.11%)	174 (26.24%)	170 (25.64%)	663
	Treino (Depois CTGAN)	319 (33.33%)	319 (33.33%)	319 (33.33%)	957
	Teste	80 (48.19%)	43 (25.90%)	43 (25.90%)	166
3	Treino (Antes)	319 (48.11%)	173 (26.09%)	171 (25.79%)	663
	Treino (Depois CTGAN)	319 (33.33%)	319 (33.33%)	319 (33.33%)	957
	Teste	80 (48.19%)	44 (26.51%)	42 (25.30%)	166
4	Treino (Antes)	319 (48.11%)	173 (26.09%)	171 (25.79%)	663
	Treino (Depois CTGAN)	319 (33.33%)	319 (33.33%)	319 (33.33%)	957
	Teste	80 (48.19%)	44 (26.51%)	42 (25.30%)	166
5	Treino (Antes)	320 (48.19%)	174 (26.20%)	170 (25.60%)	664
	Treino (Depois CTGAN)	320 (33.33%)	320 (33.33%)	320 (33.33%)	960
	Teste	79 (47.88%)	43 (26.06%)	43 (26.06%)	165

Nesta execução, a MLP exibe sobreajuste acentuado: a acurácia de treino escala de 0.45 para 0.80, enquanto a validação atinge próximo de 0.68 na época 10 e cai para 0.64 ao final, mantendo um hiato de 0.15. Trata-se de regressão em relação à rodada anterior, na qual o desvio era ≤ 0.03. Já a DNN sobe de 0.41 para 0.70 em treino; a validação ultrapassa o treino nas épocas 6–10 (0.75), estabiliza perto de 0.70 e encerra praticamente junto ao treino (perda de ≤ 0.02), representando melhora de generalização frente ao ciclo anterior, embora com acurácia absoluta ligeiramente inferior à melhor marca histórica (0.74). Os resultados podem ser observados na Figura 5.21.



(a) MLP (b) DNN

Figura 5.21: Curva de aprendizado das redes neurais construída a partir do conjunto de treinamento (Experimento 5.

Resultados do Teste

Os resultados reportados no conjunto de teste foram obtidos com os melhores hiperparâmetros definidos na etapa de otimização (Tabela 5.15). Observa-se que o *Random Forest* e o XGBoost se mantiveram próximos aos parâmetros do Experimento 4, mas com pequenas variações em profundidade e penalização. O Gradient Boosting voltou a utilizar taxa de aprendizado mais baixa e profundidade reduzida. O DNN, novamente, permaneceu inalterado.

Tabela 5.15: Melhores hiperparâmetros selecionados para o Experimento 5.

Algoritmo	**Hiperparâmetro**	**Valor**
Random Forest	Número de árvores	100
	Profundidade máxima	15
XGBoost	Número de árvores	100
	Profundidade máxima	3
	Taxa de aprendizado	0.2
Gradient Boosting	Número de árvores	200
	Taxa de aprendizado	0.2
	Profundidade máxima	3
	Subamostragem	0.6
SVM	Kernel	rbf
	Penalidade (C)	3
	Gamma	0.1

Continuação da Tabela 5.15 da página anterior

Algoritmo	**Hiperparâmetro**	**Valor**
	Balanceamento de amostras entre as classes	*balanced*
KNN	Número de vizinhos	11
	Ponderação	*distance*
	Distância (p)	1
	Tamanho da folha	15

Agora, observando os resultados no teste (Figura 5.22), o *Gradient Boosting* também manteve o melhor desempenho em *F1-score* e acurácia, seguido pelo *Random Forest* e pelo XGBoost.

No teste, os piores desempenhos ficaram por conta do SVM e do MLP, sendo o MLP o menos preciso em acurácia e o SVM o de pior precisão e revocação, além de liderar os menores *F1-score*s neste experimento.

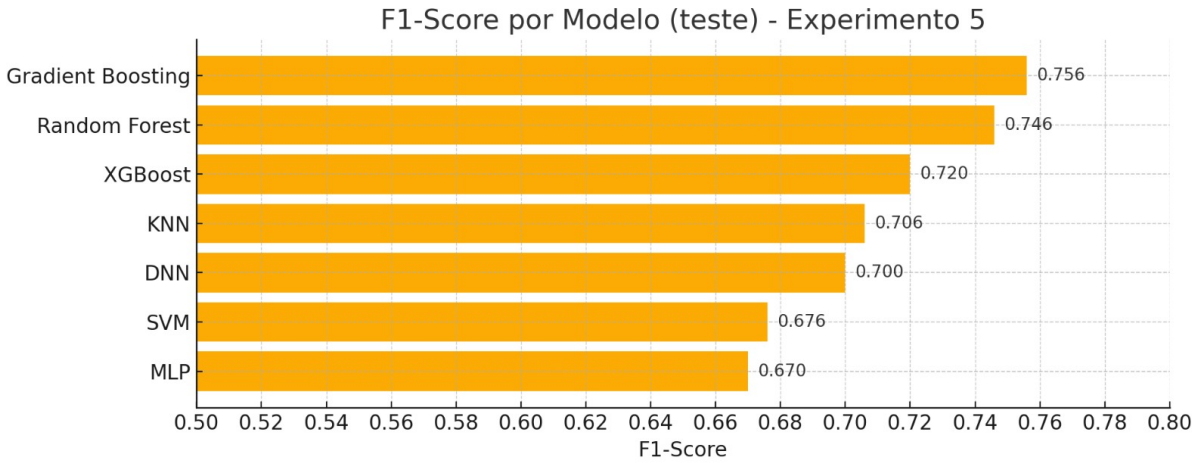
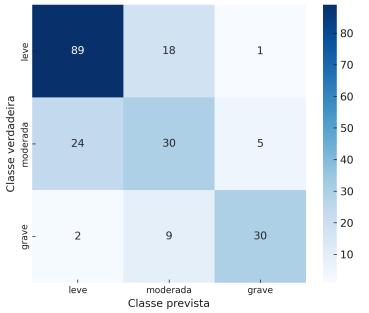
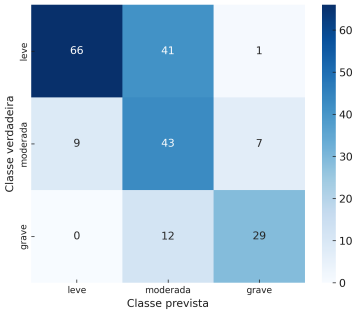


Figura 5.22: Métricas de resultado dos testes realizados para o Experimento 5.

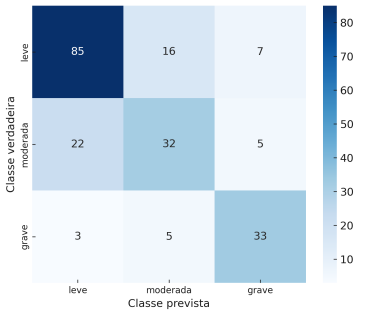
Nas matrizes de confusão observadas na Figura 5.23, percebe-se que a classe severa manteve alta taxa de acerto em todos os algoritmos. A classe moderada apresentou acurácia mediana, sendo frequentemente confundida com a leve. Já a classe leve registrou taxa de acerto intermediária, com erros direcionados principalmente para severa. Em especial, o algoritmo MLP exibiu comportamento destoante, alcançando muitos acertos em moderada, mas concentrando seus erros em leve, classificando-a erroneamente como severa.



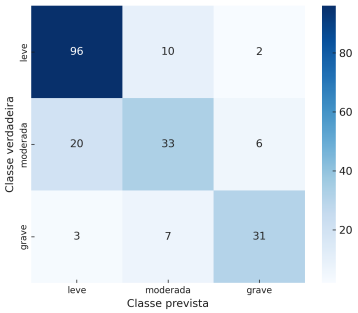
(a) DNN



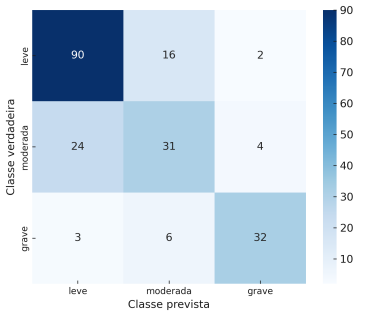
(b) MLP



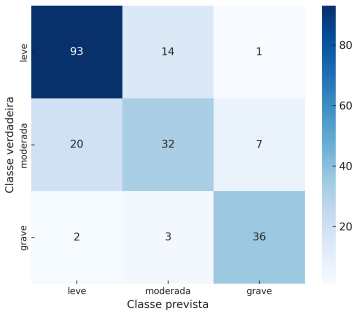
(c) KNN



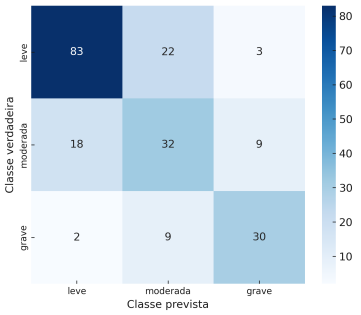
(d) *Random Forest*



(e) XGBoost



(f) *Gradient Boosting*



(g) SVM

Figura 5.23: Matriz de confusão dos algoritmos com balanceamento e aumento de dados com CTGAN para o Experimento 5.

No gráfico de erros a seguir, da Figura 5.24, podemos perceber a quantidade de erros para o algoritmo MLP que destoa dos outros, apresentando uma quantidade maior até do

que quando comparado com os resultados obtidos para o MLP em experimentos anteriores. Vale observar também que a maior parte dos erros cometidos pelo MLP foi no momento de classificar a classe leve, confundindo-a como moderada. Para os outros algoritmos, percebemos um comportamento semelhante.

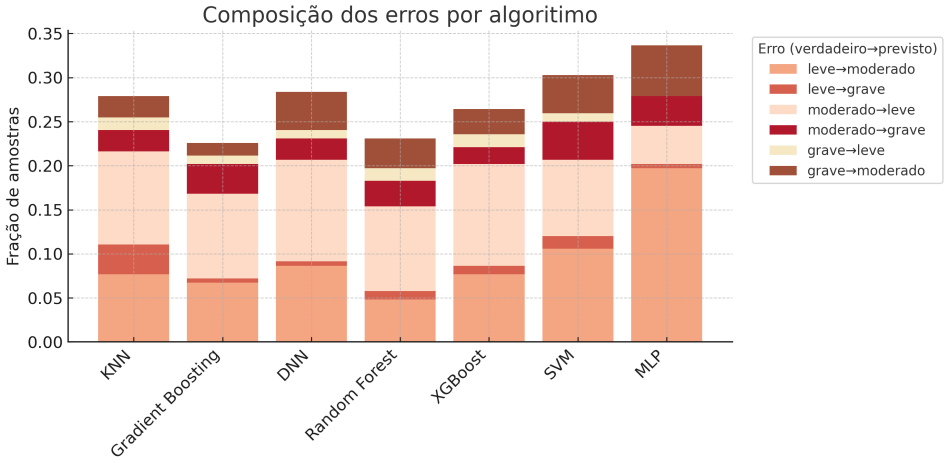


Figura 5.24: Gráfico de composição de erros no conjunto de teste do Experimento 5.

Na Figura 5.25 percebemos que o *Gradient Boosting* mantém a liderança com AUC = 0.88, igual ao valor do experimento anterior. O *Random Forest* recua ligeiramente para 0.87, enquanto o XGBoost cai para 0.86. Tanto a MLP quanto a DNN convergem em 0.85, indicando pequena perda de desempenho em relação ao experimento 4 (5.5). O destaque fica para o KNN, que se recupera de 0.78 para 0.85, aproximando-se dos algoritmos de rede neural. Já o SVM eleva-se de 0.60 para 0.64, retornando ao nível observado na base original, mas ainda distante dos demais algoritmos. O enriquecimento de dados via CTGAN proporcionou ganhos expressivos a métodos baseados em vizinhança e leves melhorias ao SVM, porém não superou o teto de desempenho imposto pelos *ensembles* de árvores.

