UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PATRICIA LOPES SILVA

MÉTODOS DE INTELIGÊNCIA ARTIFICIAL PARA DETECÇÃO DE FALHAS INDUSTRIAIS APLICADOS EM UM SISTEMA DE MANUFATURA: UMA ANÁLISE COMPARATIVA DE DESEMPENHO

DISSERTAÇÃO

CORNÉLIO PROCÓPIO
2024

PATRICIA LOPES SILVA

MÉTODOS DE INTELIGÊNCIA ARTIFICIAL PARA DETECÇÃO DE FALHAS INDUSTRIAIS APLICADOS EM UM SISTEMA DE MANUFATURA: UMA ANÁLISE COMPARATIVA DE DESEMPENHO

Artificial intelligence methods for industrial fault detection applied on a manufacturing system: a comparative performance analysis

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná como requisito para obtenção do título de "Mestre em Engenharia Elétrica".

Orientador: Prof. Dr. Paulo R. Scalassara
Coorientador: Prof. Dr. Wagner Endo

CORNÉLIO PROCÓPIO

2024

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio

PATRICIA LOPES SILVA

MÉTODOS DE INTELIGÊNCIA ARTIFICIAL PARA DETECÇÃO DE FALHAS INDUSTRIAIS APLICADOS EM UM SISTEMA DE MANUFATURA: UMA ANÁLISE COMPARATIVA DE DESEMPENHO

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Sistemas Eletrônicos Industriais.

Data de aprovação: 22 de Fevereiro de 2024

Dr. Paulo Rogerio Scalassara, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Cristiano Marcos Agulhari, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Guilherme Serpa Sestito, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Leonimer Flavio De Melo, Doutorado - Universidade Estadual de Londrina (Uel)

AGRADECIMENTOS

Este trabalho não poderia ser concluído sem a ajuda de diversas pessoas, às quais presto minha homenagem. Certamente, esses parágrafos não abrangerão todas as pessoas que fizeram parte desta fase importante da minha vida. Portanto, desde já, peço desculpas àquelas que não estão mencionadas, mas que podem ter certeza de que fazem parte dos meus pensamentos e da minha gratidão.

Primeiramente, agradeço a Deus pelo amparo e pela vida. À minha família, pelo carinho, incentivo e apoio total em todos os momentos da minha vida. Em especial, à minha mãe, Marta Lopes, que me incentivou em todos os momentos e foi meu alicerce e força.

Aos professores envolvidos na formação e desenvolvimento da pesquisa, que me indicaram os caminhos a serem seguidos e depositaram confiança em mim. Um agradecimento especial ao meu orientador, Paulo Scalassara, pelo amparo e paciência.

A todos os professores e colegas do programa que contribuíram de forma direta e indireta para a conclusão deste trabalho.

Um agradecimento também especial aos meus amigos Dr. Cláudio Fernandes e Rafael Bernini por me apoiarem na área acadêmica e na vida, sempre me lembrando que seria capaz de concluir mais essa etapa na minha jornada.

Agradeço à CAPES pelo financiamento dos meus estudos e à UTFPR pelo apoio através do programa de pós-graduação, estrutura e investimento na educação.

Enfim, agradeço a todos que, de alguma forma, contribuíram para a realização deste trabalho.

O homem é do tamanho do seu sonho.

Fernando Pessoa

RESUMO

SILVA, Patricia Lopes. **Métodos de inteligência artificial para detecção de falhas industriais aplicados em um sistema de manufatura: uma análise comparativa de desempenho**. 2024. 103 f. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2024.

Este trabalho aborda o funcionamento de processos industriais e a detecção de falhas, visando contribuir de maneira instrutiva para a indústria moderna, especialmente no contexto da fabricação de tintas. Utilizando um ambiente simulado por software que replica fundamentalmente os cenários operacionais industriais, a pesquisa justifica sua escolha nesse setor devido à sua relevância econômica e à crescente necessidade de aprimoramentos. O objetivo central é realizar uma análise comparativa de determinadas técnicas de aprendizado de máquina, como Máquinas de Vetores de Suporte, K-ésimo Vizinho mais Próximo, Perceptron Multicamadas e Floresta Aleatória, para a detecção de falhas no processo produtivo. Simultaneamente, desenvolveu-se um banco de dados para representar o funcionamento do sistema durante o processo de fabricação, contendo variáveis que reproduzem a operacionalidade típica desse processo baseado no software explorado. A implementação dessas técnicas demonstra o comportamento da detecção de falhas no sistema industrial de manufatura, propondo-se melhorias na qualidade do processo produtivo e redução de custos operacionais. A inserção de falhas no processo, com foco especialmente no sistema de produção e mistura de tintas, expôs os modelos de detecção de falhas a diversas situações corretas ou inesperadas em seu funcionamento, proporcionando análises comparativas para o desenvolvimento de uma abordagem instrutiva e aplicável, contribuindo para o avanço e aprimoramento dos métodos em processos industriais.

Palavras-chave: Detecção de Falhas. Manutenção Preventiva. Aprendizado de Máquina. Controle de Processos.

ABSTRACT

SILVA, Patricia Lopes. **Artificial intelligence methods for industrial fault detection applied on a manufacturing system: a comparative performance analysis**. 2024. 103 p. Dissertation (Master's Degree in Electrical Engineering) – Federal University of Technology - Paraná. Cornélio Procópio, 2024.

This work addresses the functioning of industrial processes and the detection of faults, aiming to contribute in an instructive way to the modern industry, especially in the context of paint manufacturing. Using a software-simulated environment that fundamentally replicates industrial operational scenarios, the research justifies its choice in this sector due to its economic relevance and the growing need for improvements. The central objective is to carry out a comparative analysis of certain machine learning techniques, such as Support Vector Machines, Kth Nearest Neighbor, Multilayer Perceptron and Random Forest, to detect failures in the production process. Simultaneously, a database was developed to represent the functioning of the system during the manufacturing process, containing variables that reproduce the typical operation of this process based on the software explored. The implementation of these techniques demonstrates the behavior of fault detection in the industrial manufacturing system, proposing improvements in the quality of the production process and reduction of operational costs. The insertion of faults in the process, focusing especially on the paint production and mixing system, exposed the fault detection models to various correct or unexpected situations in their operation, providing comparative analyzes for the development of an instructive and applicable approach, contributing for the advancement and improvement of methods in industrial processes.

Keywords: Fault Detection. Preventive maintenance. Machine Learning. Process control.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

ABREVIATURAS

AI	Inteligência Artificial, do inglês *Artificial Intelligence*
DT	Árvore de Decisão, do inglês *Decision Tree*
FMEA	Análise de Modos de Falha e Efeitos, do inglês *Failure Mode and Effect Analysis*
IoT	Internet das Coisas, do inglês *Internet of Things*
KNN	K-ésimo Vizinho mais Próximo, do inglês *K-nearest Neighbors*
LDA	Análise Discriminante Linear, do inglês *Linear Discriminant Analysis*
MCC	Manutenção Centrada em Confiabilidade
ML	Aprendizado de Máquina, do inglês *Machine Learning*
MLP	Perceptron Multicamadas, do inglês *Multi Layer Perceptron*
MPD	Medindo os Planos de Manutenção Preditiva
NCA	Análise de Componente de Vizinhança, do inglês *Neighborhood Component Analysis*
PCA	Análise de Componentes Principais, do inglês *Principal Component Analysis*
RF	Floresta de Aleatória, do inglês *Random Forests*
SL	Aprendizagem Supervisionada, do inglês *Supervised Learning*
SVM	Máquinas de Vetores de Suporte, do inglês *Support Vector Machine*
TPM	Manutenção Produtiva Total, do inglês *Total Productive Maintenance*

SUMÁRIO

1 INTRODUÇÃO

No atual contexto de globalização, onde a indústria exerce um papel primordial na paisagem econômica, o avanço tecnológico estimula a interligação de sistemas e subsistemas industriais, em meio a um mercado global cada vez mais concorrido e amplo. Essa integração, acompanhada pelos contínuos progressos nos procedimentos de produção e inovações técnicas, é essencial para impulsionar a eficiência produtiva. A importância da indústria no crescimento econômico é destacada pela maneira como as novas tecnologias são aplicadas, resultando em mudanças substanciais e abrangentes, caracterizando a chamada Revolução Industrial. Desde o início da revolução, a divisão do trabalho tem sido crucial para influenciar a qualidade e o desempenho dos produtos, impulsionando o progresso tecnológico com a introdução de inovações como o motor a vapor, a máquina têxtil, o telégrafo e o telefone. Adicionalmente, é relevante ressaltar a introdução das linhas de produção no ciclo produtivo, viabilizando uma amplificação da eficácia, uniformização dos produtos e aprimoramento da performance industrial. Nesse cenário, a automatização industrial emerge como um elemento essencial para o desenvolvimento e a obtenção de lucro das empresas, proporcionando não apenas redução de custos, mas também uma vantagem competitiva em produtos e serviços, além de maior confiabilidade e eficácia (GóES, 2024; CAPESTRO *et al.*, 2024).

O avanço da automação industrial não apenas facilita a obtenção de informações sobre o desempenho dos sistemas, mas também amplia as possibilidades para as equipes de manutenção. Ao acessar uma variedade mais abrangente de dados em tempo real, essas equipes podem aprimorar suas estratégias de análise e planejamento. No entanto, é comum que as equipes de manutenção enfrentem desafios ao lidar com volumes substanciais de dados, os quais são frequentemente analisados de forma isolada devido às limitações das ferramentas de análise convencionais. Com a introdução da Internet das Coisas, do inglês *Internet of Things* (IoT), que simplificadamente consiste na conexão de objetos físicos por meio da incorporação de componentes eletrônicos e sensores, a capacidade desses objetos para coletar e compartilhar dados é consideravelmente ampliada. Essa convergência de tecnologias reflete uma evolução no entendimento e na gestão dos processos industriais, reforçando a importância de uma abordagem integrada para otimizar a eficiência operacional e a tomada de decisões na manutenção (PEREIRA; SIMONETTO, 2018; MIRANDA *et al.*, 2024).

Atualmente, um dos principais objetivos da manutenção preditiva é unir a capacidade de

coleta de dados a uma análise eficaz. Diferentemente da manutenção preventiva, a manutenção preditiva busca realizar intervenções de manutenção apenas quando necessárias, fundamentando-se na condição dos equipamentos.

Diante desse cenário, os métodos de manutenção preventiva, preditiva e corretiva desempenham papéis distintos no enfrentamento das falhas. A manutenção preventiva atua na prevenção de falhas por meio de inspeções e intervenções regulares, visando evitar que as condições propícias para o surgimento de problemas se estabeleçam. Já a manutenção preditiva utiliza dados e análises para identificar padrões de falhas iminentes, permitindo intervenções antes que ocorram danos significativos. Por fim, a manutenção corretiva entra em ação após a falha já ter ocorrido, focando na restauração do sistema ao seu estado funcional. Essas abordagens trabalham em conjunto para minimizar o impacto das falhas e garantir a continuidade das operações industriais. Independentemente da causa, essas falhas são caracterizadas por sua natureza abrupta e instantânea (ZAYTOON; LAFORTUNE, 2013).

Diversos tipos de falhas podem afetar o funcionamento dos sistemas industriais. Falhas sensoriais, como deslocamento ou mau funcionamento de sensores, resultam em discrepâncias entre os valores medidos e os valores reais das variáveis do sistema. Falhas nos atuadores, como emperramento ou desligamento, causam diferenças entre os comandos de entrada e a saída real dos atuadores. Problemas na planta, como vazamentos ou obstruções em tubulações, alteram a dinâmica do sistema. Por fim, falhas na implementação ou execução do controlador, seja por problemas de hardware ou software, também podem ocorrer, impactando diretamente o desempenho do sistema. Essas falhas exigem uma abordagem cuidadosa e estratégias de manutenção adequadas para garantir a operação eficiente e segura dos processos industriais (ZAYTOON; LAFORTUNE, 2013; CAPESTRO *et al.*, 2024).

Quando as falhas no ambiente industrial se apresentam, o avanço na aplicação de sistemas inteligentes torna-se primordial para a eficiência e a competitividade da indústria. Isso promove uma gestão mais eficaz dos processos, identificação precoce de problemas e otimização das estratégias de manutenção. Ao integrar tecnologias avançadas, como Inteligência Artificial, do inglês *Artificial Intelligence* (AI) e Aprendizado de Máquina, do inglês *Machine Learning* (ML), esses sistemas capacitam as empresas a tomar decisões mais rápidas e precisas, melhorando a eficiência operacional, reduzindo custos e aumentando a qualidade dos produtos. Além disso, permitem uma análise mais profunda dos dados em tempo real, possibilitando a identificação precoce de falhas e a implementação de medidas corretivas de forma proativa. Com

a aplicação desses sistemas inteligentes, a identificação de falhas e o processo de manutenção desenvolveram-se significativamente, tornando-se mais eficazes e assertivos. Isso contribui não apenas para minimizar o tempo de inatividade e os custos de manutenção, mas também para aumentar a confiabilidade e a disponibilidade dos sistemas industriais, impulsionando a competitividade das empresas no mercado global. Essa abordagem inovadora não apenas impulsiona a competitividade das empresas no mercado global, mas também abre caminho para uma produção mais sustentável e responsável (CAPESTRO *et al.*, 2024; OTANI *et al.*, 2024).

A indústria desempenha um papel dominante na economia global, impulsionada por contínuos testes, inovações técnicas e avanços tecnológicos. A crescente interconexão de sistemas e subsistemas industriais, em um mercado cada vez mais competitivo, destaca a importância da automação industrial. Este estudo visa examinar a relevância da automação e seu impacto na manutenção preditiva, aproveitando a abundância de dados disponíveis. A integração de sistemas inteligentes e técnicas de ML é essencial para aprimorar a eficiência operacional e a confiabilidade na indústria contemporânea, permitindo uma detecção mais precisa de falhas e uma resposta mais rápida às mudanças no ambiente industrial. A análise subsequente dos resultados e a classificação de falhas proporcionarão contribuições valiosas para o aprimoramento contínuo dos sistemas industriais (FEDULLO *et al.*, 2022).

Neste estudo, propõe-se o desenvolvimento de um sistema de detecção de falhas em um ambiente de manufatura baseado em software que apresenta ambientes industriais, empregando técnicas de Inteligência Artificial. O sistema será projetado para analisar informações coletadas por sensores e atuadores distribuídos pelo sistema, identificando padrões que possam indicar a ocorrência de falhas.

1.1 MOTIVAÇÃO

A eficiência e excelência nos processos industriais são diretamente influenciadas pela implementação eficaz da Indústria 4.0, através da aplicação da linguagem de máquina para monitorar e identificar precocemente possíveis falhas operacionais. Esse avanço tecnológico, que envolve a integração de sistemas conectados e computação, desempenha um papel crucial na otimização de processos, proporcionando maior automação, análise de dados em tempo real e tomada de decisões mais assertiva. A admissão da Indústria 4.0 não apenas desenvolve a detecção precoce de falhas, mas também motiva a eficiência, reduzindo custos operacionais e elevando a competitividade da indústria em um cenário global. No contexto específico fabril, a

pesquisa contribuirá demonstrando como a identificação precoce de falhas por meio de algoritmos pode prevenir problemas de qualidade, reduzir retrabalho e desperdício de material, além de minimizar o tempo de inatividade do sistema. Destaca-se, assim, a importância dessa abordagem na mitigação da necessidade de manutenções corretivas que prejudicam adversamente a eficiência industrial.

1.2 OBJETIVOS

O objetivo geral da dissertação constitui na utilização de AI através dos algoritmos como Máquinas de Vetores de Suporte, do inglês *Support Vector Machine* (SVM), Perceptron Multicamadas, do inglês *Multi Layer Perceptron* (MLP), K-ésimo Vizinho mais Próximo, do inglês *K-nearest Neighbors* (KNN) e Floresta de Aleatória, do inglês *Random Forests* (RF) para percepção de falhas de controle no ambiente industrial analisado, simulado por software, sendo capaz de identificar um padrão anormal de informações fornecidas pelo sistema de controle.

1.2.1 Objetivos Específicos

- Desenvolver e contrastar o desempenho de algoritmos de AI, avaliando sua eficácia na detecção de possíveis falhas em um contexto industrial;
- Explorar e examinar estruturas, assim como técnicas e metodologias, pretendendo identificar e conter falhas de forma eficiente;
- Replicar o funcionamento regular do sistema industrial de manufatura simulado por software e introduzir estrategicamente falhas, de modo que os algoritmos possam discernir e identificar essas anomalias.

1.3 ESTRUTURA DA DISSERTAÇÃO

O Capítulo 2 apresenta a revisão teórica, incluindo Indústria 4.0, ML e Aprendizagem supervisionada.

O Capítulo 3 discute os conceitos relacionados a falhas no sistema, examinando a abordagem presente na literatura e analisando como a definição e identificação de falhas se desdobram no processo de manufatura. Além disso, são abordados os tópicos de manutenção preventiva, preditiva e corretiva.

Já o Capítulo 4 apresenta formalmente os conceitos relacionados aos métodos de detecção de falhas, destacando a aplicação de técnicas como SVM, MLP, KNN com a utilização de redução de dimensionalidade, e a abordagem de RF.

No Capítulo 5 será introduzido o sistema proposto, a identificação e correção de falhas no sistema. Além disso, será apresentada a formação do banco de dados, que fornecerá suporte para as análises realizadas. O capítulo incluirá também a exibição de plantas e aplicações que evidenciam a funcionalidade da proposta desenvolvida.

No Capítulo 6, serão apresentados os resultados e discussões, evidenciando o desempenho de cada método abordado na dissertação, a saber, KNN, SVM, RF e MLP. Posteriormente, no Capítulo 7 será realizado à conclusão, sintetizando os principais achados e encerrando de maneira sistemática o escopo do estudo.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão desenvolvidas as definições de alguns conceitos relacionados à Indústria 4.0. Tais temas são necessários para compreensão e entedimento do trabalho apresentado. Estes conceitos serão primeiramente exibidos.

2.1 INDÚSTRIA 4.0

O despertar da primeira Revolução Industrial ocorreu na Inglaterra entre os anos 1760 e 1840. Nessa época, houve uma gradual substituição dos métodos artesanais por tecnologias mecânicas, o aproveitamento do carvão como fonte energética em substituição à madeira e outros biocombustíveis, e o aumento notório do emprego de energia derivada do vapor. As transformações nos procedimentos produtivos acarretaram consequências impactantes nos âmbitos econômico e social.O trabalhador manual habilidoso, que em tempos passados tinha total controle sobre todo o processo de produção desde a extração da matéria-prima até a comercialização do produto final, viu-se agora subordinado a um empregador que governava todas as etapas, incluindo matéria-prima, produto final e ganhos (COELHO, 2016; OTANI *et al.*, 2024).

No período subsequente, até o desfecho da Segunda Guerra Mundial em 1945, testemunharam-se progressos notáveis nas esferas das indústrias química, elétrica e siderúrgica, acompanhados de uma otimização expressiva das técnicas vigentes. Emergiram embarcações confeccionadas em aço impulsionadas por motores a vapor potentes, transformando de maneira revolucionária o transporte de mercadorias, enquanto as pioneiras linhas de produção possibilitaram a fabricação em massa a custos reduzidos. Essa fase histórica representa a segunda Revolução Industrial, caracterizada por uma estreita interligação entre criação e inovação (COELHO, 2016).

Entre as décadas de 1950 e 1970, começou a se esboçar o que viria a ser reconhecido como a terceira Revolução Industrial: a revolução digital. Semicondutores, computadores, automação e robotização nas linhas de produção, a manipulação e processamento digital de dados, as telecomunicações, os dispositivos móveis e a rede mundial de computadores passaram a desempenhar funções preponderantes. No alvorecer do século XXI, com o florescimento da internet, a evolução de sensores cada vez mais compactos e potentes a preços mais acessíveis, o refinamento contínuo de software e hardware, e a capacidade aprimorada das máquinas para aprender e cooperar, deu-se início a uma metamorfose na indústria. O impacto dessa mutação

na competitividade, na sociedade e na economia é tão profundo que está redefinindo o mundo de forma substancial. Essa transição foi atualizada pelos professores Erik Brynjolfsson e Andrew McAfee do Instituto de Tecnologia de Massachusetts como a "segunda era da máquina", e em 2011, na Feira Industrial de Hannover, na Alemanha, já se cogitava sobre a "Indústria 4.0" (PEREIRA; SIMONETTO, 2018).

A antecipação de falhas, a capacidade de autoconfiguração e a adaptação a mudanças são atributos fundamentais na esfera da Indústria 4.0. Essas características emergem da interligação entre sensores, ambientes laborais, maquinários e sistemas de tecnologia da informação, empregando protocolos da Internet. Tal integração visa aprimorar a eficiência e promover a redução de custos nos procedimentos industriais (RüSSMANN *et al.*, 2015).

Com avanços tecnológicos constantes, os sistemas estão se tornando cada vez mais inteligentes, rápidos e precisos. No entanto, é importante reconhecer que nem todas as empresas têm condições de implementar totalmente a Indústria 4.0 em seus sistemas devido a limitações financeiras, tecnológicas ou outros fatores (DALLAGNOL *et al.*, 2022).

