

**UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”
CAMPUS DE ILHA SOLTEIRA**

Igor Feliciani Merizio

**TÉCNICAS DE *MACHINE LEARNING* PARA DETECÇÃO DE FALHAS
EM SISTEMAS MECÂNICOS ROTATIVOS**

Ilha Solteira - SP
2025



PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Igor Feliciani Merizio

**TÉCNICAS DE *MACHINE LEARNING* PARA DETECÇÃO DE FALHAS
EM SISTEMAS MECÂNICOS ROTATIVOS**

Dissertação apresentada à Universidade Estadual Paulista (UNESP), Faculdade de Engenharia de Ilha Solteira, Ilha Solteira, para obtenção do título de Mestre em Engenharia Mecânica.

Área de Concentração: Mecânica dos Sólidos

Orientador: Prof. Dr. Fábio Roberto Chavarrette

Ilha Solteira - SP
2025



FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

M563t Merizio, Igor Feliciani.
Técnicas de machine learning para detecção de falhas em sistemas mecânicos rotativos / Igor Feliciani Merizio. -- Ilha Solteira: [s.n.], 2025
67 f. : il.

Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Mecânica dos Sólidos, 2025

Orientador: Fábio Roberto Chavarette

Inclui bibliografia

1. Monitoramento da integridade estrutural. 2. Machine learning. 3. Data-driven. 4. Detecção de falhas. 5. Comparação de modelos.

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: TÉCNICAS DE MACHINE LEARNING PARA DETECÇÃO DE FALHAS EM SISTEMAS MECÂNICOS ROTATIVOS

AUTOR: IGOR FELICIANI MERIZIO

ORIENTADOR: FABIO ROBERTO CHAVARETTE

Aprovado como parte das exigências para obtenção do Título de Mestre em Engenharia Mecânica, área: Mecânica dos Sólidos pela Comissão Examinadora:

Prof. Dr. FABIO ROBERTO CHAVARETTE (Participação Virtual)
Departamento de Engenharia Física e Matemática / Instituto de Química - UNESP

Profa. Dra. ERICA REGINA FILLETTI NASCIMENTO (Participação Virtual)
Departamento de Engenharia, Física e Matemática / Instituto de Química - UNESP

Prof. Dr. ÉDER LUIZ OLIVEIRA (Participação Virtual)
Departamento de Engenharia Aeronáutica / Faculdade de Engenharia de São João - UNESP

Ilha Solteira, 06 de junho de 2025

*Este trabalho é dedicado a todos que me apoiaram durante minha formação,
e aqueles que trilharam esse caminho comigo.*



AGRADECIMENTOS

Agradeço aos meus pais, Iris Patricia Feliciani Merizio e Edmo Merizio (In memoriam), que junto a meus avós, Antônio Feliciani e Marilene Aparecida Gomes Feliciani, não mediram esforços para educar meu irmão e eu, desde a criação até os dias atuais. Sempre me apoiando, em todos os âmbitos da vida. Muito obrigado.

A minha noiva, Giovanna Ceola Ferreira, por todo companheirismo, paciência, compreensão, amor e apoio em todos esses anos.

Ao meu orientador, Prof. Dr. Fábio Roberto Chavarette, por ser mais que um orientador. Um amigo em quem posso confiar. Por toda orientação e conselho, na vida pessoal e profissional, e por agregar, de forma incrível, na minha formação.

A todas as amizades feitas nesse trajeto. Em especial, meus amigos de turma, os quais tive o privilégio de compartilhar diversas horas de estudo sobre inúmeros temas, e me apoiaram nessa caminhada. E aos meus amigos do Laboratório SisPlexos, uma segunda família dentro da faculdade.

Faltam palavras para expressar minha gratidão a essas pessoas, que têm papéis únicos em minha vida, e o quanto especiais elas são para mim.

A todos os meus professores e à Unesp pelo suporte.





*“Você é espectador de si mesmo.
- Clovís de Barros Filho*

RESUMO

Este estudo aborda a detecção de falhas em sistemas rotativos por meio de uma análise comparativa de três algoritmos de aprendizado de máquina: Redes Neurais Artificiais (RNA), *Random Forest* (RF) e XGBoost. O diferencial deste trabalho reside na robustez da metodologia aplicada, que abrange a extração de 29 *features* estatísticas nos domínios do tempo e da frequência, otimização sistemática de hiperparâmetros via *Grid Search* e validação rigorosa por meio de validação cruzada com 10 *folds* e conjunto de testes independente. Utilizando o conjunto de dados da Case Western Reserve University, coletado a uma taxa de amostragem de 48 kHz com falhas induzidas em elementos rolantes, pistas internas e externas, o estudo garantiu a confiabilidade e a generalização dos resultados. O foco principal foi o avanço metodológico, proporcionando uma comparação justa entre os modelos, o que possibilita sua aplicação em diferentes contextos industriais. O XGBoost destacou-se como o modelo com melhor desempenho, atingindo 95,2% de acurácia no conjunto de teste, com 92,5% de precisão e 95,5% de *recall*. Apesar disso, as contribuições deste trabalho vão além dos resultados individuais dos modelos, ao estabelecer uma abordagem sistemática para treinar, otimizar e avaliar algoritmos de aprendizado de máquina, assegurando a replicabilidade e a adaptabilidade dos métodos. O uso de parâmetros como média, variância, curtose e assimetria no domínio do tempo, além de métricas como frequência RMS e curtose espectral no domínio da frequência, permitiu identificar padrões sutis de falha. Por fim, esta pesquisa contribui significativamente para o avanço de estratégias de manutenção preditiva na Indústria 4.0, oferecendo uma metodologia robusta e generalizável que estabelece novos padrões para diagnósticos industriais baseados em aprendizado de máquina.

Palavras-chaves: monitoramento da integridade estrutural; machine learning; data-driven; detecção de falhas; comparação de modelos.



ABSTRACT

This study advances the field of fault detection in rotating mechanical systems by introducing a robust methodological framework for model training and evaluation, ensuring reliable generalization of results to diverse datasets. Three state-of-the-art machine learning algorithms—Artificial Neural Networks (ANNs), Random Forest (RF), and XGBoost—were rigorously analyzed using vibration data from defective bearings. The methodology involved the extraction of 29 statistical features from time and frequency domains, systematic hyperparameter optimization through Grid Search, and a comprehensive evaluation via 10-fold cross-validation combined with a separate test dataset. This framework guarantees fairness in comparisons and minimizes overfitting, enabling reliable performance assessments across different operational conditions. Data from the Case Western Reserve University dataset, sampled at 48 kHz, were used to capture faults induced in rolling elements, inner races, and outer races under controlled conditions. The results demonstrated that XGBoost achieved the best balance between accuracy (95.2%), precision (92.5%), and recall (95.5%), while Random Forest and Neural Networks also performed well, achieving accuracies of 93.2% and 94.1%, respectively. The proposed methodology goes beyond mere model comparison by emphasizing robust feature engineering, leveraging critical statistical parameters from time and frequency domains, and ensuring fair hyperparameter tuning through GridSearchCV. By addressing key challenges in the reproducibility and generalization of machine learning models, this study establishes a methodological benchmark for fault detection in rotating systems. The findings highlight not only the superior performance of XGBoost but also the reliability of the proposed framework in facilitating predictive maintenance within Industry 4.0. This research contributes significantly to the advancement of diagnostic tools by setting a new standard for the evaluation and generalization of machine learning algorithms in industrial applications.

Keywords: structural health monitoring; Machine Learning; Data-Driven; Fault Detection; Model comparison.



LISTA DE FIGURAS

Figura 1 – Fluxo de informações em uma Rede Neural Artificial.	24
Figura 2 – Exemplo da técnica de validação cruzada, com $K = 4$, utilizando 100 exemplos gerados aleatoriamente, pertencentes a duas classes e a 10 indivíduos.	54
Figura 3 – <i>Precision e Recall vs Threshold - Random Forest</i>	57
Figura 4 – <i>Precision e Recall vs Threshold - XGBoost</i>	57
Figura 5 – <i>Precision e Recall vs Threshold - Redes Neurais</i>	58
Figura 6 – Comprovante de publicação na revista <i>Engineering Computations</i>	63
Figura 7 – E-mail de aceite de publicação da revista <i>Trends</i>	63



LISTA DE TABELAS

Tabela 1	–	Parâmetros estatísticos no domínio do tempo.	47
Tabela 2	–	Parâmetros estatísticos no domínio da frequência.	48
Tabela 3	–	Hiperparâmetros testados para a otimização do <i>Random Forest</i>	49
Tabela 4	–	Hiperparâmetros testados para a otimização da Rede Neural.	50
Tabela 5	–	Hiperparâmetros testados para a otimização do XGBoost.	50
Tabela 6	–	Métricas de avaliação na validação cruzada <i>k-fold</i>	55
Tabela 7	–	Métricas de avaliação no conjunto de teste	56



SUMÁRIO

1	INTRODUÇÃO	15
1.1	Contexto e motivação	15
1.2	Objetivos	17
1.3	Estrutura do trabalho	17
2	REVISÃO BIBLIOGRÁFICA	19
2.1	Fundamentos Gerais dos Algoritmos	19
2.1.1	Manutenção Preditiva no Contexto da Indústria 4.0	20
2.1.2	Redes Neurais Artificiais (ANNs)	20
2.1.3	<i>Random Forest</i> (RF)	21
2.1.4	XGBoost	22
2.2	Funcionamento das Redes Neurais Artificiais (ANNs)	23
2.2.1	Introdução às Redes Neurais Artificiais	23
2.2.1.1	Fluxo de Informações em Redes Neurais	23
2.2.2	Arquitetura das Redes Neurais	24
2.2.2.1	Estrutura Básica	24
2.2.2.2	Topologias das Redes	25
2.2.3	Aprendizado das Redes Neurais	25
2.2.3.1	Operações nos Neurônios	25
2.2.3.2	Propagação Direta	26
2.2.3.3	Funções de Ativação	26
2.2.3.3.1	Importância das Funções de Ativação	29
2.2.3.4	Função de Custo	30
2.2.3.5	Gradiente Descendente	31
2.2.3.5.1	Conceito de Gradiente	31
2.2.3.5.2	Algoritmo do Gradiente Descendente	31
2.2.3.5.3	Taxa de Aprendizado	32
2.2.3.5.4	Variantes do Gradiente Descendente	32
2.2.3.5.5	Aplicação em Redes Neurais	32
2.2.4	Treinamento e Generalização	33
2.2.5	Aplicações Práticas	33
2.3	Funcionamento da <i>Random Forest</i>	33
2.3.1	Definição e Fundamentos	33
2.3.2	Ensemble Learning no <i>Random Forest</i>	34

2.3.2.1	Redução de Variância	34
2.3.3	Bagging e Bootstrap Sampling	34
2.3.4	Seleção de Variáveis e Critérios de Divisão	34
2.3.4.1	Subconjunto Aleatório de Variáveis	35
2.3.4.2	Critério de Divisão	35
2.3.4.2.1	Processo de Avaliação:	36
2.3.4.2.2	Importância dos Critérios de Divisão:	36
2.3.5	Interdependência entre Seleção de Variáveis e Critérios de Divisão . .	36
2.3.6	Métricas para Avaliação de Divisões	36
2.3.6.1	Entropia	36
2.3.6.2	Índice de Gini	37
2.3.6.3	Comparação entre Entropia e Índice de Gini	37
2.3.7	Algoritmo de Construção de Árvores no <i>Random Forest</i>	38
2.3.7.1	Critérios de Parada	38
2.3.8	Importância de Variáveis	38
2.4	Funcionamento do XGBoost	39
2.4.1	Visão Geral	39
2.4.2	Funcionamento do XGBoost	39
2.4.2.1	Estrutura Matemática do Gradient Boosting	39
2.4.2.2	Objetivo de Otimização	40
2.4.2.3	Aproximação por Taylor de Segunda Ordem	40
2.4.2.4	Divisão de Nós e Ganho de Informação	40
2.4.3	Características Avançadas do XGBoost	40
2.4.3.1	Regularização	40
2.4.3.2	Paralelismo	42
2.4.3.3	Redução de Sobrecarga Computacional	42
2.4.4	Aplicações Práticas	42
2.5	Noções sobre Estatística e Densidade Espectral	43
2.5.1	Média Aritmética	43
2.5.2	Valor Esperado	43
2.5.3	Desvio Padrão	44
2.5.4	Normalização	44
2.5.5	F_1 -Score	44
2.5.6	Densidade Espectral de Potência	45
2.5.7	Valor-t e Teste t	45
2.5.7.1	Distribuição t-Student	45
3	MATERIAIS E MÉTODOS	46
3.1	Aquisição de Dados e Configuração Experimental	46

3.2	Extração de Features	47
3.3	Treinamento e Otimização	48
3.3.1	Otimização da <i>Random Forest</i>	49
3.3.2	Otimização das Redes Neurais	49
3.3.3	Otimização do XGBoost	50
3.4	Métricas de Avaliação e Metodologia de Desempenho	50
3.4.1	Acurácia	51
3.4.2	Precisão (Precision)	51
3.4.3	Sensibilidade (Recall)	51
3.4.4	Especificidade	51
3.4.5	F1-Score	52
3.4.6	Curva ROC e AUC (Área Sob a Curva)	52
3.4.7	<i>Log Loss</i>	52
3.4.8	Validação Cruzada e Métodos de Avaliação	53
4	RESULTADOS E DISCUSSÕES	55
4.1	Resultados dos modelos comparados	55
4.1.1	<i>Random Forest</i>	55
4.1.2	XGBoost	56
4.1.3	Redes Neurais	58
4.1.4	Comparação com Estudos Existentes	59
4.1.5	Análise Estatística das Comparações de Modelos	60
5	CONSIDERAÇÕES FINAIS	61
5.1	Artigos	62
5.1.1	Artigo publicado	62
5.1.2	Artigo aceito para publicação	63
	REFERÊNCIAS	64



1 INTRODUÇÃO

Este capítulo apresenta o contexto e a motivação do trabalho através de uma breve abordagem da literatura; apresenta também o problema envolvendo a detecção de danos em estruturas e a sua proposta de solução utilizando técnicas de *Machine Learning*; apresenta os objetivos do trabalho; suas principais contribuições; e finaliza com a estrutura de divisão dos capítulos.

1.1 Contexto e motivação

A detecção de falhas em sistemas mecânicos rotativos representa um domínio crítico na engenharia mecânica, particularmente no contexto de manutenção preditiva e aprimoramento da eficiência operacional. A evolução da engenharia mecânica através das sucessivas revoluções industriais culminou na era atual da Indústria 4.0, caracterizada pela integração profunda de inteligência artificial (AI) e técnicas de *Machine Learning* (ML). Essas tecnologias aumentaram substancialmente a capacidade de monitorar e diagnosticar falhas em sistemas complexos, promovendo avanços significativos na área. Historicamente, as metodologias de detecção de falhas dependiam predominantemente de inspeções visuais e análises manuais. O advento de tecnologias modernas, como a Internet das Coisas (IoT), sistemas ciberfísicos (CPS) e computação em nuvem, revolucionou os processos de aquisição, processamento e análise de dados, permitindo monitoramento em tempo real e identificação precoce de anomalias, reduzindo o risco de falhas catastróficas. Sistemas de Monitoramento da Saúde Estrutural (SHM) tornaram-se instrumentos fundamentais em estruturas de manutenção preditiva, capacitando engenheiros a agir proativamente em relação a falhas (Chu, Chew e Chang 2022, Sardashti e Nazari 2023).

A integração de técnicas de *Machine Learning* em sistemas de detecção de falhas demonstrou significativo potencial. Redes Neurais Artificiais (ANNs) têm sido amplamente empregadas devido à sua capacidade de identificar padrões complexos em dados de sensores indicativos de falhas. Em paralelo, algoritmos baseados em árvores, como *Random Forest* e XGBoost, surgiram como alternativas robustas, frequentemente superando o desempenho das ANNs em diversos contextos de diagnóstico (Zhang et al. 2020, Angelopoulos et al. 2019). O *Random Forest*, uma metodologia de aprendizado por conjuntos, aprimora a acurácia das previsões ao agregar múltiplas árvores de decisão, mitigando os riscos de sobreajuste. Sua resiliência a dados ruidosos e escalabilidade para grandes conjuntos de dados tornam-no particularmente adequado para aplicações de detecção de falhas (Grinsztajn, Oyallon e Varoquaux 2022). Arquiteturas de aprendizado profundo,

especialmente redes neurais convolucionais (CNNs) e redes neurais recorrentes (RNNs), também têm sido utilizadas para modelar relações não lineares e temporais em dados de vibração, demonstrando eficácia notável na identificação de indicadores sutis de degradação mecânica (Han, Liang e Shi 2020, Angelopoulos et al. 2019).

Avanços recentes destacaram a utilidade de abordagens híbridas e de AI explicável. Por exemplo, frameworks de aprendizado profundo com CNNs alcançaram melhorias notáveis em acurácia ao sintetizar diferentes conjuntos de features para diagnóstico de falhas (Jalonen et al. 2023). Além disso, modelos de AI explicável aprimoraram a interpretabilidade dos diagnósticos de *Machine Learning*, aumentando a confiança nesses sistemas ao elucidar os processos de tomada de decisão (Decker, Lebacher e Tresp 2023, Team 2023).

Este estudo avalia a eficácia de algoritmos avançados de *Machine Learning*, incluindo *Random Forest*, XGBoost e redes neurais, para detecção de falhas em sistemas rotativos. Analisando dados de vibração adquiridos por sinais de acelerômetros, este trabalho identifica padrões que precedem falhas mecânicas. Pesquisas anteriores demonstraram que técnicas de *Machine Learning* podem detectar variações sutis em sinais de vibração, frequentemente indicativas de falhas iniciais (Angelopoulos et al. 2019, Han, Liang e Shi 2020).