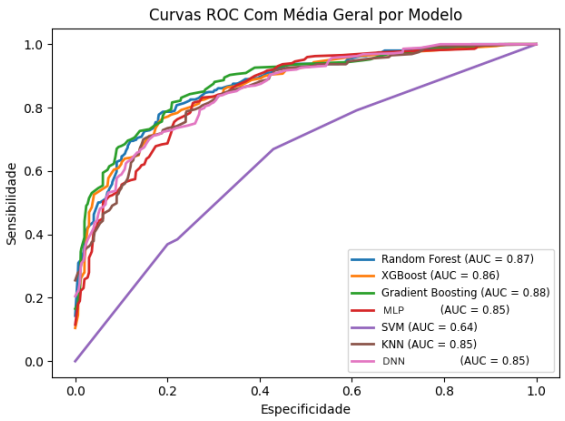


Figura 5.25: Curva ROC do Experimento 5.

5.7 Ferramenta *Web* para Diagnóstico Auxiliado por Computador

Esta seção apresenta, por meio de capturas de tela acompanhadas de explicações sobre funcionalidades e fluxos de interação, as páginas da ferramenta *web*, desenvolvida para apoiar o diagnóstico da STC. Serão descritas as seguintes telas **Login**, **Início**, **Submissão de Paciente**, **Histórico de Pacientes**, **Listagem de Pacientes**, **Listagem de Usuários** e **Listagem de Médicos**, ilustrando os resultados visuais, a interface, funcionalidades e fluxos de interação com o usuário, como descrito na Seção 4.8. Além disso, a ferramenta integra os três modelos de classificação de melhor desempenho, apresentados na Seção 6.1, garantindo assim precisão e consistência nos resultados gerados pelo sistema. Alguns casos clínicos reais, extraídos do conjunto de teste, também são apresentados com o objetivo de exemplificar a utilização da ferramenta, sendo os dados pessoais representados por informações fictícias, utilizados exclusivamente para fins de demonstração.

ADx-STC

O Assistente de Diagnóstico da Síndrome do Túnel do Carpo (ADx-STC) é uma ferramenta *web* desenvolvida para apoiar o clínico na triagem e na classificação da gravidade da STC. Por meio de uma interface intuitiva, o sistema recebe dados clínicos e ultrassonográficos do paciente, processa-os com os três algoritmos de aprendizado de máquina de melhor desempenho e exibe, em tempo real, a probabilidade de cada nível de gravidade: leve, moderada ou severa.

Como descrito na Seção 5.1, a ferramenta foi desenvolvida em Python 3.8 com *Flask*, seguindo o paradigma orientado a objetos, arquitetura MVC e exposição por *APIs* RESTful. A camada de apresentação utiliza *templates* Jinja2, os modelos treinados são serializados em *joblib* para carregamento eficiente em produção. Os dados são persistidos em *SQLite3*. O código é versionado no GitHub e o sistema é implantado em instância EC2 (Ubuntu 24.04) na AWS. Esse arranjo privilegia modularidade, reprodutibilidade e manutenibilidade, com apoio do *SDV*/CTGAN na geração de dados sintéticos durante a fase experimental.

O acesso à plataforma é restrito a usuários autenticados, por meio de uma página de login segura (Figura 5.26). O processo de autenticação adota o padrão *OAuth2*, com validação de credenciais no *backend*, emissão de *tokens* JWT para manutenção de sessão e proteção contra ataques de *replay* e falsificação de solicitações (*CSRF*). O sistema também realiza o gerenciamento de tempo de expiração e renovação de *refresh tokens*, assegurando o controle adequado do ciclo de login e logout dos usuários.

Para autenticar-se na plataforma, o usuário deve informar um e-mail e uma senha previamente cadastrados por um usuário com perfil de médico ou administrador. A ferramenta adota um sistema de controle de acesso baseado em papéis (*roles*), divididos em três categorias: paciente, médico e administrador. Cada perfil possui diferentes níveis de permissões: o paciente possui acesso mais restrito, limitado à visualização de seus próprios dados; o médico tem acesso intermediário, podendo submeter avaliações e consultar pacientes sob sua responsabilidade; e o administrador possui acesso ampliado, incluindo o gerenciamento de usuários, permissões e auditoria do sistema.

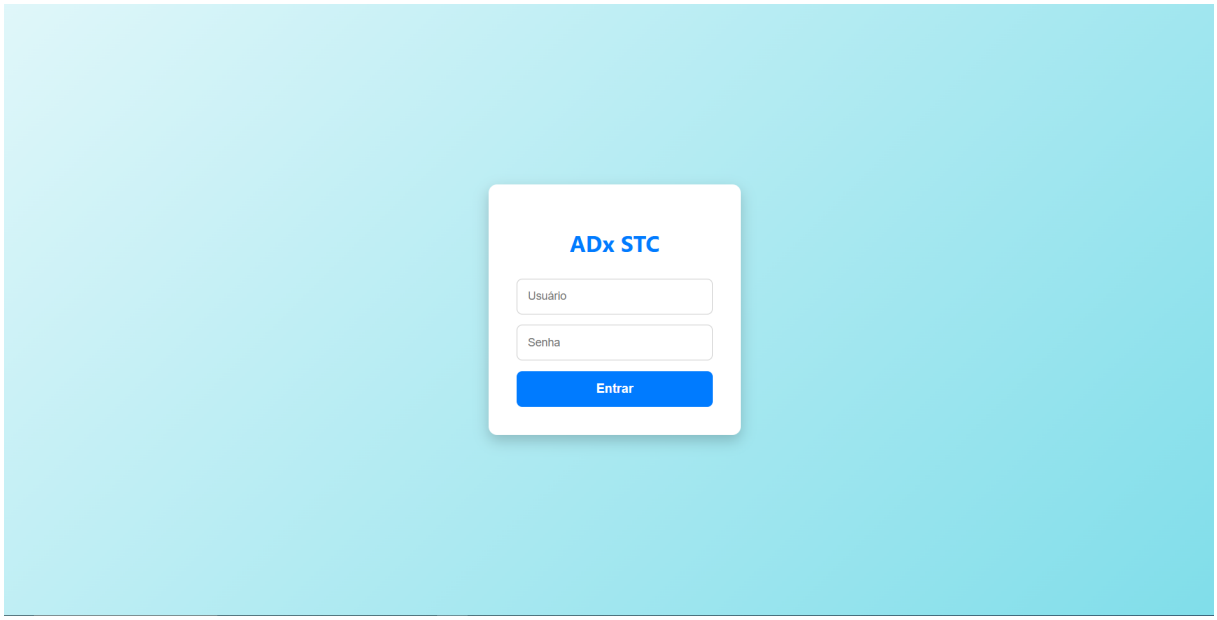


Figura 5.26: Página de *login*.

Início

A Figura 5.27 apresenta a página de início que possui uma interface acolhedora e responsiva, com mensagens de boas-vindas e um resumo das principais funcionalidades disponíveis no sistema. Ela serve como ponto de entrada para usuários autenticados, exibindo atalhos para ações como visualização de histórico e gerenciamento de perfil.

O menu lateral reúne as seguintes seções: **Início**, que retorna à página de apresentação; **ADx STC**, destinada a usuários com perfil médico para a realização do diagnóstico; **Histórico**, que exibe avaliações anteriores apenas do próprio usuário quando o perfil é paciente, e de todos os pacientes quando o perfil é médico ou administrador; **Pacientes**, visível a todos, porém com restrição de edição ao próprio registro quando o perfil é paciente; **Médicos**, que lista os profissionais cadastrados e permite alterações apenas ao próprio médico ou a administradores; e, por fim, **Sair**, para o encerramento da sessão ativa, com redirecionamento à tela de **login**, apresentada na Figura 5.26.

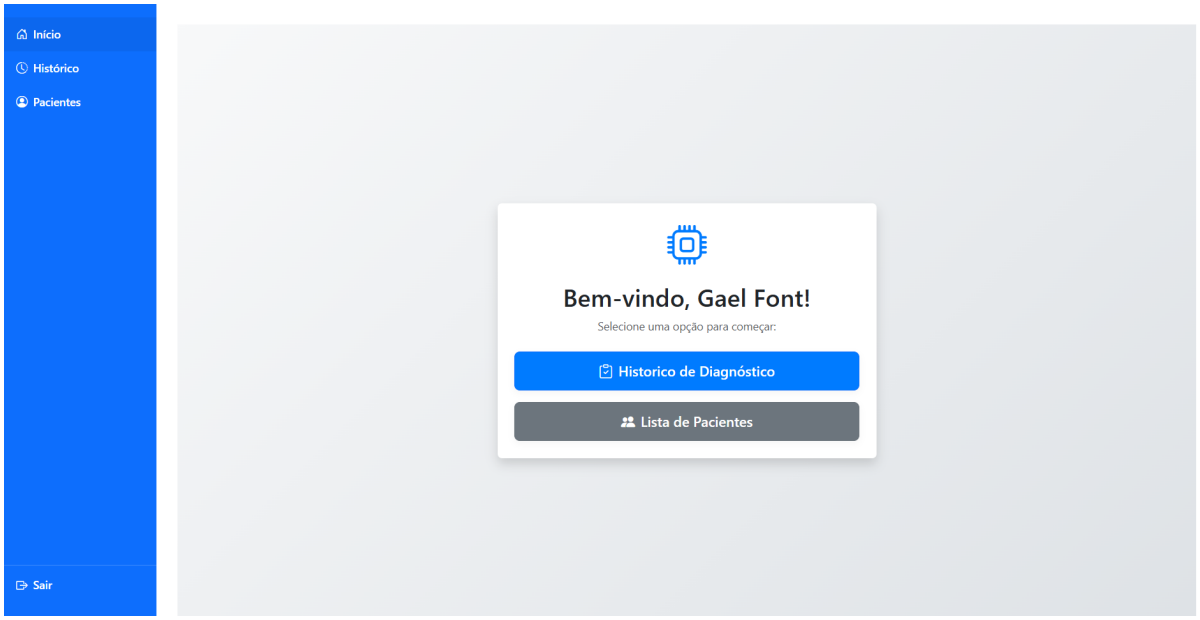


Figura 5.27: Página de início.

Submissão de Paciente

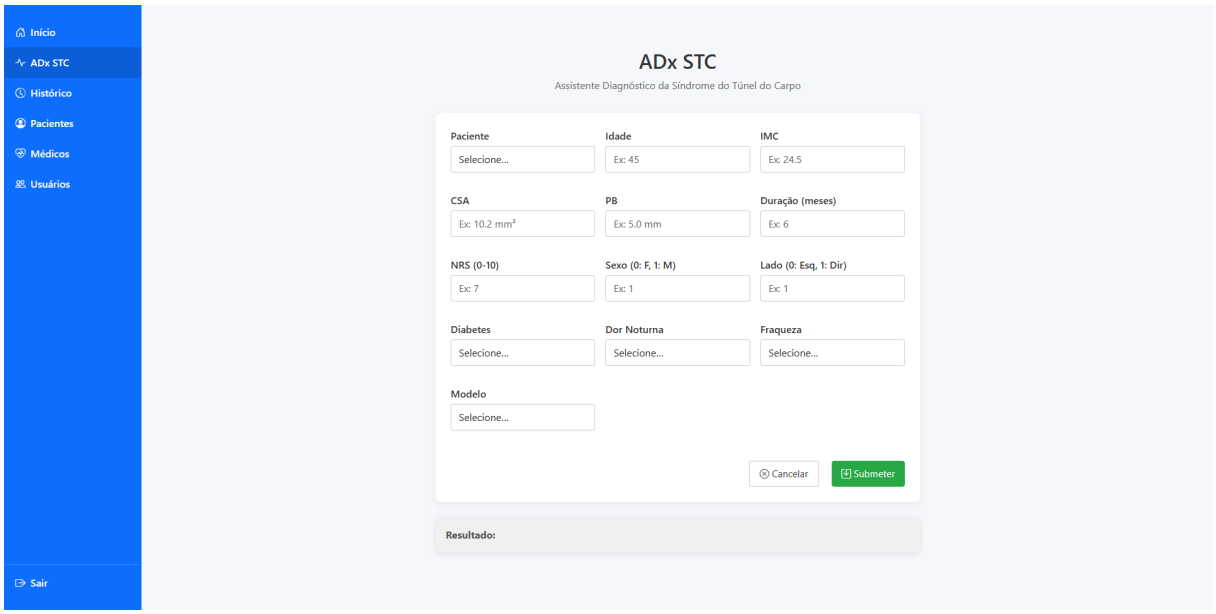


Figura 5.28: Página de submissão de dados do paciente.