Em Otani *et al.* (2024), observa-se o último relatório industruial publicado onde abordou a influência da Indústria 4.0 no contexto brasileiro. A publicação destaca que uma parcela significativa da indústria nacional está atravessando a transição entre a segunda e a terceira revoluções industriais, isto é, entre a adoção de linhas de montagem e a implementação de sistemas automatizados. Segundo o documento, o setor automotivo lidera esse movimento em direção à Indústria 4.0, com profissionais constantemente atualizados para atender às demandas. Destaca-se que a mão de obra qualificada desse setor pode ser direcionada para outros segmentos. O aprimoramento da competitividade da indústria brasileira em escala global pode ser potencializado através da digitalização, impulsionando a economia e criando uma predisposição para a adoção das tecnologias da Indústria 4.0 no cenário nacional (OTANI *et al.*, 2024; ALVAREZ-AROS; BERNAL-TORRES, 2021).

A implementação da Indústria 4.0 no contexto brasileiro apresenta desafios como: (OTANI *et al.*, 2024)

- a formulação de políticas estratégicas e incentivos governamentais;
- a colaboração ativa entre empresários e gestores;
- o progresso tecnológico e a formação de profissionais alinhados com as necessidades da indústria.

Para as empresas que não têm recursos para adotar completamente a Indústria 4.0, ainda existem formas de aproveitar os benefícios da tecnologia. A inteligência artificial, por exemplo, pode ser aplicada de maneira pontual para melhorar processos específicos, mesmo que não seja possível uma implementação abrangente. Além disso, a colaboração entre humanos e robôs desempenha um papel essencial. Embora os robôs possam automatizar tarefas, ainda é necessário o envolvimento humano na programação e supervisão desses sistemas. Essa colaboração pode reduzir custos, aumentar a produtividade e substituir trabalhos perigosos e repetitivos, trazendo benefícios para a empresa e para os trabalhadores (PEREIRA; SIMONETTO, 2018).

Em resumo, embora a implementação completa da Indústria 4.0 seja desejável, é importante reconhecer as limitações enfrentadas por algumas empresas. Ainda assim, é possível aproveitar os avanços tecnológicos de forma gradual e estratégica, buscando melhorias específicas e explorando a colaboração entre humanos e robôs para impulsionar a eficiência, produtividade e segurança na indústria (AMR, 2022).

Concluindo esta seção dedicada à Indústria 4.0, é evidente a imersão em um cenário de transformação revolucionária na maneira como as operações industriais são concebidas e executadas. A interconexão de dispositivos, a digitalização de processos e a automação inteligente delineiam um novo paradigma que promete redefinir a eficiência, a competitividade e a sustentabilidade nas indústrias.

Dando continuidade, a próxima seção direcionará nosso foco para a Inteligência 4.0, uma evolução essencial para o desenvolvimento da Indústria 4.0. Explorando como a AI, ML e análise de dados convergem para potencializar as capacidades das operações industriais.

2.2 INTELIGÊNCIA ARTIFICIAL

A AI é uma área de pesquisa relativamente recente, originada no período pós-Segunda Guerra Mundial. Atualmente, abrange uma ampla gama de subcampos, englobando desde áreas de aplicação geral, como aprendizado e percepção, até tarefas mais específicas, como jogos de xadrez, demonstração de teoremas matemáticos, criação poética e diagnóstico de doenças. A inteligência artificial visa a sistematização e automação de tarefas intelectuais, sendo, portanto, potencialmente relevante para qualquer domínio da atividade intelectual humana. Nesse sentido, ela se configura como um campo de aplicação universal (RUSSELL; NORVIG, 2013).

Quando se trata de Inteligência Artificial, é uma tarefa desafiadora definir com precisão, pois ao longo do tempo surgiram quatro abordagens principais para lidar com essa área de

estudo (GOMES, 2010):

A) Sistemas que imitam a cognição humana: Os sistemas que imitam a cognição humana são um esforço empolgante e inovador para criar computadores capazes de pensar e agir como seres humanos, possuindo mentes, de maneira literal e completa. Essa área de pesquisa busca desenvolver inteligência artificial avançada, na qual as máquinas são projetadas para realizar tarefas complexas que exigem raciocínio, aprendizado, tomada de decisões e interação com o ambiente de forma semelhante aos seres humanos. O objetivo é alcançar um nível de capacidade cognitiva que se assemelhe às habilidades humanas, permitindo que os sistemas compreendam, interpretem e respondam a informações de maneira sofisticada e adaptativa. Essa abordagem revolucionária tem o potencial de transformar diversos setores, como medicina, finanças, transporte e muito mais, ao proporcionar soluções computacionais poderosas e eficazes que rivalizam com as habilidades humanas (HAUGELAND, 1985).

B) Sistemas que executam tarefas como seres humanos: Essa abordagem busca desenvolver sistemas de inteligência artificial que possam executar tarefas complexas, como raciocinar, aprender, tomar decisões e interagir com o ambiente de forma semelhante aos seres humanos. O objetivo é capacitar as máquinas a realizar essas funções com eficiência e precisão, ampliando o leque de possibilidades em diversos setores, desde medicina e finanças até transporte e muitas outras. Essa área de estudo é fundamental para a criação de soluções computacionais avançadas, que imitam as habilidades cognitivas humanas (KURZWEIL, 2000).

C) Sistemas que usam a racionalidade para pensar: Essa área de estudo abrange a investigação das habilidades mentais por meio da utilização de representações computacionais. O objetivo é desenvolver sistemas que possam raciocinar, tomar decisões e resolver problemas de forma lógica e inteligente, de maneira semelhante à mente humana. Esses modelos computacionais são projetados para simular processos cognitivos, permitindo uma compreensão mais profunda das capacidades mentais e a criação de sistemas inteligentes que possam lidar com tarefas complexas. Ao explorar a racionalidade no pensamento, busca-se melhorar a eficiência e a eficácia das soluções computacionais em diversos domínios, como medicina, ciência, negócios e muitos outros (CHARNIAK; GOLDMAN, 1993).

D) Sistemas que usam a racionalidade para agir: A Inteligência Computacional é uma área de estudo que se dedica ao design de agentes inteligentes. Esses agentes são sistemas capazes de utilizar a racionalidade para agir de forma inteligente em diferentes situações. O objetivo da Inteligência Computacional é desenvolver métodos e algoritmos que permitam aos agentes tomar decisões baseadas em lógica e raciocínio, buscando otimizar o desempenho e alcançar objetivos específicos. Esses sistemas podem ser aplicados em diversos campos, como automação industrial, robótica, jogos, medicina e muitos outros, oferecendo soluções inteligentes e adaptativas para problemas complexos (POOLE *et al.*, 1998).

Em termos gerais, as abordagens A e C estão relacionadas ao processo de pensamento e raciocínio, enquanto as abordagens B e D se concentram no comportamento dos sistemas. Além disso, as abordagens A e B avaliam o sucesso com base na semelhança com o desempenho humano, enquanto as abordagens C e D avaliam o sucesso em relação a um conceito ideal de inteligência, chamado de racionalidade. Um sistema é considerado racional quando toma todas as decisões corretas com base nos dados disponíveis (RUSSELL; NORVIG, 2013).

A AI é um sistema altamente eficiente e capaz de executar uma variedade de tarefas. Sua eficiência não se limita apenas à rapidez na realização de tarefas, mas também permite que os sistemas simulem uma inteligência semelhante à humana e até mesmo vão além disso. Segundo um relatório de pesquisa de 2021 da Accenture, uma empresa com ampla atuação e oferta de soluções em estratégia de negócios, tecnologia e operações, 84% dos executivos reconhecem a necessidade de usar a AI para alcançar seus objetivos de desenvolvimento. No entanto, 76% enfrentam dificuldades ao implementar a AI em seus negócios. Até o momento, nenhum projeto de teste de conceito foi implantado em produção e em escala, e muitas empresas estão enfrentando desafios para realizar essa transição. Nesse ponto de transição, é crucial que as empresas tomem as medidas necessárias para amadurecer e se desenvolver (DALLAGNOL *et al.*, 2022).

No encerramento desta seção dedicada à AI, fica evidente o panorama abrangente das aplicações e potencialidades dessa disciplina em diversas esferas da atividade humana. A AI emerge como uma ferramenta poderosa, capaz de automatizar tarefas intelectuais e transformar a maneira como enfrentamos desafios complexos.

Ao introduzir-se a próxima seção desta dissertação, será apresentado uma vertente específica e profundamente conectada à AI: o aprendizado de máquina, mais conhecido como ML. Explorando as particularidades desse campo dinâmico, onde os sistemas têm a capacidade

não apenas de executar tarefas pré-programadas, mas também de aprender e evoluir com base em dados.

2.3 *MACHINE LEARNING*

Machine Learning (ML) faz parte do campo da AI e se concentra em em capacitar sistemas para aprenderem de forma autônoma. Este campo é extensamente analisado devido ao crescimento exponencial de informações disponíveis, o que torna a extração manual de dados custosa em termos de recursos financeiros e tempo (RIBEIRO, 2021).

Os algoritmos de aprendizagem de máquina atuam formando modelos com base em exemplos de entrada, e usam esses modelos para fazer previsões com base nos dados em vez de apenas seguir regras estáticas e inflexíveis. Isso permite que as máquinas extraiam informações e forneçam saídas que contenham algum nível de processamento, tornando essas informações relevantes e úteis no contexto em que são aplicadas. Em resumo, a aprendizagem de máquina capacita as máquinas a aprenderem com os dados e a tomarem decisões informadas com base nesse aprendizado (RIBEIRO, 2021).

Conforme destacado por Alpaydin (2020), algumas instâncias de aplicações do aprendizado de máquina incluem:

- Aprendizado de Associações: Utilizado na análise de compras para identificar associações entre os produtos mais obtidos pelos consumidores.
- Classificação: Aplicado na avaliação do score de crédito dos clientes, categorizando-os em grupos de alto ou baixo risco.
- Reconhecimento de Padrões: Engloba aplicações como o reconhecimento de caracteres óticos e facial.
- Regressão: Empregado na predição do valor de veículos usados.
- Aprendizado por Reforço: Sistemas capazes de aprender com ações passadas, gerando assim novas estratégias.

Esses são apenas alguns exemplos, havendo diversas outras aplicações que demonstram a versatilidade e amplitude do aprendizado de máquina na resolução de variados problemas.

A aprendizagem de máquina destaca-se como um dos fundamentais impulsionadores da transformação de um sistema industrial convencional para o nível da Indústria 4.0. Isso envolve aplicar a ML para identificar padrões como imagens, processar linguagem natural, aprimorar operações, extrair informações de dados e fazer descobertas de conhecimento (LU, 2017) (WUEST *et al.*, 2016). Desde então, o desenvolvimento e pesquisa nessa área especifica tem crescido, com um aumento significativo no número de artigos publicados.Especialmente na última década, o estado da arte das técnicas de ML deu um enorme salto em frente, como demonstrado pelos algoritmos utilizados pelos carros de condução autónoma ou pelos jogos eletrónicos de estratégia (MARTíNEZ-DíAZ; SORIGUERA, 2018). Além disso, alguns pontos de destaque são as iniciativas governamentais, como a Indústria 4.0 na Alemanha, a Fábrica Inteligente na Coreia do Sul e o Smart Manufacturing nos Estados Unidos, estão impulsionando uma mudança fundamental no modo como a produção é realizada, enfatizando a importância das Tecnologias de Informação para melhorar e otimizar os processos industriais (BERTOLINI *et al.*, 2021).

A aprendizagem de máquina destaca-se como um dos fundamentais impulsionadores da transformação de um sistema industrial convencional para o nível da Indústria 4.0. Isso envolve aplicar a ML para identificar padrões como imagens, processar linguagem natural, aprimorar operações, extrair informações de dados e fazer descobertas de conhecimento (LU, 2017). Especialmente na última década, o estado da arte das técnicas de ML deu um enorme salto em frente, como demonstrado pelos algoritmos utilizados pelos carros de condução autónoma ou pelos jogos eletrónicos de estratégia (MARTíNEZ-DíAZ; SORIGUERA, 2018). Além disso, alguns pontos de destaque são as iniciativas governamentais, como a Indústria 4.0 na Alemanha, a Fábrica Inteligente na Coreia do Sul e o Smart Manufacturing nos Estados Unidos, estão impulsionando uma mudança fundamental no modo como a produção é realizada, enfatizando a importância das Tecnologias de Informação para melhorar e otimizar os processos industriais (BERTOLINI *et al.*, 2021).

A dissertação se baseia em dois pilares essenciais: aprendizagem de máquina e aprendizagem supervisionada. O primeiro consiste na compreensão da aprendizagem de máquina, fundamental para o contexto geral da pesquisa. Em seguida, o segundo pilar aborda a aprendizagem supervisionada, incluindo suas principais técnicas e funcionamento, que serão fundamentais para as análises subsequentes.

2.3.1 APRENDIZAGEM SUPERVISIONADA

Dentro do amplo campo de ML, Aprendizagem Supervisionada, do inglês *Supervised Learning* (SL) desempenha um papel central e é amplamente aplicada em uma variedade de domínios. Nesta subseção, os princípios fundamentais da SL serão explorados, discutindo como ela funciona, suas principais técnicas e o papel crucial que desempenha na capacitação das máquinas para fazer previsões e classificações com base em exemplos rotulados. Os conceitos essenciais que sustentam essa abordagem serão desvendados, proporcionando uma base sólida para o entendimento das seções subsequentes.

A aprendizagem de máquina pode ser segmentada em dois fundamentais: aprendizagem supervisionada e não supervisionada. A dissertação se concentra no conceito de SL, onde o sistema necessita ter conhecimento do ambiente em que está inserido. Esse domínio se da atráves dos dados de entrada e saída que são inseridos através de uma sequência de instruções que o algoritmo seguirá para alcançar o objetivo desejado. Na aprendizagem não supervisionada, não há exemplos especificados da função a ser aprendida (SANTOS, 2002).

A SL ou também nomeada aprendizado preditivo, engloba uma variedade de algoritmos, sendo os mais comuns: Redes Neurais, Máquinas de Vetores de Suporte, Árvores de Decisão (bem como suas extensões, como Florestas Aleatórias e *XGBoost*), Regressão Logística e Classificadores *Naïve Bayes*. Apesar das variações na forma de operação e na execução, todos esses métodos na SL compartilham um objetivo central: aprender uma representação precisa da relação real entre os dados de entrada e os dados de saída. Isso é realizado ao utilizar informações contidas em um conjunto de exemplos de treinamento, que podem ser obtidos por meio de experimentação ou observação direta do fenômeno em análise (BERTOLINI *et al.*, 2021).

Esse conjunto de exemplos, então, divide-se em dois grupos: o conjunto de "treinamento" usado para construir o modelo, ajustando-o constantemente por meio da minimização de uma função de custo (ou perda) predefinida. O conjunto de "teste" avalia a precisão do modelo, verificando seu comportamento em dados não usados durante o treinamento (BERTOLINI *et al.*, 2021).

A importância de SL na indústria é substancial. SL é vital para melhorar os processos de fabricação e a qualidade do produto. Na dissertação em questão, a aplicação da Aprendizagem Supervisionada é particularmente relevante. Um modelo do tipo de aprendizagem acima será desenvolvido para investigar possíveis falhas no processo de fabricação. Modelos do tipo são

treinados e alimentados com dados únicos por dados históricos sobre o funcionamento normal do processo de fabricação. Com base nesse treinamento, o algoritmo desenvolve um entendimento do comportamento normal dos padrões e relações esperados entre as variáveis nível e cor. Durante a produção normal, ele identifica esses padrões e relações, quando eles mudam para se tornar imprevisíveis, o algoritmo identifica isso como uma anomalia (VALTONEN *et al.*, 2021).

Essa detecção precoce de anomalias no caso designadas falhas serão apresentadas no próximo capítulo mas se torna vital para a indústria, pois permite a intervenção antes que problemas maiores ocorram, resultando em economia de recursos e manutenção da qualidade do produto. Além disso, a SL é uma abordagem flexível que pode ser aplicada a uma ampla gama de processos de fabricação, tornando-a uma ferramenta versátil para melhorias contínuas na indústria.

Em conclusão, o entendimento da SL desempenha um papel fundamental na identificação de falhas em sistemas e processos. No próximo capítulo, serão exploradas as definições de falhas e técnicas de manutenção, enfatizando a importância de evitar a manutenção corretiva por meio de estratégias proativas baseadas em dados e aprendizado de máquina.

3 FALHAS E MANUTENÇÕES

Este capítulo tem como objetivo estabelecer o conceito de falhas em sistemas controlados, apresentando as potenciais falhas operacionais identificadas pela literatura, juntamente com estratégias para detecção e tratamento eficaz. No decorrer do capítulo, também serão explorados os tipos de manutenção existentes como preventiva, preditiva e corretiva e como a identificação de tais falhas assegura um bom funcionamento na indústria.

3.1 DEFINIÇÃO DE FALHA

As principais técnicas empregadas para identificar falhas em sistemas são classificadas com base nos tipos mais comuns de falhas e nos métodos utilizados para detectá-las (ZAYTOON; LAFORTUNE, 2013). As falhas podem ser entendidas como as causas fundamentais por trás de mudanças indesejadas no funcionamento de um sistema ou de suas partes. Até agora, as técnicas de diagnóstico de falhas têm como objetivo realizar três tarefas principais: identificar falhas, isolar sua origem e avaliar como essas falhas afetam o desempenho do sistema (NATUCCI, 2016).

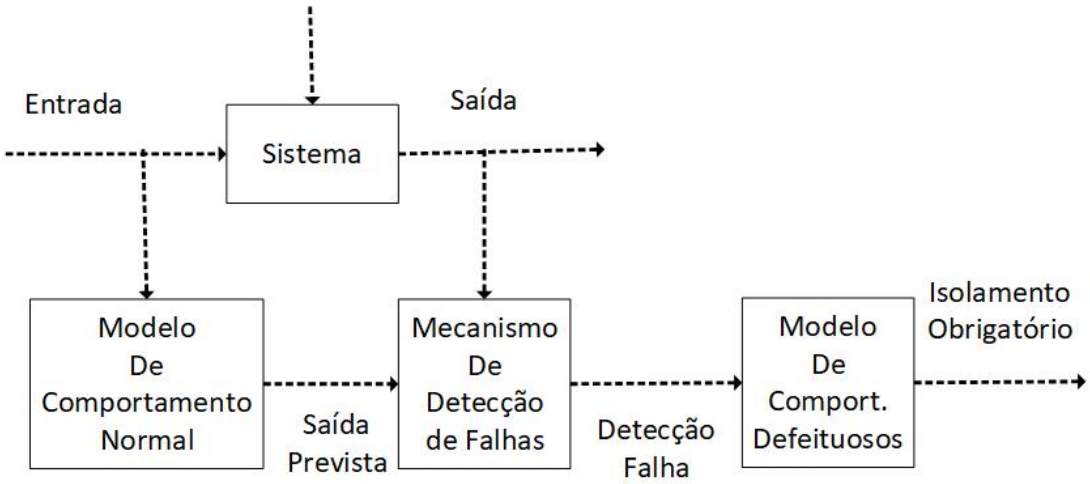


Figura 1 – Modelagem de avaliação de impacto das falhas em sistema discreto
Fonte: Adaptado de (ZAYTOON; LAFORTUNE, 2013)

Ao analisar o exemplo de modelagem apresentado na Figura1, destacam-se as técnicas que podem ser aplicadas nesse contexto diagnóstico de falhas, que têm como objetivo realizar três tipos de tarefas: detectar as falhas, identificar a sua origem e, por fim, avaliar o seu impacto no desempenho do sistema.

- Detecção de falha: técnica que se concentra em identificar quando um sistema está operando de forma desviada, ou seja, fora do comportamento esperado. Essa técnica não identifica a origem ou a gravidade do desvio;
- Isolamento de falhas: caso ocorra um desvio, o processo de localização de falhas concentra-se em identificar qual componente do sistema está causando esse desvio;
- Análise do impacto de falhas: concentra-se em identificar a natureza específica do desvio, sua gravidade e efeito no sistema. Geralmente, essa tarefa envolve a construção de modelos para representar tanto o comportamento normal do sistema quanto os efeitos de cada desvio considerado.

Os três processos mencionados são interdependentes e é notável que são apresentados em uma ordem crescente de complexidade para o diagnóstico. Além disso, cada um desses processos tem a responsabilidade de diagnosticar diferentes categorias de falhas que podem ocorrer em um sistema. Essas categorias de falhas incluem as permanentes, de transição e intermitentes, conforme ilustrado abaixo na Figura 2 (NATUCCI, 2016).

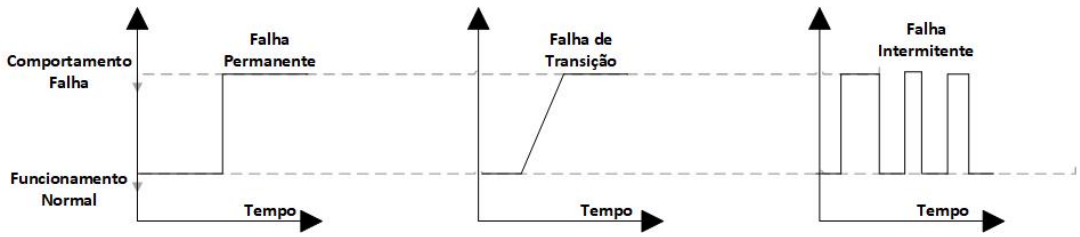


Figura 2 – Modelagem de avaliação de impacto das falhas em sistemas discretos
Fonte: Refeito de (ZAYTOON; LAFORTUNE, 2013)

As falhas permanentes resultam em perdas irreversíveis no processo, como quebras ou reações químicas irreversíveis. As falhas de transição ocorrem gradualmente em uma etapa do processo, geralmente como resultado de exposição prolongada a efeitos prejudiciais ao processo. As falhas intermitentes são causadas por elementos externos que atuam de forma intermitente, como fiação com problemas ou vibrações no processo. Em geral, a ocorrência de falhas em sistemas discretos pode ser atribuída a problemas em sensores, atuadores e controladores, além de falhas sistêmicas que afetam a dinâmica da planta (NATUCCI, 2016).