Cada algoritmo oferece vantagens e limitações distintas. O *Random Forest* é reconhecido por sua robustez e interpretabilidade, o XGBoost por sua eficiência computacional e acurácia, e as redes neurais por sua capacidade de modelar interações complexas e não lineares. Essas características os estabelecem como ferramentas essenciais no avanço de práticas de manutenção preditiva (Han et al. 2018, Angelopoulos et al. 2019, Grinsztajn, Oyallon e Varoquaux 2022).

A integração de metodologias baseadas em AI para detecção de falhas em sistemas rotativos representa uma mudança de paradigma na manutenção preditiva e no gerenciamento operacional. Ao empregar algoritmos como *Random Forest*, XGBoost e redes neurais, indústrias podem alcançar melhorias significativas na confiabilidade e eficiência de custos dos sistemas. Este estudo contribui para essas inovações ao fornecer uma comparação detalhada desses algoritmos de última geração, avaliados sob condições operacionais que replicam cenários reais. A análise incorpora técnicas avançadas de otimização de hiperparâmetros, especificamente *Grid Search* e *Randomized Search*, em conjunto com validação cruzada k-fold, para garantir avaliações de desempenho robustas e imparciais.

Esses refinamentos metodológicos oferecem insights valiosos sobre os pontos fortes e limitações de cada algoritmo, estabelecendo um padrão confiável para detecção de falhas baseada em AI em maquinário rotativo. Ao abordar questões críticas de padrões e reprodutibilidade destacadas em pesquisas anteriores, este estudo ressalta o potencial transformador da AI para melhorar ferramentas diagnósticas de sistemas mecânicos, reafirmando seu papel central na evolução de estratégias de manutenção preditiva.

O manuscrito é estruturado para fornecer uma análise comparativa rigorosa de três algoritmos proeminentes de *Machine Learning* na detecção de falhas: Redes Neurais Artificiais, *Random Forest* e XGBoost. A seção **Materiais e Métodos** detalha as técnicas de extração de features, os critérios de avaliação e os procedimentos de otimização de hiperparâmetros adotados para garantir uma comparação robusta e justa. A seção **Resultados** delinea o desempenho de cada algoritmo em um espectro de métricas, incluindo acurácia, precision, recall e F1 score. A seção **Discussão** contextualiza os achados na literatura mais ampla, enfatizando suas implicações para aplicações práticas na manutenção preditiva. Por fim, as **Conclusões** destacam as contribuições deste estudo e apontam direções para explorações futuras.

1.2 Objetivos

Este trabalho propõe uma metodologia robusta baseada em técnicas de *Machine Learning* para análise comparativa de algoritmos voltados à detecção de falhas em sistemas rotativos, garantindo resultados generalizáveis e confiáveis para diferentes conjuntos de dados. A metodologia foi desenvolvida com foco na extração de features, otimização de hiperparâmetros e avaliação imparcial dos modelos, visando estabelecer padrões metodológicos aplicáveis a cenários industriais reais.

Os objetivos gerais do presente trabalho são:

- Desenvolver uma abordagem metodológica robusta para o treinamento e avaliação de algoritmos de *Machine Learning* voltados à detecção de falhas;
- Realizar extração detalhada de 29 features estatísticas nos domínios do tempo e da frequência, destacando padrões críticos indicativos de falhas;
- Aplicar otimização de hiperparâmetros sistemática, utilizando *Grid Search* e validação cruzada k-fold, para garantir avaliações consistentes e imparciais dos modelos;
- Comparar o desempenho dos algoritmos XGBoost, *Random Forest* e Redes Neurais em termos de acurácia, precisão, recall e robustez operacional;
- Validar os resultados utilizando critérios estatísticos como *paired T-tests* para confirmar diferenças significativas entre os modelos em métricas como acurácia, *Log Loss* e AUC, assegurando a relevância estatística das análises realizadas.

1.3 Estrutura do trabalho

- **Capítulo 1 - Introdução:** Apresenta o contexto e a motivação do trabalho, enfatizando a importância da detecção de falhas em sistemas rotativos no contexto da Indústria 4.0. Este capítulo também expõe os objetivos do estudo e as principais contribuições para a área de manutenção preditiva e diagnóstico industrial.

- **Capítulo 2 - Revisão Bibliográfica:** Discute os principais conceitos relacionados aos algoritmos de aprendizado de máquina utilizados no trabalho, como Redes Neurais Artificiais, *Random Forest* e XGBoost. Também aborda fundamentos estatísticos e o conceito de densidade espectral de potência, que são fundamentais para a extração e análise de *features*.
- **Capítulo 3 - Materiais e Métodos:** Detalha o conjunto de dados utilizado, obtido da Case Western Reserve University, descreve os procedimentos experimentais para a coleta dos sinais de vibração e explica o processo de extração de *features* nos domínios do tempo e da frequência. Também aborda a metodologia de treinamento e validação dos modelos.
- **Capítulo 4 - Resultados e Discussões:** Apresenta uma análise detalhada dos resultados obtidos com os algoritmos *Random Forest*, XGBoost e Redes Neurais. Inclui comparações com estudos anteriores e análises estatísticas para validar as diferenças entre os modelos.
- **Capítulo 5 - Considerações Finais:** Resumo das principais contribuições do trabalho, com uma discussão sobre as implicações práticas dos resultados. Além disso, propõe direções futuras para a continuidade da pesquisa, incluindo o uso de novos algoritmos e a aplicação em diferentes cenários industriais.



2 REVISÃO BIBLIOGRÁFICA

A revisão bibliográfica é uma etapa fundamental para a contextualização e fundamentação teórica de qualquer pesquisa científica. Neste capítulo, são apresentados os conceitos centrais e os avanços mais recentes relacionados aos algoritmos de *Machine Learning* e seu papel na análise de dados complexos, com foco na detecção de falhas em sistemas rotativos. Além disso, são discutidos os principais desafios e oportunidades associados à aplicação dessas técnicas em cenários industriais, especialmente no contexto da Indústria 4.0.

2.1 Fundamentos Gerais dos Algoritmos

Os algoritmos de *Machine Learning* têm se consolidado como ferramentas indispensáveis em diversas áreas do conhecimento, devido à sua capacidade de processar grandes volumes de dados e identificar padrões complexos de forma automatizada. Esses algoritmos utilizam modelos matemáticos que aprendem a partir dos dados fornecidos, ajustando-se iterativamente para melhorar seu desempenho em tarefas específicas, como classificação, regressão e agrupamento.

Entre os algoritmos amplamente utilizados para análises de diagnóstico e manutenção preditiva, destacam-se:

- **Redes Neurais Artificiais (ANNs):** Inspiradas no funcionamento do cérebro humano, as ANNs são compostas por camadas de nós interconectados, conhecidos como neurônios artificiais. Elas são particularmente eficazes em modelar relações não lineares e complexas nos dados, tornando-se uma escolha frequente em problemas que envolvem variabilidade significativa e dados ruidosos.
- **Random Forest (RF):** Baseado no princípio do aprendizado em conjunto (*ensemble learning*), o RF utiliza múltiplas árvores de decisão para melhorar a precisão e reduzir o risco de *overfitting*. Este algoritmo é conhecido por sua robustez e capacidade de lidar com dados incompletos ou ruidosos.
- **XGBoost:** Uma evolução dos métodos de *boosting*, o XGBoost combina múltiplos modelos simples em uma estrutura sequencial para corrigir erros de predição anteriores. É altamente eficiente em termos computacionais e oferece alta acurácia, sendo amplamente adotado em competições de aprendizado de máquina e aplicações industriais.

Os algoritmos descritos acima foram amplamente pesquisados e aplicados em diversos campos, incluindo monitoramento de saúde estrutural (*Structural Health Monitoring - SHM*), onde são utilizados para detectar anomalias ou falhas em componentes mecânicos. O avanço desses métodos tem contribuído significativamente para a transição de abordagens tradicionais de manutenção corretiva para estratégias preditivas, alinhadas aos princípios da Indústria 4.0.

2.1.1 Manutenção Preditiva no Contexto da Indústria 4.0

A manutenção preditiva é uma abordagem que utiliza dados coletados em tempo real para prever falhas antes que elas ocorram. Diferentemente da manutenção corretiva, que só é realizada após uma falha, e da manutenção preventiva, baseada em cronogramas fixos, a manutenção preditiva emprega análises avançadas para maximizar a vida útil dos equipamentos, reduzir custos e minimizar paradas não planejadas.

Com o advento da Indústria 4.0, o uso de sensores inteligentes, *Internet of Things* (IoT) e análise de dados em larga escala revolucionou o campo da manutenção preditiva. Os algoritmos de *Machine Learning* desempenham um papel central nesse contexto, permitindo:

- A análise automática de grandes volumes de dados gerados por sensores instalados em equipamentos industriais.
- A identificação de padrões complexos que indicam o início de falhas ou degradação.
- A previsão do tempo restante antes da falha, auxiliando no planejamento eficiente de manutenções.

Ao longo deste capítulo, serão discutidos os fundamentos teóricos que sustentam os algoritmos mencionados, bem como suas vantagens e limitações em aplicações práticas. A análise detalhada desses métodos fornecerá uma visão abrangente sobre o estado da arte em detecção de falhas e sua relevância para o avanço da manutenção preditiva no cenário industrial contemporâneo.

2.1.2 Redes Neurais Artificiais (ANNs)

As Redes Neurais Artificiais (ANNs) são modelos computacionais inspirados em redes neurais biológicas, capazes de aprender e generalizar padrões complexos a partir de dados. Uma ANN típica consiste em uma camada de entrada, várias camadas ocultas e uma camada de saída. Cada camada é composta por neurônios artificiais que processam entradas e transmitem resultados para a próxima camada. Cada neurônio funciona como

uma função matemática, onde o valor de entrada é multiplicado por um peso, somado a um viés e processado por uma função de ativação para gerar a saída do neurônio, que será a entrada para o próximo.

Os principais hiperparâmetros de uma ANN incluem o número de camadas ocultas, o número de neurônios por camada, a taxa de aprendizado e a função de ativação utilizada. O número de camadas ocultas e neurônios determina a capacidade da rede de capturar padrões complexos. Redes mais profundas podem modelar interações complexas, mas correm o risco de *overfitting*. A taxa de aprendizado controla a velocidade de adaptação da rede durante o treinamento; taxas muito altas podem impedir a convergência, enquanto taxas muito baixas podem tornar o treinamento excessivamente lento (Hinton, Osindero e Teh 2006, Schmidhuber 2015).

Funções de ativação comuns incluem ReLU (*Rectified Linear Unit*), Sigmoid e Tanh. A função ReLU é amplamente utilizada devido à sua simplicidade e eficiência computacional, permitindo que a rede aprenda representações esparsas. A função Sigmoid é usada em problemas de classificação binária, transformando a saída em um valor entre 0 e 1. A função Tanh, uma versão escalada do Sigmoid, centraliza os dados em torno de zero, acelerando o treinamento de redes mais profundas (Glorot e Bengio 2010, Nair e Hinton 2010).

A principal vantagem das ANNs é sua capacidade de modelar relações não lineares complexas, o que é essencial para a detecção de falhas em sistemas rotativos, onde os padrões de falha podem ser altamente não lineares e complexos. No entanto, as ANNs podem ser propensas ao *overfitting*, especialmente quando o número de parâmetros é grande em relação ao tamanho do conjunto de dados. Técnicas como regularização L2 e *dropout* são frequentemente utilizadas para mitigar esse problema (Srivastava et al. 2014).

2.1.3 *Random Forest* (RF)

O *Random Forest* é um algoritmo de *Machine Learning* baseado em um conjunto de árvores de decisão, onde cada árvore é treinada em um subconjunto diferente dos dados. O resultado final é obtido pela agregação das previsões de todas as árvores, seja por votação majoritária (para classificação) ou pela média (para regressão).

Os principais hiperparâmetros do *Random Forest* incluem o número de árvores na floresta (*n_estimators*), a profundidade máxima das árvores (*max_depth*) e o número mínimo de amostras necessário para dividir um nó (*min_samples_split*). O número de árvores afeta a robustez do modelo; mais árvores geralmente resultam em maior acurácia, mas aumentam o tempo de treinamento e a complexidade computacional. A profundidade máxima das árvores controla a complexidade do modelo; árvores mais profundas podem capturar padrões mais detalhados, mas também são mais propensas ao *overfitting*. O número mínimo de amostras para dividir um nó ajuda a controlar a divisão da árvore,

evitando que nós pequenos causem *overfitting* (Breiman 2001, Liaw e Wiener 2002).

O *Random Forest* é conhecido por sua robustez e capacidade de lidar com dados ausentes e outliers. A variabilidade introduzida pela seleção aleatória de subconjuntos de dados e *features* durante o treinamento de cada árvore ajuda a reduzir o risco de *overfitting*, tornando o *Random Forest* um modelo robusto e eficiente para detecção de falhas (Rodriguez-Galiano et al. 2012, Probst, Wright e Boulesteix 2019).

Uma das principais vantagens do *Random Forest* é sua relativa interpretabilidade em comparação com outros métodos de conjunto, como o XGBoost. Além disso, o *Random Forest* pode ser treinado e ajustado com menor esforço em termos de seleção de hiperparâmetros, o que pode ser uma vantagem significativa em cenários industriais onde simplicidade e robustez são altamente valorizadas (Cutler et al. 2007).

2.1.4 XGBoost

O XGBoost, ou *Extreme Gradient Boosting*, é um algoritmo de *boosting* que combina o poder preditivo de múltiplas árvores de decisão fracas para formar um modelo forte. O XGBoost é renomado por sua eficiência, flexibilidade e capacidade de fornecer previsões precisas, mesmo em grandes conjuntos de dados.

Os principais hiperparâmetros do XGBoost incluem a taxa de aprendizado (*learning_rate*), o número de árvores (*n_estimators*), a profundidade máxima das árvores (*max_depth*) e os parâmetros de regularização (*reg_lambda* e *reg_alpha*). A taxa de aprendizado controla o impacto de cada nova árvore adicionada ao modelo, onde valores menores geralmente resultam em melhor desempenho, mas requerem mais iterações. A profundidade das árvores influencia a capacidade do modelo de aprender interações complexas nos dados, com árvores mais profundas capazes de capturar mais detalhes. A regularização é utilizada para penalizar a complexidade do modelo e prevenir *overfitting*, com o *reg_lambda* aplicando regularização L2 e o *reg_alpha* aplicando regularização L1 (Chen e Guestrin 2016, He, Zhao e Chu 2018).

O XGBoost é altamente eficiente devido ao uso de técnicas avançadas de regularização, redução de *shrinkage* e amostragem de *features*, que ajudam a melhorar a acurácia do modelo e reduzir o *overfitting*. A paralelização intrínseca do XGBoost permite que ele seja executado de forma eficiente em hardware moderno, tornando-o uma escolha popular para grandes conjuntos de dados (Zhang e Zhan 2017, Chen, Zhang e Gao 2021).

Uma das principais vantagens do XGBoost em comparação com outros modelos é sua capacidade de ajustar e treinar rapidamente grandes volumes de dados, mantendo alta acurácia. No entanto, como outros modelos de conjunto, a interpretabilidade do XGBoost pode ser desafiadora, já que o modelo final é uma combinação de muitas árvores de decisão (Islam e Kim 2019).

2.2 Funcionamento das Redes Neurais Artificiais (ANNs)

As Redes Neurais Artificiais (ANNs) são modelos computacionais amplamente inspirados no funcionamento do cérebro humano. Sua principal característica é a capacidade de aprender padrões a partir de dados, ajustando-se automaticamente para realizar tarefas como classificação, regressão e agrupamento. Este capítulo fornece uma explicação detalhada sobre o funcionamento das ANNs, cobrindo seus aspectos estruturais, algoritmos de aprendizado, fundamentos matemáticos e aplicações práticas. A apresentação é enriquecida com conceitos avançados e exemplos para garantir que o leitor compreenda integralmente os princípios subjacentes (Wang e Wang 2003, Clark 1991, Hammerstrom 1993, Specht et al. 1991, Vogels, Rajan e Abbott 2005).

2.2.1 Introdução às Redes Neurais Artificiais

As ANNs são sistemas compostos por múltiplas unidades processadoras, chamadas neurônios artificiais, interligadas em camadas. Inspiradas no sistema nervoso biológico, as ANNs utilizam conexões ponderadas para transmitir informações, ajustando esses pesos durante o processo de aprendizado para otimizar sua capacidade de generalizar padrões. As ANNs são particularmente úteis em situações onde as relações entre variáveis são não lineares ou desconhecidas.

2.2.1.1 Fluxo de Informações em Redes Neurais

As Redes Neurais Artificiais (ANNs) são compostas por nós semelhantes a neurônios que estão organizados em camadas. Essas camadas são interligadas por conexões ponderadas, que são ajustadas durante o processo de aprendizado. O aprendizado ocorre por meio da modificação iterativa dos pesos dessas conexões, permitindo que a rede modele relações complexas e qualquer função computável com os pesos apropriados.