A Figura 5.28 ilustra a interface disponibilizada aos usuários com perfil médico, permitindo o preenchimento de um formulário estruturado com os dados clínicos e sintomas do paciente, conforme descrito previamente na Tabela 4.1. Ao serem submetidos, os dados são direcionados ao modelo de ML escolhido entre os três de melhor desempenho, previ-

amente selecionados na *pipeline* experimental, o qual retorna a classificação da STC nos níveis de gravidade: leve, moderada ou grave.

Histórico de Pacientes

A página de histórico apresenta todas as avaliações já realizadas, listando data, hora, resultados obtidos e parâmetros utilizados. É possível aplicar filtros por médico responsável pelo diagnóstico, por paciente avaliado e pela data em que o exame foi realizado, permitindo consultas específicas conforme a necessidade do usuário. Usuários com perfil de paciente possuem acesso restrito a esse módulo, podendo visualizar apenas seus próprios diagnósticos anteriores e realizar filtragens com base no médico responsável. Além disso, há suporte para exportação em CSV, o que favorece análises comparativas e o monitoramento clínico dos pacientes a partir do modelo de ML.

Para pacientes

Como visto na Figura 5.29, a página de histórico permite ao paciente visualizar exclusivamente suas próprias avaliações, apresentando de forma clara a data de realização do exame, os parâmetros utilizados e os resultados obtidos em cada predição.

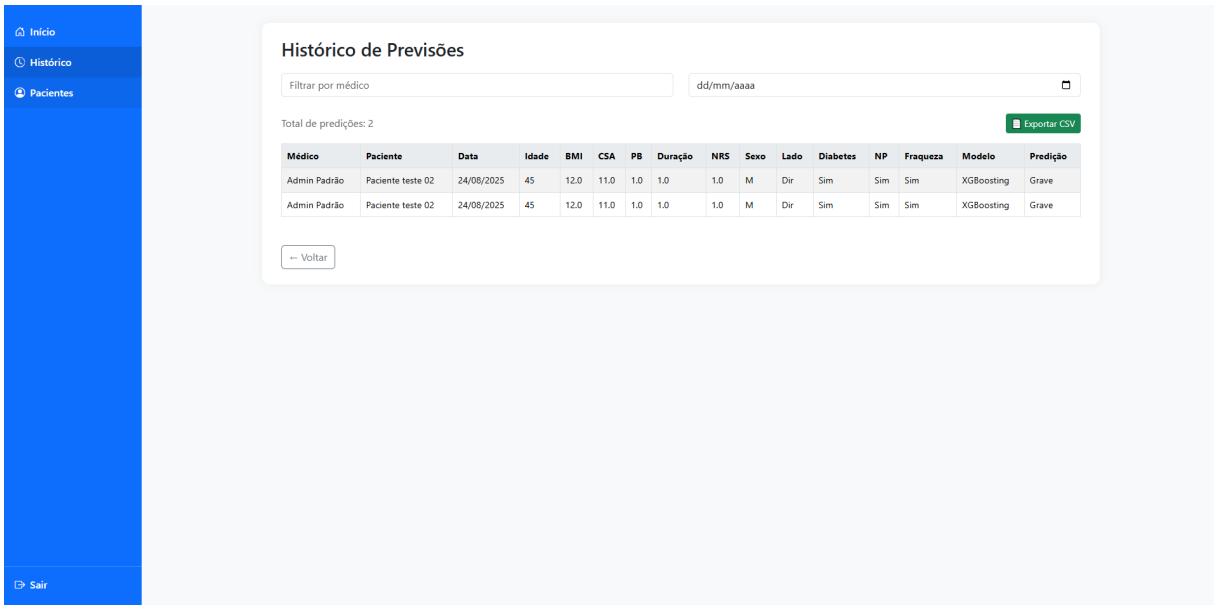


Figura 5.29: Página de histórico visão paciente.

Para médicos

A Figura 5.30 ilustra a visão de médico ou administrador na página de histórico. Nessa visão, são apresentadas todas as avaliações dos pacientes sob seu acompanhamento, com

informações de data, hora, parâmetros clínicos e classificação de gravidade do melhor algoritmo de ML.

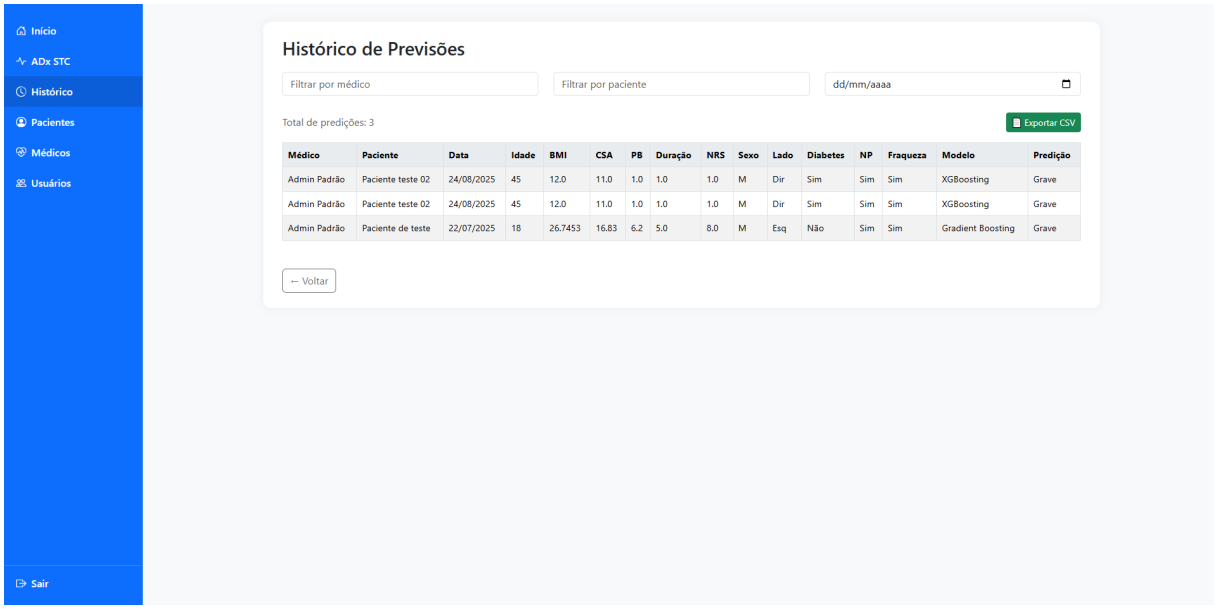


Figura 5.30: Página de histórico visão médico.

Listagem de Pacientes

Figura 5.31: Página de listagem de pacientes.

A Figura 5.31 demonstra como a página aparece quando acessada por um paciente. A página de listagem exibe apenas o seu próprio cadastro, possibilitando a edição de suas

informações pessoais diretamente nessa interface. Já na visão de médico ou administrador, são apresentados todos os pacientes do sistema, com botões dedicados para adicionar novos registros ou editar os cadastros existentes.

Listagem de Médicos

A página mostrada na Figura 5.32 está disponível apenas para usuários com perfil de médico ou administrador. O administrador visualiza todos os médicos cadastrados, podendo adicionar novos registros e editar qualquer perfil. Já o médico visualiza a lista completa de colegas, acessa os históricos vinculados a cada um e pode editar apenas o próprio cadastro.

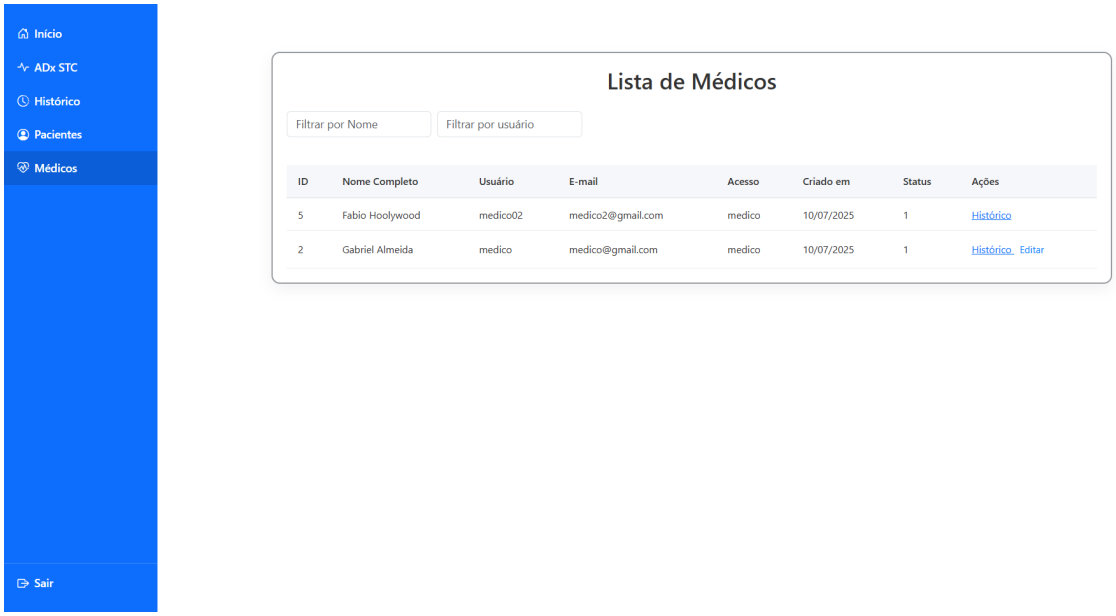


Figura 5.32: Página de listagem de médicos.

Listagem de Usuários

A página mostrada na Figura 5.33 está disponível apenas para usuários com perfil de administrador. A página de listagem exibe todos os usuários cadastrados no sistema, permitindo buscar, filtrar e ordenar registros. Nesta interface, o administrador pode cadastrar novos usuários de qualquer tipo (paciente, médico ou administrador) e também alterar o nível de acesso de usuários já existentes. São fornecidos botões de ação para editar, desativar ou redefinir perfis, que acionam modais de confirmação ou edição *inline*.