Concluindo, falha pode ser definida como um evento indesejado que causa impactos negativos no desempenho e na operação adequada de um sistema. Quando ocorrem falhas, os comportamentos do sistema podem se desviar do esperado, resultando em efeitos indesejáveis. Essas falhas podem causar pausas na produção, perda de qualidade, danos aos equipamentos ou

até mesmo colocar em risco a segurança dos operadores e usuários. Portanto, a detecção precoce e eficaz dessas falhas é de extrema importância, visando minimizar seus efeitos negativos. Ao detectar as falhas de forma antecipada, é possível tomar medidas corretivas rapidamente, evitando danos maiores ao sistema e reduzindo os custos de reparo.

3.2 MANUTENÇÃO PREVENTIVA, PREDITIVA E CORRETIVA

Iniciando a seção subsequente, abordará as práticas de manutenção, mais especificamente as modalidades preventiva, preditiva e corretiva. Enquanto a seção anterior se concentrou nas falhas e sua definição, este segmento visa explorar as estratégias e abordagens utilizadas para mitigar essas falhas e assegurar a operação eficiente dos sistemas. A manutenção preventiva, preditiva e corretiva desempenham um papel crucial na gestão de falhas e na promoção da confiabilidade contínua dos equipamentos e processos industriais. Buscando examinar detalhadamente essas práticas e entender como elas contribuem para a sustentabilidade e eficácia das operações.

Desde 1997, a gestão da manutenção, possivelmente mais do que qualquer outra atividade de administração, passou por notáveis transformações em seus métodos ao longo de sua evolução, notadamente nos últimos trinta anos. As alterações ocorridas nesse período, seja pelo aumento das expectativas em relação à manutenção, pelas mudanças nas perspectivas sobre a origem de defeitos e falhas, ou nas técnicas de manutenção, podem ser identificadas em três gerações distintas. Todas essas mudanças surgem, como sempre, da necessidade de aprimoramento e otimização exigida por períodos de crise (SIMEóN, 2008; SANTOS *et al.*, 2023).

Analisando as distintas eras da manutenção, destaca-se, na contemporaneidade, a terceira geração, que teve início em 1970 e se estendeu até os anos 2000. Esse período foi impulsionado pelas transformações significativas nas indústrias da época. Caracterizou-se por avanços como aprimoramento da disponibilidade e confiabilidade de maquinaria, aperfeiçoamento da segurança, melhorias na qualidade dos produtos, redução do impacto ambiental, ampliação da vida útil dos equipamentos e uma relação custo-eficácia mais vantajosa (SANTOS *et al.*, 2023; SIMEóN, 2008).

Na era atual, sobressaem as contribuições notáveis relacionadas às metodologias de gestão da manutenção. Essas contribuições abrangem desde o surgimento das primeiras técnicas de monitoramento de condição Medindo os Planos de Manutenção Preditiva (MPD), a utilização de ferramentas de apoio à tomada de decisões e análise de riscos; a introdução do Análise de Modos de Falha e Efeitos, do inglês *Failure Mode and Effect Analysis* (FMEA),

sistemas especialistas, redes neurais, lógica nebulosa, até uma maior ênfase na fase de projeto, considerando aspectos de confiabilidade e manutenibilidade. Além disso, observa-se a formação de grupos de trabalho multidisciplinares, envolvendo todos os níveis hierárquicos da empresa, para estabelecer metodologias mais eficientes na gestão de ativos, a exemplo da Manutenção Produtiva Total, do inglês *Total Productive Maintenance* (TPM) e da Manutenção Centrada em Confiabilidade (MCC). Esses avanços representam um marco na evolução das práticas de manutenção, consolidando uma abordagem mais abrangente e estratégica no gerenciamento de ativos industriais (PEREIRA, 2023; SIMEóN, 2008).

Após examinar as diversas eras da manutenção, torna-se imperativo compreender os três principais métodos empregados: manutenção preventiva, preditiva e corretiva. Essas abordagens desempenham papéis distintos na gestão eficiente dos ativos industriais, sendo a manutenção preventiva especialmente destacada pela sua capacidade de antecipar e prevenir falhas antes de sua manifestação. Nesse contexto, a dissertação direcionará seu foco para analisar a evolução dessas práticas e explorar as notáveis contribuições das últimas décadas, enfatizando a importância estratégica da prevenção na otimização do desempenho e na sustentabilidade dos equipamentos industriais (INABA, 2021)

- Manutenção Preventiva: Esta abordagem é proativa e planejada, realizada periodicamente para evitar o surgimento de falhas. Supervisões, substituições e ajustes são coordenados regularmente para garantir a eficiência operacional e prevenir paralisações não programadas.
- Manutenção Preditiva: Utilizando técnicas avançadas de monitoramento e análise, a manutenção preditiva busca identificar sinais precoces de desgaste ou possíveis falhas. A coleta de dados em tempo real permite intervenções específicas quando necessário, otimizando recursos e minimizando paradas não planejadas.
- Manutenção Corretiva: Essa abordagem reativa entra em ação após a identificação de falhas, envolvendo a correção do problema já ocorrido. Apesar de necessária em certos casos, pode resultar em períodos de inatividade e custos adicionais.

No âmbito desta dissertação, destaca-se a crucial manutenção preditiva. Antecipar e mitigar falhas potenciais antes de sua manifestação busca estabelecer um ambiente operacional confiável e eficiente. Dada a evolução tecnológica, a implementação de técnicas como *machine learning* para antecipar falhas torna-se fundamental. Ao adotar a manutenção preditiva como método para antecipar e evitar falhas, torna-se evidente que esta se configura como a estratégia

fundamental para assegurar a disponibilidade, confiabilidade e durabilidade dos equipamentos industriais. Nesse sentido, será examinada a evolução das técnicas de manutenção, com destaque para a prevenção, destacando-se as significativas contribuições das últimas décadas e sua aplicação estratégica na gestão de ativos industriais.s (ABREU, 2023).

Em síntese, a manutenção de sistemas, equipamentos e máquinas é vital para assegurar a eficiência e a segurança operacional nas indústrias. Através da revisão da literatura, fica clara a relevância da adoção de estratégias de manutenção preditiva e preventiva, além da incorporação de tecnologias emergentes, como a IoT e a análise de *Big Data*. Contudo, surge a necessidade de pesquisas adicionais para explorar os desafios na implementação dessas estratégias em diversos contextos industriais e desenvolver abordagens eficazes para superá-lo (ABREU, 2023).

Diante desse cenário, o próximo capítulo desta dissertação se dedicará a apresentar as técnicas de *machine learning* aplicadas à identificação proativa de falhas. Exploraremos como essas inovações tecnológicas têm o potencial de revolucionar a gestão de ativos industriais, proporcionando uma visão mais clara e preditiva das condições operacionais. A ênfase será na prevenção, destacando a importância estratégica de antecipar e mitigar falhas, contribuindo assim para a otimização do desempenho e a prolongação da vida útil dos equipamentos.

4 MATERIAIS E MÉTODOS

Neste capítulo, serão apresentados algoritmos de detecção de falhas que serão aplicados ao sistema, visando identificar e considerar tais funcionamentos anormais (falhas) no modelo de simulação. Serão descritos os métodos utilizados, bem como as diferentes falhas que podem ocorrer no sistema, permitindo sua identificação e registro adequados.

4.1 PROPOSTA DE DISSERTAÇÃO

Conforme apresentado nos capítulos anteriores, o desenvolvimento de sistemas de detecção de falhas em processos de manufatura é um tópico de suma importância, visando assegurar a eficácia e a segurança das operações produtivas. A utilização de métodos provenientes da Inteligência Artificial tem se tornado uma abordagem cada vez mais difundida para enfrentar esse desafio.

Neste capítulo, serão introduzidos os métodos de identificação de falhas que serão implementados no sistema de avaliação. A proposta envolve a utilização de algoritmos como SVM, MLP, KNN, RF. Serão fornecidos dados em relação ao funcionamento dessas técnicas de identificação, seus modelos operacionais, e serão apresentados exemplos de suas aplicações, em conjunto com ilustrações de sua base de execução. O intuito é explorar essas abordagens eficazes de detecção de falhas, visando aprimorar o desempenho do sistema de avaliação. O entendimento dessas técnicas e seu correto emprego serão cruciais para aprimorar a identificação precoce de falhas, contribuindo assim para a confiabilidade e qualidade dos resultados obtidos. Evitar manutenções corretivas é essencial para manter a eficiência do processo industrial e minimizar custos associados a interrupções não planejadas.

4.2 MÉTODO DE DETECÇÃO DE FALHAS

Este capítulo apresenta os métodos de detecção de falhas que serão aplicados na dissertação. É crucial que o algoritmo esteja em constante análise e comparação com o modelo pré-estabelecido de funcionamento, compreendendo o contexto em que está inserido. Espera-se que seja capaz de identificar padrões anormais de funcionamento ou comportamento indesejável entre as variáveis do sistema. A detecção precoce dessas falhas é de suma importância para

proteger o sistema contra instabilidades no processo, paradas abruptas ou falhas de equipamentos. Os algoritmos utilizados nesta pesquisa, como SVM,MLP, KNN e RF, serão apresentados de modo que sua compreensão de funcionamento seja clara para o entendimento dos resultados expressos pela dissertação, contribuindo assim para aprimorar a identificação e prevenção de falhas em sistemas de controle industrial.

4.2.1 Support Vector Machines (SVM)

O SVM, significa Máquina de Vetores de Suporte, foi apresentado inicialmente por Cortes e Vapnik (1995). Trata-se de um algoritmo de aprendizado de máquina supervisionado que emprega a ideia de planos de decisão para categorizar um conjunto de dados em um espaço de múltiplas dimensões. Esse método pode utilizar o conceito de *Kernels* para realizar essa classificação (DAMASCENO, 2021).

A resolução do problema do SVM é simplificada ao formato de otimização convexa, onde a meta é minimizar uma função quadrática sujeita a restrições convexas. Nesse processo de otimização convexa, ajustamos a margem geométrica por meio da margem funcional, mantendo-a igual a 1, enquanto minimizamos a norma do vetor de pesos associado ao hiperplano (w;b)(essa norma permanece inalterada se redimensionada). A Figura 3 fornece uma representação visual do hiperplano de margem máxima, destacando as margens geométricas $H1$ e $H2$, a mediana $H0$ entre essas margens, os Vetores de Suporte (SV) localizados nas margens $H1$ e $H2$, e as distâncias $d+$ e $d-$, que representam as menores distâncias aos pontos positivos e negativos mais próximos, respectivamente (MELLO *et al.*, 2019).

Assim, um vetor de peso w é projetado para obter uma margem funcional de 1 em instâncias positivas $x+$ e negativas $x-$, sendo que sua margem geométrica é calculada conforme as equações (1) e (2) apresentadas.

$$H1 : \langle w.x^{+} \rangle + b \ = 1 \tag{1}$$

$$H2 : \langle w.x^{-} \rangle + b \ = \ -1 \tag{2}$$

O parâmetro b, denominado termo de limiar ou viés, é recalculado após cada iteração. A fim de determinar a margem geométrica, é necessário normalizar o vetor w, sendo esta margem calculada com base no valor da margem funcional do seguinte classificador apresentado na

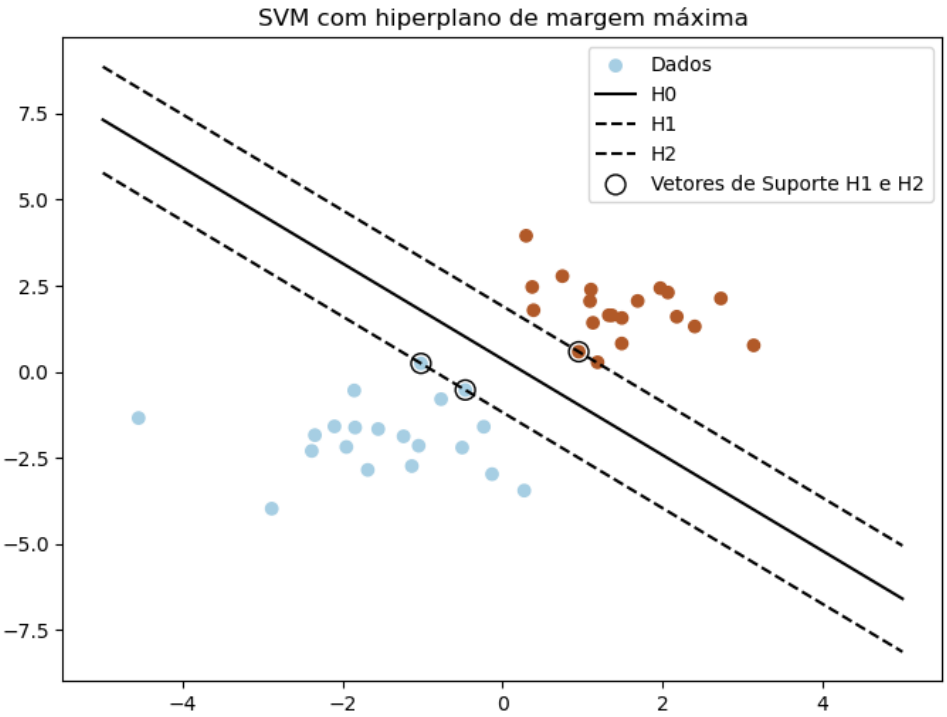


Figura 3 – Hiperplano de Margem Máxima
Fonte: Fonte Própria

Equação (3): (MELLO *et al.*, 2019).

$$\gamma = \frac{1}{\| w \|_2} \tag{3}$$

Outra aplicação utilizada no SVM foi a utilização do *kernel*, que desempenha um papel fundamental na transformação do espaço de características. Isso permite a separação de padrões que podem não ser linearmente separáveis no espaço original, tornando-os separáveis em um espaço onde a separação linear é possível. O *kernel* no SVM permite que o algoritmo trabalhe eficientemente em problemas de classificação não lineares, aumentando a capacidade de generalização do modelo (AMÂNCIO, 2023; ZHANG; SONG, 2015).

Existem vários tipos de *kernel*: linear, polinomial, gaussiano (RBF), sigmoidal, entre outros. Cada tipo de *kernel* possui características distintas e é mais apropriado para diferentes conjuntos de dados e problemas específicos. No caso do *kernel* linear aplicado na dissertação, ele é geralmente empregado quando os dados são linearmente separáveis, o que significa que é possível desenhar uma linha reta para separar claramente as diferentes classes no espaço de características (AMÂNCIO, 2023; ZHANG; SONG, 2015).

Dentro do contexto de alcançar melhores resultados de generalização e desempenho do modelo, foi utilizada a técnica de validação cruzada. Esta técnica segue a seguinte estrutura:

divide o conjunto de dados em partes iguais (k grupos), onde cada grupo é utilizado alternadamente como conjunto de teste e treinamento em cada iteração do loop de validação cruzada. Isso permite avaliar o desempenho do modelo de forma mais robusta, uma vez que todas as observações do conjunto de dados são utilizadas tanto para treinamento quanto para teste em diferentes momentos, reduzindo assim o viés e a possibilidade de sobreajuste. Essa abordagem é crucial para garantir que o modelo seja capaz de generalizar bem para novos dados não vistos, o que é fundamental em aplicações de aprendizado de máquina (KUCHERYAVSKIY *et al.*, 2020).

Em resumo, o SVM é uma técnica de aprendizado de máquina usada para classificação. No SVM, busca-se encontrar um hiperplano ótimo que maximize a margem geométrica entre as classes, garantindo uma separação clara e eficaz. Isso é alcançado através da resolução de um problema de otimização convexa, onde o objetivo é minimizar a função de custo enquanto ainda mantém as restrições de que todos os exemplos de treinamento estejam do lado correto do hiperplano ou dentro da margem de separação, com penalidades para violações. O vetor de peso (w) é ajustado para definir a orientação do hiperplano e o termo de viés (b) é dinamicamente recalculado. O SVM destaca-se por sua capacidade de lidar com conjuntos de dados em espaços de alta dimensão, sendo uma ferramenta versátil e eficiente para tarefas de classificação (GONÇALVES, 2009).

4.2.2 Rede Neural Artificial

As redes neurais são estruturas computacionais que têm a capacidade de aprender e reconhecer padrões em dados, inspirados no funcionamento do cérebro humano. Essas redes são compostas por neurônios artificiais interconectados, que geram saídas com base em seus pesos e funções de ativação. Recentemente, avanços em neurofisiologia permitiram representar matematicamente os mecanismos relacionados ao fluxo e ao processamento de informações no cérebro humano, permitindo a criação de algoritmos computacionais que simulam de forma simplificada a estrutura fundamental do neurônio (BUENO, 2006).

Redes neurais são circuitos artificiais que simulam o sistema nervoso humano e buscam resolver problemas de AI. Elas têm a capacidade de aprender e agir diante de diferentes situações e adquirir conhecimento através da experiência e observação (CARRASCO, 2005). O processo de aprendizado ocorre através de um algoritmo que ajusta os pesos das conexões entre neurônios, para produzir saídas desejadas a partir de entradas fornecidas. Esse ajuste é feito por meio de um algoritmo de *backpropagation* que compara as saídas geradas pela rede com as saídas desejadas

e atualiza os pesos das conexões de acordo com o erro obtido.

Inicialmente MCCULLOCH e PITTS (1943) apresentaram o modelo artificial de um neurônio biológico. O neurônio proposto por McCulloch é um instrumento binário, que produz uma saída binária e tem entradas com ganhos arbitrários, que podem ser excitatórios ou inibitórios. A Figura 4 apresenta o neurônio artificial proposto. É possível identificar três componentes básicos em um neurônio: um conjunto de sinapses, onde um sinal x_j é recebido na entrada da sinapse j, conectada ao neurônio k, que é então multiplicado pelo peso sináptico w_{kj} correspondente; um somador que é responsável pela soma dos sinais de entrada que são multiplicados pelos respectivos pesos sinápticos dos neurônios; e uma função de ativação para limitar a amplitude da saída do neurônio. O bias, representado por "bk", tem como objetivo promover um deslocamento de curva da função de ativação, ou seja, aumentar ou diminuir a entrada da função de ativação (BUENO, 2006).

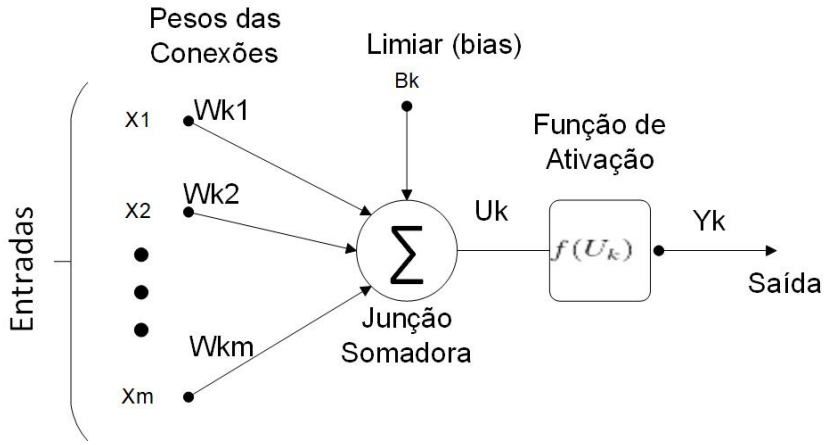


Figura 4 – Estrutura de Neurônio - RNA
Fonte: (BUENO, 2006)

O arranjo dos neurônios na rede neural está diretamente ligado ao algoritmo de aprendizagem utilizado no treinamento. Existem alguns tipos principais de arquitetura de MLP:
I - Redes de uma camada de *forward propagation*: possuem uma camada de entrada e uma camada de saída.
II - Redes de múltiplas camadas de *forward propagation*: têm uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída (BUENO, 2006).

Após o treinamento, a rede neural pode ser utilizada para efetuar tarefas como classificação, reconhecimento de tipos e previsão, entre outras coisas. Como, em uma tarefa de classificação, a rede recebe uma entrada e atribui uma classe a essa entrada com base nos padrões que aprendeu durante o treinamento. Em uma tarefa de previsão, a rede recebe uma série temporal

de entradas e tenta prever a saída futura, como apresentado na Figura 5 com base em padrões identificados nos dados históricos. O método desenvolvido neste estudo é o MLP, o qual pode ser considerado como uma das técnicas mais empregadas na área de engenharia.