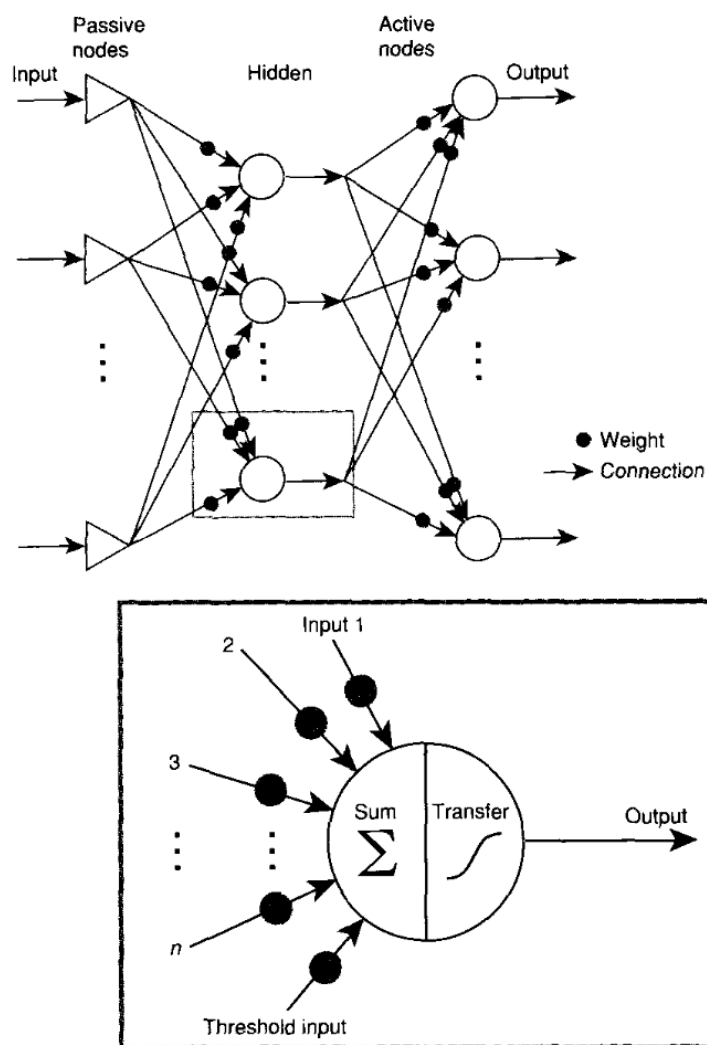
Em cada nó de uma ANN, o funcionamento típico envolve:

- Multiplicar cada entrada pelo peso correspondente.
- Somar os produtos resultantes para gerar um valor agregado.
- Aplicar uma função de transferência não linear ao valor agregado para produzir a saída do nó.

O cálculo fundamental em uma ANN é, portanto, o produto escalar entre o vetor de entradas e o vetor de pesos, seguido por operações de soma e aplicação de uma função de ativação. A eficiência computacional de uma rede depende diretamente da velocidade com que essas operações de multiplicação e acumulação são realizadas.

Uma visualização básica do fluxo de informações em uma ANN pode ser extraída de [Hammerstrom 1993](#), destacando as conexões entre camadas e os ajustes nos pesos durante o treinamento. Essa figura ilustra de forma clara como os dados fluem pela rede e como os pesos são ajustados para minimizar o erro.

Figura 1 – Fluxo de informações em uma Rede Neural Artificial.



Fonte: Adaptado de [Hammerstrom 1993](#).

Na Figura 1, é possível observar a organização das camadas de entrada, ocultas e saída, bem como o papel das conexões ponderadas. Esse diagrama destaca a importância dos pesos e da retropropagação no treinamento da rede, sendo fundamental para compreender como as ANNs aprendem a partir de dados.

2.2.2 Arquitetura das Redes Neurais

2.2.2.1 Estrutura Básica

A arquitetura de uma ANN é definida por três tipos principais de camadas:

- **Camada de Entrada:** Cada neurônio desta camada representa uma variável do conjunto de dados de entrada. Ela apenas transmite os valores recebidos para a camada seguinte.
- **Camadas Ocultas:** Estas camadas realizam transformações nos dados por meio de combinações lineares ponderadas, seguidas de funções de ativação não lineares.
- **Camada de Saída:** Produz a resposta final da rede. Pode representar probabilidades, classes ou valores contínuos, dependendo da tarefa (Wang e Wang 2003).

2.2.2.2 Topologias das Redes

Existem diferentes topologias de redes, dependendo do fluxo de informações:

- **Redes Feedforward:** Os dados fluem em uma única direção, da entrada para a saída, sem ciclos.
- **Redes Recorrentes (RNNs):** Permitem conexões entre neurônios de camadas anteriores, possibilitando a modelagem de dependências temporais em séries temporais (Vogels, Rajan e Abbott 2005).
- **Redes Convolucionais (CNNs):** Utilizam operações de convolução para analisar padrões locais em imagens ou sinais.

2.2.3 Aprendizado das Redes Neurais

As Redes Neurais Artificiais (ANNs) aprendem a partir dos dados por meio de um processo iterativo de ajuste dos pesos das conexões entre os neurônios. Esse aprendizado baseia-se em minimizar a diferença entre a saída produzida pela rede e o valor esperado, utilizando algoritmos matemáticos eficientes. O processo é composto por três etapas principais: propagação direta, cálculo do erro e retropropagação do erro.

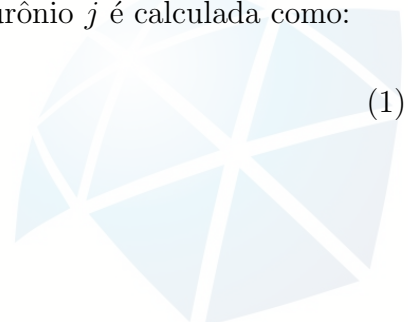
2.2.3.1 Operações nos Neurônios

Cada neurônio realiza operações matemáticas fundamentais, combinando as entradas recebidas por meio de uma soma ponderada. O resultado é ajustado por um termo de viés e submetido a uma função de ativação. A saída do neurônio j é calculada como:

$$y_j = f \left(\sum_{i=1}^n w_{ij} x_i + b_j \right) \quad (1)$$

onde:

- x_i : Entrada do neurônio i .



- w_{ij} : Peso da conexão entre os neurônios i e j .
- b_j : Viés adicionado ao neurônio j .
- f : Função de ativação (Clark 1991).

2.2.3.2 Propagação Direta

Na propagação direta, os dados de entrada fluem da camada de entrada para as camadas ocultas e, por fim, para a camada de saída. Cada neurônio calcula sua saída somando as entradas ponderadas pelos pesos e aplicando uma função de ativação não linear:

$$y_j = f \left(\sum_{i=1}^n w_{ij} x_i + b_j \right) \quad (2)$$

onde:

- y_j é a saída do neurônio j .
- w_{ij} é o peso associado à conexão entre os neurônios i e j .
- x_i é a entrada fornecida pelo neurônio i .
- b_j é o viés do neurônio j .
- f é a função de ativação.

Essa etapa produz uma predição inicial com base nos pesos e vieses atuais da rede. No início do treinamento, esses pesos são inicializados aleatoriamente, o que significa que as predições iniciais geralmente apresentam erros significativos.

2.2.3.3 Funções de Ativação

As funções de ativação são componentes essenciais nas Redes Neurais Artificiais (ANNs), responsáveis por introduzir não-linearidades no modelo. Sem essas funções, as redes neurais seriam incapazes de capturar relações complexas nos dados, limitando-se a modelar apenas relações lineares. A seguir, detalhamos algumas das funções de ativação mais utilizadas:

- **Função Sigmoid:**

- **Definição:** A função sigmoide, também conhecida como função logística, é definida por:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3)$$

onde e representa a base do logaritmo natural. Essa função mapeia qualquer valor real para o intervalo $(0, 1)$, tornando-a especialmente útil em tarefas de classificação binária, onde a saída pode ser interpretada como uma probabilidade (Wang e Wang 2003).

– **Propriedades:**

- * **Não-linearidade:** Permite que a rede capture relações complexas nos dados.
- * **Diferenciabilidade:** A função é suave e possui derivada contínua, facilitando o processo de treinamento por métodos como a retropropagação.
- * **Saturação:** Para valores de entrada muito positivos ou negativos, a função aproxima-se de 1 ou 0, respectivamente, resultando em gradientes próximos de zero. Esse fenômeno pode levar ao problema do *gradiente desaparecido*, dificultando o treinamento de redes profundas (Wang e Wang 2003)..

• **Função Tangente Hiperbólica (Tanh):**

– **Definição:** A função Tanh é matematicamente representada por:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (4)$$

Diferentemente da sigmoide, a Tanh mapeia os valores de entrada para o intervalo $(-1, 1)$, sendo útil em situações onde se deseja que as ativações sejam centradas em zero (Wang e Wang 2003).

– **Propriedades:**

- * **Não-linearidade:** Assim como a sigmoide, permite a modelagem de relações complexas.
- * **Diferenciabilidade:** Possui derivada contínua, facilitando o treinamento.
- * **Saturação:** Também sofre com o problema do gradiente desaparecido em valores extremos de entrada, embora, por ser centrada em zero, possa apresentar desempenho superior em determinadas aplicações.

• **Unidade Linear Retificada (ReLU):**

– **Definição:** A função ReLU é definida por:

$$\text{ReLU}(x) = \max(0, x) \quad (5)$$

Ou seja, retorna zero para entradas negativas e o próprio valor para entradas positivas.

– **Propriedades:**

- * **Simplicidade Computacional:** Devido à sua forma simples, é eficiente em termos de cálculo.

- * **Não-linearidade:** Apesar de sua simplicidade, introduz não-linearidade no modelo.
- * **Problema de *Dying ReLU*:** Neurônios podem "morrer" durante o treinamento se receberem entradas negativas constantes, resultando em gradientes nulos e impedindo o aprendizado (Wang e Wang 2003).

- **Unidade Linear Exponencial (ELU):**

- **Definição:** A função ELU é dada por:

$$\text{ELU}(x) = \begin{cases} x & \text{se } x > 0 \\ \alpha(e^x - 1) & \text{se } x \leq 0 \end{cases} \quad (6)$$

onde α é um hiperparâmetro que define o valor para entradas negativas.

- **Propriedades:**

- * **Não-linearidade:** Permite a modelagem de relações complexas.
- * **Diferenciabilidade:** Possui derivada contínua em todo o seu domínio.
- * **Centragem em Zero:** As ativações podem ser negativas, ajudando na centralização dos dados e potencialmente acelerando o treinamento.
- * **Mitigação do *Dying ReLU*:** Devido à sua definição para valores negativos, reduz a chance de neurônios "mortos" (Maier, Shupe e Fetz 2005).

- **Unidade Linear Retificada com Fuga (Leaky ReLU):**

- **Definição:** A função Leaky ReLU é expressa como:

$$\text{Leaky ReLU}(x) = \begin{cases} x & \text{se } x > 0 \\ \alpha x & \text{se } x \leq 0 \end{cases} \quad (7)$$

onde α é um pequeno valor positivo, geralmente $\alpha = 0.01$, permitindo que mesmo entradas negativas resultem em um pequeno gradiente.

- **Propriedades:**

- * **Mitigação do Problema de *Dying ReLU*:** Diferentemente da ReLU padrão, a Leaky ReLU evita que os neurônios "morram", já que entradas negativas produzem pequenas saídas em vez de zero.
- * **Eficiência Computacional:** Mantém a simplicidade computacional da ReLU.
- * **Hiperparâmetro α :** O valor de α pode ser ajustado para controlar a magnitude da fuga (leak) em entradas negativas (Maier, Shupe e Fetz 2005).

- **Softmax:**

- **Definição:** A função Softmax é amplamente utilizada na camada de saída de redes neurais projetadas para tarefas de classificação multiclasse. Ela converte os valores brutos (logits) em probabilidades. É definida como:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}} \quad (8)$$

onde z_i é o valor de entrada para a i -ésima classe, e n é o número total de classes.

- **Propriedades:**

- * **Probabilidades Normalizadas:** As saídas da função são valores entre 0 e 1, cuja soma é igual a 1, representando uma distribuição de probabilidade.
- * **Adequada para Classificação Multiclasse:** Facilita a interpretação das saídas do modelo como probabilidades atribuídas a cada classe.
- * **Sensível a Diferenças Relativas:** Amplifica diferenças entre os valores de entrada, destacando a classe com maior probabilidade (Wang e Wang 2003).

- **Swish:**

- **Definição:** Introduzida mais recentemente, a função Swish é definida como:

$$\text{Swish}(x) = x \cdot \sigma(x) \quad (9)$$

onde $\sigma(x)$ é a função sigmoide. Essa função combina características da ReLU e da sigmoide.

- **Propriedades:**

- * **Suavidade:** É uma função contínua e diferenciável, o que facilita a convergência durante o treinamento.
- * **Não-linearidade Adaptativa:** Diferente da ReLU, a Swish pode produzir saídas negativas ou amplificadas para entradas específicas, dependendo do valor de x .
- * **Desempenho Empírico:** Estudos mostraram que o Swish pode superar outras funções de ativação em redes profundas devido à sua flexibilidade e suavidade (Sorsa, Koivo e Koivisto 1991).

2.2.3.3.1 Importância das Funções de Ativação

As funções de ativação desempenham um papel crítico em redes neurais, pois permitem que o modelo aprenda e represente relações complexas nos dados. Cada função possui características específicas que a tornam mais adequada para determinadas aplicações. Por exemplo:

- A **sigmoide** é ideal para saídas probabilísticas binárias.
- A **ReLU** e suas variantes (Leaky ReLU, ELU) são amplamente utilizadas em camadas ocultas devido à sua eficiência computacional e capacidade de evitar a saturação para entradas positivas.
- A **softmax** é essencial para classificação multiclasse.

A escolha da função de ativação deve levar em consideração o problema em questão, a arquitetura da rede e os requisitos de treinamento (Specht et al. 1991).

2.2.3.4 Função de Custo

A função de custo, também conhecida como função de perda ou erro, quantifica a discrepância entre as previsões do modelo e os valores reais esperados. Em outras palavras, ela mede o quão bem o modelo está performando em relação aos dados de treinamento. A escolha apropriada da função de custo é crucial, pois influencia diretamente o processo de aprendizado e a eficácia do modelo. Algumas das funções de custo mais comuns incluem:

- **Erro Quadrático Médio (MSE - Mean Squared Error):** Amplamente utilizada em problemas de regressão, o MSE calcula a média dos quadrados das diferenças entre as previsões do modelo e os valores reais. Matematicamente, é representado por:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (10)$$

onde y_i representa o valor real, $\hat{y}_i$ a previsão do modelo e n o número total de amostras. O MSE penaliza severamente grandes discrepâncias, tornando-o sensível a outliers (Clark 1991).

- **Entropia Cruzada (Cross-Entropy):** Predominantemente utilizada em tarefas de classificação, especialmente quando há múltiplas classes, a entropia cruzada mede a diferença entre duas distribuições de probabilidade: a distribuição real dos dados e a distribuição prevista pelo modelo. Para classificação binária, é definida como:

$$\text{CE} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (11)$$

onde y_i é o rótulo real (0 ou 1) e $\hat{y}_i$ é a probabilidade prevista de y_i ser 1. A entropia cruzada aumenta à medida que as previsões se distanciam dos valores reais, incentivando o modelo a produzir probabilidades próximas dos rótulos verdadeiros (Hammerstrom 1993).

A minimização da função de custo é realizada por meio de algoritmos de otimização, sendo o **Gradiente Descendente** um dos mais utilizados. Este algoritmo ajusta iterativamente os parâmetros do modelo (por exemplo, os pesos em uma rede neural) para minimizar a função de custo, melhorando assim a precisão das previsões.

2.2.3.5 Gradiente Descendente

O **Gradiente Descendente** é um algoritmo de otimização fundamental no aprendizado de máquina, utilizado para minimizar funções de custo ajustando iterativamente os parâmetros do modelo. A essência do gradiente descendente reside na ideia de mover-se na direção oposta ao gradiente da função de custo em relação aos parâmetros, visando encontrar o ponto de mínimo dessa função (Nguyen 2017).

2.2.3.5.1 Conceito de Gradiente

Em cálculo diferencial, o **gradiente** de uma função escalar multivariada é um vetor que aponta na direção do maior aumento da função. Cada componente desse vetor corresponde à derivada parcial da função em relação a uma das variáveis. No contexto de aprendizado de máquina, o gradiente indica como a função de custo varia em relação a cada parâmetro do modelo. Movendo-se na direção oposta ao gradiente, é possível reduzir o valor da função de custo, aproximando-se do mínimo (Nguyen 2017).

2.2.3.5.2 Algoritmo do Gradiente Descendente

O processo iterativo do gradiente descendente pode ser descrito pelos seguintes passos:

1. **Inicialização:** Definir valores iniciais para os parâmetros do modelo. Essa inicialização pode ser feita de forma aleatória ou baseada em algum critério específico.
2. **Cálculo do Gradiente:** Calcular o gradiente da função de custo em relação aos parâmetros atuais. Este gradiente indica a direção e a magnitude da maior taxa de aumento da função de custo.
3. **Atualização dos Parâmetros:** Atualizar os parâmetros do modelo movendo-os na direção oposta ao gradiente, com um passo proporcional à **taxa de aprendizado** (α):

$$\theta_{\text{novo}} = \theta_{\text{atual}} - \alpha \cdot \nabla J(\theta_{\text{atual}}) \quad (12)$$

onde θ representa os parâmetros do modelo e $J(\theta)$ é a função de custo.

4. **Convergência:** Repetir os passos de cálculo do gradiente e atualização dos parâmetros até que a mudança na função de custo entre iterações sucessivas seja menor que um limiar pré-definido, indicando que o algoritmo convergiu para um mínimo (Nguyen 2017).

2.2.3.5.3 Taxa de Aprendizado

A **taxa de aprendizado** (α) é um hiperparâmetro crucial que determina o tamanho dos passos dados na direção oposta ao gradiente durante a atualização dos parâmetros. Uma taxa de aprendizado muito alta pode fazer com que o algoritmo oscile em torno do mínimo ou até mesmo diverja, enquanto uma taxa muito baixa pode resultar em uma convergência extremamente lenta. A escolha adequada da taxa de aprendizado é, portanto, vital para o desempenho eficiente do algoritmo (Wang e Wang 2003) (Nguyen 2017).