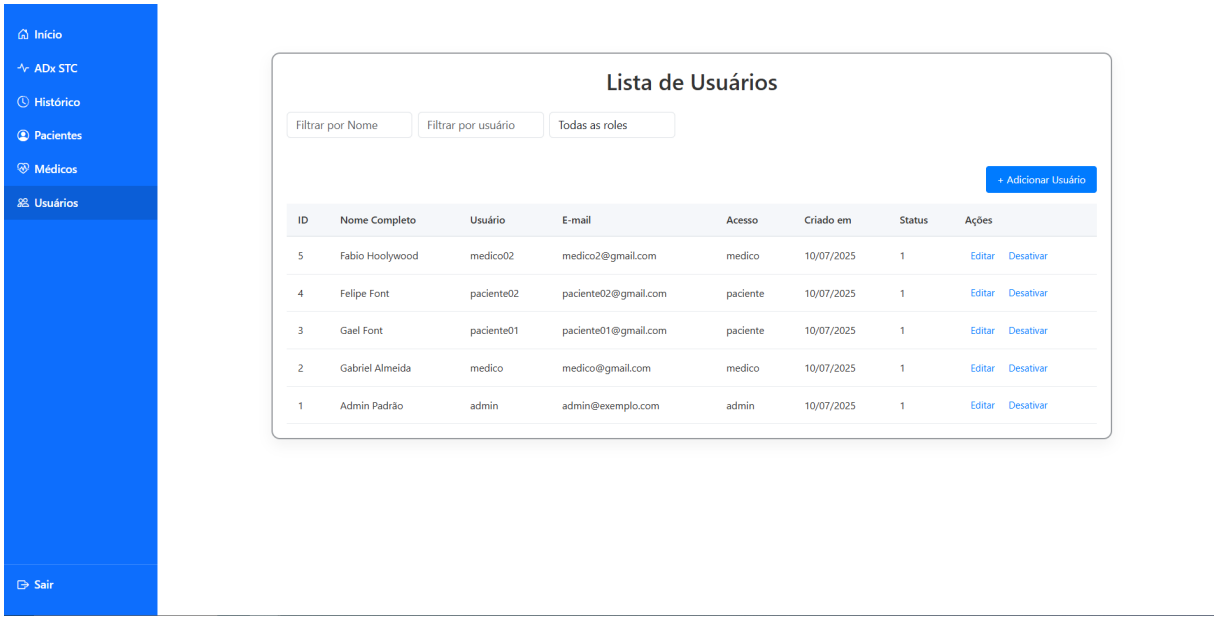


Figura 5.33: Página de listagem de usuários.

Cadastro de Usuários

A página mostrada na Figura 5.34 representa o formulário de cadastro e edição, é responsável por coletar informações básicas (nome, e-mail, senha) e dados complementares (perfil de acesso), com validação em tempo real. Campos obrigatórios são destacados, e mensagens de erro são exibidas de forma clara para orientar o usuário durante o preenchimento.

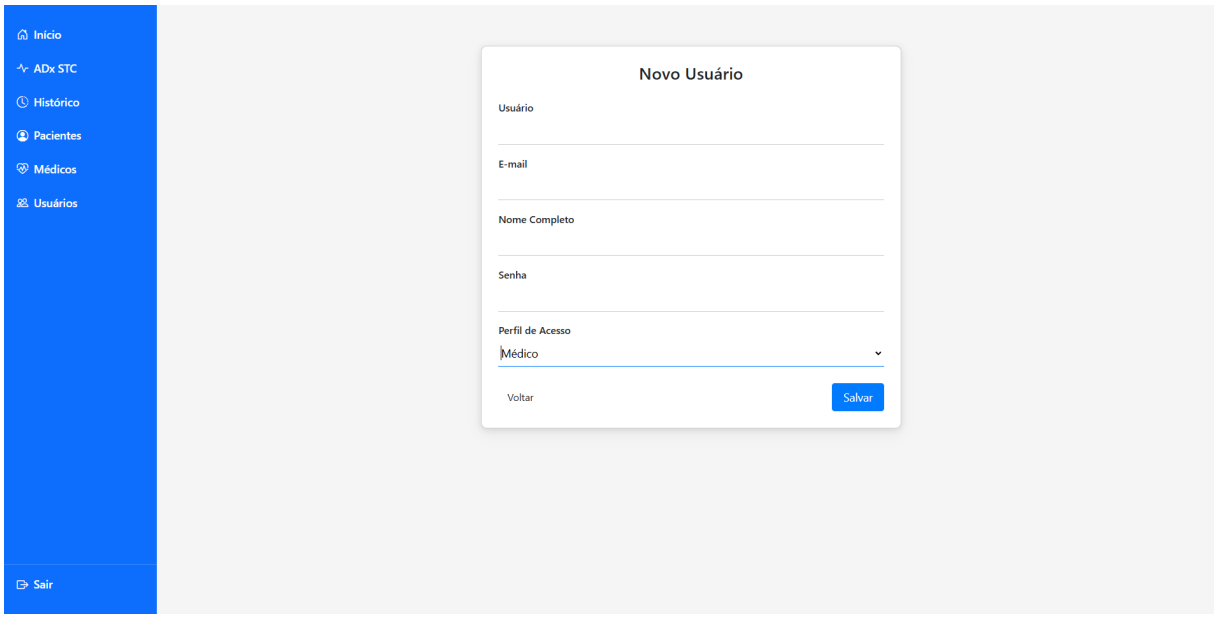


Figura 5.34: Página de cadastro de usuários.

Capítulo 6

Discussão de Resultados

Os resultados apresentados no Capítulo 5 serão discutidos de maneira sistemática neste capítulo, em prol de entender como os cinco cenários apresentados (Experimentos 1 ao 5 - Seções de 5.2 a 5.6) se comportaram ao tentar solucionar o problema de classificação da gravidade da STC no conjunto de teste. Além disso, os principais resultados serão discutidos, apontando limitações e implicações clínicas.

6.1 Desempenho Geral dos Algoritmos

Nos experimentos conduzidos, os métodos baseados em *ensemble* de árvores apresentaram desempenho superior, com *f1-score* variando de 0.71 a 0.77. Em conjuntos tabulares de dimensão reduzida, *Random Forest* e XGBoost destacaram-se por reduzir a variância e capturar relações não lineares de forma eficiente. Entre esses métodos, o *Gradient Boosting* obteve o melhor resultado, alcançando *f1-score* de 0.77, graças à combinação de árvores aditivas capazes de modelar interações complexas e aos mecanismos de regularização implícita que mitigam o sobreajuste sem comprometer o poder preditivo.

Quanto aos algoritmos de MLP e DNN, foi alcançado um *f1-score* promissor entre 0.67 e 0.74 ao longo dos experimentos, com uma pequena diferença de 0.03 no melhor resultado quando comparado com os métodos baseados em árvores. Essa diferença pode ser explicada pela quantidade moderada de amostras disponíveis, insuficiente para o treinamento eficaz dos modelos, e pela heterogeneidade das variáveis clínicas e ultrassonográficas, que favorece algoritmos de *bagging* e *boosting* na redução de variância. Além disso, observou-se o desempenho superior de ambos os algoritmos nos experimentos sem aumento e sem balanceamento, pois as configurações com essas técnicas introduziram ruído sintético, resultando em degradação do desempenho. Dessa forma, é possível observar os comportamentos descritos na Figura 6.1.

Considerando todos os experimentos, os três algoritmos de melhor desempenho foram: (i) *Gradient Boosting* com *f1-score* igual a 0.77 no Experimento 4 ; (ii) *Random Forest* com *f1-score* de 0.76 no Experimento 2 ; e (iii) XGBoost apresentando *f1-score* de 0.743 no Experimento 3. Em termos de comportamento, o *Gradient Boosting* manteve boa capacidade de generalização mesmo diante do ruído inerente à combinação de aumento e balanceamento, enquanto que o *Random Forest* mostrou robustez à superamostragem, preservando estabilidade e o XGBoost aproveitou a variabilidade introduzida exclusivamente pelo aumento sintético. Dessa forma, esses três algoritmos foram os selecionados para serem utilizados posteriormente na ferramenta web desenvolvida.

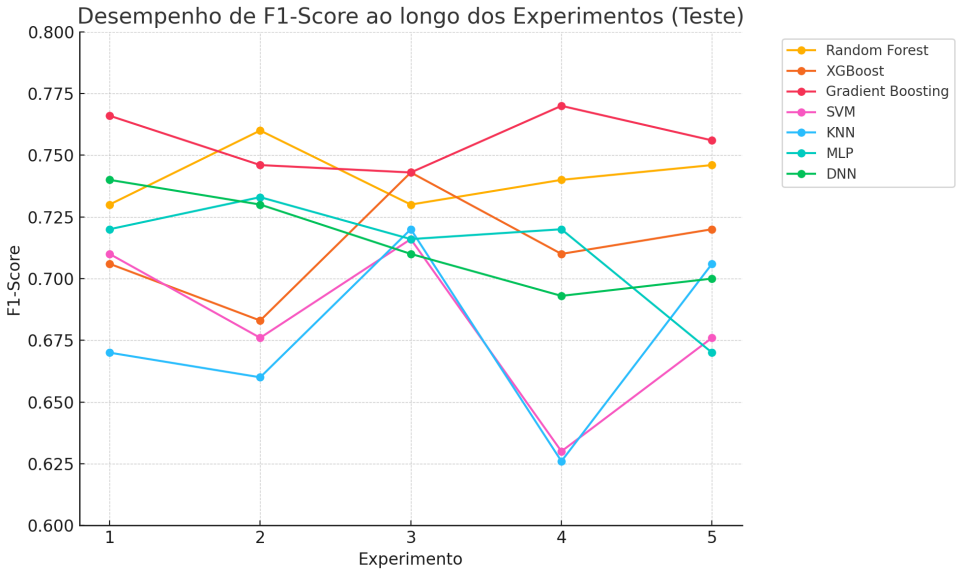


Figura 6.1: Gráfico com o comportamento do *f1-score* dos algoritmos executados no conjunto de teste em cada experimento.

6.2 Efeito do Balanceamento de Classes e Aumento de Dados

Ao realizar o aumento de dados utilizando SMOTE no Experimento 2 (Seção 5.3), não se observou ganho consistente no desempenho dos modelos. Embora tenha ocorrido uma leve elevação da revocação para a classe grave, verificou-se simultaneamente redução na precisão e, em parte, no *f1-score*. Esses resultados sugerem que o balanceamento puramente sintético via SMOTE pode melhorar o desempenho de predição em classes minoritárias, mas tende a degradar a exatidão geral do classificador. Portanto, ao aumentar apenas as classes minoritárias, cresce a densidade local e as fronteiras de decisão se deslocam a seu favor, gerando mais confusões de leve como moderada/grave e explicando o ganho de

desempenho de predição em moderada (e pontualmente em grave) com perda de precisão, além da queda de revocação em leve.

Ao aplicar o CTGAN isoladamente no Experimento 3 (Seção 5.4), preservando as proporções, observaram-se ganhos mais equilibrados. Em termos de acurácia, o maior avanço ocorreu no XGBoost (+2.88 p.p.) e no KNN (+2.88 p.p.), enquanto, entre os modelos neurais, o MLP apresentou desempenho ligeiramente superior ao DNN. Por classe, os efeitos médios foram contidos, mas informativos: na classe leve houve pequenas melhorias em precisão (+0.33 p.p.) e revocação (+0.53 p.p.); na moderada, ganhos heterogêneos, com médias de (+0.58 p.p.) em precisão e (+0.97 p.p.) em revocação; e na grave, observou-se discreta alta em precisão (+0.57 p.p.) com revocação quase estável (-1.05 p.p.).

Dessa forma, ao densificar todas as classes, elimina-se a assimetria de representação do experimento anterior, reduzindo o "arrasto" da fronteira para as minorias e estabilizando o compromisso entre revocação e precisão. Isso favorece modelos sensíveis à densidade (KNN, com ganhos marcantes em moderada) e de margem (SVM, com ganhos consistentes em leve e moderada), enquanto *ensembles* reagem de forma distinta (*Random Forest* estável; *Gradient Boosting* mais suscetível a ruído sintético) e MLP/DNN dependem de regularização e da fidelidade do gerador.