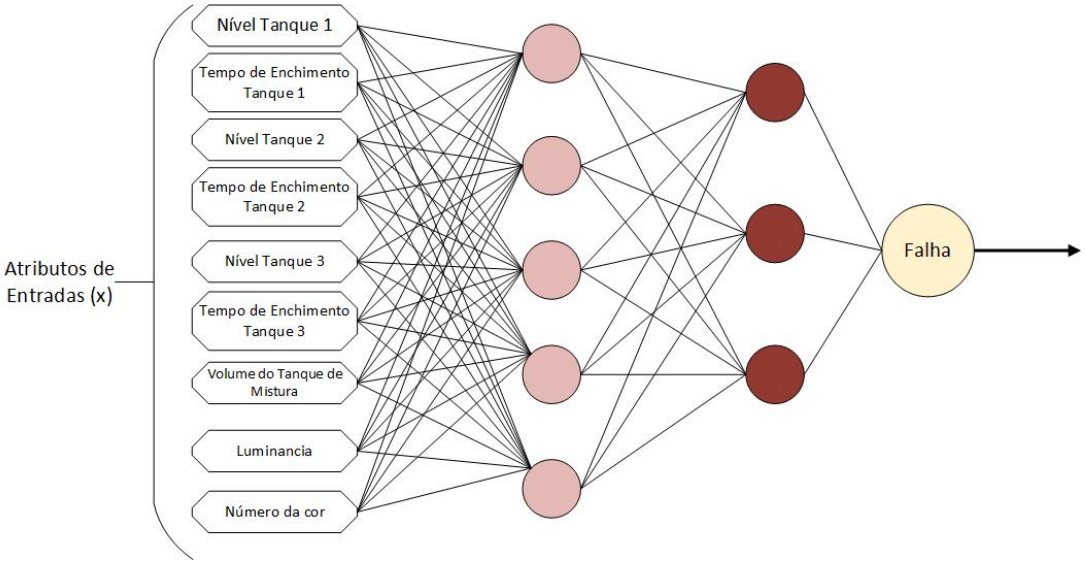


Figura 5 – Modelo de RNA MLP
Fonte: Fonte Própria

4.2.3 K-Nearest Neighbors

O algoritmo KNN, é uma técnica de aprendizado supervisionado do tipo *lazy* (vagaroso), introduzida por Aha *et al.* (1991). O princípio básico deste algoritmo consiste em encontrar os K exemplos rotulados mais próximos de um exemplo não classificado. Apesar de um funcionamento basicamente simples, o método KNN obteve sucesso em um grande número de problemas de classificação, tanto com dígitos manuscritos e cenas de imagens de satélite. Por não ser um método não paramétrico, é normalmente bem sucedido em situações de classificação (FERRERO, 2009; GOLDBERGER *et al.*, 2004).

A classificação baseada em vizinhos constitui uma modalidade de aprendizado fundamentada em instâncias, marcada pela ausência de generalização. No contexto do KNN, essa ausência de generalização se manifesta quando o algoritmo não consegue se adaptar adequadamente a novos dados fora do conjunto de treinamento, resultando em uma eficácia limitada em situações complexas com grandes volumes de dados ou alta dimensionalidade, além de ser sensível a ruídos e distribuições não representativas. Nesse contexto, não se busca construir um modelo interno amplo, visto que o método se limita a armazenar instâncias dos dados de treinamento. A determinação da classificação ocorre por meio de uma votação por maioria simples

dos vizinhos mais próximos de cada ponto. Dessa forma, a um ponto de consulta é atribuída a classe de dados que apresenta maior representatividade dentro do conjunto dos vizinhos mais próximos associados a esse ponto. Ao contrário de alguns métodos, os algoritmos da família KNN demandam baixa complexidade durante a fase de treinamento. O número de amostras que dever ser usadas durante o treinamento pode ser definido pelo usuário, ou variar com base na densidade local de pontos (GOLDBERGER *et al.*, 2004).

De forma geral, o KNN necessita do armazenamento de todos os padrões de treinamento $w^{(i)} \in \mathbb{R}^K$, associados com suas respectivas respostas desejadas $y^{(i)}$, onde $i = 0, ..., N-1$. Portanto, para a inserção de um novo dado de entrada w', a saída proporcionada pelo KNN requer as saídas associadas aos K padrões de treinamento que estão mais próximos à entrada w' no espaço de atributos. Isso é realizado, por exemplo, através do cálculo da média para obter a saída dos vizinhos mais próximos:

$$\hat{y}(w') = \frac{1}{K} \sum_{w(i) \in M_k(w')} y(i), \tag{4}$$

No qual $M_k(x')$ indica a vizinhança de w', constituida pelos padrões de treinamento $w(i)$ que retratam aos K vizinhos mais próximos de w' (ALPAYDIN, 2020).

Dessa forma, o método KNN utiliza as seguintes propriedades:

- Para encontrar os vizinhos mais próximos, usamos uma métrica de distância dos atributos com o objetivo de determinar os vizinhos mais próximos;
- A definição do parâmetro K no algoritmo KNN é relevante, pois ele determina o número de vizinhos mais próximos a serem considerados para gerar a saída do algoritmo. Esse parâmetro influencia justamente na suavidade da fronteira de decisão do KNN valores menores de K tendem a gerar fronteiras de decisão mais confusas e irregulares, enquanto valores maiores de K tendem a gerar fronteiras de decisão mais suaves e simplificadas. Isso resulta em desempenho e capacidade de generalização do algoritmo em diferentes conjuntos de dados, pois uma escolha ajustada de K é essencial para obter resultados precisos e robustos (MENESES *et al.*, 2019).

4.2.3.1 Redução da dimensionalidade KNN

O desempenho do algoritmo KNN revela-se satisfatório quando se lida com um número reduzido de variáveis de entrada. No entanto, entra em conflito diante de um aumento significativo no número de entradas, pois cada variável de entrada passa a representar uma dimensão em um espaço de entrada. Ilustrativamente, ao considerarmos duas variáveis de entrada, denominadas $e1$ e $e2$, o espaço de entrada assume uma configuração bidimensional. À medida que o número de variáveis de entrada aumenta, o volume do espaço de entrada expande-se de maneira exponencial (REZENDE *et al.*, 2021).

Com base nesse princípio, foram aplicadas três técnicas de redução de dimensionalidade nos resultados do algoritmo, visando uma análise mais eficiente de desempenho. Essas técnicas incluem a Análise de Componentes Principais, do inglês *Principal Component Analysis* (PCA), Análise Discriminante Linear, do inglês *Linear Discriminant Analysis* (LDA) e a Análise de Componente de Vizinhança, do inglês *Neighborhood Component Analysis* (NCA).

Primeiramente, abordaremos o modo de operação do PCA que, no contexto da estatística multivariada, é um método não supervisionado que busca maximizar a variância dos dados. O componente principal, denominado $p1$, é escolhido para espalhar as amostras após a projeção, tornando mais evidentes as diferenças entre os dados. A busca por uma solução única enfatiza a direção como um fator crucial na escolha, onde $\| \ p1 \ \| = 1$ indica a normalização do vetor. Essa ideia é expressa pela Equação (5) e pela notação $Conv(x) = \Sigma$: (ALPAYDIN, 2020)

$$s = p^T x \tag{5}$$

$$Var(s1) = \mathrm{p}_1^t \ \Sigma p1 \tag{6}$$

O objetivo é buscar um vetor (p_1) de forma que a variância $(\mathrm{Var}(s_1))$ seja maximizada, sujeito à restrição $(p_1^T p1 = 1)$. Esta condição é expressa formalmente através de um problema de otimização com a técnica de *Lagrange* (REZENDE *et al.*, 2021).

$$max \ p_1^T \Sigma \ p1 - \alpha \ (p1^T p1 - 1) \tag{7}$$

A otimalidade da Equação(7) está condicionada à condição de (p_1) ser um autovetor de (Σ), onde (α) representa o autovalor correspondente. Nesse contexto, o critério de escolha recai sobre o autovetor que possui o maior autovalor, uma vez que isso resulta na maximização da

variância. Logo, o componente principal é definido como o autovetor da matriz de covariância dos dados de entrada, correspondente ao maior autovalor, $(\lambda_1 = \alpha)$ (REZENDE *et al.*, 2021).

Para o segundo componente principal, (p_2), é imperativo que ele também maximize a variância, possua comprimento unitário e seja ortogonal a (p_1). A ortogonalidade é crucial para garantir que, após a projeção $(s_2 = \omega^T x)$, (s_2) não apresente correlação com (s_1). O vetor (p_2) é então escolhido como o autovetor de (Σ) associado ao segundo maior autovalor, $(\lambda_2 = \alpha)$. Esse processo pode ser generalizado para as demais dimensões, onde os autovetores são determinados pelos autovalores em ordem decrescente (LYRA *et al.*, 2010).

O primeiro autovetor, (p_1), correspondente ao componente principal, descreve a maior parte da variância, seguido pelo segundo autovetor, que representa a segunda maior parcela, e assim por diante. Dessa forma, a (5) pode ser reformulada como mostrado na (8), consolidando a abordagem da PCA na seleção dos autovetores mais significativos para maximizar a informação retida em cada componente (REZENDE *et al.*, 2021).

$$s = P^T \, (x - m) \tag{8}$$

Na Equação (8), o vetor m representa o vetor médio dos dados originais. Onde s é o vetor de dados transformados, P^T é a matriz de autovetores transposta, x é o vetor de dados original, e m é o vetor médio.

A interpretação dos termos na equação é a seguinte:

- s: representa o vetor de dados transformados, isto significa, os dados originais x projetados no espaço de componentes principais definido pelos autovetores em P^T.
- P^T: é a matriz de autovetores transposta, que indica a transformação linear dos dados originais para o espaço de componentes principais.
- x: é o vetor de dados original que está sendo projetado no espaço de componentes principais.
- m: é o vetor médio dos dados originais. Ao subtrair m de x, os dados são centralizados em torno da média antes de aplicar a transformação para o espaço de componentes principais.

Essa centralização realizada pela Equação(8) é essencial para o PCA, pois permite que os autovetores capturem as direções de máxima variância nos dados (REZENDE *et al.*, 2021).

A aplicação da PCA implica na transformação de um conjunto original de variáveis em outro conjunto de componentes principais de mesma dimensão. Esses componentes apresentam propriedades relevantes, sendo cada um uma combinação linear independente de todas as variáveis originais. A estimação dos componentes tem como objetivo reter a máxima informação possível em relação à variação total dos dados. Associada à ideia de redução de massa de dados, a PCA redistribui a variação nos eixos originais para obter um conjunto de eixos ortogonais não correlacionados. Isso é especialmente útil para a geração de índices e o agrupamento de indivíduos, classificando-os com base em sua variação e comportamento na população (LYRA *et al.*, 2010).

Ao considerarmos a simplicidade e eficácia da PCA, observamos que esse método se destaca como uma das transformações lineares ótimas entre as técnicas de análise de imagens. No âmbito do reconhecimento de padrões, a PCA é amplamente empregada, evidenciando sua relevância na comunidade dedicada a essa área de estudo (LYRA *et al.*, 2010; VARELLA, 2008).

Neste ponto, introduziremos uma segunda abordagem para a redução de dimensionalidade, conhecida como LDA. Tal técnica representa uma extensão do discriminante linear de Fisher, um método utilizado em reconhecimento de padrões e aprendizado de máquina. Seu propósito é identificar uma combinação linear de características que possa caracterizar ou distinguir eficientemente duas ou mais classes de objetos ou eventos. Considerando amostras provenientes de duas classes distintas, $A1$ e $A2$, o objetivo do LDA é determinar uma direção representada pelo vetor (ω). Esta direção é escolhida de maneira a maximizar a separação entre os exemplos das duas classes quando os dados, (x), são projetados ao longo de (ω). A projeção de (x) na direção de (p) é expressa pela Equação (5) (ALPAYDIN, 2020).

A matriz inter-classe (S_b), definida pela Equação (9), quantifica as diferenças entre as médias das amostras de duas classes, $A1$ e $A2$, ao considerar sua transposta e dividir pelo número total de amostras (N).

$$S_b = (m_1 - m_2)(m_1 - m_2)^T \tag{9}$$

$$S_1 = \sum_{t=1}^{N} r^t (x^t - m_1)(x^t - m_1)^t \tag{10}$$

$$S_2 = \sum_{t=1}^{N} (1 - r^t)(x^t - m_2)(x^t - m_2)^t \tag{11}$$

As matrizes intraclasse ($S_1 e S_2$), especificadas pelas Equações (10) e (11), são calculadas individualmente para cada classe. Essas matrizes avaliam as variações entre cada amostra (x^t) e a média correspondente da classe ($m1$ ou $m2$), sendo multiplicadas pela transposta dessa diferença (REZENDE *et al.*, 2021).

O fator de classificação (r^t) é uma ferramenta crucial para determinar a classe de cada amostra. Se (x^t) pertence à classe A1, (r^t) é igual a 1; caso contrário, (r^t) é igual a 0 (REZENDE *et al.*, 2021) (SILVA *et al.*, 2017). A matriz intra-classe total (S_w), expressa pela Equação (12), é obtida somando as matrizes intraclasse de ambas as classes (S1 + S2).

$$S_w = S_1 + S_2 \tag{12}$$

Com base nessas matrizes, o vetor (ω) é derivado usando a fórmula:

$$(\omega = cS^{-1}(m1 - m2)) \tag{13}$$

Onde (c) é uma constante. A escolha comum é assumir $(c = 1)$, uma vez que o foco reside na direção do vetor, e não em sua magnitude. Essa abordagem analítica proporciona percepções valiosos para a LDA no contexto da redução de dimensionalidade (SILVA *et al.*, 2017; REZENDE *et al.*, 2021).

No contexto de C classes, com $C > i > 2$, as Equações (14) e (15) proporcionam uma abordagem para calcular as matrizes inter-classe (S_b) e intra-classe (S_i) referentes à i-ésima classe:

$$[Si = \sum_t r_{i,t}(x_t - m_i)(x_t - m_i)^T] \tag{14}$$

$$[SW = \sum_{i=1}^{C} Si] \tag{15}$$

$$[Sb = \sum_{i=1}^{C} N_i(m_i - m)(m_i - m)^T] \tag{16}$$

Onde $N_i = \sum_t r_{t,i}$. As matrizes de dispersão de interclasses e intraclasses após a projeção são representadas por $W^T S_b W$ e $W^T S_W W$ respectivamente. A fim de maximizar a dispersão entre as classes e minimizar a dispersão dentro das classes, opta-se por utilizar os maiores autovetores de $S^{-1}S_b$ como solução. A matriz S_b é formada pela soma de K matrizes de ordem 1, representadas por $(m_i - m)(m_i - m)^T$, sendo que apenas $C - 1$ delas são independentes.

Consequentemente, S_b possui uma classificação máxima de $C-1$. Assim, define-se um novo subespaço dimensional de $(C-1)$, onde o discriminante é construído. Apesar de o LDA basear-se na separabilidade de classes como critério de optimalidade, é importante ressaltar que qualquer método ou classificação pode ser aplicado nesse novo subespaço para estimar os discriminantes (REZENDE *et al.*, 2021; SILVA *et al.*, 2017).

Em resumo, a LDA é um método de redução de dimensionalidade que busca encontrar um subespaço onde as classes dos dados sejam separáveis. Isso é alcançado ao maximizar a dispersão entre as classes e minimizar a dispersão dentro de cada classe. Utilizando as matrizes inter-classe (Sb) e intraclasse (Si), o LDA identifica os autovetores de $S^{-1}S_b$ para definir um novo espaço discriminante. O resultado é um conjunto de direções que otimiza a distinção entre as classes no novo subespaço, proporcionando uma representação eficiente e discriminativa dos dados.

Em sequência aos métodos aplicados, agora a NCA, um modelo adicional de redução de dimensionalidade que se destaca por sua abordagem única. Diferentemente das técnicas anteriores, a NCA enfoca a aprendizagem diretamente a partir dos dados, utilizando informações sobre a vizinhança para definir uma transformação que preserva as relações locais. Este método oferece uma perspectiva inovadora ao adaptar a representação dos dados com base nas interações entre as instâncias vizinhas, proporcionando uma abordagem mais adaptável e sensível às características locais do conjunto de dados.

O NCA é um classificador que tem como propósito encontrar, a partir de um conjunto de dados, uma transformação linear para um novo espaço e aplicar a estratégia KNN nesse novo espaço. Essa transformação linear visa aprimorar o desempenho do KNN no novo espaço em comparação ao desempenho no espaço original. A avaliação da performance do kNN é realizada por meio da validação cruzada *leave-one-out (LOO)*. Essa abordagem inovadora destaca-se por adaptar a representação dos dados com base nas interações entre as instâncias vizinhas, proporcionando uma perspectiva mais personalizada e sensível às características locais do conjunto de dados (LEITE, 2019).

Devido à natureza do KNN em particionar o espaço, a função que calcula o erro de classificação LOO do KNN para um conjunto $x_1, ..., x_n - x_i$ produz um valor diferente do obtido para o conjunto $x_1, ..., x_n - x_j$. Essa discrepância revela uma descontinuidade na função do erro LOO. Para abordar essa descontinuidade, a Análise de Componente de Vizinhança (NCA) adota uma estratégia de suavização na atribuição de vizinhos no novo espaço. Essa suavização é

implementada de forma que a seleção de qual observação é vizinha a cada ponto não seja mais determinística, como ocorre originalmente no KNN, mas sim probabilística. Assim, considerando p_ij como a probabilidade de o ponto i escolher o ponto j como seu vizinho, o valor de pij é definido por meio de uma abordagem probabilística.

Nas expressões (17) e (18), $(D(xi,xj) = \|Axi - Axj\|^2$ denota a distância euclidiana no espaço transformado por (A), ao passo que $K(z) = e^{\frac{z}{\sigma}}$ é uma função de *kernel*, onde (σ) define a amplitude do *kernel*.

$$pij = \left\{ \frac{k(D(x_i,x_j))}{\Sigma_{k \neq i} k(D(x_i,x_k))}, i \neq j \right\} \tag{17}$$

$$pij = \{ \ 0 \ , i = j \} \tag{18}$$

Tomando $C_i = \{j | c_i = c_j\}$ como o conjunto de pontos com a mesma classe que o ponto (i), a possibilidade de o algoritmo classificar corretamente esse ponto é expressa pela Equação (19) (LEITE, 2019).

$$pi = \sum_{j \in C_i} p_i j \tag{19}$$

A obtenção da transformação (D) ocorre por meio da maximização da probabilidade de cada ponto no conjunto ser classificado corretamente, conforme expresso na (20).

$$f(D) = \sum_i \sum_{j \in C_i} p_i j = \sum_i p_i \tag{20}$$

Ao derivar esta função em relação a D, resulta na seguinte expressão:

$$\frac{d_f}{d_D} = 2A \sum_i (p_i \sum_k p_{ik} x_{ik} x_{ik}^T - \sum_{j \in Ci} p_{ij} x_{ij} x_{ij}^T) \tag{21}$$

As duas funções expostas podem ser adaptadas para incorporar um parâmetro de regularização. A introdução desse parâmetro tem o propósito de reduzir o erro na classificação de novas observações não vistas, prevenindo o overfitting. As funções ajustadas com a regularização são estabelecidas pelas Equações (22) e (23), onde N representa o número de elementos em D, como descrito anteriormente.

$$f(D) = \sum_i pi - \frac{\lambda}{N} \parallel D \parallel_F^2 \tag{22}$$

$$\frac{df}{dD} = 2D \sum_i (p_i \sum_k p_{ik} x_{ik} x_{ik}^T \; - \sum_{j \in C_i} p_{ij} x_{ij} x_{ij}^T) - \frac{\lambda}{N} D \tag{23}$$

Para esclarecer o funcionamento do NCA de maneira mais acessível, ao otimizar a função objetivo (22) mediante algum método de otimização de primeira ou segunda ordem, como os algoritmos que fazem uso da informação do gradiente, obtém-se a matriz D. Neste estudo, a escolha recaiu sobre o método de gradiente conjugado. Devido à sua propriedade de segunda ordem, o método obtém convergência com um número mínimo de iterações, comparado aos métodos de otimização de primeira ordem. Com a transformação D em mãos, procedemos à sua aplicação nos dados. Para classificar uma nova observação, efetuamos a transformação desse ponto e atribuímos o rótulo associado à maior probabilidade na sua vizinhança nesse novo espaço. Uma particularidade interessante do NCA surge da transformação linear obtida. Caso D tenha dimensão $Q \times q$ com $q < Q$, é viável encontrar uma representação dos pontos em um espaço de dimensão inferior. Notadamente, com $q = 2$ ou $q = 3$, é possível visualizar essa nova representação (LEITE, 2019).

Em resumo, a NCA se destaca como um método eficiente de redução de dimensionalidade, priorizando a aprendizagem direta a partir dos dados. A maximização da probabilidade de classificação correta, conforme refletida na Equação (22), é alcançada por meio da matriz D, cuja obtenção é facilitada pelo emprego do método de gradiente conjugado, um algoritmo de segunda ordem. Essa abordagem, ao amenizar a atribuição de vizinhos no novo espaço, contribui para evitar o sobreajuste e possibilita um desempenho mais ajustável e sensível às relações locais. A utilidade prática do NCA é ainda evidenciada pela possibilidade de visualização em espaços de menor dimensão, facilitando a interpretação dos dados.