2.2.3.5.4 Variantes do Gradiente Descendente

Existem diferentes variantes do gradiente descendente, cada uma com suas características específicas:

- **Gradiente Descendente em Batch:** Utiliza todo o conjunto de dados de treinamento para calcular o gradiente da função de custo. Embora forneça uma estimativa precisa do gradiente, pode ser computacionalmente caro para conjuntos de dados grandes.
- **Gradiente Descendente Estocástico (SGD - Stochastic Gradient Descent):** Atualiza os parâmetros usando o gradiente calculado a partir de uma única amostra de treinamento selecionada aleatoriamente. Isso torna o processo de atualização mais rápido, porém mais ruidoso, podendo causar flutuações na função de custo.
- **Gradiente Descendente em Mini-Batch:** Compromisso entre as abordagens anteriores, onde o gradiente é calculado usando um pequeno subconjunto (mini-batch) do conjunto de dados de treinamento. Essa abordagem equilibra a eficiência computacional e a estabilidade das atualizações (Wang e Wang 2003).

2.2.3.5.5 Aplicação em Redes Neurais

No contexto de **Redes Neurais Artificiais (ANNs)**, o gradiente descendente é utilizado em conjunto com o algoritmo de **Retropropagação** para treinar a rede. A retropropagação calcula os gradientes da função de custo em relação aos pesos da rede, propagando o erro da camada de saída para as camadas anteriores (Wang e Wang 2003).

2.2.4 Treinamento e Generalização

O treinamento consiste em ajustar os pesos da rede para minimizar o erro no conjunto de treinamento. Para evitar overfitting, várias técnicas podem ser empregadas:

- **Regularização L2:** Penaliza pesos elevados, incentivando modelos mais simples.
- **Dropout:** Desativa aleatoriamente neurônios durante o treinamento, melhorando a generalização.
- **Validação Cruzada:** Garante que o modelo generalize bem para novos dados (Clark 1991).

2.2.5 Aplicações Práticas

As ANNs são amplamente utilizadas em:

- **Reconhecimento de Imagens:** Classificação e segmentação.
- **Processamento de Linguagem Natural:** Tradução e análise de sentimentos.
- **Sistemas de Recomendação:** Personalização de produtos e serviços (Hammerstrom 1993).

As Redes Neurais Artificiais combinam fundamentos matemáticos robustos e flexibilidade computacional, tornando-se ferramentas indispensáveis em diversas áreas. Este capítulo detalhou os aspectos do funcionamento das ANNs, desde suas estruturas até as técnicas de treinamento, oferecendo uma base sólida para compreender e aplicar essas tecnologias em problemas reais.

2.3 Funcionamento da *Random Forest*

Este capítulo apresenta uma análise técnica aprofundada do algoritmo *Random Forest*, com foco nos mecanismos de seleção de variáveis, cálculo de métricas como entropia e índice de Gini, e sua aplicação em árvores de decisão dentro do modelo de ensemble. Esses elementos são cruciais para compreender como o *Random Forest* realiza a divisão dos nós e mantém sua robustez em tarefas de classificação e regressão.

2.3.1 Definição e Fundamentos

O *Random Forest* é um modelo de aprendizado supervisionado baseado em um ensemble de árvores de decisão. Ele combina as vantagens da construção de árvores independentes e da agregação estatística para melhorar a generalização e reduzir a variância

do modelo (Biau e Scornet 2016). O processo de decisão em cada árvore é guiado pela escolha das variáveis mais informativas, com base em métricas como entropia e índice de Gini.

2.3.2 Ensemble Learning no *Random Forest*

O *Random Forest* é construído sobre o conceito de ensemble learning, que combina múltiplos modelos individuais para produzir uma predição agregada. Em vez de depender de uma única árvore de decisão, o *Random Forest* utiliza um conjunto de árvores treinadas em subconjuntos aleatórios dos dados e das variáveis disponíveis, promovendo robustez e diversidade no ensemble (Biau e Scornet 2016).

2.3.2.1 Redução de Variância

O ensemble learning reduz a variância do modelo ao combinar as predições de múltiplas árvores independentes. Matematicamente, a variância da predição do modelo pode ser expressa como:

$$\text{Var}(\hat{f}(x)) = \frac{1}{M} \text{Var}(T(x)) + \left(1 - \frac{1}{M}\right) \text{Cov}(T_i(x), T_j(x)), \quad (13)$$

onde M é o número de árvores, $T_i(x)$ e $T_j(x)$ são árvores individuais, e Cov representa a correlação entre árvores. À medida que $M \rightarrow \infty$ e a correlação diminui, a variância do ensemble se aproxima de zero.

2.3.3 Bagging e Bootstrap Sampling

O bagging (*bootstrap aggregating*) é o mecanismo central no *Random Forest*. Ele cria subconjuntos de dados ao selecionar amostras com reposição (*bootstrap*), permitindo que cada árvore seja treinada em uma versão diferente dos dados originais. Isso melhora a diversidade do modelo e reduz o risco de overfitting (Rigatti 2017).

Cada subconjunto contém aproximadamente 63.2% das amostras originais, enquanto 36.8% são deixadas como *out-of-bag* (OOB). Essas amostras OOB são usadas para avaliar o desempenho interno do modelo, eliminando a necessidade de um conjunto de validação separado.

2.3.4 Seleção de Variáveis e Critérios de Divisão

A seleção de variáveis e os critérios de divisão são os elementos centrais que garantem o desempenho e a robustez do algoritmo *Random Forest*. Esses mecanismos são responsáveis por construir árvores de decisão diversificadas e equilibradas, promovendo a redução do *overfitting* e uma melhor generalização do modelo. Neste contexto, a introdução de aleatoriedade no processo de seleção de variáveis desempenha um papel crucial.

2.3.4.1 Subconjunto Aleatório de Variáveis

No processo de construção de cada árvore, o *Random Forest* seleciona um subconjunto aleatório de variáveis em cada nó para avaliar as possíveis divisões. Este subconjunto, denotado por m , é uma fração do número total de variáveis p no conjunto de dados. A escolha de m segue uma fórmula prática que varia com a tarefa em questão:

$$m = \begin{cases} \sqrt{p}, & \text{para classificação,} \\ \frac{p}{3}, & \text{para regressão.} \end{cases} \quad (14)$$

Os benefícios dessa abordagem são múltiplos:

- **Diversidade no Ensemble:** A seleção aleatória de variáveis promove diferenças estruturais significativas entre as árvores individuais, aumentando a diversidade do *ensemble* e, conseqüentemente, sua robustez.
- **Redução de Correlação entre Árvores:** Árvores construídas com variáveis ligeiramente diferentes reduzem a correlação entre elas, resultando em um modelo que evita viés de dominância.
- **Exploração de Variáveis Subutilizadas:** Variáveis que podem não ser imediatamente relevantes para as primeiras divisões ganham a oportunidade de contribuir em outras árvores do *ensemble*, melhorando a abrangência do modelo.
- **Eficiência Computacional:** Avaliar um subconjunto reduzido de variáveis em cada nó diminui significativamente o custo computacional, tornando o *Random Forest* escalável para conjuntos de dados de alta dimensionalidade (Biau e Scornet 2016, Schonlau e Zou 2020).

A introdução de aleatoriedade no subconjunto de variáveis é uma característica distintiva do *Random Forest*, garantindo que o modelo não dependa excessivamente de variáveis específicas e promovendo uma construção mais equilibrada das árvores.

2.3.4.2 Critério de Divisão

Depois de selecionar o subconjunto aleatório de variáveis, o algoritmo *Random Forest* avalia todas as possíveis divisões em cada nó para identificar a que melhor separa os dados. O critério de divisão é escolhido com base em métricas de pureza do nó, como a entropia e o índice de Gini, que medem a qualidade da separação.

2.3.4.2.1 Processo de Avaliação:

1. **Cálculo do Ganho de Informação:** A qualidade de uma divisão é avaliada pelo aumento na pureza do nó resultante, medido pelo ganho de informação (*information gain*) ou pela redução do índice de Gini.
2. **Escolha da Melhor Divisão:** A divisão que maximiza o ganho de informação ou minimiza a impureza é escolhida como a melhor divisão para aquele nó.
3. **Iteração ao Longo dos Nós:** Este processo é repetido para cada nó em todas as árvores do *ensemble*, garantindo que cada árvore seja construída de maneira independente e diversificada.

2.3.4.2.2 Importância dos Critérios de Divisão:

- **Equilíbrio entre *Bias* e Variância:** Critérios de divisão bem definidos ajudam a evitar árvores excessivamente complexas (*overfitting*) ou excessivamente simplistas (*underfitting*).
- **Eficiência na Identificação de Padrões:** Divisões bem escolhidas garantem que o modelo capture padrões significativos nos dados sem se desviar para ruídos (Rigatti 2017, Bonissone et al. 2010).

2.3.5 Interdependência entre Seleção de Variáveis e Critérios de Divisão

A seleção de variáveis e os critérios de divisão são processos interdependentes e complementares. Enquanto a seleção de variáveis aleatórias promove a diversidade entre as árvores, os critérios de divisão asseguram que cada nó seja construído de forma a maximizar a pureza dos grupos resultantes ou minimizar a variância. Essa sinergia é essencial para a eficácia do *Random Forest*, equilibrando diversidade, robustez e desempenho computacional.

2.3.6 Métricas para Avaliação de Divisões

2.3.6.1 Entropia

A entropia mede o grau de desordem ou incerteza em um conjunto de dados. Ela é amplamente utilizada em árvores de decisão para avaliar a pureza dos nós. Para um conjunto de dados com K classes, a entropia H é definida como:

$$H = - \sum_{k=1}^K p_k \log_2(p_k) \quad (15)$$

onde p_k é a proporção de observações pertencentes à classe k no nó avaliado.

A redução de entropia resultante de uma divisão (*information gain*) é calculada como:

$$IG = H_{\text{pai}} - \sum_{j=1}^m \frac{|N_j|}{|N|} H_j \quad (16)$$

onde:

- H_{pai} é a entropia do nó antes da divisão.
- H_j é a entropia dos nós filhos após a divisão.
- $|N_j|$ e $|N|$ representam o número de observações nos nós filho j e no nó pai, respectivamente.

Divisões que maximizam o *information gain* são preferidas, pois resultam em nós mais puros (Rigatti 2017).

2.3.6.2 Índice de Gini

O índice de Gini é outra métrica amplamente utilizada para avaliar a pureza de nós. Ele mede a probabilidade de classificar incorretamente uma observação escolhida aleatoriamente no conjunto de dados, caso a classificação seja baseada na distribuição de classes do nó. O índice de Gini G é definido como:

$$G = 1 - \sum_{k=1}^K p_k^2 \quad (17)$$

Assim como na entropia, o objetivo é minimizar o índice de Gini nos nós resultantes de uma divisão. A redução do Gini (*Gini gain*) é calculada por:

$$GG = G_{\text{pai}} - \sum_{j=1}^m \frac{|N_j|}{|N|} G_j \quad (18)$$

onde os termos são equivalentes aos definidos anteriormente.

Comparado à entropia, o índice de Gini é computacionalmente mais simples, já que evita o cálculo do logaritmo. Ele é, portanto, preferido em situações onde o desempenho computacional é crítico (Bonissone et al. 2010).

2.3.6.3 Comparação entre Entropia e Índice de Gini

Embora ambas as métricas avaliem a pureza dos nós, há diferenças práticas entre elas:

- A **entropia** tende a penalizar mais fortemente nós com distribuições uniformes entre as classes.
- O **índice de Gini** é mais sensível a grandes diferenças de probabilidade entre as classes.

2.3.7 Algoritmo de Construção de Árvores no *Random Forest*

O processo de construção de árvores no *Random Forest* pode ser resumido nos seguintes passos:

1. Selecionar um subconjunto aleatório de variáveis m para cada divisão.
2. Avaliar todas as divisões possíveis com base em uma métrica de pureza (entropia ou Gini).
3. Escolher a divisão que maximiza o ganho de informação (*information gain*) ou reduz mais significativamente o índice de Gini.
4. Repetir o processo para cada nó até atingir um critério de parada, como profundidade máxima ou número mínimo de observações no nó.

2.3.7.1 Critérios de Parada

Os critérios de parada evitam o crescimento excessivo das árvores. Eles incluem:

- **Profundidade máxima** (*max_depth*): Define o limite máximo de divisões.
- **Número mínimo de amostras por nó** (*min_samples_split*): Impede a criação de nós pequenos.
- **Número mínimo de amostras em folhas** (*min_samples_leaf*): Garante folhas significativas.

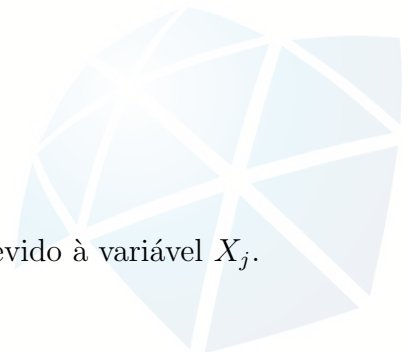
2.3.8 Importância de Variáveis

O *Random Forest* fornece uma estimativa da importância de cada variável com base em sua contribuição para a pureza dos nós em todas as árvores. A importância de uma variável X_j é calculada como:

$$\text{Imp}(X_j) = \frac{1}{M} \sum_{m=1}^M \sum_{t \in T_m} \Delta C_t(X_j) \quad (19)$$

onde:

- M é o número total de árvores.
- T_m é o conjunto de nós na árvore m .
- $\Delta C_t(X_j)$ é a redução no critério de divisão no nó t devido à variável X_j .



O *Random Forest* utiliza métricas matematicamente fundamentadas, como entropia e índice de Gini, para selecionar divisões que maximizam a pureza dos nós. A introdução de aleatoriedade no processo de seleção de variáveis contribui para a robustez do modelo, reduzindo correlações entre as árvores e melhorando a generalização. A compreensão detalhada dessas métricas e seus cálculos é essencial para interpretar os resultados do *Random Forest* e ajustá-lo para diferentes aplicações.

2.4 Funcionamento do XGBoost

O algoritmo XGBoost (*Extreme Gradient Boosting*) é uma implementação de Gradient Boosting que se destaca por sua eficiência, escalabilidade e robustez. Desenvolvido com base em princípios teóricos sólidos e otimizados para cenários práticos, o XGBoost é amplamente utilizado em competições de aprendizado de máquina e aplicações industriais. Este capítulo apresenta os fundamentos teóricos e práticos que suportam o funcionamento do XGBoost.

2.4.1 Visão Geral

O XGBoost é uma extensão do método de Gradient Boosting que utiliza árvores de decisão como base. O algoritmo constrói árvores sequencialmente, onde cada nova árvore é treinada para corrigir os erros das árvores anteriores. O objetivo é minimizar uma função de custo que combina a perda preditiva e um termo de regularização que penaliza a complexidade do modelo (Chen 2015).

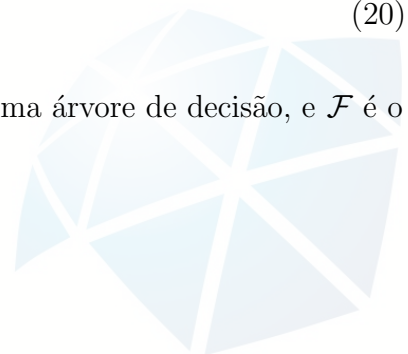
2.4.2 Funcionamento do XGBoost

2.4.2.1 Estrutura Matemática do Gradient Boosting

O XGBoost é baseado no Gradient Boosting, que é uma técnica de aprendizado de máquina para problemas de classificação e regressão. O objetivo é encontrar um modelo preditivo $\hat{y}$ que minimize uma função de perda $L(y, \hat{y})$ em relação às variáveis dependentes e independentes. Formalmente, o modelo é descrito por:

$$\hat{y} = \sum_{m=1}^M f_m(x), \quad f_m \in \mathcal{F} \quad (20)$$

onde $f_m(x)$ representa uma função de decisão, geralmente uma árvore de decisão, e $\mathcal{F}$ é o conjunto de todas as funções possíveis (Nielsen 2016).



2.4.2.2 Objetivo de Otimização

A função de custo do XGBoost combina um termo de perda preditiva com um termo de regularização que penaliza a complexidade do modelo:

$$Obj(\Theta) = \sum_{i=1}^n L(y_i, \hat{y}_i) + \sum_{m=1}^M \Omega(f_m), \quad (21)$$

onde $\Omega(f) = \gamma T + \frac{1}{2} \lambda \|w\|^2$, sendo T o número de folhas na árvore, w os pesos das folhas, e γ e λ os parâmetros de regularização (Chen e Guestrin 2016).

2.4.2.3 Aproximação por Taylor de Segunda Ordem

Para resolver a função objetivo, o XGBoost utiliza uma aproximação de Taylor de segunda ordem para a função de perda:

$$Obj \approx \sum_{i=1}^n \left[g_i f(x_i) + \frac{1}{2} h_i f(x_i)^2 \right] + \Omega(f), \quad (22)$$

onde $g_i = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}$ e $h_i = \frac{\partial^2 L(y_i, \hat{y}_i)}{\partial \hat{y}_i^2}$ são o gradiente e o hessiano, respectivamente. Essa abordagem permite uma otimização mais eficiente (Chen e Guestrin 2016).

2.4.2.4 Divisão de Nós e Ganho de Informação

Em cada nó de uma árvore, o XGBoost avalia possíveis divisões para maximizar o ganho de informação. O ganho é definido como a redução na perda após a divisão, dado por:

$$Gain = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma, \quad (23)$$

onde G_L , H_L , G_R e H_R são os gradientes e hessianos das partições esquerda e direita (Chen 2015).