Em contraste, a estratégia híbrida que combina CTGAN e SMOTE no Experimento 4 (Seção 5.5) revelou-se a mais eficaz entre as abordagens testadas, proporcionando ganhos modestos, porém consistentes, no desempenho dos modelos. Ao aumentar sinteticamente todas as classes, observou-se um comportamento mais uniforme do que no aumento direcionado às minoritárias, porém sem ganho global de desempenho. Em acurácia, o destaque positivo foi o *Random Forest* (0.00 p.p.), que manteve estabilidade, enquanto entre os modelos neurais o MLP apresentou desempenho ligeiramente superior ao DNN. A análise por classe confirmou padrões distintos: na classe leve, houve queda em precisão (-0.50 p.p.) e revocação (-7.67 p.p.); na moderada, redução em precisão (-6.45 p.p.) acompanhada de aumento em revocação (+2.42 p.p.); e na grave, queda de precisão (-4.67 p.p.) com revocação praticamente estável (+0.35 p.p.).

Ademais, aumentar todas as classes reduz a assimetria de representação introduzida quando somente as minoritárias são aumentadas, mas mantém o prior em favor da classe leve. O aumento indiscriminado mitiga o viés estrutural do aumento só nas minorias, mas os ganhos não são uniformes: há perda de desempenho de predição da classe majoritária e oscilações de precisão/revocação nas minorias.

No Experimento 5 (Seção 5.6), o CTGAN foi ajustado para gerar, de forma condicional, um número igual de amostras da classe majoritária (leve). Essa estratégia não resultou em ganho uniforme de desempenho. Em acurácia, o destaque positivo foi o *Random Forest*, com leve melhora (+0.96 p.p.), enquanto entre os modelos neurais o DNN apresentou

desempenho superior ao MLP. A análise por classe revelou efeitos distintos: na leve, houve precisão de (+2.34 p.p.) e revocação de (-8.33 p.p.), indicando predição mais conservadora, porém menos sensível; na moderada, registraram-se (-5.57 p.p.) em precisão e (+6.30 p.p.) em revocação, sinal de recuperação de positivos reais à custa de mais falsos positivos; e na grave, a precisão caiu (-3.90 p.p.), enquanto a revocação apresentou leve ganho de (+1.05 p.p.).

Comparado ao Experimento 2 (Seção 5.3), o Experimento 5 (Seção 5.6) reduz a assimetria de representação e melhora a cobertura das regiões de decisão de moderada e grave. Ainda assim, como o prior de classe se mantém, o adensamento sintético aproxima as fronteiras: leve perde apresenta queda de desempenho, moderada ganha e grave exibe o *trade-off* com a precisão caindo e a revocação aumentando. O impacto depende do viés indutivo: KNN é muito sensível à densidade local; SVM tende a mover a margem em favor da classe grave; *Random Forest* mostrou maior resiliência (acurácia +0.96 p.p. e balanço razoável), enquanto XGBoost foi o único a melhorar simultaneamente a precisão e a revocação em grave.

Dessa forma, a análise dos cinco experimentos mostra que a base original já oferecia um bom equilíbrio entre desempenho e precisão, alcançando o melhor *f1-score* de (0.766) com o *Gradient Boosting*. Apenas um arranjo de aumento de dados conseguiu superá-la: o Experimento 4 (Seção 5.5), cujo melhor *f1-score* atingiu (0.770; +0.4 p.p.). Nos demais casos, os efeitos foram mais localizados: no Experimento 2 (Seção 5.3), houve ganhos em revocação da classe grave, mas acompanhados de perda em precisão e redução do *f1-score* final (0.760; -0.8 p.p.); o Experimento 3 (Seção 5.4) mitigou a assimetria de representação, mas ampliou a sobreposição entre classes, resultando em queda global; já o Experimento 5 recuperou parcialmente a classe moderada e, em menor grau, a grave, porém reduziu a revocação da leve e o *f1-score* global. No conjunto, isso explica por que a base original frequentemente performou melhor: com poucos dados (1037 amostras), os geradores sintéticos (SMOTE/CTGAN) tendem a replicar padrões locais e introduzir ruído entre classes; muitos classificadores (KNN, SVM) passam então a confundir fronteiras, pagando com queda de precisão nas minorias e/ou perda de revocação na maioria. O Experimento 4 foi a exceção porque combinou diversidade sintética utilizando o CTGAN com refinamento local do SMOTE em todas as classes, reduzindo o viés estrutural do Experimento 2, e modelos mais robustos a ruído (como *Gradient Boosting*) conseguiram capitalizar essa diversidade extra sem perder calibragem. É possível observar esse comportamento descrito previamente na Tabela 6.1, onde cada linha mostra, para um experimento, o melhor *f1-score* obtido no teste, o algoritmo responsável e o ganho em pontos percentuais (p.p.) em relação ao melhor *f1-score* do Experimento 1 (Seção 5.2).

Tabela 6.1: Resultado de ponto percentual (p.p.) para cada experimento baseado no *f1-score* resultante da avaliação do conjunto de teste.

Experimento	Melhor *f1-score*	Algoritmo	Ganho
1	0.766	*Gradient Boosting*	—
2	0.760	*Random Forest*	−0.8 p.p.
3	0.743	*Gradient Boosting/XGBoost*	−2.3 p.p.
4	**0.770**	*Gradient Boosting*	**+0.4** p.p.
5	0.756	*Gradient Boosting*	−1.0 p.p.

6.3 Análise por Classe: Dificuldade com a Classe Moderada

Os gráficos de barras apresentados nas Figuras 6.2 e 6.3 ilustram o comportamento da revocação e f1-*score* para cada classe (leve, moderada, grave) ao longo dos cinco cenários experimentais (Seções 5.2 - 5.6) para os melhores algoritmos apontados na Tabela 6.1 da Seção anterior 6.2. Observa-se que, enquanto leve e grave mantêm uma identificação estável, a moderada exibe os menores desempenhos, confirmando a persistente dificuldade de discriminação dessa categoria.

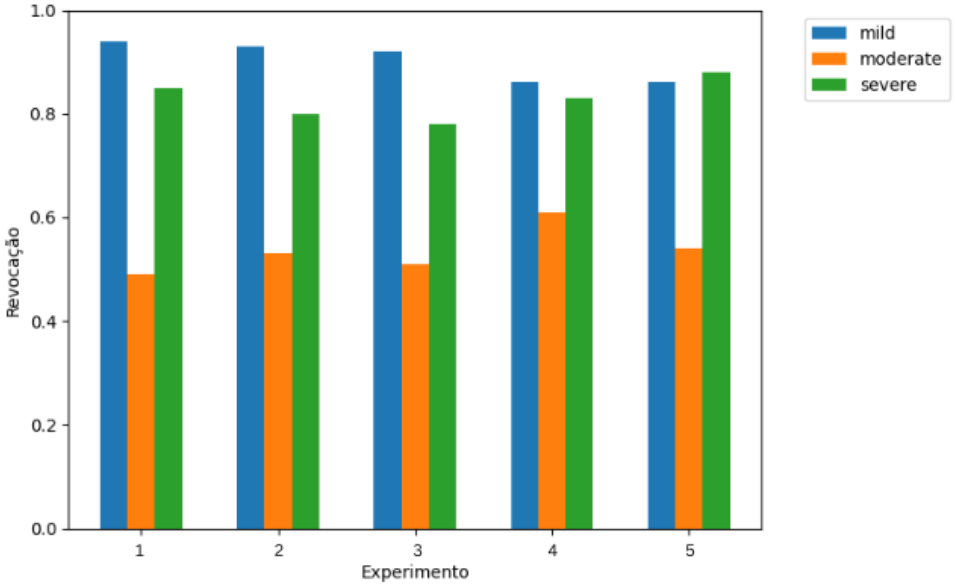


Figura 6.2: Revocação por classe em cada Experimento.

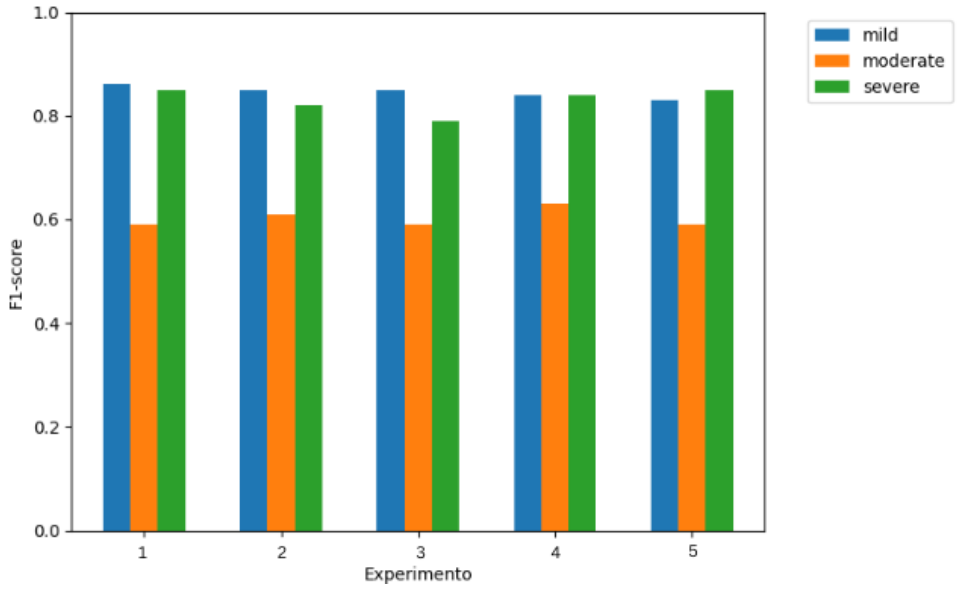


Figura 6.3: f1-score por classe em cada Experimento.

No Experimento 1 (Seção 5.2), a classe moderada apresentou os piores resultados, com revocação de apenas 0.49 e *f1-score* de 0.59. Esses valores foram significativamente inferiores aos observados em leve (revocação de 0.94, f1-score de 0.86) e grave (revocação de 0.85, *f1-score* de 0.85). Com a aplicação do SMOTE para o balanceamento das classes no Experimento 2 (Seção 5.3), houve uma leve melhora nos resultados para a classe moderada, com revocação subindo para 0.53. O uso isolado do CTGAN para o aumento dos dados no Experimento 3 (Seção 5.4) também gerou um pequeno ganho, elevando a revocação para 0.51. No entanto, em ambos os casos, observou-se uma queda no desempenho de predição da classe grave, e as disparidades entre as classes permaneceram elevadas (acima de 0.40 p.p.), indicando que os métodos de balanceamento aplicados não foram suficientes para nivelar o desempenho entre as categorias.

A estratégia de aumentar os dados com CTGAN e balancear as classes com SMOTE no Experimento 4 (Seção 5.5) produz o melhor equilíbrio: a revocação da classe moderada salta para 0.61, enquanto leve e grave permanecem em 0.86 e 0.83, respectivamente. O f1-score acompanha o ganho, atingindo 0.63 em moderada e mantendo acima de 0.84 nas outras classes. Por fim, a utilização do CTGAN como técnica para aumentar e balancear os dados (Experimento 5 - Seção 5.6) privilegia a classe grave (0.88 de revocação), mas não supera o SMOTE puro na classe moderada (0.54).