4.2.4 Floresta Aleatória

A Floresta Aleatória, do inglês *Random Forest* (RF), constitui em uma técnica proeminente de aprendizado de máquina, se destaca por sua abordagem inovadora, empregando múltiplas árvores de decisão para tarefas de classificação ou regressão. Cada árvore é treinada independentemente, e a decisão final é obtida pela média ou votação das saídas individuais. Essa técnica introduz aleatoriedade durante o treinamento, selecionando subconjuntos aleatórios dos dados de treinamento para cada árvore e características aleatórias para cada divisão de nó. Essa diversidade mitigadora de especializações excessivas promove uma generalização

robusta (SILVA, 2022).

Para entender o funcionamento de uma RF é necessário recapitular que Árvore de Decisão, do inglês *Decision Tree* (DT) é uma estrutura de dados não linear, que usa uma representação baseada em árvores. Ela é composta por um número finito de nós ou elementos. Cada árvore tem um elemento especial chamado elemento raiz, que é o pai dos elementos de suas subárvores. Todos os elementos com filhos são considerados elementos terminais e são chamados de folhas. O nível de um elemento é a distância do mesmo até a raiz, ou seja, o número de ligações percorridas até chegar à raiz. A profundidade de uma DT é definida pela maior distância entre uma folha e a raiz, podendo ter árvores com profundidade uniforme ou não. Dependendo do número de filhos de cada elemento, a árvore pode ser considerada binária, ternária ou mista (MURTHY, 1995). A Figura 6 apresenta uma estrutura de funcionamento da DT.

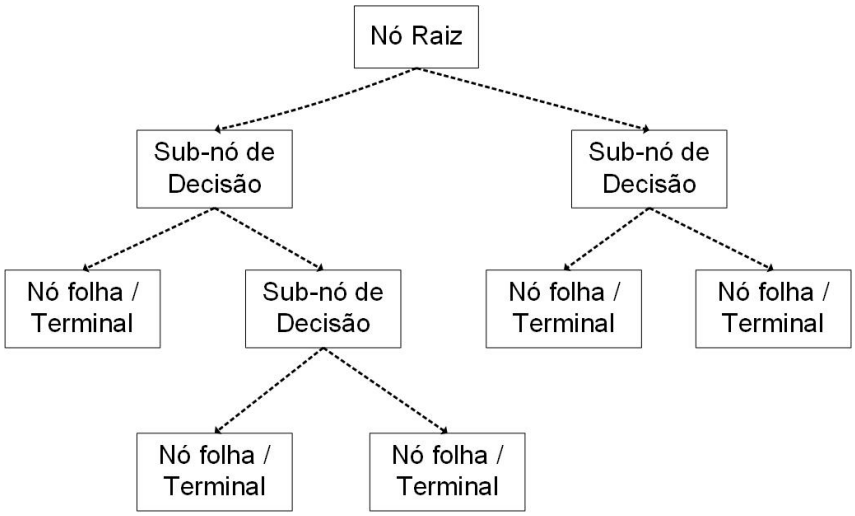


Figura 6 – Estrutura exemplo de dados: Árvore
Fonte: Autoria Própria

No contexto da RF composta por três árvores como ilustrado na Figura 7, cada árvore é cuidadosamente construída, atentando para a profundidade dos ramos e o número de folhas. A profundidade dos ramos, indicando o número de níveis da árvore, é limitada para evitar o sobreajuste, enquanto o número de folhas é controlado para modular a complexidade do modelo. Este enfoque visa equilibrar a capacidade de generalização e complexidade do modelo (MENEGAZZO, 2021; SILVA, 2022).

O funcionamento básico da RF destaca-se pela utilização de estruturas de seleção *If* e *Else* em cada árvore de decisão. Cada nó na árvore representa uma decisão baseada em uma condição específica, criando bifurcações sucessivas até chegar a uma folha que representa

a decisão final de classificação ou regressão. A aleatoriedade na seleção de características e subconjuntos de dados durante a construção da árvore confere diversidade e robustez ao conjunto (SILVA, 2022).

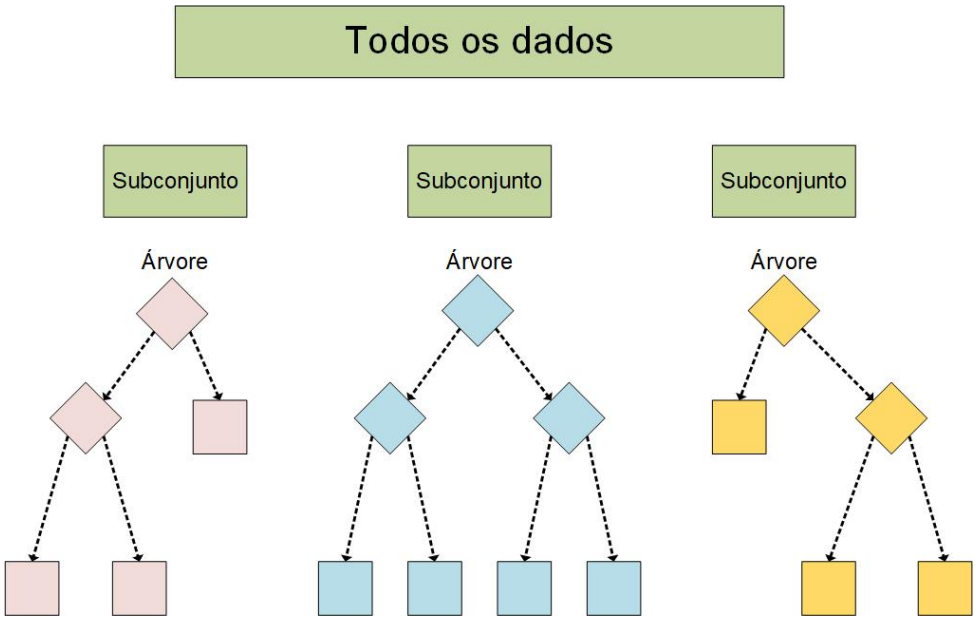


Figura 7 – Estrutura exemplo de dados: Floresta de Decisão
Fonte: Atualizado de (SILVA, 2022)

Dessa forma, a Floresta Aleatória é uma ferramenta versátil, destacando-se por sua capacidade de integração de múltiplas árvores de decisão, cada uma fundamentada em estruturas de seleção If e Else, promovendo um modelo preditivo eficiente e resistente a sobreajuste. Essa abordagem tem demonstrado sucesso em diversas aplicações de aprendizado de máquina, tornando-se uma escolha valiosa para abordar desafios complexos em análise de dados.

Na indústria, a aplicação da técnica de Floresta Aleatória estende-se à detecção de falhas em sistemas de produção. Nesse contexto, cada nó da rede treina um modelo de Árvore de Decisão com um conjunto de dados específico. Após o treinamento, os modelos são compartilhados entre os nós, culminando na construção de uma floresta de decisão distribuída que reúne as árvores mais precisas na previsão do conjunto de dados. Essa floresta é então empregada para prever os fluxos do sistema, fazendo uso do conceito de Bagging (MENEGAZZO, 2021).

O conceito de *Bagging*, ou *Bootstrap Aggregating*, é uma técnica de *ensemble learning* utilizada para melhorar a precisão e a estabilidade de modelos de ML, como as DT. O *Bagging* consiste em criar múltiplas amostras de treinamento a partir do conjunto de dados original através de reamostragem com substituição *(bootstrap)*, treinar um modelo em cada amostra e, por fim, combinar as previsões dos modelos individuais para obter uma previsão final mais robusta e

geralmente mais precisa. Essa estratégia é especialmente benéfica em situações onde há grande variação nos dados ou um potencial risco de *overfitting*, pois ela minimiza a sensibilidade do modelo às flutuações nos dados de treinamento (LEITE *et al.*, 2020).

Ao trabalharem em conjunto, as DT na RF proporcionam uma previsão mais robusta e precisa. Esse método revela-se crucial na detecção oportuna de falhas, permitindo a implementação de medidas preventivas antes que o sistema sofra impactos significativos. Além disso, a técnica exibe eficácia na redução do sobreajuste, fenômeno no qual o modelo se ajusta excessivamente aos dados de treinamento, comprometendo sua capacidade de generalização para novos dados (MENEGAZZO, 2021).

Concluindo, a Floresta Aleatória surge como uma ferramenta valiosa na indústria, destacando-se não apenas por sua capacidade de detectar falhas de maneira eficiente, mas também por sua habilidade em promover previsões mais robustas e, ao mesmo tempo, evitar ajustes excessivos aos dados de treinamento. Essa abordagem integrativa e distribuída contribui significativamente para a confiabilidade e estabilidade dos sistemas produtivos.

4.3 LUMINOSIDADE RGB

No âmbito deste estudo, uma atenção especial é dedicada à luminosidade relativa como um elemento fundamental na avaliação deste processo industrial. O cálculo para se obter a luminosidade relativa em modelos RGB (*Red, Green, Blue*), desempenha um papel crucial no monitoramento e na garantia da qualidade pois através desse método serão obtidos valores que irão assegurar a qualidade do processo.

Esta seção explora o conceito de luminosidade relativa, destacando como essa abordagem pode ser representativa da operação e do desempenho de sensores de luminosidade. O entendimento detalhado garante o entendimento do desenvolvimento desse trabalho, pois na prática seria utilizado um sensor óptico mas no ambiente computacional em que foi desenvolvido o banco de dados se fez necessário a utilização dessa função.

Na prática, um sensor de luminosidade é um dispositivo óptico que usa um emissor e um receptor fotossensível para medir a quantidade de luz presente. Um exemplo de tal sensor é um sistema de barreira, como o mostrado na Figura 8, em que o emissor e o receptor são posicionados em lados opostos para obtenção dos valores.

Para exemplificar a ação do sensor óptico de luminosidade no sistema, uma função é utilizada como base para transformar os níveis dos tanques (Vermelho, Azul e Verde) em

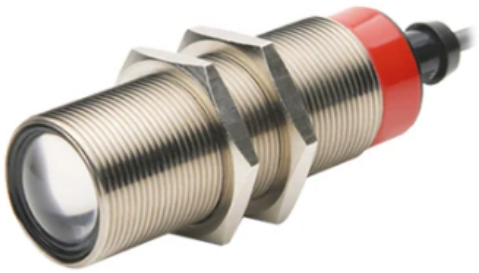


Figura 8 – Sensor Óptico (WEG, 2023)

variáveis para a função de luminância relativa (R,G,B). Essa função mede o brilho relativo de qualquer ponto em um espaço de cores, normalizado 0 (zero) para o preto mais escuro e 1 (um) para o branco mais claro, que pode ser renormalizado em uma escala de 0 (zero) a 100 (cem) por cento de luminosidade na tinta (NGUYEN; BROWN, 2017; FARINI, 2023).

No caso do espaço de cores RGB, a luminância relativa de uma cor é definida como:

$$luminance = 0.2126 * h1 + 0.7152 * h2 + 0.0722 * h3; \tag{24}$$

Analisando os dados obtidos pelo sistema para a fabricação das cores se baseando na Equação 24, conseguimos obter os valores de luminância. Esses valores são fundamentais para a identificação de possíveis falhas no sistema industrial estudado, desempenhando um papel crucial no controle de qualidade. A porcentagem de luz presente na tinta permite distinguir se o produto resultante deve ser descartado ou encaminhado para o envase, evidenciando a importância da luminosidade RGB como indicador confiável pois se torna o único item automatizado no processo de tomada de decisões (NGUYEN; BROWN, 2017).

Concluindo, torna-se evidente que a aplicação dessa função base no contexto industrial estudado, especialmente no processos de fabricação de tintas, mesmo sendo pouco em comparação a processos industriais inseridos completamente na Indústria 4.0, oferece vantagens significativas. A luminosidade relativa emerge como uma ferramenta essencial para monitorar e controlar a qualidade da produção. Seja na detecção de desvios no processo, na otimização da eficiência produtiva ou na garantia da uniformidade do produto final, a luminosidade RGB destaca-se como uma contribuição valiosa para a excelência operacional na indústria. Essa abordagem, aliada a estratégias de controle de qualidade, posiciona-se como um elemento diferenciador no cenário industrial obtendo a confiabilidade dos processos fabris.

5 SISTEMA DE TESTE

Neste capítulo, é apresentado o sistema que servirá como base para os testes dos algoritmos de identificação de falhas. Serão descritas as características gerais do sistema, incluindo sua estrutura e funcionamento, bem como os diferentes tipos de falhas que podem ocorrer durante sua operação.

Na fábrica de produção de tintas em que se baseia o software de simulação, foi aplicado um contexto operacional que envolve a ausência de recursos avançados de automação. Nesse cenário, a fábrica enfrenta limitações significativas em termos de modernização e incorporação de tecnologias inovadoras de produção.

A infraestrutura produtiva da fábrica não está totalmente alinhada aos princípios da Indústria 4.0. No entanto, como um primeiro passo em direção à automação e ao monitoramento da qualidade, foi implementado um sensor de luminosidade no final da linha de produção. Este sensor desempenha um papel crucial ao avaliar a correta composição da tinta com base em critérios luminosos. Essa decisão estratégica representa o único investimento acessível no momento para aprimorar o processo, permitindo uma avaliação mais objetiva e automatizada da qualidade final do produto.

A inserção do sensor de luminosidade no final da produção oferece uma solução pontual para aliviar os desafios enfrentados pela fábrica, possibilitando uma avaliação mais eficaz e objetiva da qualidade da tinta. O sistema de fabricação é baseado em malha aberta, o que significa que não há uma realimentação ou controle contínuo dos resultados. Essa abordagem, embora represente apenas um primeiro passo em relação aos avanços propostos pela Indústria 4.0, demonstra a disposição da fábrica em adotar medidas graduais em direção à automação e ao controle de qualidade em seus processos produtivos, visando maior eficiência, precisão e redução de erros.

5.1 DESCRIÇÃO DO SISTEMA DE TESTE

O principal objetivo da implementação dos algoritmos nesse processo é identificar possíveis falhas que possam ocorrer neste sistema. Para isso, foi escolhido um processo de mistura de tintas como caso de estudo. Esse processo é inspirado em um software e hardware chamado ITS (GAMES; LTDA, 2006), que é um sistema de treinamento interativo. O ITS PLC

possui cinco sistemas de simulação virtuais, incluindo o sistema de mistura de tintas, que foi utilizado para montagem e entendimento do processo de fabricação de tintas em um ambiente industrial.

No sistema de mistura de tintas, o objetivo é misturar três cores primárias (vermelho, verde e azul) para obter outras cores. Para conseguir isso, é necessário utilizar uma quantidade específica de cada cor. O sistema industrial estudado é composto por três reservatórios principais que são responsáveis pela mistura de tintas para produzir diferentes cores. O operador inicia o processo selecionando a cor desejada e acionando as válvulas correspondentes a cada uma das três cores primárias (vermelho, verde e azul) como na Figura 9.

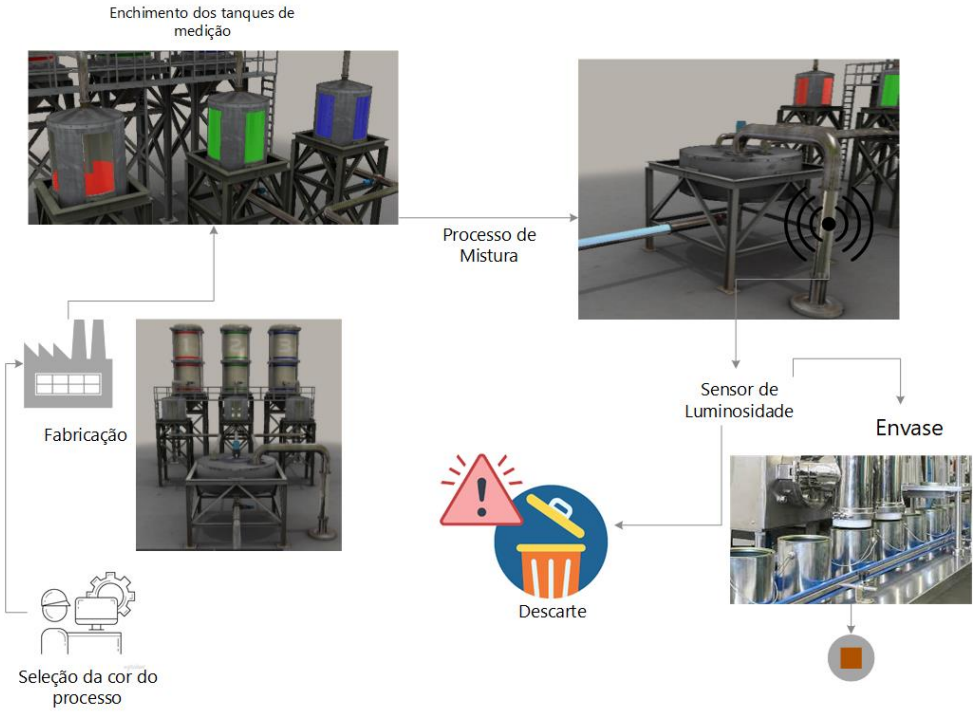


Figura 9 – Fluxograma do processo
Fonte: Autoria Própria

Em seguida, é liberada a quantidade pré-programada de tinta para produzir a cor escolhida. Esses tanques principais fornecem a tinta para os tanques de medição, que possuem sensores de nível alto e médio. Esses sensores são relevantes para garantir a segurança do sistema e proteção contra acionamento incorreto de válvulas. Após o enchimento dos tanques de medição, as tintas são enviadas para o tanque de mistura, onde ocorre a mistura para obter a cor desejada. Neste tanque, existem também dois sensores de nível alto e médio para garantir a segurança do sistema. No entanto, devido à possibilidade de falhas no sistema, um sensor de luminosidade foi instalado na saída do tanque de mistura. O sensor recebe um valor específico para cada cor

produzida e verifica se a tinta resultante está de acordo com o previsto. Caso a cor esteja fora do esperado, o sensor desliga a válvula do tanque de mistura e aciona a válvula do tubo de descarga.

Inicialmente, os reservatórios principais são preenchidos com as tintas necessárias para produzir a cor desejada. A partir daí, a quantidade de tinta necessária é liberada para cada tanque de medição. É importante que o tempo de cada atuador seja cronometrado com precisão, pois isso é essencial para o funcionamento correto do sistema e para a obtenção da cor escolhida. Os atuadores são posicionados de acordo com a Figura 10 que representa o sistema.

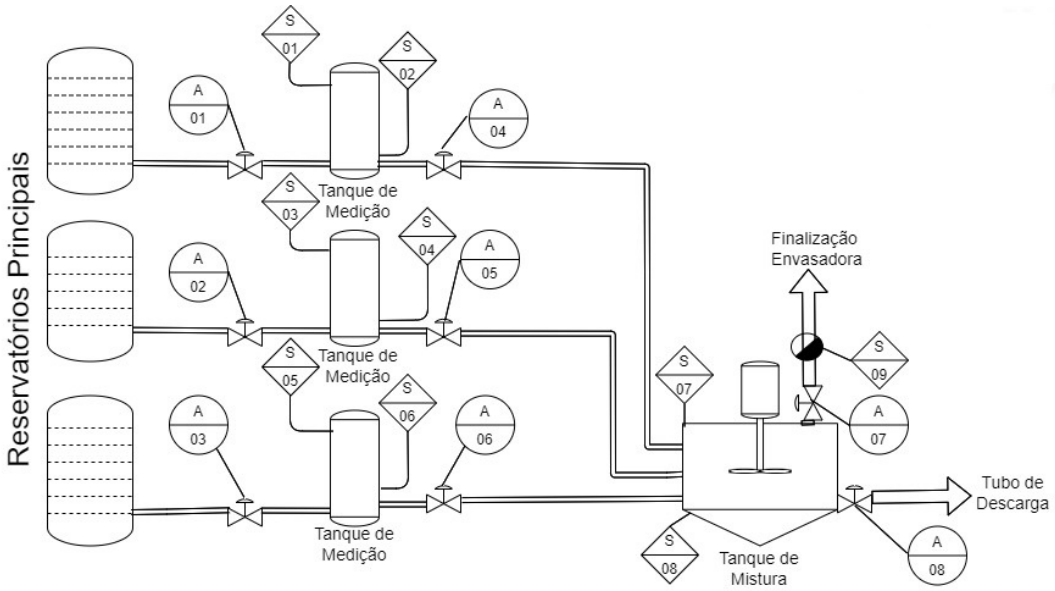


Figura 10 – Sistema de Teste, Autoria Própria baseado no (GAMES; LTDA, 2006).

Tabela 1 – Sensores e Atuadores do Sistema de Teste

Nome	Função
A01	Atuador do reservatório principal 1
A02	Atuador do reservatório principal 2
A03	Atuador do reservatório principal 3
A04	Atuador de saída do tanque de medição A
A05	Atuador de saída do tanque de medição B
A06	Atuador de saída do tanque de medição C
A07	Atuador de finalização envazadora
A08	Atuador do tubo de descarga
S01	Nível alto do tanque de medição A
S02	Nível médio do tanque de medição A
S03	Nível alto do tanque de medição B
S04	Nível médio do tanque de medição B
S05	Nível alto do tanque de medição C
S06	Nível médio do tanque de medição C
S07	Nível inferior do misturador
S08	Nível superior do misturador
S09	Sensor de Luminosidade

Fonte: Autoria própria.