2.4.3 Características Avançadas do XGBoost

2.4.3.1 Regularização

A regularização no XGBoost é uma técnica fundamental para controlar a complexidade do modelo e prevenir o overfitting. Ela é implementada através de penalizações adicionadas à função objetivo, especificamente os termos de regularização L1 (Lasso) e L2 (Ridge). Esses termos são controlados pelos hiperparâmetros `alpha` (α) e `lambda` (λ), respectivamente.

- **Regularização L1 (Lasso):** A regularização L1 adiciona à função objetivo a soma dos valores absolutos dos pesos das folhas das árvores. Matematicamente, o termo

de penalização L1 é expresso como:

$$\Omega_1(f) = \alpha \sum_{j=1}^T |w_j|$$

onde T é o número de folhas na árvore, w_j representa o peso da folha j , e α é o parâmetro de regularização L1. Essa penalização incentiva a esparsidade dos pesos, promovendo modelos mais simples e interpretáveis ao forçar alguns pesos a serem exatamente zero. Isso é particularmente útil em cenários com alta dimensionalidade, onde muitas características podem ser irrelevantes ou redundantes (Chen 2015).

- **Regularização L2 (Ridge):** A regularização L2 adiciona à função objetivo a soma dos quadrados dos pesos das folhas das árvores. O termo de penalização L2 é dado por:

$$\Omega_2(f) = \lambda \sum_{j=1}^T w_j^2$$

onde λ é o parâmetro de regularização L2. Essa penalização desencoraja pesos excessivamente grandes, promovendo uma distribuição mais uniforme dos pesos e aumentando a robustez do modelo contra variações nos dados. Diferentemente da regularização L1, a L2 não força os pesos a zero, mas sim os reduz proporcionalmente, mantendo todas as características no modelo, o que é benéfico quando todas as características têm alguma relevância (Chen 2015).

A função objetivo regularizada do XGBoost combina ambos os termos de regularização, além da função de perda, conforme mostrado abaixo:

$$Obj(\Theta) = \sum_{i=1}^n L(y_i, \hat{y}_i) + \Omega(f)$$

onde:

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2 + \alpha \sum_{j=1}^T |w_j|$$

Aqui, $L(y_i, \hat{y}_i)$ representa a função de perda que mede a diferença entre a previsão $\hat{y}_i$ e o valor real y_i , e γ é um parâmetro que penaliza a complexidade do modelo com base no número de folhas T .

A inclusão dos termos de regularização L1 e L2 permite ao XGBoost:

- **Controlar a complexidade do modelo:** Reduzindo o risco de overfitting, especialmente em conjuntos de dados ruidosos ou de alta dimensionalidade.

- **Promover a generalização:** Melhorando o desempenho em dados de teste não vistos anteriormente.
- **Adaptar-se a diferentes cenários:** Combinando esparsidade e robustez por meio do ajuste cuidadoso dos parâmetros α e λ .

Essas técnicas tornam o XGBoost um dos algoritmos mais eficazes e versáteis para aprendizado supervisionado ([Chen 2015](#)).

2.4.3.2 Paralelismo

O XGBoost foi projetado para aproveitar eficientemente a computação paralela, acelerando significativamente o processo de treinamento. Durante a construção das árvores de decisão, o algoritmo realiza buscas exaustivas para determinar o melhor ponto de divisão em cada nó. Este processo envolve cálculos independentes que podem ser distribuídos entre múltiplos threads ou núcleos de processamento. Ao paralelizar a avaliação das possíveis divisões, o XGBoost reduz consideravelmente o tempo necessário para treinar modelos complexos, especialmente em conjuntos de dados volumosos ([Chen e Guestrin 2016](#)).

Além disso, o XGBoost suporta treinamento distribuído em clusters de computadores, permitindo que grandes volumes de dados sejam processados de forma eficiente. Essa capacidade de paralelismo e distribuição torna o XGBoost altamente escalável e adequado para aplicações em larga escala ([Chen e Guestrin 2016](#)).

2.4.3.3 Redução de Sobrecarga Computacional

Para lidar com conjuntos de dados extensos que excedem a capacidade de memória disponível, o XGBoost implementa técnicas de computação fora do núcleo (out-of-core computing). Este método envolve o processamento dos dados em blocos que são carregados e descarregados da memória conforme necessário, permitindo que o treinamento ocorra mesmo em sistemas com recursos limitados. Além disso, o XGBoost utiliza algoritmos conscientes de cache, otimizando o acesso à memória e minimizando latências associadas à movimentação de dados. Essas otimizações estruturais garantem que o XGBoost mantenha um desempenho eficiente e escalável, mesmo quando confrontado com desafios computacionais significativos ([Nielsen 2016](#)).

2.4.4 Aplicações Práticas

Devido à sua eficiência e precisão, o XGBoost tem sido amplamente utilizado em diversas áreas, como bioinformática, detecção de fraudes e previsões financeiras. Ele também se destaca em competições de aprendizado de máquina, como as organizadas pela plataforma Kaggle ([Chen 2015](#)).

O XGBoost é uma ferramenta poderosa e versátil para aprendizado supervisionado. Suas características avançadas, como regularização e paralelismo, combinadas com um sólido fundamento teórico, explicam seu sucesso em uma ampla gama de aplicações.

2.5 Noções sobre Estatística e Densidade Espectral

Este capítulo busca apresentar os principais fundamentos estatísticos e conceitos teóricos aplicados ao longo deste trabalho. Abordam-se tópicos como média aritmética, valor esperado, desvio padrão, normalização e F_1 -score, que são essenciais para a análise e interpretação de dados. Também são introduzidos conceitos avançados, como a Densidade Espectral de Potência (PSD), que desempenha papel crucial na extração de características do *dataset* CONCEPT, e o valor- t , elemento central em testes de hipóteses estatísticas.

2.5.1 Média Aritmética

A média aritmética é uma das medidas de tendência central mais utilizadas em estatística, servindo para descrever o valor médio de um conjunto de dados. Representa o ponto de equilíbrio de uma distribuição e é frequentemente usada como uma referência simples e compreensível para dados quantitativos.

Definição: Seja n o número de observações e x_i cada valor, com $i = 1, \dots, n$. A média aritmética $\bar{x}$ é dada pela soma de todos os valores dividida pelo número total de observações:

$$\bar{x} = \frac{x_1 + x_2 + \dots + x_n}{n} = \frac{1}{n} \sum_{i=1}^n x_i \quad (24)$$

Esse conceito é amplamente aplicado em análises estatísticas, seja para interpretar distribuições de frequências, calcular índices econômicos ou mesmo comparar desempenhos em diferentes cenários.

2.5.2 Valor Esperado

O valor esperado, também conhecido como esperança matemática, é uma generalização da média aritmética para variáveis aleatórias. Ele reflete o valor médio que se espera obter em experimentos repetidos e é essencial em análises de probabilidade e estatística.

Definição: Seja X uma variável aleatória com possíveis valores $x_1, \dots, x_n$ e suas respectivas probabilidades $p(x_i)$. O valor esperado $E[X]$ é dado por:

$$E[X] = \sum_{i=1}^n x_i p(x_i) \quad (25)$$

Quando todas as probabilidades são iguais ($p(x_i) = 1/n$), o valor esperado coincide com a média aritmética:

$$\mu = E[X] = \frac{1}{n} \sum_{i=1}^n x_i \quad (26)$$

O conceito de valor esperado é amplamente utilizado em áreas como teoria dos jogos, análise de risco financeiro e engenharia de sistemas.

2.5.3 Desvio Padrão

O desvio padrão é uma medida de dispersão que indica o grau de variação ou consistência em um conjunto de dados. Um valor baixo de desvio padrão sugere que os dados estão próximos da média, enquanto um valor alto indica maior variabilidade.

Definição: Para uma variável aleatória X com média μ , o desvio padrão σ é dado por:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2} \quad (27)$$

O desvio padrão é amplamente utilizado em diversas aplicações, como controle de qualidade, análise financeira e monitoramento de processos industriais, devido à sua capacidade de descrever a variabilidade dos dados.

2.5.4 Normalização

A normalização é uma técnica fundamental para ajustar dados a uma escala comum, eliminando discrepâncias causadas por unidades ou magnitudes diferentes. No contexto de aprendizado de máquina, ela é essencial para melhorar o desempenho de algoritmos.

Definição: Cada valor x de um conjunto de dados é normalizado subtraindo-se sua média e dividindo pelo desvio padrão:

$$\hat{x} = \frac{x - \mu_x}{\sigma_x} \quad (28)$$

Farrar e Worden 2012 destaca que a normalização desempenha um papel crítico na mitigação de variações ambientais e operacionais em estruturas monitoradas, o que é particularmente relevante em sistemas de monitoramento de integridade estrutural (*Structural Health Monitoring* - SHM).

2.5.5 F_1 -Score

O F_1 -score é uma métrica que equilibra precisão e sensibilidade em modelos de classificação. É especialmente útil em contextos onde há classes desbalanceadas.

Definição: O F_1 -score é dado pela média harmônica entre precisão (*precision*) e sensibilidade (*recall*):

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (29)$$

Farrar e Worden 2012 destaca a relevância dessa métrica para avaliações robustas de desempenho em sistemas de detecção.

2.5.6 Densidade Espectral de Potência

A densidade espectral de potência (PSD) descreve como a energia de um sinal é distribuída ao longo das frequências. É uma ferramenta essencial na análise de sinais vibracionais em sistemas rotativos.

Definição: A PSD é definida como o valor esperado do quadrado da transformada de Fourier do sinal:

$$S_{xx}(\omega) = E[|X(\omega)|^2] \quad (30)$$

Essa técnica é amplamente utilizada para identificar padrões e anomalias em sinais industriais, contribuindo para diagnósticos precisos de falhas (Farrar e Worden 2012).

2.5.7 Valor-t e Teste t

O valor-t é uma estatística usada para comparar médias ou avaliar a significância de uma amostra em relação a uma população.

2.5.7.1 Distribuição t-Student

Antes de calcular o valor-t, é necessário compreender a distribuição t-Student, uma distribuição de probabilidade usada quando o tamanho da amostra é pequeno ou a variância populacional é desconhecida.

Definição: A estatística t é calculada como:

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}} \quad (31)$$

onde $\bar{x}$ é a média amostral, μ é a média populacional, s é o desvio padrão amostral e n é o tamanho da amostra.

O teste t avalia a probabilidade de uma diferença observada ser devido ao acaso, sendo amplamente utilizado em estudos comparativos e análises experimentais (Farrar e Worden 2012).

3 MATERIAIS E MÉTODOS

3.1 Aquisição de Dados e Configuração Experimental

Para o treinamento e validação do modelo, foi utilizado o conjunto de dados da Case Western Reserve University (CWRU). Este conjunto de dados é publicamente disponível e pode ser acessado no URL: <<https://engineering.case.edu/bearingdatacenter>>. A coleta de dados foi realizada pela equipe da CWRU, conforme descrito em [Chen, Zhang e Gao 2021](#).

O conjunto de dados foi coletado de um motor elétrico de 1,5 kW (2 hp) equipado com rolamentos de esferas modelo JEM SKF 6205-2RS instalados no eixo do motor. Os rolamentos foram submetidos a falhas induzidas por descarga elétrica (*electro-discharge machining* - EDM) em três componentes distintos: elementos rolantes, pista interna e pista externa, com três tamanhos de falha (0,007, 0,014 e 0,021 polegadas). Além dos dados de falha, também foram coletados dados de rolamentos em condição normal ([Chen, Zhang e Gao 2021](#)).

Os dados de vibração foram coletados por acelerômetros posicionados na extremidade motriz (DE) do motor, com uma frequência de amostragem de 48 kHz. Durante a coleta de dados, o motor operou a uma potência constante de 3 hp e uma velocidade de 1730 rpm ([Han et al. 2018](#)).

O sistema experimental utilizado para gerar os dados de falha nos rolamentos consiste em:

- Motor de 1,5 kW (2 hp) com rolamentos de esferas instalados no eixo;
- Sensor de torque para coleta de dados de velocidade e potência do motor;
- Dinamômetro para medição de potência;
- Diversos componentes eletrônicos de controle (não mostrados na figura).

Os dados de treinamento e teste foram derivados dos acelerômetros posicionados na extremidade motriz do motor, com amostragem a 48 kHz. As falhas nos rolamentos foram induzidas utilizando EDM com três diâmetros diferentes de falha (0,007, 0,014 e 0,021 polegadas), resultando em três tipos de falhas: falha nos elementos rolantes, falha na pista interna e falha na pista externa, cada uma com três tamanhos de falha. Além disso, dados de rolamentos em condição normal também foram coletados ([Chen, Zhang e Gao 2021](#)).

3.2 Extração de Features

A extração de features é uma etapa fundamental na construção de modelos de diagnóstico de falhas em sistemas rotativos. Diferentes features podem levar a resultados diagnósticos distintos, sendo, portanto, importante realizar uma comparação abrangente. Neste trabalho, foram extraídas duas classes populares de features para o treinamento do modelo: features estatísticas no domínio do tempo (TDF - *time-domain statistical features*) e features estatísticas no domínio da frequência (FDF - *frequency-domain statistical features*) (Han et al. 2018, Dragomiretskiy e Zosso 2014).

Quando ocorre uma falha em um sistema mecânico, a forma de onda do sinal de vibração no domínio do tempo pode mudar tanto em amplitude quanto em distribuição. Portanto, dezesseis parâmetros estatísticos no domínio do tempo foram extraídos, conforme mostrado na Tabela 1 (Guo et al. 2016). Esses parâmetros incluem, entre outros, a média, variância, curtose e assimetria do sinal de vibração. A análise desses parâmetros possibilita identificar mudanças sutis no comportamento do sistema que podem indicar a presença de falhas.

Tabela 1 – Parâmetros estatísticos no domínio do tempo.

Número	Expressão da Feature	Número	Expressão da Feature
tp1	$\frac{\sum_{n=1}^N x(n)}{N}$	tp9	tp7 - tp8
tp2	$\sqrt{\frac{1}{N} \sum_{n=1}^N x_n^2}$	tp10	$\frac{\sum_{n=1}^N (x(n)-tp1)^2}{N-1}$
tp3	$\left(\frac{1}{N} \sum_{n=1}^N \sqrt{ x_n }\right)^2$	tp11	$\frac{tp2}{tp4}$
tp4	$\frac{\sum_{n=1}^N x(n) }{N}$	tp12	$\frac{tp7}{tp2}$
tp5	$\frac{\sum_{n=1}^N (x(n)-tp1)^3}{N}$	tp13	tp7
tp6	$\frac{\sum_{n=1}^N (x(n)-tp1)^4}{N}$	tp14	tp4
tp7	$\max(x(n))$	tp15	$\frac{tp5}{(tp2)^3}$
tp8	$\min(x(n))$	tp16	$\frac{tp6}{(tp2)^4}$

Fonte: Elaborado pelo autor.

Assim como no domínio do tempo, o espectro de frequência do sinal de falha também pode sofrer alterações, já que o impacto das falhas provoca ressonâncias no sistema e pode gerar componentes de alta frequência. Portanto, treze parâmetros estatísticos no domínio da frequência foram extraídos, conforme mostrado na Tabela 2 (Tang et al. 2016). Esses parâmetros incluem frequência média, frequência RMS e frequência centróide, que são úteis para capturar características ressonantes do sistema e identificar padrões de falha.

A próxima etapa envolve o diagnóstico inteligente utilizando técnicas de reconhecimento de padrões. O modelo de classificação foi treinado com as features extraídas do

Tabela 2 – Parâmetros estatísticos no domínio da frequência.

Número	Expressão da Feature	Número	Expressão da Feature
fp1	$\frac{\sum_{k=1}^K y(k)}{K}$	fp8	$\sqrt{\frac{\sum_{k=1}^K f_k^4 y(k)}{\sum_{k=1}^K f_k^2 y(k)}}$
fp2	$\sqrt{\frac{\sum_{k=1}^K (y(k) - fp1)^2}{K-1}}$	fp9	$\frac{\sum_{k=1}^K f_k^2 y(k)}{\sqrt{\sum_{k=1}^K y(k) \sum_{k=1}^K f_k^2 y(k)}}$
fp3	$\frac{\sum_{k=1}^K (y(k) - fp1)^3}{K(\sqrt{fp2})^3}$	fp10	$\frac{fp6}{fp5}$
fp4	$\frac{\sum_{k=1}^K (y(k) - fp1)^4}{K(fp2)^4}$	fp11	$\sum_{k=1}^K \frac{(f_k - fp5)^3 y(k)}{K(fp6)^3}$
fp5	$\sqrt{\sum_{k=1}^K (f_k - fp5)^2 y(k) / K}$	fp12	$\sum_{k=1}^K \frac{(f_k - fp5)^4 y(k)}{K(fp6)^4}$
fp6	$\sum_{k=1}^K \frac{f_k y(k)}{\sum_{k=1}^K y(k)}$	fp13	$\sum_{k=1}^K \frac{(f_k - fp5)^{1/2} y(k)}{K\sqrt{fp6}}$
fp7	$\sqrt{\frac{\sum_{k=1}^K f_k^2 y(k)}{\sum_{k=1}^K y(k)}}$		

Fonte: Elaborado pelo autor.

conjunto de treinamento. Posteriormente, o conjunto de teste foi utilizado para examinar e comparar o desempenho dos modelos.

A seleção de features foi inspirada em estudos anteriores que demonstraram a eficácia dessas features na detecção de falhas em maquinário rotativo. Especificamente, o artigo de [Han et al. 2018](#) foi fundamental na definição da metodologia de extração de features utilizada neste estudo ([Bogoevska et al. 2017](#)). A aplicação de features estatísticas nos domínios do tempo e da frequência é bem estabelecida na literatura e confirmada por diversos estudos como eficiente para a detecção de falhas em sistemas rotativos ([Dong et al. 2016](#)).