Em complemento aos gráficos, o CV (Seção 2.6) calculado entre as três classes e apresentado nas tabelas 6.3 e 6.5, oferece uma métrica única de consistência. O Experimento 4 (Seção 5.5) apresenta o menor desvio relativo (CV = 0.1454 para a revocação; e CV = 0.1286 para o *f1-score*), confirmando quantitativamente que SMOTE junto ao CTGAN é o cenário que melhor equaliza o desempenho multiclasses. Ademais, a persistente difi-

culdade com a classe moderada decorre de sua alta similaridade com as classes vizinhas em termos de características clínicas e ultrassonográficas (como é possível observar nos gráficos das Figuras 5.5, 5.9, 5.14, 5.19 e 5.24). Do ponto de vista clínico, essa dificuldade de separação pode refletir a transição fisiopatológica gradual do nervo mediano: a área de secção transversal (CSA) aproxima-se do limiar diagnóstico de 13–15 mm^2 justamente nessa faixa, reduzindo a discriminabilidade estatística entre as classes adjacentes. Embora técnicas de balanceamento reduzam disparidades, elas não eliminam totalmente a ambiguidade intrínseca. É interessante que futuras investigações explorem modelos que incorporem a estrutura ordinal dos estágios e aprimorem a representação de atributos contínuos para aumentar a discriminabilidade da categoria intermediária.

Tabela 6.3: Revocação por classe e CV entre classes.

Cenário	**Revocação**			**CV**
	Leve	**Moderada**	**Grave**	
5.2 – Base de Dados Desbalanceada e Sem Aumento	0.94	0.49	0.85	0.2558
5.3 – Balanceamento de Classes utilizando SMOTE	0.93	0.53	0.80	0.2212
5.4 – Aumento de Dados utilizando CTGAN	0.92	0.51	0.78	0.2310
5.5 – Aumento de Dados utilizando CTGAN e Balanceamento de Classes utilizando SMOTE	0.86	0.61	0.83	0.1454
5.6 – Balanceamento de Classes utilizando CTGAN	0.86	0.54	0.88	0.2050

Tabela 6.5: *f1-score* por classe e e CV entre classes.

Cenário	***f1-score***			**CV**
	Leve	**Moderada**	**Grave**	
5.2 – Base de Dados Desbalanceada e Sem Aumento	0.86	0.59	0.85	0.1630
5.3 – Balanceamento de Classes utilizando SMOTE	0.85	0.61	0.82	0.1405
5.4 – Aumento de Dados utilizando CTGAN	0.85	0.59	0.79	0.1495
5.5 – Aumento de Dados utilizando CTGAN e Balanceamento de Classes utilizando SMOTE	0.84	0.63	0.84	0.1286
5.6 – Balanceamento de Classes utilizando CTGAN	0.83	0.59	0.85	0.1561

6.4 Limitações

Ao decorrer do desenvolvimento deste trabalho, foram enfrentadas algumas limitações, como o tamanho amostral reduzido que limitou a capacidade de treinamento dos algoritmos de ML e os de DL em dados tabulares e comprometendo a robustez dos exemplos sintéticos gerados pelo CTGAN, recomendando-se a ampliação da base e a realização de validação externa.

Ademais, observa-se uma ambiguidade intrínseca na classe moderada, decorrente da sobreposição de características ultrassonográficas em relação às classes leve e grave, o que dificulta sua distinção; como mitigação, sugere-se a incorporação de variáveis adicionais (clínicas e/ou eletrofisiológicas) e o emprego de modelagem ordinal que respeite o ordenamento das categorias.

O fato do conjunto de dados ser proveniente de um único hospital universitário limita a validade externa e a generalização dos dados, sendo desejável a avaliação de mais de um hospital.

Por fim, o foco exclusivo em dados estruturados restringe o *pipeline* às variáveis clínicas, sem explorar informações potencialmente contidas nas imagens de ultrassom; investigações futuras devem considerar abordagens utilizando dados tabulares juntamente com imagens, com extração automática de atributos por CNNs e estratégias de fusão apropriadas. Tais aspectos devem ser ponderados na interpretação dos resultados e no delineamento de estudos subsequentes.

Capítulo 7

Conclusão

Este trabalho desenvolveu e avaliou um *pipeline* completo de algoritmos de aprendizado de máquina e aprendizado profundo para auxiliar o diagnóstico e a classificação da gravidade da STC a partir de dados clínicos. Foram comparados sete algoritmos supervisionados em uma base de dados com 1037 amostras rotuladas nas classes leve, moderada e grave, submetidos a cinco cenários de reamostragem (Experimentos 1 ao 5 - Seções 5.2 a 5.6) e avaliados via validação cruzada em cinco *folds*. Os modelos tiveram seus hiperparâmetros otimizados e foram testados em um conjunto de teste isolado. Foi possível observar ganhos da otimização de hiperparâmetros em comparação com os modelos base, evidenciando a relevância dessa etapa para o ajuste fino dos classificadores. Além disso, foi possível identificar os classificadores com maior potencial para a classificação da gravidade da STC.

Foram comparadas abordagens tradicionais e contemporâneas, destacando-se que a combinação de estratégias gerou ganhos de desempenho relevantes, como detalhado na seção de resultados (5). Observou-se ainda que os efeitos da reamostragem variaram entre as classes, reforçando a importância dessa etapa para o balanceamento adequado dos dados e para a melhoria da capacidade de generalização dos modelos.

O *Gradient Boosting* destacou-se como o melhor modelo, atingindo *F1-score* macro de 0.77 no experimento utilizando CTGAN para aumentar os dados e o SMOTE para balancear as classes, equilibrando precisão e revocação entre as classes e elevando a revocação da classe grave até 0.82, mesmo que o ganho em relação à base original tenha sido modesto (+0.4 p.p.), porém relevante para a sensibilidade a casos graves. Por outro lado, o uso isolado de CTGAN para o balanceamento das classes mostrou-se nocivo, introduzindo ruído sintético que ampliou a disparidade entre moderada e grave, evidenciando a importância de avaliações objetivas da qualidade dos dados gerados. A análise por classe confirmou o desafio persistente em discriminar o estágio moderado, resultado da sobreposição intrínseca das medidas ultrassonográficas e clínicas nessa faixa de gravidade.

A partir dos resultados experimentais, desenvolveu-se integralmente uma aplicação *web* baseada na arquitetura MVC, conteinerizada e com autenticação JWT, disponibilizando rotas RESTful. O sistema adota boas práticas de usabilidade e compatibilidade, além de integrar de forma robusta, segundo os princípios de MLOps, os três modelos de melhor desempenho, selecionados pelo *F1-score* no conjunto de teste. A interface implementa funcionalidades de login, cadastro e gerenciamento de pacientes, submissão de dados e visualização dos resultados de classificação, visando facilitar sua adoção em fluxos de trabalho hospitalares, conforme descrito na Seção 5.7.

Sugestões Futuras

É recomendado explorar abordagens de modelagem ordinal aliadas a funções de perda customizadas que considerem explicitamente a distância entre as classes. Ao aplicar penalidades crescentes para erros mais graves (por exemplo, classificar grave como leve) espera-se melhorar a separação da classe moderada e, consequentemente, a acurácia global do sistema.

Outra direção promissora consiste na normalização de variáveis contínuas e na incorporação de novos biomarcadores. A CSA do nervo pode ser ajustada por métricas individuais, como estatura ou circunferência do punho, reduzindo a variabilidade inter-paciente. Além disso, a integração de exames adicionais, como estudos de condução nervosa e escores funcionais, tem potencial para enriquecer o vetor de atributos e aumentar o poder preditivo do modelo.

Para assegurar a qualidade dos dados sintéticos gerados pelo CTGAN, é fundamental aplicar métricas objetivas que avaliem sua fidelidade estatística e clínica. Essas métricas devem quantificar a preservação das distribuições marginais, das correlações entre variáveis e da plausibilidade fisiológica dos registros sintetizados.

Sugerem-se também colaborações interinstitucionais que possibilitem coletar dados de diferentes hospitais, populações e protocolos de imagem. A diversidade resultante favorecerá a generalização do modelo, sobretudo se forem combinados dados clínicos, ultrassonográficos, ENMG e histórico de tratamentos em um conjunto multimodal coeso.

Adicionalmente, a realização de ensaios clínicos prospectivos ou estudos-piloto permitirá avaliar o desempenho do sistema em cenários reais, medir a aceitação entre profissionais de saúde e quantificar o impacto em tempo de diagnóstico e desfechos clínicos. Essas evidências são cruciais para demonstrar benefícios concretos antes de uma implementação em larga escala.

Por fim, inspirando-se em sistemas de CAD de outras áreas, propõe-se o monitoramento longitudinal de pacientes, integrando o modelo de IA a *dashboards* interativos que

acompanhem a evolução da CSA e dos demais biomarcadores ao longo do tratamento. Essa funcionalidade poderá apoiar decisões terapêuticas personalizadas e fornecer *feedback* contínuo sobre a eficácia das intervenções.

Referências

[1] Park, Dougho, Byung Hee Kim, Sang Eok Lee, Dong Young Kim, Mansu Kim, Heum Dai Kwon, Mun Chul Kim, Ae Ryoung Kim, Hyoung Seop Kim e Jang Woo Lee: *Machine learning-based approach for disease severity classification of carpal tunnel syndrome*. Sci Rep, 2021. `https://www.nature.com/articles/s41598-021-97043-7`. ix, xi, 2, 6, 27, 30, 35, 36, 37

[2] Network, Mozilla Developer: *Model–view–controller diagram*, 2025. `https://developer.mozilla.org/en-US/docs/Glossary/MVC`. ix, 9

[3] Yoshii, Yuichi, Chunfeng Zhao e Peter C. Amadio: *Recent advances in ultrasound diagnosis of carpal tunnel syndrome*. Diagnostics, 10(8), 2020, ISSN 2075-4418. `https://www.mdpi.com/2075-4418/10/8/596`. 1, 5, 6, 7

[4] Mekki, Y. M., H. C. Rhim, D. Daneshvar, A. N. Pouliopoulos, C. Curtin e E. Hagert: *Applications of artificial intelligence in ultrasound imaging for carpal-tunnel syndrome diagnosis: a scoping review*. International Orthopaedics, 49(4):965–973, 2025. 1, 6, 7

[5] Faeghi, A., M. Sharifi e S. Faeghi: *Accurate diagnosis of carpal tunnel syndrome using machine learning models*. Computers in Biology and Medicine, 134:104456, 2021. 1, 6, 7

[6] Elseddik, Marwa, Khaled Alnowaiser, Reham R. Mostafa, Ahmed Elashry, Nora El-Rashidy, Shimaa Elgamal, Ahmed Aboelfetouh e Hazem El-Bakry: *Deep learning-based approaches for enhanced diagnosis and comprehensive understanding of carpal tunnel syndrome*. Diagnostics, 13(20):3211, 2023. 1, 2

[7] Bishop, Christopher M.: *Pattern Recognition and Machine Learning*. Springer, 2006. 1, 15, 16

[8] LeCun, Yann, Yoshua Bengio e Geoffrey Hinton: *Deep learning*. Nature, 521(7553):436–444, 2015. 2

[9] Esteva, Andre, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Kristin Chou, Claire Cui, Greg Corrado, Sebastian Thrun e Jeff Dean: *A guide to deep learning in healthcare*. Nature Medicine, 25(1):24–29, 2019. 2, 6, 12

[10] Yetiş, Mehmet, Hikmet Kocaman, Mehmet Canlı, Hasan Yıldırım, Aysu Yetiş e İsmail Ceylan: *Carpal tunnel syndrome prediction with machine learning algorithms*

using anthropometric and strength-based measurement. PLOS ONE, 19(4):e0300044, 2024. 2, 38

[11] Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall e William P. Kegelmeyer: *Smote: Synthetic minority over-sampling technique*. Em *Journal of Artificial Intelligence Research*, volume 16, páginas 321–357, 2002. 2, 20

[12] Xu, Lei, Maria Skoularidou, Alfredo Cuesta-Infante e Kalyan Veeramachaneni: *Modeling tabular data using conditional gan*. Em *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. `https://papers.nips.cc/paper_files/paper/2019/hash/254edc0180f3f3c30d611a3e4b6420fd-Abstract.html`. 2, 20