Os sensores de nível médio e alto nos tanques de medição são componentes importantes

do sistema porque indicam não só o tempo, mas também se o nível está correto. Depois que esses pontos são preenchidos e os tanques de medição são cheios, a válvula do tanque de mistura é ativada e permanece aberta por 2 minutos até que a tinta esteja pronta. Se o sensor de luminosidade detectar que a tinta não está dentro dos padrões adequados, a válvula do tanque de mistura será fechada e a válvula do tanque de descarga será ativada para descartar a tinta resultando em prejuízo e perda de tempo na produção.

As cores a serem simuladas serão apresentadas ao final da Seção 5.2, sendo considerada qualquer anomalia ou desvio como um caso preciso de falha do sistema. O objetivo é identificar qualquer anormalidade de cor o mais rápido possível, sendo a identificação a função principal para preservar a qualidade de todo o processo.

Na programação utilizada a representação do sensor de luminosidade foi através de valores em (r,g,b) onde os vetores serão lidos em *pixels* e transformados na cor resultante. Durante a fabricação podem ocorrer algumas falhas no sistema, como nos sensores, válvulas entre outras variáveis do processo que atrapalham o resultado final, sendo capturada pelo sensor de luminosidade.

Em termos simples, a redução da luminosidade da tinta pode ser um problema difícil de ser detectado no sistema de produção. Portanto, a inclusão de um sensor de luminosidade é fundamental para garantir a qualidade e a confiabilidade do processo.

Na Seção 5.2, exploraremos a estrutura fundamental que sustenta o sistema de manufatura em análise. O grupo de dados é concebido para viabilizar o processo produtivo, especialmente no desenvolvimento de 10 cores distintas, representando um conjunto diversificado de produtos finais. Um elemento crucial integrado a esse cenário é o único item de automação, que desempenha um papel crucial no controle operacional: o sensor de luminosidade.

A configuração específica do grupo de dados reflete não apenas a complexidade do processo produtivo, com suas 10 cores distintas, mas também destaca a ênfase na automação através do sensor de luminosidade.

5.2 BANCO DE DADOS

O banco de dados foi desenvolvido englobando informações sobre o funcionamento regular do sistema, bem como a introdução controlada de falhas nos tanques. Especificamente, as falhas foram deliberadamente inseridas no funcionamento dos tanques, manipulando seus níveis. Esta abordagem foi adotada devido à complexidade do processo, onde a cor final produzida

não utiliza a capacidade total do tanque. Dessa forma, ao variar os níveis dos tanques, mesmo que a configuração do processo esteja aparentemente correta, não se tem certeza se o tanque de medição está operando na medida adequada, uma vez que os sensores especificados na Tabela 1 estão posicionados no início e no final do tanque, como ilustrado na Figura 11.

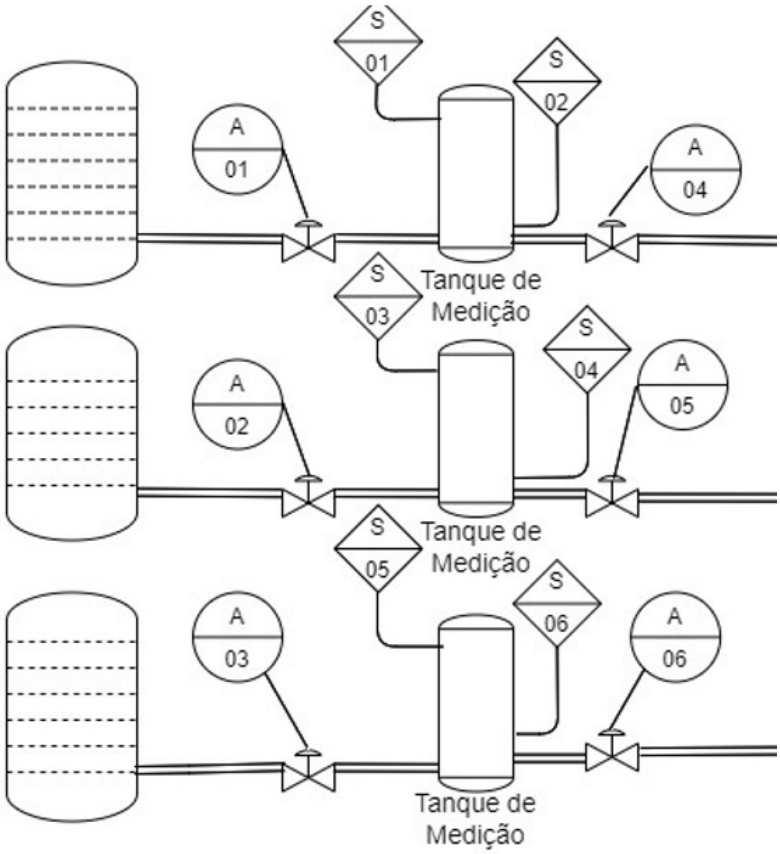


Figura 11 – Posição dos Sensores no Tanque
Fonte: Fonte Própria

Nas falhas inseridas nos tanques 1, 2 e 3, e em alguns casos exclusivamente em algum tanque individual, a aplicação de falhas variou em todos os tanques, com valores oscilando entre 9% e 12% acima ou abaixo, e até 35% de falhas. Esses valores foram escolhidos como critério para representar uma falha, pois abaixo desses valores podem existir outras possibilidades de aplicação do produto. Por exemplo, valores abaixo de 9% seriam destinados à *outlet*, enquanto casos intermediários como apresentado na Figura 12, entre o certo e errado ainda seriam comercializados, não resultando em uma perda total. Essa escolha foi feita para o desenvolvimento da dissertação, que se concentrou na aplicação computacional, sem embasamento específico na literatura.

No processo de enchimento dos tanques, as falhas foram aplicadas de maneira diversificada, mantendo, no entanto, o tempo de enchimento dentro dos parâmetros normais do

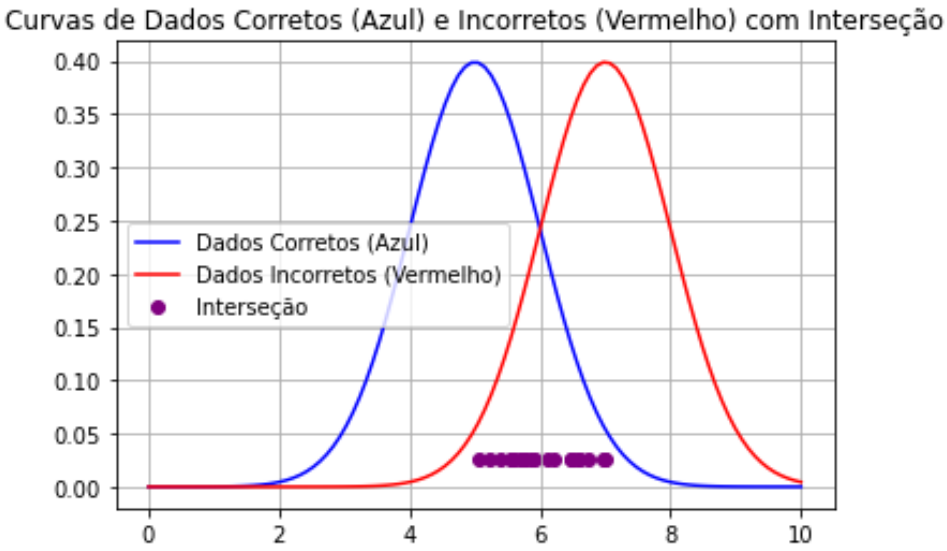


Figura 12 – Representação dos Dados
Fonte: Fonte Própria

processo. Essa decisão foi tomada para isolar as variações nos níveis dos tanques, possibilitando a identificação de problemas de vazão ainda não identificados. Os valores pré-programados para o tempo de enchimento estavam normais, e qualquer variação no nível seria indicativa de uma possível falha no sensor de luminância, já que não estaria dentro dos limites esperados para a comercialização.

A criação do banco de dados foi realizada com o objetivo de preservar sua integridade e minimizar viés que poderia afetar a eficácia dos algoritmos durante o treinamento. Durante a inserção de falhas, foram adotadas práticas para evitar a formação de um bando de dados tedencioso na captura das falhas. A inserção deliberada de falhas no funcionamento dos tanques foi realizada de forma controlada e variada, abrangendo diferentes porcentagens de variação nos níveis dos tanques, além de cenários intermediários que desafiaram a identificação das falhas sem comprometer a qualidade final do produto dentro dos parâmetros aceitáveis para comercialização. Essa abordagem estratégica assegura que o banco de dados não apresente padrões facilmente identificáveis durante o treinamento dos algoritmos, promovendo assim a robustez e a generalização dos modelos de detecção de falhas no contexto do processo de manufatura.

Os parâmetros dos tanques foram estabelecidos com base no tempo de enchimento registrado pelo software ITS Games e Ltda (2006), que desempenhou um papel central no desenvolvimento do projeto. Essa abordagem foi orientada pelos nossos objetivos, que incluíam a garantia de que cada tanque, ao atingir sua capacidade máxima, contivesse mil litros, enquanto

o tanque de mistura mantivesse um nível máximo de três mil litros. A calibração teve como objetivo proporcionar uma representação precisa das condições operacionais, especialmente nos casos intermediários em que o tanque não estivesse completamente cheio. Isso foi crucial para compreender o funcionamento adequado do sistema, considerando que as cores produzidas frequentemente utilizam esses casos intermediários. Conhecer os valores na capacidade máxima de cada tanque e no tanque de mistura contribuiu significativamente para a formulação precisa do banco de dados.

O banco de dados é composto por diversas variáveis cruciais, incluindo o nível e volume dos tanques 1, 2 e 3, o tempo de enchimento de cada tanque, o volume do tanque de mistura, a luminosidade, e, fundamentalmente, a classificação final indicando se há falha ou não. A categorização das falhas é realizada atribuindo o número 0 (zero) para condições normais e 1 (um) para identificação de falhas. Essas variáveis abrangentes e a codificação clara proporcionam uma base para a análise e treinamento dos modelos, visando otimizar a eficácia do sistema de detecção de falhas.

Na Figura 13, são demonstradas as cores produzidas, e seus valores RGB correspondentes estão listados na Tabela 2. Durante a etapa de desenvolvimento do algoritmo, decidiu-se excluir as variáveis relacionadas ao volume dos tanques (Volume do Tanque 1, Volume do Tanque 2, Volume do Tanque 3), por ser uma variável de correlação direta com o nível dos tanques e o tempo de enchimento, visando evitar interferências no processo de treinamento e teste do algoritmo.

Tabela 2 – Valores RGB das cores comercializadas

Cor	Valor RGB
Verde Amarelo	173, 255,47
Gramado Verde	124,252,0
Centaurea Azul	100,149,237
Azul Real	65, 105,225
Salmão	250,128,114
Magenta	255,0,255
Orquídea	218,112,214
Ameixa	221,160,221
Ouro	255,215,0
Laranja	255,165,0

Fonte: Autoria própria.

A estratégia de remover essas variáveis correlacionadas faz parte do pré processamento dos dados, fundamental na abordagem de *machine learning*, visando aprimorar o desempenho do algoritmo. A exclusão de informações redundantes contribui para a eficiência do modelo, evitando sobrecarga no processamento de dados que não agregariam valor significativo ao

treinamento. Essa prática é essencial para garantir que o algoritmo se concentre em padrões relevantes, promovendo uma análise mais precisa e eficiente (BATISTA *et al.*, 2003).

As variáveis selecionadas para o treinamento do modelo foram determinadas com base na Tabela 3, considerando sua importância na análise e detecção de falhas no sistema. Essas variáveis críticas englobam o nível dos tanques 1, 2 e 3, o tempo de enchimento dos tanques 1, 2 e 3, o volume do tanque de mistura e a luminosidade relativa. Cada uma dessas variáveis desempenha um papel essencial na representação abrangente das condições operacionais do sistema de fabricação. O Apêndice B exibe uma parcela do banco de dados que foi composto por 136 amostras, com 13 atributos conforme apresentado na mesma tabela, e uma classe numérica de Falha que categoriza as amostras como 0 (zero) sem falha e 1 (com falha) com falha, referentes à amostra de tinta.

Tabela 3 – Definição dos parâmetros dos tanques.

Parâmetros	**Valor**	**Significado**	**Unidade**
V1	250	Vazão do tanque 1	$litros$
V2	250	Vazão do tanque 2	$litros$
V3	250	Vazão do tanque 3	$litros$
Nível tanque 1	<=1000	Definido pela cor produzida	$litros$
Nível tanque 2	<=1000	Definido pela cor produzida	$litros$
Nível tanque 3	<=1000	Definidio pela cor produzida	$litros$
tempo de enchimento tanque 1	<= 4	Definido pela cor produzida	$segundos$
tempo de enchimento tanque 2	<= 4	Definido pela cor produzida	$segundos$
tempo de enchimento tanque 3	<= 4	Definido pela cor produzida	$segundos$
Volume tanque de Mistura	<= 3.000	Definido pela cor produzida	$litros$
Luminosidade relativa	<=100	Definido pela cor produzida	$porcentagem$

Fonte: Autoria própria.

A inclusão desses parâmetros no treinamento visa garantir que o modelo desenvolvido seja capaz de identificar padrões e anomalias com precisão, contribuindo assim para a eficácia do sistema de detecção de falhas. Essa abordagem estratégica na seleção de variáveis visa criar um modelo robusto e abrangente, capaz de lidar com as complexidades próprias ao processo industrial em questão. No próximo capítulo, serão apresentados os resultados e discussões, destacando como essas variáveis características foram essenciais para o treinamento do modelo e sua influência nos resultados obtidos.

Figura 13 – Cores produzidas

(a) Verde Amarelo **(b) Gramado Verde** **(c) Centaurea Azul**

(d) Azul Real **(e) Salmão** **(f) Magenta**

(g) Orquídea **(h) Ameixa** **(i) Ouro**

(j) Laranja

Fonte: Fonte Própria

6 RESULTADOS E DISCUSSÕES

Neste Capítulo, serão apresentados os resultados obtidos pelos algoritmos, seguidos de uma análise de seu desempenho. Será examinada a capacidade dos modelos de *machine learning* em detectar falhas no sistema de manufatura estudado e classificar as diversas situações operacionais simuladas. Será dada atenção à precisão do modelo em identificar corretamente eventos de falha, bem como em distinguir situações intermediárias em que a cor pode não ser ideal, mas não atinge o critério de falha.

6.1 RESULTADOS K-NEAREST NEIGHBORS

Nesta seção, são apresentados e discutidos os resultados obtidos por meio da aplicação do KNN. Esta técnica de aprendizado supervisionado foi utilizada para realizar classificações em um conjunto de dados previamente processado. A análise abordará diferentes configurações do KNN, incluindo sua aplicação sem redução de dimensionalidade e comparando seu desempenho quando combinado com métodos de redução, como PCA, LDA e NCA. A exposição detalhada dos resultados permitirá uma avaliação abrangente do impacto dessas abordagens na eficácia do KNN na tarefa de classificação.

Na dissertação, quatro métricas foram aplicadas para avaliar o desempenho do modelo de classificação. Abaixo estão suas funcionalidades e métodos empregados para calcular cada uma delas.:

1. Acurácia: Mede a proporção de previsões corretas feitas pelo modelo em todos os tipos de previsões. A acurácia é calculada como o número de previsões corretas (Verdadeiros positivos e Verdadeiros Negativos), dividido pelo total de previsões feitas pelo modelo (certo e errado) (UDDIN *et al.*, 2022).

$$\text{Acurácia} = \frac{\text{Previsões Corretas}}{\text{Total de Previsões}}$$

2. *Recall* (Sensibilidade): O *recall*, também conhecido como sensibilidade, mede a parcela de exemplos positivos identificados corretamente quanto ao total de exemplos positivos existentes. O *recall* é calculado como o número de exemplos positivos identificados corretamente dividido pelo total de exemplos positivos existentes (UDDIN *et al.*, 2022).

$$\text{Recall} = \frac{\text{Positivos Verdadeiros}}{\text{Positivos Verdadeiros}+\text{Falsos Negativos}}$$

3. *Precision* (Precisão):

   A precisão representa o número de exemplos positivos identificados corretamente dividido pelo total de exemplos positivos identificados pelo modelo (UDDIN *et al.*, 2022).

$$\text{Precisão} = \frac{\text{Positivos Verdadeiros}}{\text{Positivos Verdadeiros}+\text{Falsos Positivos}}$$

4. *F1 Score*: O *F1 Score* é uma métrica que combina precisão e *recall* em um único número. É especialmente útil quando há um desequilíbrio significativo entre as classes. O *F1 Score* é calculado a partir da média harmônica entre precisão e *recall* (ZüFLE *et al.*, 2022).

$$F1 \text{ Score} = \frac{2\times\text{Precisão}\times\text{Recall}}{\text{Precisão}+\text{Recall}}$$

   O *F1-Score* oferece uma maneira simplificada de avaliar o desempenho de um modelo, combinando precisão e *recall* em uma única métrica. Ele é calculado como uma média harmônica entre precisão e *recall*, sendo mais sensível a variações menores do que uma média aritmética simples. Assim, um baixo *F1-Score* indica que tanto a precisão quanto o *recall*, ou um deles, estão em níveis inferiores.

Tais métricas desempenham um papel crucial na aplicação industrial e na identificação de falhas em processos produtivos. Em um ambiente industrial, onde a precisão e confiabilidade são fundamentais, essas métricas ajudam a avaliar a eficácia dos modelos de classificação na detecção de falhas e anomalias nos sistemas de produção. O *recall* é especialmente importante, pois mede a capacidade do modelo em identificar todas as falhas existentes, garantindo que nenhum problema crítico passe despercebido. Por outro lado, a precisão assegura que as falhas detectadas sejam reais, evitando falsos positivos que poderiam resultar em interrupções desnecessárias na produção. A acurácia fornece uma visão geral do desempenho do modelo em termos de previsões corretas, enquanto o *F1-Score* oferece um equilíbrio entre precisão e *recall*, sendo especialmente útil quando há desigualdade entre as classes de falhas. Portanto, ao utilizar essas métricas em conjunto, as empresas podem otimizar seus processos de identificação de falhas, garantindo uma produção mais eficiente, confiável e com menos interrupções (ZüFLE *et al.*, 2022; UDDIN *et al.*, 2022).