3.3 Treinamento e Otimização

Para garantir a construção de modelos robustos e eficazes, a aplicação de boas práticas de modelagem é essencial. Neste trabalho, empregamos técnicas de divisão do conjunto de dados, validação cruzada e *Grid Search* para otimizar os hiperparâmetros dos modelos. Cada uma dessas práticas é detalhada a seguir, com as devidas referências que sustentam nossa abordagem.

O primeiro passo na modelagem de dados é dividir o conjunto de dados em partes de treinamento e teste. Utilizamos uma divisão de 75% dos dados para treinamento e 25% para teste. Essa divisão é amplamente aceita na literatura e permite que o modelo aprenda com a maior parte dos dados disponíveis, reservando uma porção significativa para avaliar o desempenho do modelo em dados não vistos anteriormente. Essa prática ajuda a evitar o *overfitting*, quando o modelo se ajusta excessivamente aos dados de treinamento e falha

em generalizar para novos dados (Pedregosa et al. 2011, Agrawal 2021).

Para assegurar que o modelo não apenas se ajuste bem ao conjunto de treinamento, mas também generalize adequadamente para dados não vistos, foi empregada a técnica de validação cruzada. Em particular, utilizamos a validação cruzada *k-fold*, na qual o conjunto de treinamento é dividido em k subconjuntos. O modelo é treinado k vezes, cada vez usando $k - 1$ subconjuntos como dados de treinamento e o subconjunto restante como dados de validação. A média dos resultados dessas k execuções fornece uma estimativa robusta do desempenho do modelo (Fu 2011, Esling e Agon 2012, Merizio et al. 2021). Neste trabalho, utilizamos $k = 10$.

A otimização de hiperparâmetros é uma etapa vital para melhorar o desempenho de modelos de *Machine Learning*. A técnica de *Grid Search* foi utilizada para encontrar a melhor combinação de hiperparâmetros para os modelos *Random Forest* (RF), XGBoost e Redes Neurais (NN). O *Grid Search* envolve definir um conjunto de valores possíveis para cada hiperparâmetro e avaliar exaustivamente todas as combinações possíveis desses valores.

3.3.1 Otimização da *Random Forest*

Para o *Random Forest*, o *Grid Search* foi configurado com hiperparâmetros que incluíam o número de estimadores, profundidade máxima da árvore, taxa de aprendizado, número mínimo de amostras por folha, método de *bootstrap* e o número máximo de *features* consideradas para divisão. Em cada iteração, o modelo foi treinado e avaliado usando validação cruzada para cada combinação de hiperparâmetros. Os hiperparâmetros testados estão apresentados na Tabela 3.

Tabela 3 – Hiperparâmetros testados para a otimização do *Random Forest*.

Parâmetro	Valores
n_estimators	[25, 50, 100]
max_depth	[8, 10, 20]
min_samples_leaf	[10, 20, 30]
bootstrap	[True, False]
max_features	['sqrt', 5, 10]

Fonte: Elaborado pelo autor.

3.3.2 Otimização das Redes Neurais

Para Redes Neurais, foi utilizada a funcionalidade *GridSearchCV* da biblioteca *scikit-learn*, que combina de forma eficiente o *Grid Search* com validação cruzada. Foram avaliados o número de camadas ocultas, número de neurônios por camada, função de

ativação, taxa de regularização (α), otimizador (*solver*) e taxa de aprendizado. Os hiperparâmetros testados estão apresentados na Tabela 4.

Tabela 4 – Hiperparâmetros testados para a otimização da Rede Neural.

Parâmetro	Valores
hidden_layer_sizes	[(50,50), (100,50), (50,50,50), (50,50,50,50), (100,50,50)]
activation	['tanh', 'relu']
alpha	[0.001, 0.01]
solver	['adam']
learning_rate	['constant', 'adaptive']

Fonte: Elaborado pelo autor.

3.3.3 Otimização do XGBoost

Para o XGBoost, o *GridSearchCV* também foi empregado. Este método assegura uma busca exaustiva em todo o espaço de hiperparâmetros pré-definido, enquanto mantém uma avaliação robusta por meio da validação cruzada. Essa abordagem é particularmente vantajosa para modelos complexos como o XGBoost, cujos hiperparâmetros como número de árvores, profundidade das árvores, taxa de aprendizado, *subsample* e *colsample_bytree* desempenham um papel crítico no desempenho do modelo. Os hiperparâmetros testados para o XGBoost estão apresentados na Tabela 5.

Tabela 5 – Hiperparâmetros testados para a otimização do XGBoost.

Parâmetro	Valores
n_estimators	np.arange(50, 500, 50)
max_depth	np.arange(3, 10)
learning_rate	np.linspace(0.01, 0.3, 10)
subsample	np.linspace(0.5, 1.0, 6)
colsample_bytree	np.linspace(0.5, 1.0, 6)

Fonte: Elaborado pelo autor.

A aplicação uniforme dessa abordagem de otimização garantiu consistência metodológica e equidade entre todos os modelos. Essa exploração detalhada ressalta a importância de um ajuste sistemático para alcançar resultados confiáveis e eficazes.

3.4 Métricas de Avaliação e Metodologia de Desempenho

Para avaliar de forma abrangente o desempenho dos algoritmos aplicados neste estudo — Redes Neurais Artificiais, *Random Forest* e XGBoost —, utilizamos diversas métricas amplamente reconhecidas. Essas métricas proporcionam uma análise detalhada da capacidade dos modelos em distinguir corretamente entre falhas e não falhas, além de avaliar sua confiabilidade e robustez preditiva.

3.4.1 Acurácia

A *acurácia* mede a proporção de previsões corretas em relação ao total de previsões realizadas. Embora seja uma métrica intuitiva, pode ser enganosa em conjuntos de dados desbalanceados, onde uma classe é significativamente mais prevalente que a outra. Nesses casos, a acurácia pode não refletir o verdadeiro desempenho do modelo (Powers 2020). É calculada como:

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \quad (32)$$

onde:

- TP : verdadeiros positivos;
- TN : verdadeiros negativos;
- FP : falsos positivos;
- FN : falsos negativos.

3.4.2 Precisão (Precision)

A *precisão* representa a proporção de verdadeiros positivos entre todas as previsões positivas, indicando a capacidade do modelo em evitar falsos positivos. É especialmente útil em situações onde o custo de um falso positivo é elevado (Powers 2020). Calcula-se da seguinte forma:

$$\text{Precisão} = \frac{TP}{TP + FP} \quad (33)$$

3.4.3 Sensibilidade (Recall)

Também conhecida como *recall*, a *sensibilidade* mede a proporção de verdadeiros positivos corretamente identificados pelo modelo, refletindo sua eficácia em detectar casos positivos. É crucial em contextos onde a identificação de todos os casos positivos é fundamental (Powers 2020). Sua fórmula é:

$$\text{Sensibilidade} = \frac{TP}{TP + FN} \quad (34)$$

3.4.4 Especificidade

A *especificidade* indica a proporção de verdadeiros negativos corretamente identificados, avaliando a capacidade do modelo em reconhecer casos negativos. É particularmente

importante quando o objetivo é minimizar falsos positivos (Saito e Rehmsmeier 2015). Calcula-se por:

$$\text{Especificidade} = \frac{TN}{TN + FP} \quad (35)$$

3.4.5 F1-Score

O *F1-Score* é a média harmônica entre precisão e recall, proporcionando uma única métrica que equilibra ambas. É especialmente útil quando há um desbalanceamento entre as classes, pois considera tanto falsos positivos quanto falsos negativos (Saito e Rehmsmeier 2015). Sua fórmula é:

$$F1 = 2 \times \frac{\text{Precisão} \times \text{Sensibilidade}}{\text{Precisão} + \text{Sensibilidade}} \quad (36)$$

3.4.6 Curva ROC e AUC (Área Sob a Curva)

A *Curva ROC* (*Receiver Operating Characteristic*) ilustra a relação entre a taxa de verdadeiros positivos (*True Positive Rate* - TPR) e a taxa de falsos positivos (*False Positive Rate* - FPR) em diversos limiares de decisão, avaliando a capacidade do modelo em distinguir entre classes. A *Área Sob a Curva* (*AUC*) quantifica essa habilidade; um valor próximo de 1 indica excelente desempenho, enquanto um valor próximo de 0,5 sugere desempenho similar ao acaso (Fawcett 2006).

3.4.7 Log Loss

O *Log Loss* avalia a qualidade das probabilidades preditivas de um modelo, penalizando previsões incorretas com alta confiança. Um valor menor de *Log Loss* indica melhores probabilidades preditivas, sendo essencial para modelos que fornecem saídas probabilísticas (Fawcett 2006). É calculado como:

$$\text{Log Loss} = -\frac{1}{N} \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \quad (37)$$

onde:

- N : número de amostras;
- y_i : rótulo verdadeiro da amostra i ;
- p_i : probabilidade prevista para a classe positiva na amostra i .



3.4.8 Validação Cruzada e Métodos de Avaliação

Para assegurar a robustez dos resultados e evitar problemas como o *overfitting*, empregamos a técnica de validação cruzada *k-fold* com $k = 10$. Esse método divide o conjunto de dados T de tamanho N em K subconjuntos disjuntos aleatórios, onde $T_1, T_2, \dots, T_K \subset T$ e $T_i \cap T_j = \emptyset$ para todo $i \neq j$, com $T_i = N/K$. Durante o processo, cada subconjunto é utilizado uma única vez como conjunto de validação, enquanto os $K - 1$ subconjuntos restantes são utilizados para treinamento (James et al. 2013).

Na prática, o método *k-fold* é amplamente adotado devido à sua capacidade de reduzir a variância nos resultados de avaliação, proporcionando estimativas mais confiáveis da capacidade de generalização dos modelos. Como destacado por Hawkins 2004, o *overfitting* é um problema recorrente em *machine learning*, onde o modelo apresenta excelente desempenho nos dados de treinamento, mas falha ao generalizar para novos dados. A validação cruzada minimiza esse risco ao expor o modelo a diferentes subconjuntos do conjunto de dados, forçando-o a se adaptar a diversos cenários.

Em cada iteração do processo de validação cruzada, o subconjunto destinado à validação (também chamado de conjunto de teste interno) permanece completamente desconhecido para o modelo durante o treinamento, simulando um cenário de generalização para dados "novos". Esse procedimento é repetido K vezes, e a média dos resultados das K iterações é utilizada como a métrica de desempenho final. Essa abordagem é essencial para evitar vieses na avaliação do modelo e garantir que os resultados reflitam seu comportamento real em condições desconhecidas.

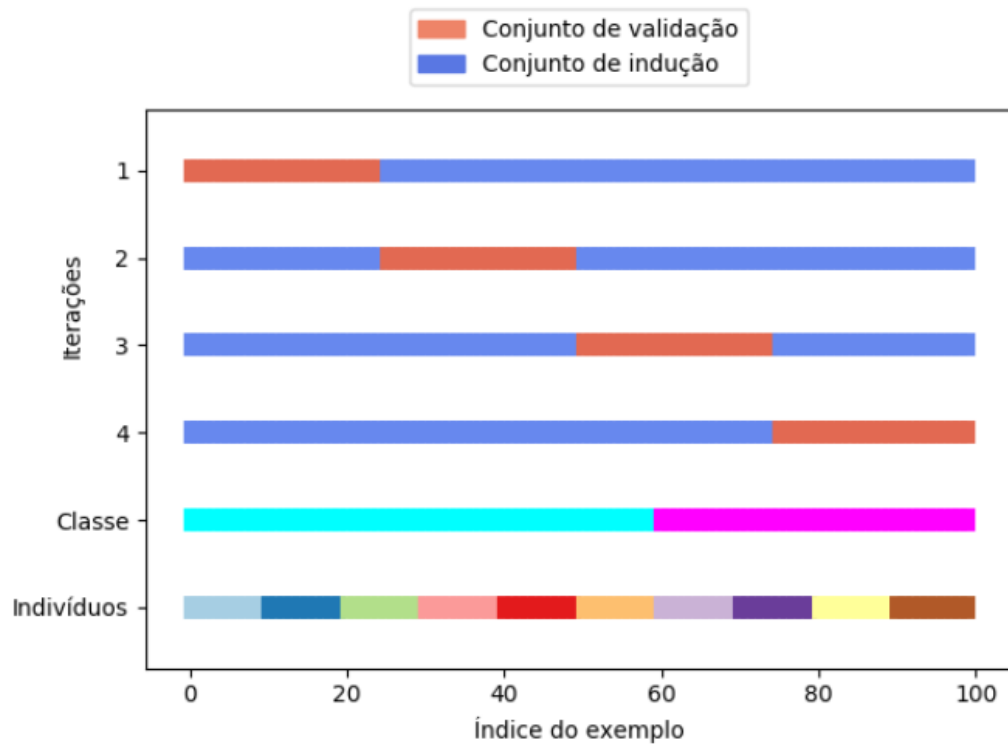
Além disso, o valor de K é estrategicamente escolhido. De acordo com Oliveira 2020, valores entre 5 e 10 são comumente utilizados, sendo $k = 10$ o mais popular devido a razões empíricas que sugerem um equilíbrio adequado entre viés e variância. Um K muito pequeno pode levar a estimativas com alta variância, enquanto valores muito altos podem aumentar o custo computacional sem melhorias significativas na confiabilidade.

A Figura 2 ilustra o procedimento de validação cruzada para $k = 4$, destacando a alternância entre os subconjuntos utilizados para treinamento e validação. Esse esquema visualiza o processo iterativo de dividir os dados em subconjuntos e medir o desempenho do modelo.

A validação cruzada também se mostrou crucial para selecionar os hiperparâmetros mais adequados. A otimização foi realizada utilizando *Grid Search*, que avalia sistematicamente todas as combinações possíveis de hiperparâmetros dentro de um espaço de busca pré-definido. Esse processo garante que as melhores configurações sejam identificadas para maximizar o desempenho do modelo em diferentes cenários.

Por fim, a metodologia descrita é fundamental para estabelecer a confiabilidade dos

Figura 2 – Exemplo da técnica de validação cruzada, com $K = 4$, utilizando 100 exemplos gerados aleatoriamente, pertencentes a duas classes e a 10 indivíduos.



Fonte: Adaptado de [Oliveira, Chavarette e Lima 2020](#).

resultados relatados neste estudo. Ao combinar a validação cruzada com testes estatísticos, conseguimos validar as diferenças de desempenho entre os modelos de forma rigorosa, assegurando que os resultados sejam generalizáveis e relevantes para aplicações industriais. A adoção dessa abordagem robusta é especialmente importante no contexto de sistemas rotativos, onde a detecção precisa de falhas desempenha um papel crítico na manutenção preditiva e na eficiência operacional.



4 RESULTADOS E DISCUSSÕES

Nesta seção, é apresentada uma análise comparativa dos resultados obtidos entre os três modelos treinados para este estudo: *Random Forest*, XGBoost e Redes Neurais. Cada modelo foi treinado e avaliado utilizando o mesmo conjunto de dados, e várias métricas de desempenho foram calculadas para comparar a eficácia de cada um. Os resultados são discutidos em termos de acurácia, *log loss*, área sob a curva ROC (ROC AUC), *precision*, *recall* e *F1 score*.

4.1 Resultados dos modelos comparados

A Tabela 6 apresenta os resultados das métricas avaliadas durante a validação cruzada *k-fold*, com $k = 10$, enquanto a Tabela 7 apresenta as métricas de desempenho avaliadas no conjunto de teste.

Tabela 6 – Métricas de avaliação na validação cruzada *k-fold*

Métrica	Rede Neural	XGBoost	<i>Random Forest</i>
fit_time	49,854	8,598	4,577
score_time	0,015	0,018	0,044
test_accuracy	0,945	0,953	0,934
train_accuracy	0,962	0,988	0,962
test_neg_log_loss	-0,139	-0,116	-0,163
train_neg_log_loss	-0,094	-0,040	-0,096
test_roc_auc	0,985	0,990	0,982
train_roc_auc	0,993	0,999	0,997
test_precision_macro	0,924	0,928	0,899
train_precision_macro	0,948	0,977	0,934
test_recall_macro	0,929	0,954	0,939
train_recall_macro	0,952	0,992	0,973
test_f1_macro	0,926	0,940	0,916
train_f1_macro	0,950	0,984	0,951

Fonte: Elaborado pelo autor.

4.1.1 *Random Forest*

O modelo *Random Forest* apresentou um tempo de treinamento de 4,577 segundos e um tempo de avaliação (*score_time*) de 0,044 segundos. A acurácia no conjunto de teste foi de 0,932, enquanto no conjunto de treinamento foi de 0,962, indicando um leve *overfitting*, mesmo com as precauções tomadas.

Tabela 7 – Métricas de avaliação no conjunto de teste

Métrica	Rede Neural	XGBoost	<i>Random Forest</i>
Acurácia	0,941	0,952	0,932
<i>Log Loss</i>	0,150	0,125	0,172
ROC AUC	0,983	0,989	0,980
<i>Precision</i> (Macro)	0,917	0,925	0,897
<i>Recall</i> (Macro)	0,929	0,955	0,938
Especificidade	0,873	0,960	0,850
F1 <i>Score</i> (Macro)	0,923	0,938	0,914

Fonte: Elaborado pelo autor.

A área sob a curva ROC (ROC AUC) para o conjunto de teste foi 0,980, demonstrando uma excelente capacidade de discriminar entre classes de falha e não falha. A *macro precision* foi 0,897, com *macro recall* de 0,938 e F1 *score* macro de 0,914. Esses resultados indicam que o modelo *Random Forest* possui um equilíbrio adequado entre *precision* e *recall*, crucial para a tarefa de detecção de falhas, onde tanto a identificação correta de falhas quanto a minimização de falsos positivos são importantes.