[13] Sethanan, Kanchana, Rapeepan Pitakaso, Thanatkij Srichok, Surajet Khonjun, Nantawatana Weerayuth, Chutinun Prasitpuriprecha, Thanawadee Preeprem, Sirima Suvarnakuta Jantama, Sarayut Gonwirat, Prem Enkvetchakul, Chutchai Kaewta e Natthapong Nanthasamroeng: *Computer-aided diagnosis using embedded ensemble deep learning for multiclass drug-resistant tuberculosis classification*. Frontiers in Medicine, Volume 10 - 2023, 2023, ISSN 2296-858X. `https://www.frontiersin.org/journals/medicine/articles/10.3389/fmed.2023.1122222`. 2, 31, 33, 47

[14] Novo, Jorge, José Rouco, Noelia Barreira, Marcos Ortega, Manuel G. Penedo e Aurélio Campilho: *Wivern: a web-based system enabling computer-aided diagnosis and interdisciplinary expert collaboration for vascular research*. Journal of Medical and Biological Engineering, 37(6):920–935, 2017, ISSN 2199-4757. `https://doi.org/10.1007/s40846-017-0256-y`. 2, 8, 9, 31, 33

[15] Papaléo, Ricardo M. e Daniel S. de Souza: *Ultrassonografia: Princípios físicos e controle da qualidade*. Revista Brasileira de Física Médica, 13(1):14–23, 2019. 5

[16] Bakhru, Rita N. e William D. Schweickert: *Intensive care ultrasound: I. physics, equipment, and image quality*. Annals of the American Thoracic Society, 10(5):540–548, 2013. 5

[17] Zaki, H. A., E. Shaban, W. Salem, F. Bilal, M. Fayed, M. Hendy e colleagues: *A comparative analysis between ultrasound and electromyographic and nerve conduction studies in diagnosing carpal tunnel syndrome (cts): A systematic review and meta-analysis*. Cureus, 14(10):e30476, 2022. 6

[18] Battaglia, F., L. Troisi, E. Cigna, F. Stagno d'Alcontres, V. Rizzo e G. Delia: *Standardized high-resolution ultrasound protocol for the diagnosis and monitoring of carpal tunnel syndrome: A mixed-design observational study*. Diagnostics, 15(13):1593, 2025. 6

[19] Elseddik, Amal, Ayman Soliman, Hoda M. El-Bakry e Ayman Eldeib: *Machine learning model for the diagnosis and prognosis of carpal tunnel syndrome using clinical, sonographic, and electrophysiological data*. Diagnostics, 13(3):492, 2023. 6, 7, 28, 30

[20] Kim, Min Kyu, Hong Jun Jeon, Se Hyuck Park, Dong Sik Park e Hee Seung Nam: *Value of ultrasonography in the diagnosis of carpal tunnel syndrome: Correlation with electrophysiological abnormalities and clinical severity.* Journal of Korean Neurosurgical Society, 55(2):78–82, 2014. 6

[21] Peng, Jiayu, Jiajun Zeng, Manlin Lai, Ruobing Huang, Dong Ni e Zhenzhou Li: *One-stop automated diagnostic system for carpal tunnel syndrome in ultrasound images using deep learning.* Ultrasound in Medicine & Biology, 50(2):304–314, 2024, ISSN 0301-5629. https://www.sciencedirect.com/science/article/pii/S030156292300340X. 7

[22] Simon, Benjamin D., Kutsev B. Ozyoruk, David G. Gelikman, Stephanie A. Harmon e Barış Türkbey: *The future of multimodal artificial intelligence models for integrating imaging and clinical metadata: a narrative review.* Diagnostic and Interventional Radiology, 31(4):303–312, 2025. 8

[23] Moser, Florentin, Sébastien Muller, Torgrim Lie, Thomas Langø e Mari Hoff: *Automated segmentation of the median nerve in patients with carpal tunnel syndrome.* Scientific Reports, 14(1):16757, 2024, ISSN 2045-2322. https://doi.org/10.1038/s41598-024-65840-5. 8

[24] Necula, Sabina Cristiana: *Exploring the model–view–controller (mvc) architecture: A broad analysis of market and technological applications.* Preprints, (202404.1860), 2024. Systematic review highlighting widespread MVC adoption across Java, .NET and JavaScript ecosystems. 8

[25] Gamma, Erich, Richard Helm, Ralph Johnson e John Vlissides: *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994. 9

[26] Fielding, Roy T.: *Architectural Styles and the Design of Network-based Software Architectures.* Ph.d. dissertation, University of California, Irvine, 2000. https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. 9

[27] Richardson, Leonard e Sam Ruby: *RESTful Web Services.* O'Reilly Media, 2008. 10

[28] Mucci, Tim e Cole Stryker: *O que é mlops?* https://www.ibm.com/br-pt/topics/mlops. Acesso em: 15 abr. 2025. 10

[29] Eken, Beyza, Samodha Pallewatta, Nguyen Khoi Tran, Ayse Tosun e Muhammad Ali Babar: *A multivocal review of mlops practices, challenges and open issues.* arXiv preprint, arXiv:2406.09737, 2024. https://arxiv.org/abs/2406.09737. 10

[30] Mohri, Mehryar, Afshin Rostamizadeh e Ameet Talwalkar: *Foundations of Machine Learning.* MIT Press, 2nd edição, 2018. 10, 11

[31] Pichler, Maximilian e Florian Hartig: *Machine learning and deep learning—a review for ecologists.* Methods in Ecology and Evolution, 14:994–1016, 2023. 10

[32] Fawcett, Tom: *An introduction to roc analysis.* Pattern Recognition Letters, 27(8):861–874, 2006. 11, 20, 22, 24

[33] Gulshan, Varun, Lily Peng, Marc Coram, Martin C. Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, Rajiv Raman, Philip C. Nelson, Jessica L. Mega e Dale R. Webster: *Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs*. JAMA, 316(22):2402–2410, 2016. 12

[34] Tang, Lei, Ye Li, Ji Zhang, Feng Zhang, Qiaoling Tang, Xiangbin Zhang, Sai Wang, Yupeng Zhang, Siyuan Ma, Ran Liu, Lei Chen, Junyi Ma, Xuelun Zou, Tianxing Yao, Rongmei Tang, Huifang Zhou, Lianxu Wu, Yexiang Yi, Yi Zeng, Duolao Wang e Le Zhang: *Machine learning model to predict sepsis in icu patients with intracerebral hemorrhage*. Scientific Reports, 15:16326, 2025. 12

[35] Yeghaian, Melda, Zuhir Bodalal, Daan van den Broek, John B. A. G. Haanen, Regina G. H. Beets-Tan, Stefano Trebeschi e Marcel A. J. van Gerven: *Multimodal integration of longitudinal noninvasive diagnostics for survival prediction in immunotherapy using deep learning*. Journal of the American Medical Informatics Association, 2025. Pre-print available at textttarXiv:2411.18253. 12

[36] Quinlan, J. R.: *Induction of decision trees*. Machine Learning, 1(1):81–106, 1986. 12

[37] Hastie, Trevor, Robert Tibshirani e Jerome Friedman: *The Elements of Statistical Learning*. Springer, 2nd edição, 2009. 12, 13, 14, 42

[38] Géron, Aurélien: *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, 2nd edição, 2019. 12, 14

[39] Breiman, Leo: *Random forests*. Machine Learning, 45(1):5–32, 2001. 12, 13

[40] Friedman, Jerome H.: *Greedy function approximation: A gradient boosting machine*. Annals of Statistics, 29(5):1189–1232, 2001. 12, 13

[41] Chen, Tianqi e Carlos Guestrin: *Xgboost: A scalable tree boosting system*. Em *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, páginas 785–794, 2016. 12, 13, 42

[42] Natekin, Alexey e Alois Knoll: *Gradient boosting machines, a tutorial*. Frontiers in Neurorobotics, 7:21, 2013. 13

[43] Shwartz, Matthew, Alex Yeo, Alan Wang e Honghuang Chen: *Application of xgboost in medical data analysis and prediction*. Healthcare Informatics Research, 28(1):3–12, 2022. 13

[44] Cortes, Corinna e Vladimir Vapnik: *Support-vector networks*. Machine Learning, 20(3):273–297, 1995. 14

[45] Cover, Thomas M. e Peter E. Hart: *Nearest neighbor pattern classification*. IEEE Transactions on Information Theory, 13(1):21–27, 1967. 14

[46] Goodfellow, Ian, Yoshua Bengio e Aaron Courville: *Deep Learning*. MIT Press, 2016. 14, 15, 16, 42

[47] Ruder, Sebastian: *An overview of gradient descent optimization algorithms*. arXiv preprint arXiv:1609.04747, 2016. 14, 17

[48] Bishop, Christopher M.: *Pattern Recognition and Machine Learning*. Springer, 2006, ISBN 978-0387310732. 15

[49] Kunc, Vladimír e Jiří Kléma: *Three decades of activations: A comprehensive survey of 400 activation functions for neural networks*. arXiv preprint, arXiv:2402.09092, 2024. `https://arxiv.org/abs/2402.09092`. 15

[50] Nair, Vinod e Geoffrey E. Hinton: *Rectified linear units improve restricted boltzmann machines*. Em *Proceedings of the 27th International Conference on Machine Learning (ICML)*, páginas 807–814, 2010. 15

[51] Maas, Andrew L., Awni Y. Hannun e Andrew Y. Ng: *Rectifier nonlinearities improve neural network acoustic models*. Em *Proceedings of the 30th International Conference on Machine Learning (ICML)*, páginas 3–11, 2013. `https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf`, Introduz e avalia a *Leaky ReLU* (LReLU) como alternativa ao ReLU convencional em modelos acústicos. 16

[52] Clevert, Djork Arné, Thomas Unterthiner e Sepp Hochreiter: *Fast and accurate deep network learning by exponential linear units (elus)*. International Conference on Learning Representations (ICLR), 2016. `https://arxiv.org/abs/1511.07289`, Propõe a *Exponential Linear Unit* (ELU) e demonstra ganhos de velocidade e acurácia em redes profundas. 16

[53] He, Kaiming, Xiangyu Zhang, Shaoqing Ren e Jian Sun: *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*. Em *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, páginas 1026–1034, 2015. `https://arxiv.org/abs/1502.01852`, Apresenta a *Parametric ReLU* (PReLU) e alcança desempenho recorde no ImageNet. 16

[54] Terven, Juan, Diana Margarita Cordova-Esparza, Julio Alejandro Romero-González, Alfonso Ramírez-Pedraza e E. A. Chávez-Urbiola: *A comprehensive survey of loss functions and metrics in deep learning*. Artificial Intelligence Review, 58(195), 2025. 16, 17

[55] Abdulkadirov, Ruslan, Pavel Lyakhov e Nikolay Nagornov: *Survey of optimization algorithms in modern neural networks*. Mathematics, 11(11):2466, 2023. 17

[56] Kingma, Diederik P. e Jimmy Ba: *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980, 2014. `https://arxiv.org/abs/1412.6980`. 18

[57] Loshchilov, Ilya e Frank Hutter: *Sgdr: Stochastic gradient descent with warm restarts*. arXiv preprint arXiv:1608.03983, 2016. `https://arxiv.org/abs/1608.03983`. 18

[58] He, Haibo e Edward A. Garcia: *Learning from imbalanced data.* IEEE Transactions on Knowledge and Data Engineering, 21(9):1263–1284, 2009. 19

[59] Salmi, Mabrouka, Dalia Atif, Diego Oliva, Ajith Abraham e Sebastian Ventura: *Handling imbalanced medical datasets: Review of a decade of research.* Artificial Intelligence Review, 57:273, 2024. 19

[60] Carvalho, Miguel, Armando J. Pinho e Susana Brás: *Resampling approaches to handle class imbalance: A review from a data perspective.* Journal of Big Data, 12:71, 2025. 19