No modelo KNN os resultados obtidos foram expressados atráves da Tabela 4, na qual todas as métricas estão apresentadas:

Tabela 4 – Valores obtidos e métodos utilizados no KNN

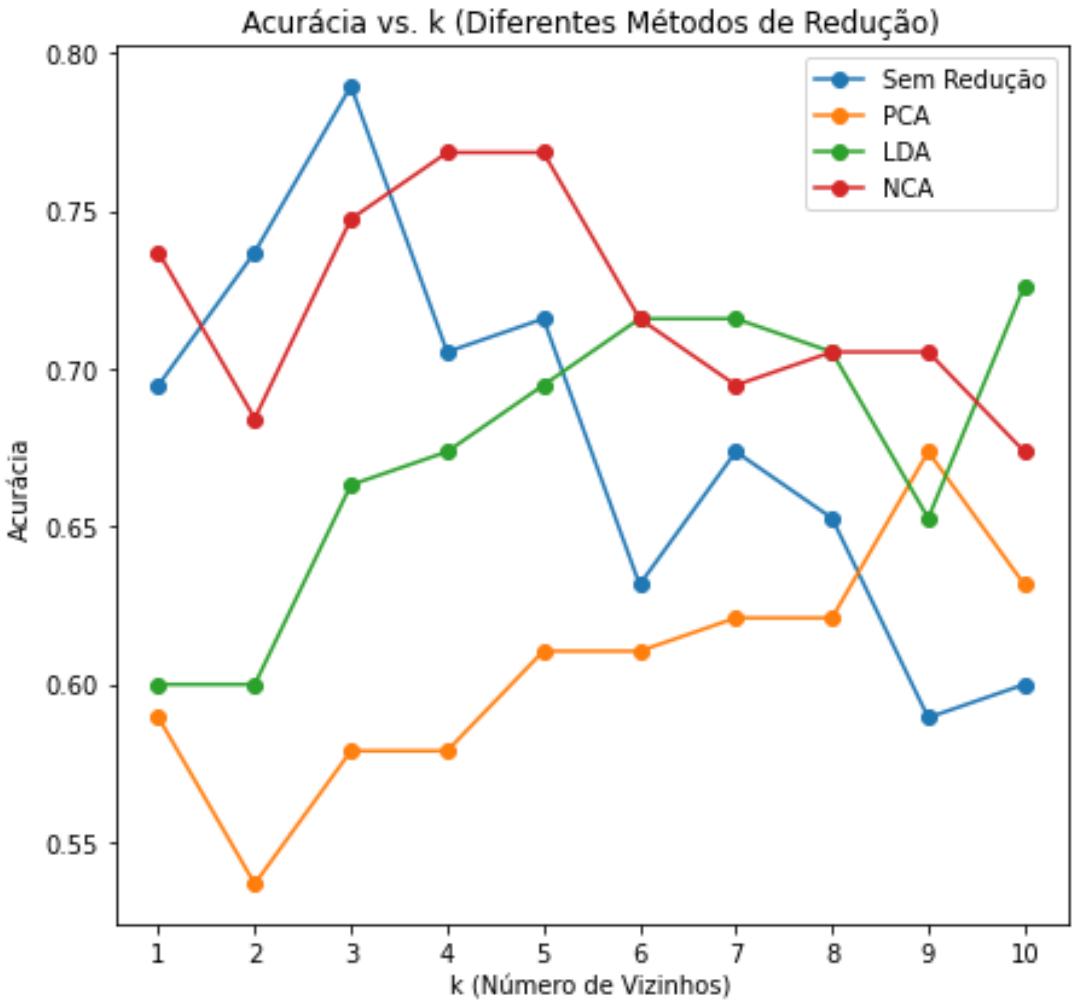
Método	N° de Vizinhos	Acurácia	F1	Precisão	Recall
Sem Redução	1	0.694737	0.652678	0.659787	0.658537
Sem Redução	2	0.736842	0.593018	0.658116	0.609756
Sem Redução	3	0.789474	0.801530	0.814384	0.804878
Sem Redução	4	0.705263	0.732027	0.733217	0.731707
Sem Redução	5	0.715789	0.754625	0.756815	0.756098
Sem Redução	6	0.631579	0.731063	0.731295	0.731707
Sem Redução	7	0.673684	0.754625	0.756815	0.756098
Sem Redução	8	0.652632	0.727195	0.768043	0.731707
Sem Redução	9	0.589474	0.755517	0.769530	0.756098
Sem Redução	10	0.600000	0.677595	0.714447	0.682927
PCA	1	0.589474	0.652678	0.659787	0.658537
PCA	2	0.536842	0.593018	0.658116	0.609756
PCA	3	0.578947	0.801530	0.814384	0.804878
PCA	4	0.578947	0.732027	0.733217	0.731707
PCA	5	0.610526	0.754625	0.756815	0.756098
PCA	6	0.610526	0.731063	0.731295	0.731707
PCA	7	0.621053	0.754625	0.756815	0.756098
PCA	8	0.621053	0.727195	0.768043	0.731707
PCA	9	0.673684	0.755517	0.769530	0.756098
PCA	10	0.631579	0.677595	0.714447	0.682927
LDA	1	0.600000	0.652678	0.659787	0.658537
LDA	2	0.600000	0.593018	0.658116	0.609756
LDA	3	0.663158	0.801530	0.814384	0.804878
LDA	4	0.673684	0.732027	0.733217	0.731707
LDA	5	0.694737	0.754625	0.756815	0.756098
LDA	6	0.715789	0.731063	0.731295	0.731707
LDA	7	0.715789	0.754625	0.756815	0.756098
LDA	8	0.705263	0.727195	0.768043	0.731707
LDA	9	0.652632	0.755517	0.769530	0.756098
LDA	10	0.726316	0.677595	0.714447	0.682927
NCA	1	0.736842	0.652678	0.659787	0.658537
NCA	2	0.684211	0.593018	0.658116	0.609756
NCA	3	0.747368	0.801530	0.814384	0.804878
NCA	4	0.768421	0.732027	0.733217	0.731707
NCA	5	0.768421	0.754625	0.756815	0.756098
NCA	6	0.715789	0.731063	0.731295	0.731707
NCA	7	0.694737	0.754625	0.756815	0.756098
NCA	8	0.705263	0.727195	0.768043	0.731707
NCA	9	0.705263	0.755517	0.769530	0.756098
NCA	10	0.673684	0.677595	0.714447	0.682927

Fonte: Autoria própria.

Na análise dos resultados obtidos, observou-se que o método de redução de dimensionalidade NCA demonstrou um desempenho superior em termos de acurácia quando comparado aos métodos concorrentes como apresentado na Figura 14, incluindo o KNN sem redução e os métodos PCA e LDA. O KNN sem redução, embora seja uma técnica robusta, pode enfrentar desafios em conjuntos de dados de alta dimensionalidade, resultando em uma menor eficácia na

identificação de padrões e relações. O PCA, embora eficiente na redução de dimensionalidade, se concentra na maximização da variância global, podendo perder informações discriminativas importantes para o KNN. O LDA, por sua vez, busca maximizar a separação entre classes, mas pode enfrentar limitações quando as classes não são linearmente separáveis.

Figura 14 – Resultado dos Métodos KNN



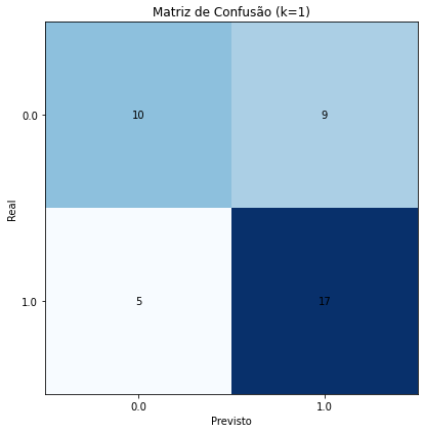
Fonte: Fonte Própria

O destaque para o desempenho superior do NCA pode ser atribuído à sua abordagem inovadora, que se baseia na aprendizagem direta das relações locais entre instâncias, adaptando-se de forma mais sensível às características específicas do conjunto de dados. Ao suavizar a atribuição de vizinhos no novo espaço, o NCA evita o overfitting, permitindo que a classificação de novas observações seja mais precisa. A capacidade do NCA de aprender transformações lineares que preservam relações locais específicas contribui para uma representação mais discriminativa dos dados, resultando em um desempenho aprimorado no contexto do KNN. Esses resultados evidenciam a importância de considerar métodos de redução de dimensionalidade,

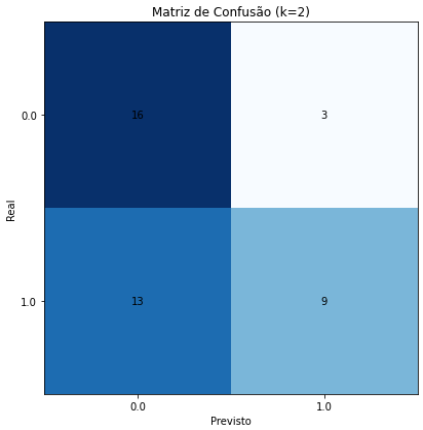
como o NCA, para otimizar o desempenho do KNN em conjuntos de dados complexos.

Ao examinar a Tabela 4, que exibe os resultados de todos os métodos aplicados, destaca-se o desempenho superior alcançado pelo método sem redução com 3 vizinhos. Para uma análise mais aprofundada deste método com melhor desempenho, foram apresentadas as matrizes de confusão nas Figuras 15 e 16, proporcionando uma visão mais detalhada dos resultados do KNN sem redução. A questão dos vizinhos refere-se à proximidade dos pontos semelhantes em nosso caso de identificação de falhas. Quando utilizado um número específico de vizinhos no modelo KNN, está considerando a classe dos vizinhos mais próximos para fazer a previsão da classe de um novo ponto. No contexto da identificação de falhas, isso significa que o modelo teve um desempenho melhor nas quatro métricas aplicadas (*recall*, precisão, acurácia e *F1-Score*) ao considerar 3 vizinhos para fazer suas previsões. Esse resultado destaca a importância de escolher o número adequado de vizinhos para maximizar a capacidade do modelo em detectar e classificar corretamente as falhas no processo industrial.

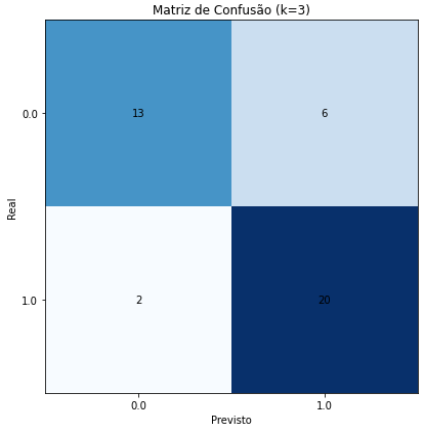
Figura 15 – Resultado das matrizes de confusão - (Sem Redução)



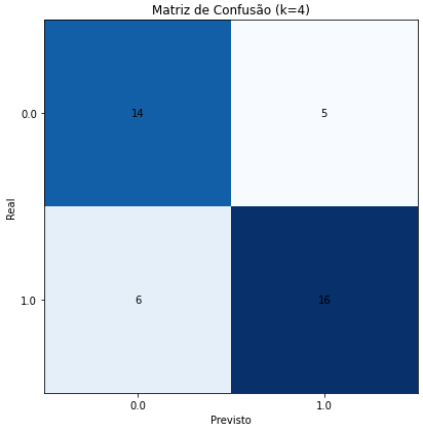
(a) Matriz de Confusão KNN - 1 Vizinho - Sem Redução



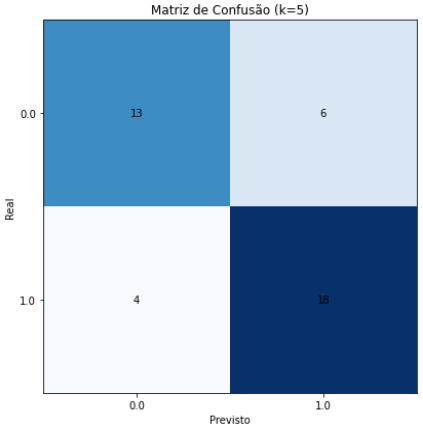
(b) Matriz de Confusão KNN - 2 Vizinhos - Sem Redução



(c) Matriz de Confusão KNN - 3 Vizinhos - Sem Redução

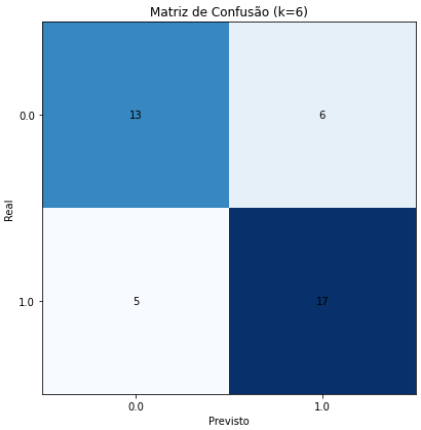


(d) Matriz de Confusão KNN - 4 Vizinhos - Sem Redução

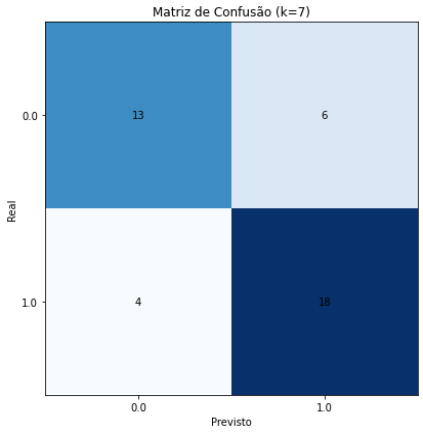


(e) Matriz de Confusão KNN - 5 Vizinhos - Sem Redução

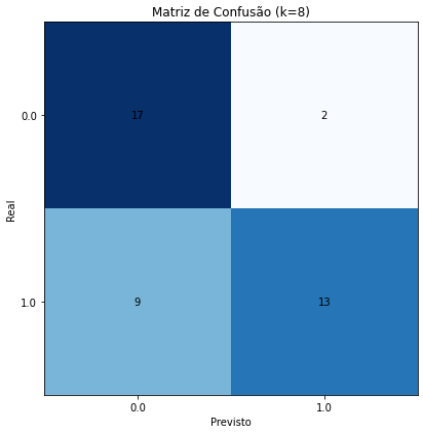
Figura 16 – Resultado das Matrizes de Confusão - (Sem Redução)



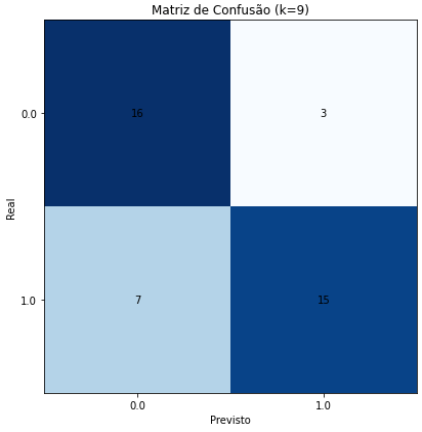
(a) Matriz de Confusão KNN - 6 Vizinhos - Sem Redução



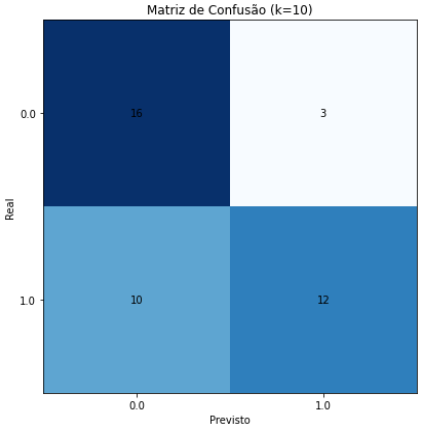
(b) Matriz de Confusão KNN - 7 Vizinhos - Sem Redução



(c) Matriz de Confusão KNN - 8 Vizinhos - Sem Redução



(d) Matriz de Confusão KNN - 9 Vizinhos - Sem Redução



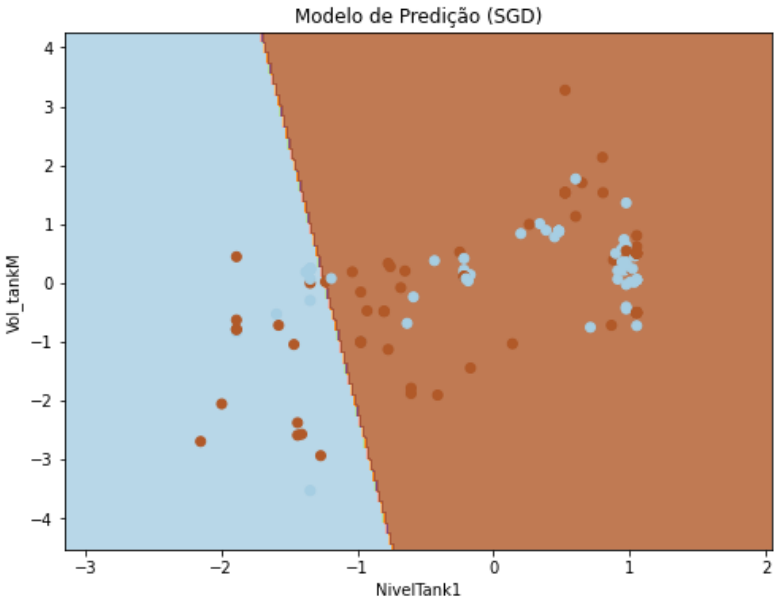
(e) Matriz de Confusão KNN - 10 Vizinhos - Sem Redução

6.2 RESULTADOS MÁQUINAS DE VETORES DE SUPORTE

Nesta seção, serão apresentados os resultados do SVM, conforme descrito no Capítulo 4. A análise do desempenho do SVM foi conduzida em diversos folds, empregando a validação cruzada como método uniforme em todos os algoritmos avaliados.

Os resultados revelaram uma notável variabilidade na eficácia do SVM em diferentes conjuntos de dados, conforme evidenciado na Tabela 5. Essa variação pode ser influenciada por diversos fatores, e uma análise mais aprofundada é necessária para compreender as nuances desses resultados. Existem várias razões pelas quais isso pode ocorrer: modelo inadequado ao meu banco de dados, desiquilíbrios significativos entre as classes, ocorrência de sobreajuste ou adaptação induficiente, hiperparâmetros mal ajustados, como tipo de *kernel* que foi utilizado, como apresentado na Figura 17 onde o parâmetro utilizado foi '*kernel=linear*' podemos ver suas principais caracteristicas por uma linha que atravessa o gráfico, dividindo os dados em duas classes distintas (Eixo (x) = Nível do Tanque 1 e Eixo (y) = Volume do Tanque Médio). Essa linha é traçada de forma que separe claramente as diferentes classes no espaço de características (AMÂNCIO, 2023).

Figura 17 – Modelo de Predição



Fonte: Fonte Própria

A seguir na Tabela 5 serão apresentados os indicadores de desempenho, incluindo acurácia, *F1 Score*, precisão e *recall*, em cada grupo. Os grupos na programação referem-se à técnica de validação cruzada conhecida como *k-grupos cross-validation*. Essa técnica é

essencial para avaliar a capacidade de generalização de um modelo de aprendizado de máquina, especialmente quando o conjunto de dados disponível é limitado, no caso foram utilizados 137 amostras.

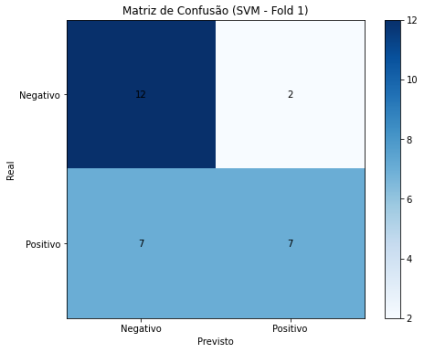
Tabela 5 – Resultados por Fold do SVM

Fold	Acurácia	F1 Score	Precisão	Recall
1	0.6785	0.6086	0.7777	0.5
2	0.4814	0.3636	0.5	0.2857
3	0.4444	0.4	0.4545	0.3571
4	0.6666	0.6086	0.7	0.5384
5	0.7407	0.6666	0.875	0.5384

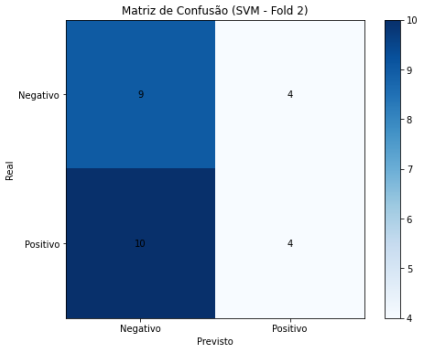
Fonte: Autoria própria.

Os resultados sugerem uma variabilidade nos desempenhos do algoritmo em diferentes conjuntos de dados. Para uma análise mais detalhada do desempenho do modelo, foram também desenvolvidos os resultados da Figura 18, apresentando o desempenho de cada grupo através das matrizes de confusão. A análise desses resultados aponta para um comportamento menos consistente do SVM, evidenciado principalmente pelos resultados mais baixos em alguns grupos. Uma possível explicação para esse comportamento pode estar relacionada aos hiperparâmetros. Pode-se inferir que, em alguns casos, com poucos dados, isso pode ter impactado negativamente no desempenho do SVM, tornando-o menos eficiente em algumas configurações específicas.

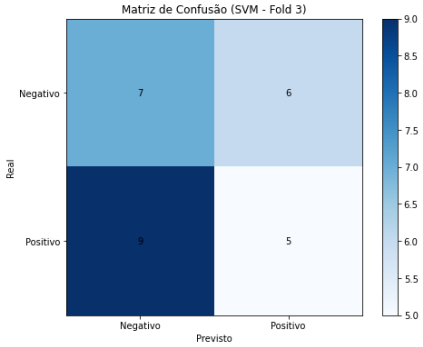
Figura 18 – Resultado das matrizes de confusão



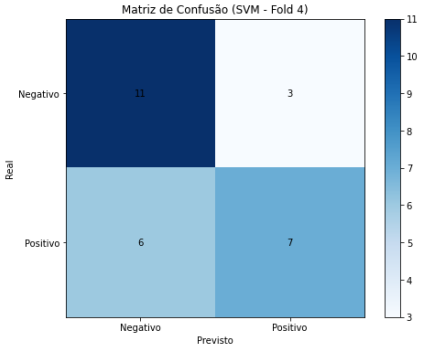
(a) Matriz de Confusão SVM - Fold 1



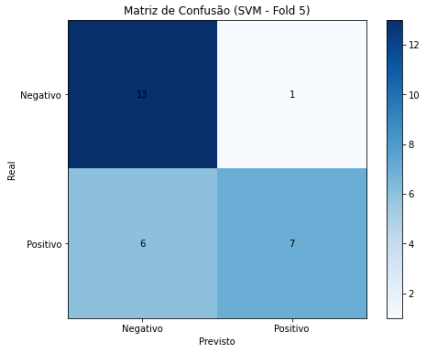
(b) Matriz de Confusão SVM - Fold 2



(c) Matriz de Confusão SVM - Fold 3



(d) Matriz de Confusão SVM - Fold 4



(e) Matriz de Confusão SVM - Fold 5

Na programação utilizada do SVM, os hiperparâmetros definidos são:

- *kernel*=linear: Especifica o tipo de *kernel* a ser utilizado no SVM, neste caso, o kernel linear.
- *random-state* = 1: Define a semente para a geração de números aleatórios, garantindo a reprodutibilidade dos resultados.
- *max-iter*= 1000: Define o número máximo de iterações para o algoritmo SGD (*Stochastic Gradient Descent*) no caso do modelo *SGDClassifier*.

- tol= 1e-3: Define a tolerância para critério de parada do algoritmo SGD no modelo *SGDClassifier*.