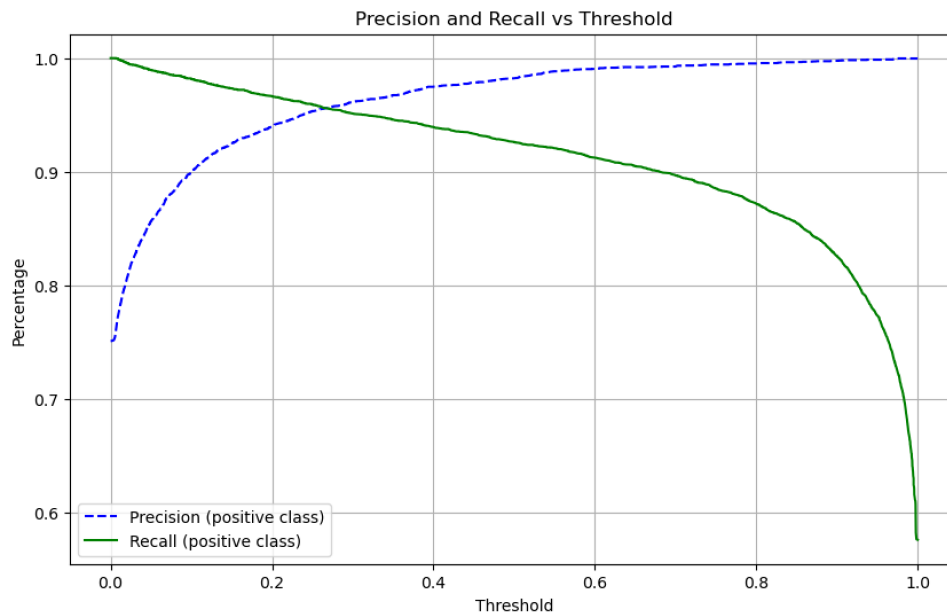
Os melhores parâmetros encontrados para o modelo *Random Forest* foram: *bootstrap: False*, *max_depth: 20*, *max_features: 10*, *min_samples_leaf: 10* e *n_estimators: 100*. A configuração desses hiperparâmetros foi essencial para atingir o equilíbrio entre *precision* e *recall*, maximizando o desempenho do modelo na detecção de falhas.

A Figura 3 ilustra a relação entre *precision* e *recall* para o modelo *Random Forest*. O comportamento do *Random Forest* destaca sua eficiência em identificar corretamente falhas (*recall*) com limiares mais baixos, mas compromete a *precision*. Isso sugere que, para aplicações onde é crucial identificar todas as falhas, o *Random Forest* pode ser ajustado com um limiar mais baixo. No entanto, se a prioridade for reduzir falsos positivos, o limiar deve ser ajustado para aumentar a *precision*, aceitando uma menor taxa de *recall*.

4.1.2 XGBoost

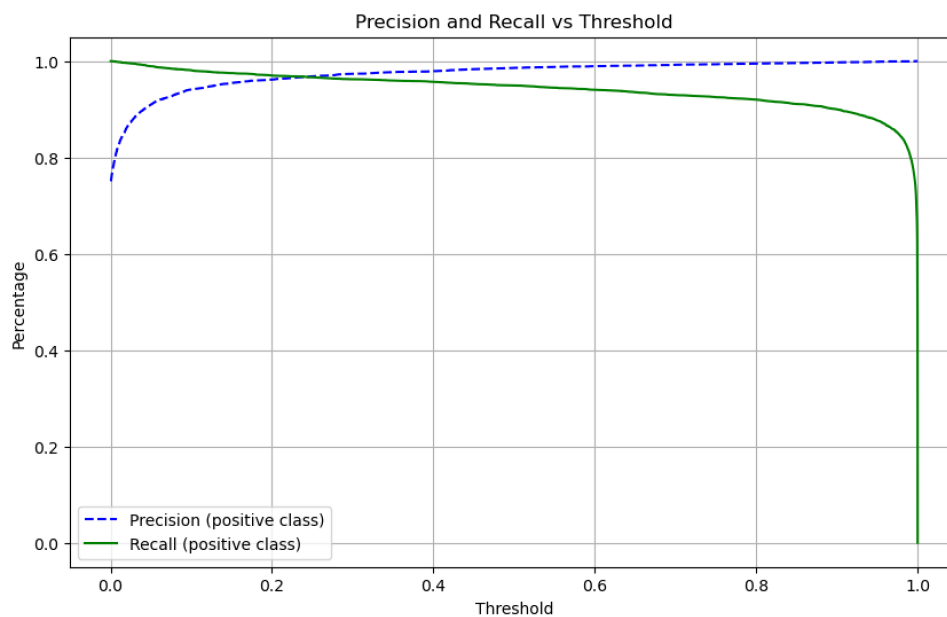
O modelo XGBoost apresentou um tempo de treinamento de 8,598 segundos e um tempo de avaliação (*score_time*) de 0,018 segundos. A acurácia no conjunto de teste foi de 0,952, enquanto no conjunto de treinamento foi de 0,988, indicando leve *overfitting*.

A área sob a curva ROC (ROC AUC) para o conjunto de teste foi 0,990, indicando uma capacidade superior de discriminar entre classes de falha e não falha. A *macro precision* foi 0,925, com *macro recall* de 0,955 e F1 *score* macro de 0,938. Esses resultados destacam o XGBoost como o modelo de melhor desempenho entre os três, especialmente em termos de *precision* e *recall*, aspectos críticos para a detecção de falhas.

Figura 3 – *Precision e Recall vs Threshold - Random Forest*

Fonte: Elaborado pelo autor.

Os melhores parâmetros encontrados para o modelo XGBoost foram: *subsample*: 0.8, *n_estimators*: 450, *max_depth*: 6, *learning_rate*: 0.0744 e *colsample_bytree*: 0.7. Esses hiperparâmetros permitiram ao XGBoost lidar de maneira eficiente com interações complexas e não lineares nos dados e prevenir o *overfitting*, resultando em desempenho superior.

Figura 4 – *Precision e Recall vs Threshold - XGBoost*

Fonte: Elaborado pelo autor.

A robustez do XGBoost em manter um bom equilíbrio entre *precision* e *recall* em

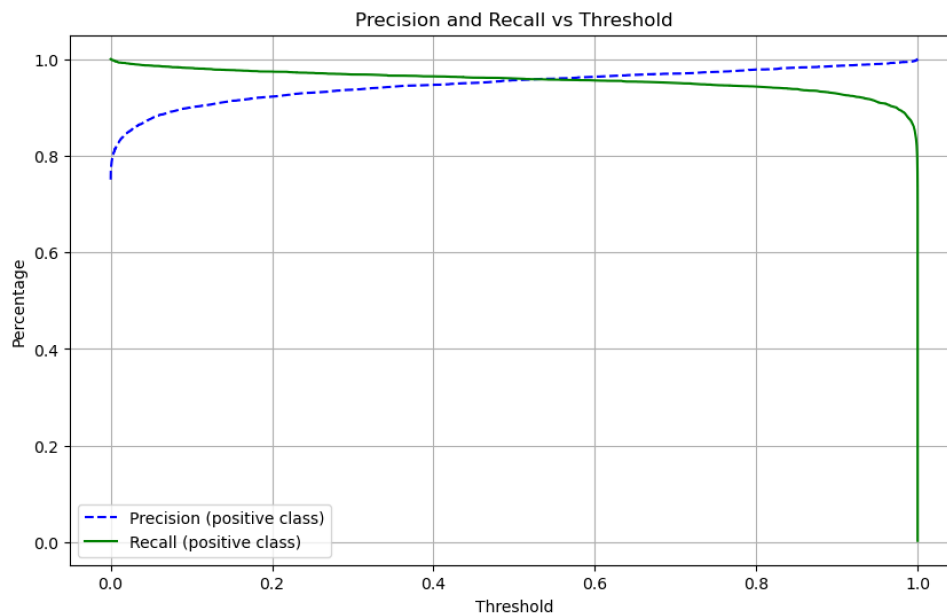
uma ampla faixa de limiares é uma vantagem significativa em aplicações industriais onde a variabilidade dos dados é alta.

4.1.3 Redes Neurais

O modelo de Redes Neurais apresentou um tempo de treinamento de 49,854 segundos e um tempo de avaliação (*score_time*) de 0,015 segundos. A acurácia no conjunto de teste foi de 0,945, enquanto no conjunto de treinamento foi de 0,962, indicando leve *overfitting*.

A área sob a curva ROC (ROC AUC) foi 0,985, indicando uma excelente capacidade de discriminar entre classes. A *macro precision* foi 0,924, com *macro recall* de 0,929 e F1 *score* macro de 0,926. Apesar de altamente eficaz, o desempenho das Redes Neurais foi ligeiramente inferior ao do XGBoost em termos de *precision* e *recall*.

Figura 5 – *Precision e Recall vs Threshold* - Redes Neurais



Fonte: Elaborado pelo autor.

Ao comparar os três modelos, observa-se que o XGBoost apresenta o melhor desempenho geral em termos de acurácia, ROC AUC, *macro precision*, *macro recall* e F1 *score* macro. O *Random Forest*, apesar de apresentar métricas ligeiramente inferiores, é uma escolha atraente para cenários industriais devido à sua simplicidade e eficiência computacional. As Redes Neurais, embora competentes, foram menos eficientes em termos de tempo de treinamento e especificidade, o que pode reduzir sua adequação em aplicações onde a identificação correta de negativos é crítica.

4.1.4 Comparação com Estudos Existentes

Para contextualizar nossos achados, comparamos os resultados obtidos com estudos existentes que aplicaram modelos de *Machine Learning* para detecção de falhas em maquinário rotativo. Nosso foco foi nos algoritmos *Random Forest* (RF), Redes Neurais (NN) e XGBoost.

O estudo de [Han et al. 2018](#) avaliou RF, NN e Máquinas de Vetores de Suporte (SVM) para diagnóstico inteligente de maquinário rotativo. Eles observaram que, embora as NNs capturassem padrões não lineares de forma eficaz, o RF demonstrou maior robustez e foi menos propenso ao *overfitting*. Similarmente, em nosso estudo, o RF exibiu eficiência e robustez, alcançando excelentes resultados com custos de treinamento significativamente mais baixos, apesar de *precision* e *recall* ligeiramente inferiores em comparação com outros modelos.

[Zhang, Wang e Habetler 2019](#) conduziram uma análise abrangente de algoritmos de aprendizado profundo para diagnóstico de falhas em rolamentos, incluindo RF, NN e Autoencoders. Eles concluíram que, apesar das vantagens das NNs em modelar relações complexas, modelos de conjunto como RF ofereciam um equilíbrio favorável entre desempenho e robustez, especialmente em cenários com dados ruidosos ou incompletos. Nossos achados corroboram essa conclusão, já que o RF lidou de forma eficaz com dados ruidosos e forneceu resultados interpretáveis, tornando-se prático para aplicações industriais.

[Amruthnath e Gupta 2018](#) exploraram algoritmos de aprendizado não supervisionado para detecção precoce de falhas em manutenção preditiva, incluindo RF e NNs. Eles destacaram que, enquanto as NNs possuem maior capacidade para capturar padrões complexos, o RF foi mais eficaz em cenários com dados não rotulados ou parcialmente rotulados, oferecendo detecção de anomalias consistente e confiável. Isso reforça a versatilidade do RF em diferentes cenários de rotulagem de dados.

O estudo de [He, Zhao e Chu 2018](#) avaliou o XGBoost em relação a outros modelos, como RF e NNs, em diversas aplicações de manutenção preditiva. Eles concluíram que o XGBoost frequentemente supera outros modelos em termos de acurácia e eficiência computacional, devido às suas capacidades de otimização e técnicas avançadas de regularização que previnem *overfitting*. Nosso estudo confirma essa observação, pois o XGBoost alcançou métricas de desempenho superiores e tempos de treinamento reduzidos, tornando-se adequado para cenários industriais que demandam detecção rápida e precisa de falhas.

[Chen e Guestrin 2016](#) demonstraram a eficácia do XGBoost em uma ampla gama de tarefas de aprendizado de máquina, incluindo detecção de falhas. Eles destacaram que o XGBoost não apenas oferece alta acurácia, mas também reduz significativamente o tempo de treinamento em comparação com NNs, tornando-o uma escolha preferida em muitos cenários industriais. Nossos resultados são consistentes com essa conclusão, pois o

XGBoost apresentou uma combinação convincente de acurácia e eficiência.

Vakharia, Gupta e Kankar 2015 compararam métodos de aprendizado supervisionado e não supervisionado para diagnóstico de falhas em rolamentos. Eles descobriram que algoritmos de aprendizado supervisionado alcançaram maior acurácia na classificação de falhas quando dados rotulados suficientes estavam disponíveis, destacando a eficácia de abordagens supervisionadas em ambientes controlados. Isso apoia nosso uso de modelos supervisionados, que entregaram alta *precision* em nosso estudo.

Por fim, Lu, Wang e Xiao 2021 propuseram um sistema de diagnóstico inteligente de falhas baseado em transferência de *features* para aprendizado não supervisionado. Seus resultados sugerem que, embora métodos não supervisionados possam detectar anomalias sem dados rotulados, eles podem não igualar a acurácia dos algoritmos supervisionados quando os rótulos estão disponíveis. Isso destaca o equilíbrio entre métodos supervisionados e não supervisionados, reforçando nossa escolha pelo aprendizado supervisionado devido à sua superioridade em acurácia na detecção de falhas com dados rotulados.

Em resumo, os achados do nosso estudo estão alinhados com a literatura existente, demonstrando que, enquanto métodos não supervisionados oferecem vantagens em cenários sem dados rotulados, algoritmos supervisionados como XGBoost e *Random Forest* fornecem maior acurácia e confiabilidade na detecção de falhas, atendendo às demandas de precisão e dependabilidade da Indústria 4.0.

4.1.5 Análise Estatística das Comparações de Modelos

Realizamos testes T pareados nas métricas de desempenho obtidas tanto na validação cruzada quanto nos conjuntos de teste para validar a significância das diferenças observadas entre os modelos (Redes Neurais, XGBoost e *Random Forest*). Os resultados indicam que o XGBoost consistentemente supera os outros dois modelos em várias métricas, incluindo acurácia, *Log Loss* e ROC AUC, com valores-p abaixo de 0,05, confirmando a superioridade estatística deste modelo sobre as Redes Neurais e o *Random Forest*.

As comparações entre XGBoost e *Random Forest* revelaram que, embora o XGBoost tenha se destacado na maioria das métricas, algumas métricas, como *Recall* e *F1 Score*, apresentaram desempenho similar entre os dois modelos, como indicado por valores-p maiores que 0,05. Por outro lado, as comparações entre as Redes Neurais e os outros modelos mostraram consistentemente diferenças estatisticamente significativas, com as Redes Neurais apresentando o desempenho mais fraco. Esses resultados são consistentes tanto nos conjuntos de validação cruzada quanto nos de teste.

5 CONSIDERAÇÕES FINAIS

Neste estudo, comparamos o desempenho de três modelos de *Machine Learning*, *Random Forest*, XGBoost e Redes Neurais, na detecção de falhas em sistemas rotativos. Os resultados indicam que o XGBoost apresenta o melhor desempenho geral, seguido pelas Redes Neurais e, por fim, pelo *Random Forest*, embora todos os modelos tenham demonstrado alta eficácia em diversas métricas de avaliação, como acurácia, ROC AUC, *precision*, *recall* e F1 *score*.

O XGBoost destacou-se em termos de acurácia, *log loss* e outras métricas, demonstrando uma robusta capacidade de discriminar entre classes de falha e não falha. Sua superioridade pode ser atribuída aos parâmetros de regularização e às técnicas avançadas de gerenciamento de variáveis, que previnem *overfitting* e melhoram a capacidade de generalização do modelo. O ajuste fino de hiperparâmetros, como *subsample* e *colsample_bytree*, foi essencial para alcançar desempenho consistente e robusto, tornando-o ideal para aplicações industriais onde acurácia e confiabilidade são cruciais.

As Redes Neurais mostraram-se consideravelmente menos eficientes em termos de tempo de treinamento para alcançar resultados similares aos outros modelos, exigindo 479,83% e 989,23% mais tempo de treinamento em relação ao XGBoost e ao *Random Forest*, respectivamente. Sendo apenas 2,71% melhores que o *Random Forest* e com resultados inferiores ao XGBoost nas métricas de avaliação, as Redes Neurais apresentaram bom desempenho geral. Uma boa busca por parâmetros, como ativação *tanh* e solver Adam, contribuiu significativamente para seus bons resultados. No entanto, a menor especificidade observada em comparação ao XGBoost implica um maior número de falsos positivos, o que pode ser um fator limitante em certas aplicações.

O *Random Forest* mostrou-se robusto e eficiente, exigindo 90,82% e 46,77% menos tempo de treinamento em relação às Redes Neurais e ao XGBoost, respectivamente, com resultados ligeiramente inferiores nas métricas de avaliação, apenas 2,63% piores na pior métrica em relação às Redes Neurais. Além disso, é um modelo que pode ser aplicado especialmente em cenários com dados ruidosos ou incompletos. Sua simplicidade e eficiência computacional, combinadas com uma boa capacidade de discriminação e equilíbrio entre *precision* e *recall*, fazem dele uma opção prática e confiável para a detecção de falhas em ambientes industriais. A menor sensibilidade de seus parâmetros e o menor tempo de treinamento tornam o *Random Forest* uma escolha atrativa para muitas aplicações onde interpretabilidade e robustez são valorizadas.