[61] He, Haibo e Edward A. Garcia: *Learning from imbalanced data.* IEEE Transactions on Knowledge and Data Engineering, 21(9):1263–1284, 2009. 19, 41

[62] Chen, Wuxing, Kaixiang Yang, Zhiwen Yu, Yifan Shi e C. L. Philip Chen: *A survey on imbalanced learning: Latest research, applications and future directions.* Artificial Intelligence Review, 57:137, 2024. 19

[63] Vargas, Vitor Werner de, Jorge Arthur Schneider Aranda, Ricardo dos Santos Costa, Paulo Ricardo da Silva Pereira e Jorge Luis Victória Barbosa: *Imbalanced data preprocessing techniques for machine learning: a systematic mapping study.* Knowledge and Information Systems, 65:31–57, 2023. `https://doi.org/10.1007/s10115-022-01772-8`. 19

[64] Shorten, Connor e Taghi M. Khoshgoftaar: *A survey on image data augmentation for deep learning.* Journal of Big Data, 6(1):60, 2019. 19

[65] Xu, Lei, Maria Skoularidou, Alfredo Cuesta-Infante e Kalyan Veeramachaneni: *Modeling tabular data using conditional gan.* Em *Advances in Neural Information Processing Systems (NeurIPS)*, páginas 7333–7343, 2019. `https://proceedings.neurips.cc/paper/2019/file/254ed7d2de3e4ef5f6b1165b3f1f53c3-Paper.pdf`. 19

[66] Sokolova, Milena e Guy Lapalme: *A systematic analysis of performance measures for classification tasks.* Information Processing & Management, 45(4):427–437, 2009. 22, 23

[67] Tharwat, Alaa: *Classification assessment methods.* Applied Computing and Informatics, 17(1):168–192, 2021. 22, 24

[68] Powers, David M. W.: *Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation.* Journal of Machine Learning Technologies, 2(1):37–63, 2011. 23

[69] Sasaki, Yutaka: *The truth of the F-measure.* Em *Proceedings of the ICDAR Workshop on Evaluating Information Retrieval Systems*, 2007. 23

[70] Hand, David J. e Richard J. Till: *A simple generalisation of the area under the ROC curve for multiple class classification problems.* Machine Learning, 45(2):171–186, 2001. 24

[71] Neyman, Jerzy: *Outline of a theory of statistical estimation based on the classical theory of probability.* Philosophical Transactions of the Royal Society A, 236:333–380, 1937. 25

[72] Cumming, Geoff: *Replication and p intervals: p-values predict the future only vaguely, but confidence intervals do much better.* Perspectives on Psychological Science, 3(4):286–300, 2008. 25

[73] *Coefficient of variation: Definition and how to use it.* https://www.investopedia.com/terms/c/coefficientofvariation.asp. Accessed July 2025. 25

[74] Eurostat: *Beginners: Statistical concept – percentage change and percentage points.* Relatório Técnico, 2024. 26

[75] Tsamis, Konstantinos I., Prokopis Kontogiannis, Ioannis Gourgiotis, Stefanos Ntabos, Ioannis Sarmas e George Manis: *Automatic electrodiagnosis of carpal tunnel syndrome using machine learning.* Bioengineering, 8(11), 2021, ISSN 2306-5354. https://www.mdpi.com/2306-5354/8/11/181. 27, 30

[76] Harrison, Conrad J., Luke Geoghegan, Chris J. Sidey-Gibbons, Paul H. C. Stirling, Jane E. McEachan e Jeremy N. Rodrigues: *Developing machine learning algorithms to support patient-centered, value-based carpal tunnel decompression surgery.* Plastic and Reconstructive Surgery – Global Open, 10(4), 2022. https://journals.lww.com/prsgo/fulltext/2022/04000/developing_machine_learning_algorithms_to_support.35.aspx, ID: 01720096-202204000-00035. 28, 30

[77] Smerilli, Gianluca, Edoardo Cipolletta, Gianmarco Sartini, Erica Moscioni, Mariachiara Di Cosmo, Maria Chiara Fiorentino, Sara Moccia, Emanuele Frontoni, Walter Grassi e Emilio Filippucci: *Development of a convolutional neural network for the identification and the measurement of the median nerve on ultrasound images acquired at carpal tunnel level.* Arthritis Research Therapy, 24(1):38, 2022, ISSN 1478-6362. https://doi.org/10.1186/s13075-022-02729-6, Published: 2022/02/08. 28, 30

[78] Elseddik, Marwa, Khaled Alnowaiser, Reham R. Mostafa, Ahmed Elashry, Nora El-Rashidy, Shimaa Elgamal, Ahmed Aboelfetouh e Hazem El-Bakry: *Deep learning-based approaches for enhanced diagnosis and comprehensive understanding of carpal tunnel syndrome.* Diagnostics, 13(20), 2023, ISSN 2075-4418. https://www.mdpi.com/2075-4418/13/20/3211. 29, 30

[79] Kreuzberger, Dominik, Niklas Kühl e Sebastian Hirschl: *Machine learning operations (mlops): Overview, definition, and architecture.* IEEE Access, 11:31866–31879, 2023. 31

[80] Shethiya, Aditya S.: *Deploying ai models in .net web applications using azure kubernetes service (aks).* Spectrum of Research, 5(1), 2025. https://spectrumofresearch.com/index.php/sr/article/view/13, Acesso em: 15 abr. 2025. 32, 33, 47

[81] Aghda, Ali Key, Mohammad Asheghan e Ali Amanollahi: *Comparisons of electrophysiological and clinical findings between young and elderly patients with carpal tunnel syndrome*. Revue Neurologique, 176(5):387–392, 2020. 36

[82] Grönfors, Henri, Sari-Leena Himanen, Lauri Martikkala, Mika Kallio e Katri Mäkelä: *Median nerve ultrasound cross-sectional area and wrist-to-forearm ratio in relation to carpal tunnel syndrome related axonal damage and patient age*. Clinical Neurophysiology Practice, 8:81–87, 2023. 36

[83] Cazares-Manríquez, Melissa Airem, Claudia Camargo Wilson, Ricardo Vardasca, Jorge Luis García-Alcaraz, Jesús Everardo Olguín-Tiznado, Juan Andrés López-Barreras e Blanca Rosa García-Rivera: *A review of carpal tunnel syndrome and its association with age, body mass index, cardiovascular risk factors, hand dominance, and sex*. Applied Sciences, 10(10), 2020, ISSN 2076-3417. `https://www.mdpi.com/2076-3417/10/10/3488`. 36, 37

[84] Lee, H., J., K. Kwon, H., H. Kim, D. e B. Pyun, S.: *Nerve conduction studies of median motor and sensory branches according to the severity of carpal tunnel syndrome*. Annals of Rehabilitation Medicine, 37:254–262, 2013. 37

[85] Lin, Ting Yu, Ke Vin Chang, Wei Ting Wu e Levent Özçakar: *Ultrasonography for the diagnosis of carpal tunnel syndrome: An umbrella review*. Journal of Neurology, 269(9):4663–4675, 2022. 37

[86] Casas, María P. Murciano, María Rodríguez Piñero Durán, José M. Delgado Mendilivar, Juan A. Expósito Tirado e Ana S. Jiménez Sarmiento: *Análisis de los parámetros ecográficos descritos en el estudio del síndrome del túnel carpiano: Revisión sistemática*. Rehabilitación, 58(1):100822, 2024. 37

[87] Braham, S., R. de Freitas, T. Silva, A. Neto e L. Santos: *Exploring ultrasound and electromyography for carpal tunnel syndrome diagnosis: a comprehensive comparative study and implications for occupational medicine*. Frontiers in Neurology, 15:1490873, 2024. 37

[88] Ng, A. W. H., J. F. Griffith, R. K. L. Lee, W. L. Tse, C. W. Y. Wong e P. C. Ho: *Ultrasound carpal tunnel syndrome: Additional criteria for diagnosis*. Clinical Radiology, 73(2):214.e11–214.e18, 2018. 37

[89] Upadhyaya, V. e H. N. Choudur: *Revisiting ultrasound assessment of the median nerve in carpal tunnel syndrome: A review*. Indian Journal of Musculoskeletal Radiology, 6(2):88–94, 2024. 37

[90] Padua, Luca, Marco Lo Monaco, Raffaele Padua, Bruno Gregori e Pietro Tonali: *Neurophysiological classification of carpal tunnel syndrome: Assessment of 600 symptomatic hands*. Italian Journal of Neurological Sciences, 18(3):145–150, 1997. 38

[91] Bland, Jeremy D.: *A neurophysiological grading scale for carpal tunnel syndrome*. Muscle & Nerve, 23(8):1280–1283, 2000. 38

[92] Jablecki, Charles K., Michael T. Andary, Mary K. Floeter, Robert G. Miller, Catherine A. Quartly, Michael J. Vennix e James R. Wilson: *Practice parameter: Electrodiagnostic studies in carpal tunnel syndrome. report of the american association of electrodiagnostic medicine, american academy of neurology, and the american academy of physical medicine and rehabilitation.* Neurology, 58(11):1589–1592, 2002. 38

[93] American Academy of Orthopaedic Surgeons: *Clinical practice guideline on the diagnosis of carpal tunnel syndrome.* `http://www.aaos.org/research/guidelines/CTS_guideline.pdf`, 2007. Approved May 2007. 38

[94] Keith, Michael W., Victoria Masear, Kevin Chung, Kent Maupin, Michael Andary, Peter C. Amadio, Richard W. Barth, William C. Watters, Michael J. Goldberg, Robert H. Haralson, Charles M. Turkelson e Janet L. Wies: *Diagnosis of carpal tunnel syndrome.* Journal of the American Academy of Orthopaedic Surgeons, 17(6):389–396, 2009. 38

[95] Vabalas, Andrius, Emma Gowen, Ellen Poliakoff e Alexander J. Casson: *Machine learning algorithm validation with a limited sample size.* PLOS ONE, 14(11):e0224365, 2019. 38

[96] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay: *Scikit-learn: Machine learning in python.* Journal of Machine Learning Research, 12:2825–2830, 2011. `https://jmlr.org/papers/v12/pedregosa11a.html`. 39

[97] Alzubaidi, Laith, Jinshuai Bai, Aiman Al-Sabaawi, José Santamaría, A. S. Albahri, Bashar Sami Nayyef Al-Dabbagh, Mohammed A. Fadhel, Mohamed Manoufali, Jinglan Zhang, Ali H. Al-Timemy, Ye Duan, Amjed Abdullah, Laith Farhan e Felix Albu: *A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications.* Journal of Big Data, 10:46, 2023. `https://journalofbigdata.springeropen.com/articles/10.1186/s40537-023-00727-2`. 41

[98] Rather, Ishfaq Hussain, Sushil Kumar e Amir H. Gandomi: *Breaking the data barrier: a review of deep learning techniques for democratizing AI with small datasets.* Artificial Intelligence Review, 57:226, 2024. `https://doi.org/10.1007/s10462-024-10859-3`. 41

[99] Yousef, Malik e Jens Allmer: *Deep learning in bioinformatics.* Turkish Journal of Biology, 47(6):366–382, 2023. `https://doi.org/10.55730/1300-0152.2671`. 41

[100] Johnson, Justin M. e Taghi M. Khoshgoftaar: *Survey on deep learning with class imbalance.* Journal of Big Data, 6:27, 2019. 41

[101] Chen, Chao, Andy Liaw e Leo Breiman: *Using random forest to learn imbalanced data.* Technical report 666, Department of Statistics, University of California, Berkeley, 2004. `https://statistics.berkeley.edu/tech-reports/666`. 41

[102] Pressman, Roger S. e Bruce R. Maxim: *Engenharia de Software: uma abordagem profissional*. McGraw-Hill Brasil, São Paulo, 7ª edição, 2014. 50