Embora o período de coleta de dados para este estudo tenha sido relativamente

curto (aproximadamente 10,07 segundos por condição), o conjunto de dados de alta resolução (48.000 amostras por segundo) forneceu informações suficientes para treinar modelos robustos e produzir resultados estatisticamente significativos. A validação cruzada rigorosa com 10 *folds* e testes T pareados garantiram a confiabilidade e a generalização das métricas relatadas, demonstrando que conjuntos de dados compactos bem processados podem gerar resultados significativos para a detecção de falhas industriais.

Este estudo buscou preencher lacunas metodológicas, implementando um processo de otimização de hiperparâmetros adaptado e adotando um framework abrangente de validação cruzada *k-fold*. Essas medidas garantiram comparações justas entre os modelos e avaliações de desempenho imparciais. Esse rigor metodológico contribui para estabelecer benchmarks confiáveis para a detecção de falhas em sistemas rotativos, enfatizando a aplicabilidade prática e a reprodutibilidade nos diagnósticos industriais.

É importante observar que as conclusões apresentadas são preliminares e baseadas em conjuntos de dados e configurações específicas. Embora o XGBoost tenha demonstrado desempenho promissor no tratamento de dados desbalanceados e o *Random Forest* tenha apresentado interpretabilidade robusta, validações adicionais com conjuntos de dados diversificados e testes estatísticos são necessárias para confirmar esses achados de forma abrangente.

Estudos futuros podem se concentrar em aprimorar técnicas de engenharia de *features*, integrar métodos de conjunto para capitalizar os pontos fortes de múltiplos algoritmos e expandir a aplicação dos modelos para outros tipos de maquinário rotativo. Além disso, a incorporação de aprendizado por transferência (*transfer learning*) pode melhorar a adaptabilidade dos modelos a novos conjuntos de dados, minimizando esforços de re-treinamento e ampliando o escopo das metodologias de detecção de falhas.

5.1 Artigos

5.1.1 Artigo publicado

Destaca-se nesta seção a contribuição científica que este trabalho visou desenvolver. O autor teve artigo publicado na revista *Engineering Computations* (ISSN: 0264-4401), com o título *Machine learning techniques for fault detection in rotating mechanical systems*, em coautoria com Fabio Roberto Chavarette e Estevão Fuzaro de Almeida. Acesse pela URL <<<https://www.emerald.com/insight/content/doi/10.1108/ec-09-2024-0896/full/html>>>.

A Figura 6 apresenta o comprovante de publicação do presente trabalho.

Figura 6 – Comprovante de publicação na revista *Engineering Computations*.

The screenshot shows the Emerald Insight website interface. At the top, there's a navigation bar with 'emerald insight' logo and 'Discover Journals, Books & Case Studies'. To the right are links for 'Browse our content', 'Publish with us', 'Register for a profile', 'Cart', and 'Login'. Below this is a search bar with the placeholder 'Enter your search terms here' and an 'Advanced search' button. The breadcrumb trail reads: 'Home / Journals / Engineering Computations / Volume 42 Issue 3 / Machine learning techniques for fault detection in rotating mechanical systems'. The article title 'Machine learning techniques for fault detection in rotating mechanical systems' is prominently displayed. Below the title, the authors 'Igor Feliciani Merizio, Fabio Roberto Chavarette, Estevão Fuzaro de Almeida' are listed. The journal 'Engineering Computations' and its ISSN '0264-4401' are also shown. Publication dates are provided: 'Article publication date: 3 April 2025' and 'Issue publication date: 11 April 2025'. A 'Permissions' button and a 'Downloads' icon with the number '20' are visible. The 'Abstract' section is titled 'Purpose' and describes the work on developing and evaluating AI models (neural networks, random forest, XGBoost) for fault detection and localization in dynamic systems, aiming to identify the most effective approach based on accuracy, precision, recall, and other metrics.

Fonte: Elaborado pelo autor.

5.1.2 Artigo aceito para publicação

Além disso, o autor teve outro artigo aceito para publicação na revista *Trends in Computational and Applied Mathematics* com título *Analysis and Numeric Formation of a Stationary Wave*.

Figura 7 – E-mail de aceite de publicação da revista *Trends*.

The screenshot shows an email interface. The subject line is '[TCAM] Editor Decision'. The sender is 'Prof. Geraldo L. Diniz via Open Journal S...' and the recipient is 'para mim, Fabio, Roberto, Luiz, Thiago'. The date and time are 'ter., 3 de dez. de 2024, 12:44'. There are icons for star, smiley, reply, and more options. A button 'Traduza para o português' is visible. The body of the email starts with 'Igor Feliciani Merizio, Fabio Chavarette, Roberto Outa, Luiz Roefero, Thiago Moro:' followed by 'We have reached a decision regarding your submission to Trends in Computational and Applied Mathematics, "Analysis and Numeric Formation of a Stationary Wave".' and 'Our decision is to: Accept Submission'. The email is signed 'Geraldo L. Diniz'.

Fonte: Elaborado pelo autor.

REFERÊNCIAS

- Agrawal 2021 AGRAWAL, T. Hyperparameter optimization using scikit-learn. *Hyperparameter Optimization in Machine Learning*, Apress, Berkeley, CA, 2021. 49
- Amruthnath e Gupta 2018 AMRUTHNATH, N.; GUPTA, T. A research on unsupervised machine learning algorithms for early fault detection in predictive maintenance. In: IEEE. *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*. [S.l.], 2018. p. 355–361. 59
- Angelopoulos et al. 2019 ANGELOPOULOS, A. et al. Tackling faults in the industry 4.0 era—a survey of machine-learning solutions and key aspects. *Sensors*, MDPI, v. 20, n. 1, p. 109, 2019. 15, 16
- Biau e Scornet 2016 BIAU, G.; SCORNET, E. A random forest guided tour. *Test*, Springer, v. 25, p. 197–227, 2016. 34, 35
- Bogoevska et al. 2017 BOGOEVSKA, S. et al. A data-driven diagnostic framework for wind turbine structures: A holistic approach. *Sensors*, v. 17, n. 4, p. 1–28, 2017. 48
- Bonissone et al. 2010 BONISSONE, P. et al. A fuzzy random forest. *International Journal of Approximate Reasoning*, Elsevier, v. 51, n. 7, p. 729–747, 2010. 36, 37
- Breiman 2001 BREIMAN, L. *Random forests*. [S.l.: s.n.], 2001. v. 45. 5-32 p. 22
- Chen 2015 CHEN, T. Xgboost: extreme gradient boosting. *R package version 0.4-2*, v. 1, n. 4, 2015. 39, 40, 41, 42
- Chen e Guestrin 2016 CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: ACM. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. [S.l.], 2016. p. 785–794. 22, 40, 42, 59
- Chen, Zhang e Gao 2021 CHEN, X.; ZHANG, B.; GAO, D. Bearing fault diagnosis base on multi-scale cnn and lstm model. *Journal of Intelligent Manufacturing*, Springer, v. 32, n. 4, p. 971–987, 2021. 22, 46
- Chu, Chew e Chang 2022 CHU, K.; CHEW, K.; CHANG, Y. Fault diagnosis system of hall sensor in brushless dc motor based on neural networks approach. In: IEEE. *2022 IEEE 18th International Colloquium on Signal Processing & Applications (CSPA)*. [S.l.], 2022. p. 35–40. 15
- Clark 1991 CLARK, J. Neural network modelling. *Physics in Medicine & Biology*, IOP Publishing, v. 36, n. 10, p. 1259, 1991. 23, 26, 30, 33
- Cutler et al. 2007 CUTLER, D. R. et al. Random forests for classification in ecology. *Ecology*, v. 88, n. 11, p. 2783–2792, 2007. 22
- Decker, Lebacher e Tresp 2023 DECKER, T.; LEBACHER, M.; TRESP, V. Does your model think like an engineer? explainable ai for bearing fault detection with deep learning. *arXiv preprint arXiv:2310.12967*, arXiv, 2023. 16

- Dong et al. 2016 DONG, Y. et al. Comparison of random forest, random ferns and support vector machine for eye state classification. *Multimedia Tools and Applications*, v. 75, n. 19, p. 11763–11783, 2016. [48](#)
- Dragomiretskiy e Zosso 2014 DRAGOMIRETSKIY, K.; ZOSSO, D. Variational mode decomposition. *IEEE Transactions on Signal Processing*, v. 62, n. 3, p. 531–544, 2014. [47](#)
- Esling e Agon 2012 ESLING, P.; AGON, C. Time-series data mining. *ACM Computing Surveys*, v. 45, n. 1, p. 12:1–12:34, 2012. [49](#)
- Farrar e Worden 2012 FARRAR, C. R.; WORDEN, K. *Structural health monitoring: a machine learning perspective*. [S.l.]: John Wiley & Sons, 2012. [44](#), [45](#)
- Fawcett 2006 FAWCETT, T. An introduction to roc analysis. *Pattern recognition letters*, Elsevier, v. 27, n. 8, p. 861–874, 2006. [52](#)
- Fu 2011 FU, T. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, v. 24, n. 1, p. 164–181, 2011. [49](#)
- Glorot e Bengio 2010 GLOTOT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. [S.l.: s.n.], 2010. p. 249–256. [21](#)
- Grinsztajn, Oyallon e Varoquaux 2022 GRINSZTAJN, L.; OYALLON, E.; VAROQUAUX, G. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, v. 35, p. 507–520, 2022. [15](#), [16](#)
- Guo et al. 2016 GUO, L. et al. Multifeatures fusion and nonlinear dimension reduction for intelligent bearing condition monitoring. *Shock and Vibration*, p. 1–10, 2016. [47](#)
- Hammerstrom 1993 HAMMERSTROM, D. Working with neural networks. *IEEE spectrum*, IEEE, v. 30, n. 7, p. 46–53, 1993. [23](#), [24](#), [30](#), [33](#)
- Han, Liang e Shi 2020 HAN, D.; LIANG, K.; SHI, P. Intelligent fault diagnosis of rotating machinery based on deep learning with feature selection. *Journal of Low Frequency Noise, Vibration and Active Control*, SAGE Publications Sage UK: London, England, v. 39, n. 4, p. 939–953, 2020. [16](#)
- Han et al. 2018 HAN, T. et al. Comparison of random forest, artificial neural networks and support vector machine for intelligent diagnosis of rotating machinery. *Transactions of the Institute of Measurement and Control*, SAGE Publications Sage UK: London, England, v. 40, n. 8, p. 2681–2693, 2018. [16](#), [46](#), [47](#), [48](#), [59](#)
- Hawkins 2004 HAWKINS, D. M. The problem of overfitting. *Journal of chemical information and computer sciences*, ACS Publications, v. 44, n. 1, p. 1–12, 2004. [53](#)
- He, Zhao e Chu 2018 HE, X.; ZHAO, K.; CHU, X. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, Elsevier, v. 212, p. 106622, 2018. [22](#), [59](#)
- Hinton, Osindero e Teh 2006 HINTON, G. E.; OSINDERO, S.; TEH, Y. W. A fast learning algorithm for deep belief nets. *Neural computation*, v. 18, n. 7, p. 1527–1554, 2006. [21](#)

- Islam e Kim 2019 ISLAM, M. M.; KIM, J. M. Automated bearing fault diagnosis scheme using 2d representation of wavelet packet transform and deep convolutional neural network. *Computers Industrial Engineering*, v. 106, p. 142–153, 2019. [22](#)
- Jalonen et al. 2023 JALONEN, T. et al. Real-time vibration-based bearing fault diagnosis under time-varying speed conditions. *arXiv preprint arXiv:2311.18547*, arXiv, 2023. [16](#)
- James et al. 2013 JAMES, G. et al. *An introduction to statistical learning*. [S.l.]: Springer, 2013. v. 112. [53](#)
- Liaw e Wiener 2002 LIAW, A.; WIENER, M. Classification and regression by randomforest. *R news*, v. 2, n. 3, p. 18–22, 2002. [22](#)
- Lu, Wang e Xiao 2021 LU, N.; WANG, S.; XIAO, H. An unsupervised intelligent fault diagnosis system based on feature transfer. *Mathematical Problems in Engineering*, Wiley Online Library, v. 2021, n. 1, p. 6686057, 2021. [60](#)
- Maier, Shupe e Fetz 2005 MAIER, M. A.; SHUPE, L. E.; FETZ, E. E. Dynamic neural network models of the premotoneuronal circuitry controlling wrist movements in primates. *Journal of computational neuroscience*, Springer, v. 19, p. 125–146, 2005. [28](#)
- Merizio et al. 2021 MERIZIO, I. F. et al. Machine learning applied in the detection of faults in pipes by acoustic means. *Journal of The Institution of Engineers (India): Series C*, Springer, v. 102, n. 4, p. 975–980, 2021. [49](#)
- Nair e Hinton 2010 NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. [S.l.: s.n.], 2010. p. 807–814. [21](#)
- Nguyen 2017 NGUYEN, L. Global optimization with descending region algorithm. *Applied and Computational Mathematics*, Science Publishing Group, v. 6, n. 4-1, p. 72–82, 2017. [31](#), [32](#)
- Nielsen 2016 NIELSEN, D. *Tree boosting with xgboost-why does xgboost win "every" machine learning competition?* Dissertação (Mestrado) — NTNU, 2016. [39](#), [42](#)
- Oliveira 2020 OLIVEIRA, B. R. d. Reconhecimento de contração prematura ventricular utilizando separação cega de fontes e comitê de máquinas bayesianas. Universidade Estadual Paulista (Unesp), 2020. [53](#)
- Oliveira, Chavarette e Lima 2020 OLIVEIRA, D. C. D.; CHAVARETTE, F. R.; LIMA, F. P. dos A. Structural health monitoring using artificial immune system/monitoramento estrutural da saúde usando sistema imunológico artificial. *Brazilian Journal of Development*, v. 6, n. 4, p. 16948–16963, 2020. [54](#)
- Pedregosa et al. 2011 PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. [49](#)
- Powers 2020 POWERS, D. M. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020. [51](#)

- Probst, Wright e Boulesteix 2019 PROBST, P.; WRIGHT, M. N.; BOULESTEIX, A. L. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, v. 9, n. 3, p. e1301, 2019. [22](#)
- Rigatti 2017 RIGATTI, S. J. Random forest. *Journal of Insurance Medicine*, American Academy of Insurance Medicine 1700 Magnavox Way, Fort Wayne, IN 46804, v. 47, n. 1, p. 31–39, 2017. [34](#), [36](#), [37](#)
- Rodriguez-Galiano et al. 2012 RODRIGUEZ-GALIANO, V. F. et al. An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, v. 67, p. 93–104, 2012. [22](#)
- Saito e Rehmsmeier 2015 SAITO, T.; REHMSMEIER, M. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, Public Library of Science San Francisco, CA USA, v. 10, n. 3, p. e0118432, 2015. [52](#)
- Sardashti e Nazari 2023 SARDASHTI, A.; NAZARI, J. A learning-based approach to fault detection and fault-tolerant control of permanent magnet dc motors. *Journal of Engineering and Applied Science*, Springer, v. 70, n. 1, p. 109, 2023. [15](#)
- Schmidhuber 2015 SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks*, v. 61, p. 85–117, 2015. [21](#)
- Schonlau e Zou 2020 SCHONLAU, M.; ZOU, R. Y. The random forest algorithm for statistical learning. *The Stata Journal*, SAGE Publications Sage CA: Los Angeles, CA, v. 20, n. 1, p. 3–29, 2020. [35](#)
- Sorsa, Koivo e Koivisto 1991 SORSA, T.; KOIVO, H. N.; KOIVISTO, H. Neural networks in process fault diagnosis. *IEEE Transactions on systems, man, and cybernetics*, IEEE, v. 21, n. 4, p. 815–825, 1991. [29](#)
- Specht et al. 1991 SPECHT, D. F. et al. A general regression neural network. *IEEE transactions on neural networks*, Citeseer, v. 2, n. 6, p. 568–576, 1991. [23](#), [30](#)
- Srivastava et al. 2014 SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, v. 15, n. 1, p. 1929–1958, 2014. [21](#)
- Tang et al. 2016 TANG, G. et al. Underdetermined blind source separation with variational mode decomposition for compound roller bearing fault signals. *Sensors*, v. 16, n. 6, p. 1–17, 2016. [47](#)
- Team 2023 TEAM, S. E. Ai in fault diagnosis for rotating machinery. *Sensors*, MDPI, v. 21, n. 12, p. 4070, 2023. [16](#)
- Vakharia, Gupta e Kankar 2015 VAKHARIA, V.; GUPTA, V.; KANKAR, P. Ball bearing fault diagnosis using supervised and unsupervised machine learning methods. *Int J Acoust Vib*, v. 20, n. 4, p. 244–250, 2015. [60](#)
- Vogels, Rajan e Abbott 2005 VOGELS, T. P.; RAJAN, K.; ABBOTT, L. F. Neural network dynamics. *Annu. Rev. Neurosci.*, Annual Reviews, v. 28, n. 1, p. 357–376, 2005. [23](#), [25](#)

- Wang e Wang 2003 WANG, S.-C.; WANG, S.-C. Artificial neural network. *Interdisciplinary computing in java programming*, Springer, p. 81–100, 2003. [23](#), [25](#), [27](#), [28](#), [29](#), [32](#)
- Zhang et al. 2020 ZHANG, C. et al. A novel intelligent fault diagnosis method based on variational mode decomposition and ensemble deep belief network. *IEEE Access*, IEEE, v. 8, p. 36293–36312, 2020. [15](#)
- Zhang, Wang e Habetler 2019 ZHANG, S.; WANG, B.; HABETLER, T. G. Deep learning algorithms for bearing fault diagnostics—a comprehensive review. *arXiv preprint arXiv:1901.08247*, 2019. [59](#)
- Zhang e Zhan 2017 ZHANG, Y.; ZHAN, Y. Machine learning in rock facies classification: An application of xgboost. In: *SEG Technical Program Expanded Abstracts 2017*. [S.l.: s.n.], 2017. p. 2141–2145. [22](#)