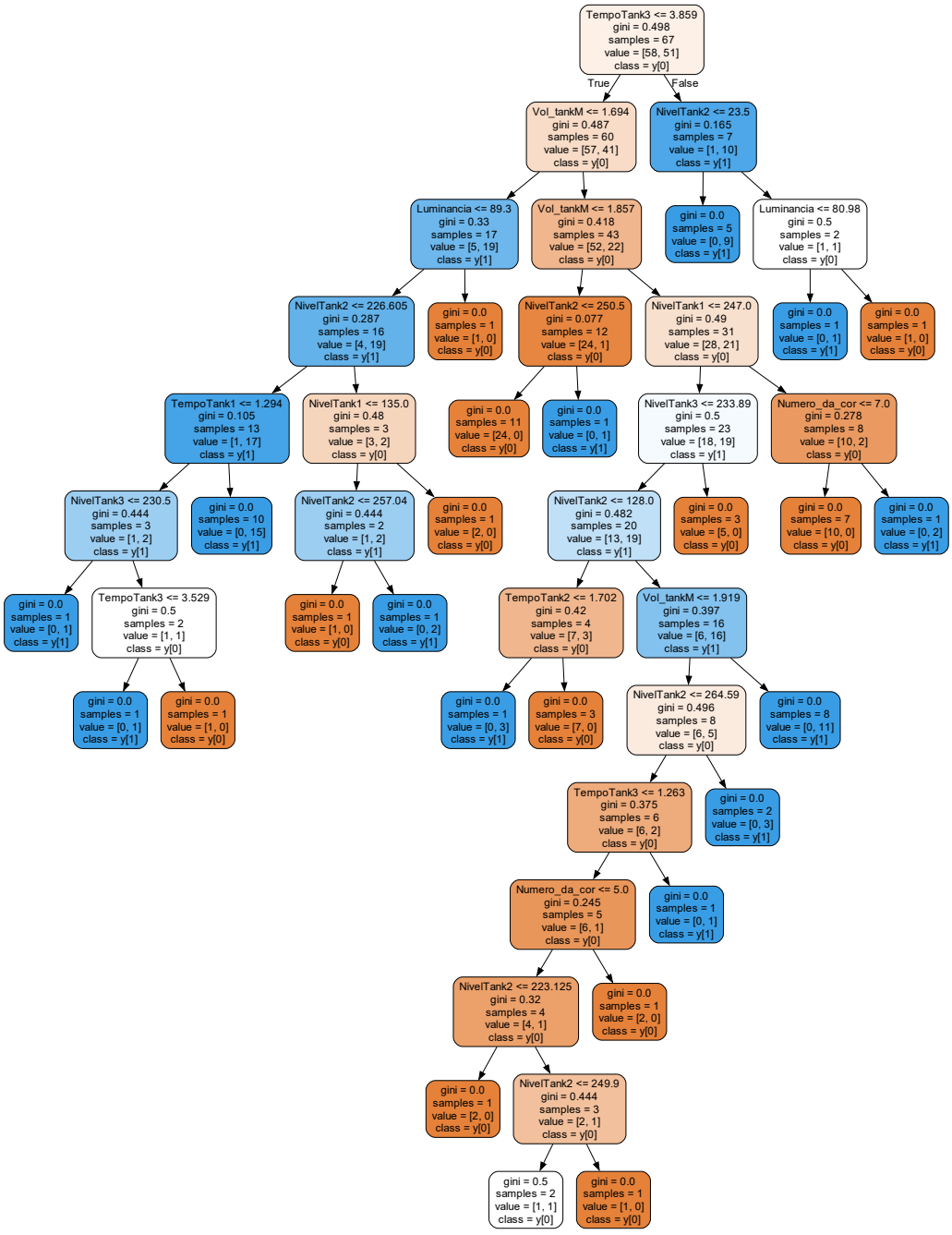
Além desses hiperparâmetros, também foram utilizados outros métodos e funções específicos do *sklearn*, como a normalização dos dados, a seleção de características com *SelectKBest*, a validação cruzada com *StratifiedKFold*, entre outros. Estes elementos contribuem para o ajuste e otimização do modelo SVM para a classificação dos dados.

Além disso, é relevante considerar a natureza binária do banco de dados utilizado. O SVM é conhecido por seu desempenho eficaz em problemas de classificação binária, porém a utilização do *kernel* linear pode ter atrapalhado sua classificação. Essa observação sugere a necessidade de uma análise mais aprofundada e possíveis ajustes na configuração do algoritmo para otimizar seu desempenho em cenários multifacetados.

6.3 RESULTADOS FLORESTA ALEATÓRIA

Na fase de apresentação dos dados referentes à RF, um algoritmo baseado em conjuntos de DT, é essencial compreender primeiramente o seu funcionamento. A *Random Forest* opera por meio da construção de múltiplas árvores de decisão independentes e combina os resultados para produzir uma predição mais robusta e confiável, conforme ilustrado na Figura 19, que representa uma das três árvores resultantes da aplicação do algoritmo. A árvore destaca a relevância dos atributos e quais exercem maior influência no processo de classificação. Demonstrar o caminho percorrido desde a raiz até o nó folha é fundamental para a interpretação do modelo, identificando combinações específicas de valores que levam a diferentes resultados e obtendo um entendimento mais profundo sobre as relações nos dados.

Figura 19 – Uma das arvores criada pelo modelo



Fonte: Fonte Própria

Ao analisarmos os resultados obtidos em cada grupo, observamos uma performance consistente da RF. Em particular, a acurácia variou entre $66{,}67\%$ e $81{,}48\%$, como apresentado na Tabela 6 foi obtido um bom desempenho de classificação. Destaca-se também a precisão, *recall* e *F1-Score*, indicando que o algoritmo manteve um equilíbrio satisfatório entre a identificação correta das instâncias positivas e negativas.

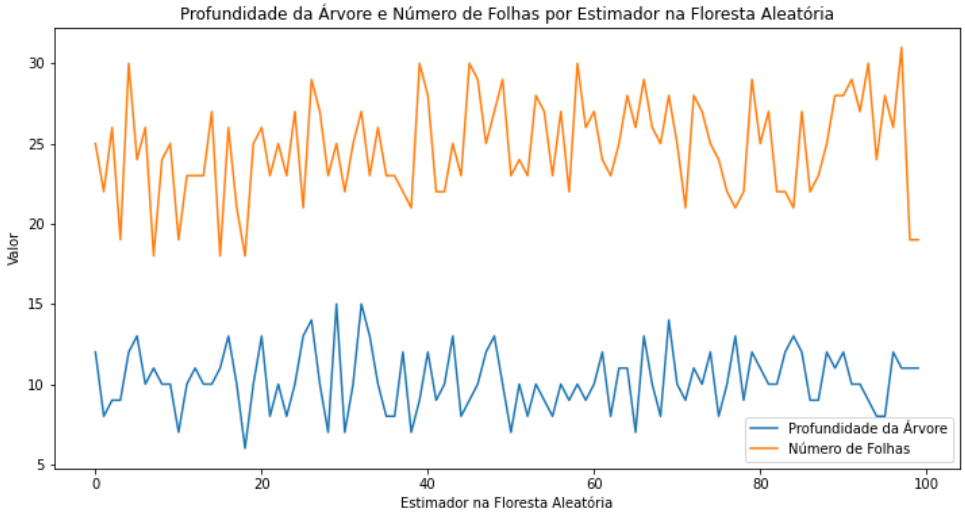
Tabela 6 – Resultados obtidos na Floresta de Decisão

Fold	Acurácia	Precisão	Recall	F1-Score
1	0.7142	0.7187	0.7142	0.7128
2	0.7037	0.7577	0.7037	0.6910
3	0.6666	0.6684	0.6666	0.6666
4	0.7407	0.7552	0.7407	0.7385
5	0.8148	0.8160	0.8148	0.8143

Fonte: Autoria própria.

Uma característica relevante é a análise gráfica que exibe a associação entre a quantidade de folhas e a profundidade da árvore na *Random Forest* como apresentado na Figura 20. Essa análise é crucial para compreender a complexidade do modelo. Notavelmente, o número de folhas variou entre aproximadamente 16, com picos em 30, enquanto a profundidade da árvore ficou na faixa de 6 a 15 em alguns pontos do estimador de profundidade.

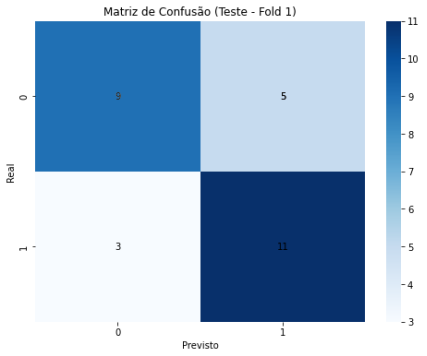
Figura 20 – Profundidade dos dados criados pela Floresta de Decisão



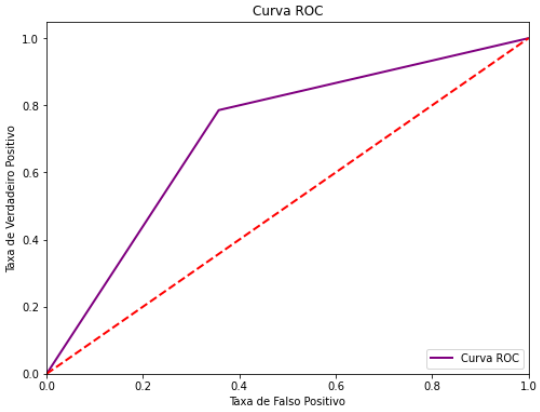
Fonte: Fonte Própria

A importância dessa análise reside na busca por um equilíbrio ideal entre a complexidade do modelo e a capacidade de generalização. Um número excedente de folhas ou uma profundidade muito grande podem levar a um sobreajuste, prejudicando a capacidade do modelo de se adaptar a novos dados. Por outro lado, uma complexidade insuficiente pode resultar em subajuste, comprometendo a precisão do modelo.

Figura 21 – Random Resultado Fold 1

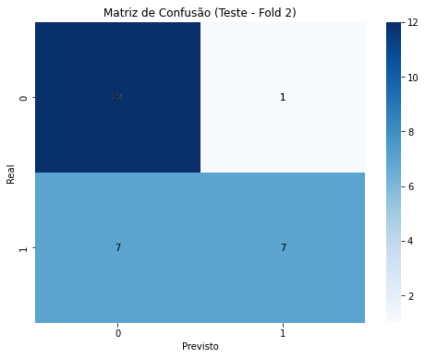


(a) Teste - Fold 1

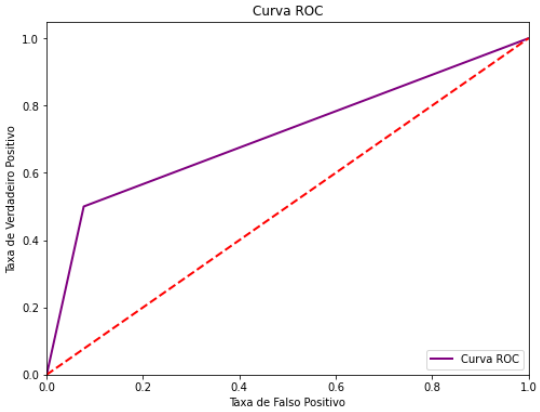


(b) Curva ROC - Fold 1

Figura 22 – Random Resultado Fold 2

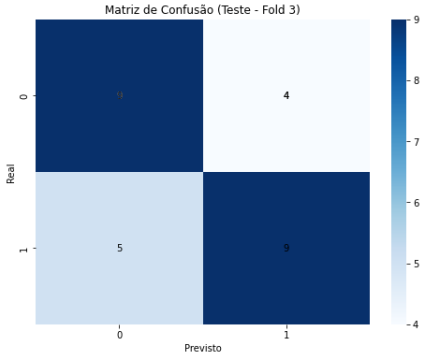


(a) Teste - Fold 2

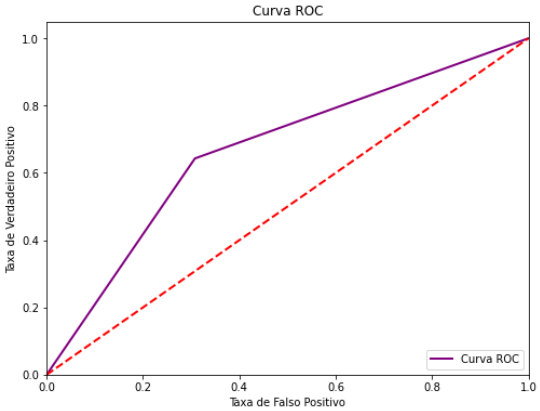


(b) Curva ROC - Fold 2

Figura 23 – Random Resultado Fold 3



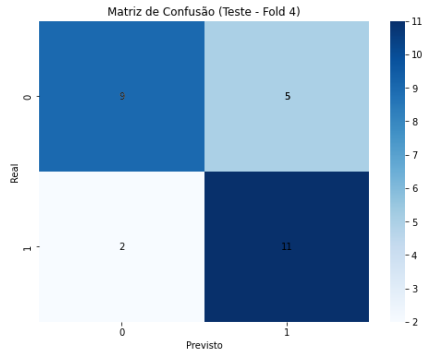
(a) Teste - Fold 3



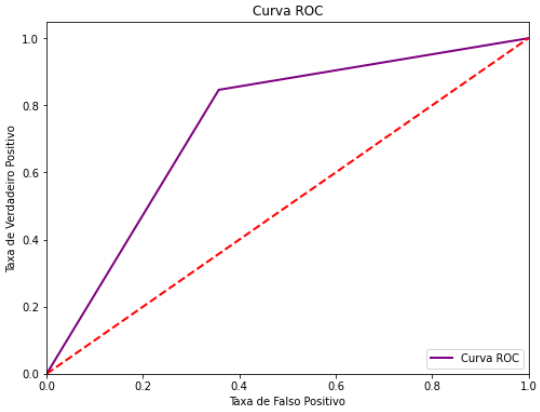
(b) Curva ROC - Fold 3

Portanto, ao analisar as Figuras de 21 a 25, observa-se a dinâmica do modelo em capturar os resultados verdadeiros positivos e falsos através da matriz de confusão. Outro ponto

Figura 24 – Random Resultado Fold 4

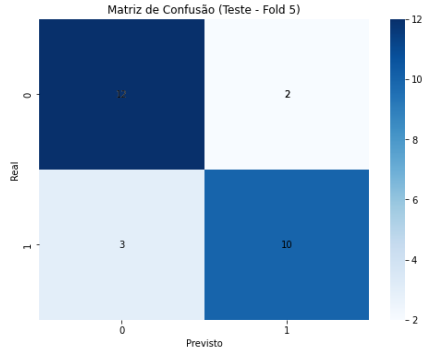


(a) Teste - Fold 4

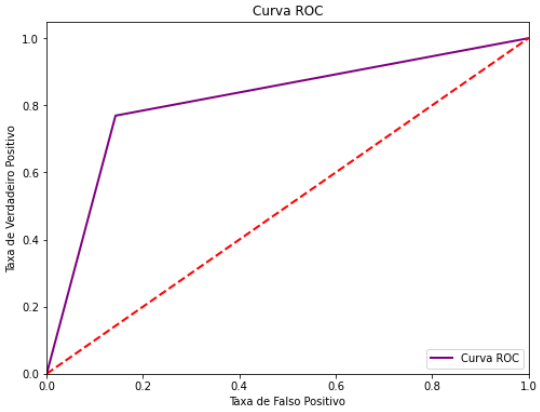


(b) Curva ROC - Fold 4

Figura 25 – Random Resultado Fold 5



(a) Teste - Fold 5



(b) Curva ROC - Fold 5

apresentado é a utilização da curva ROC obtida por cada grupo, essa curva é valiosa para avaliar o quão bem o modelo está discriminando entre as classes positivas e negativas. Quanto mais próxima a curva ROC estiver do canto superior esquerdo do gráfico (ponto ideal), melhor será o desempenho do modelo. Esses parâmetros de resultados fornecem insights cruciais para demonstrar a aplicabilidade do modelo RF, contribuindo para uma tomada de decisão mais informada em relação à sua implementação e ajuste.

6.4 RESULTADOS REDE NEURAL - MLP

Na análise dos resultados da MLP, um algoritmo de aprendizado de máquina inspirado no funcionamento do cérebro humano, foi observado um desempenho notável. O modelo de RNA utilizou duas camadas de neurônios, com 94 na primeira camada e 38 na segunda camada, tais valores foram obtidos com base nos cálculos realizados pelo *Hyperopt*, uma biblioteca de

otimização de hiperparâmetros amplamente utilizada em diversas áreas da ciência de dados, incluindo aprendizado de máquina e *deep learning*. Essa ferramenta é relevante devido ao seu suporte abrangente para otimização de hiperparâmetros em vários algoritmos de ML. No *Hyperopt*, foi empregado o algoritmo TPE *(Tree-structured Parzen Estimator)*, um método de otimização que se baseia na construção de modelos probabilísticos.

O algoritmo TPE se destaca por sua eficácia ao lidar com espaços de busca desconhecidos ou complexos. A forma como funciona inicialmente, constrói duas funções de probabilidade, uma para as configurações consideradas as melhores até o momento e outra para as configurações restantes. Essas funções de probabilidade modelam a chance de uma determinada configuração de hiperparâmetros ser a ideal (VALSECCHI *et al.*, 2021).

Em seguida, o algoritmo TPE realiza a amostragem de configurações do espaço de busca com base nessas funções de probabilidade. Ele utiliza a função restante para escolher configurações de baixa probabilidade e a função melhor para selecionar aquelas de alta probabilidade. As configurações selecionadas são então avaliadas através de uma função de avaliação, que mede o desempenho do modelo com cada configuração (VALSECCHI *et al.*, 2021).

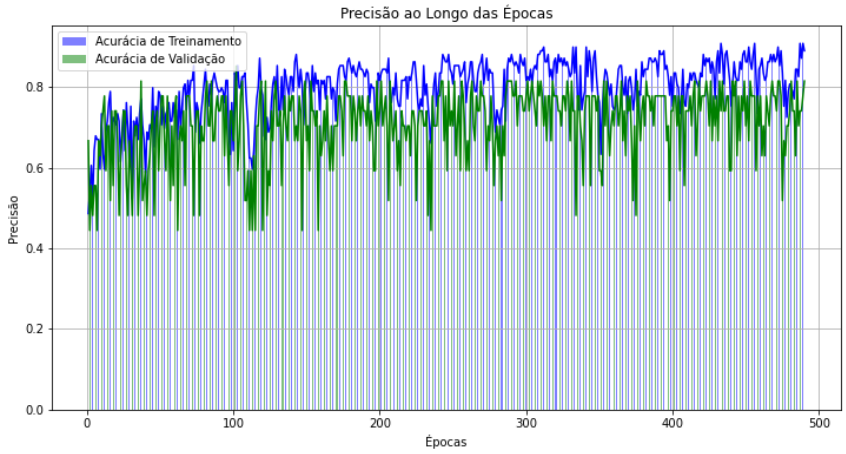
Por meio dessa programação, foi possível automatizar a busca por configurações ideais de hiperparâmetros, aumentando o desempenho do modelo. Esse método se destaca por sua capacidade de explorar totalmente o espaço de busca, resultando em um modelo mais preciso para as tarefas de classificação realizadas durante a pesquisa.

Na programação realizada, foi adotada a função de ativação *ReLU (Rectified Linear Activation)* como uma escolha em redes neurais. Essa função introduz não linearidade ao ativar a saída se a entrada for positiva e zerá-la se for negativa, conferindo eficiência computacional e contribuindo para o processo. Além disso, a *ReLU* auxilia na diminuição de desafios como o desvanecimento do gradiente, promovendo um treinamento mais ágil e eficiente. É importante ressaltar que a seleção da *ReLU* pode variar conforme o contexto específico da rede e do problema em questão. Posteriormente, a programação incorporou a técnica de validação cruzada, fundamental no cenário de aprendizado de máquina, onde *"fold"* se refere à divisão do conjunto de dados original em subconjuntos para treinamento e teste. A validação cruzada é uma ferramenta essencial para avaliar o desempenho de um modelo de *machine learning* e sua capacidade de generalização (ATTO *et al.*, 2022; TAKEKAWA *et al.*, 2021).

Após a aplicação dos métodos de programação, foram gerados gráficos que proporcionam uma visão mais abrangente do desempenho da RNA. A Figura 26 sobre a acurácia de

treinamento e validação permite observar a evolução do modelo durante o treinamento, indicando a capacidade de generalização e adaptação aos dados de validação.

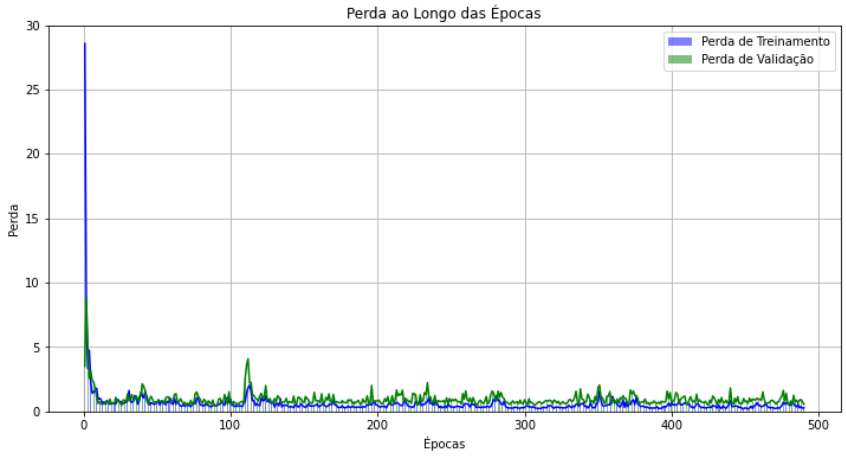
Figura 26 – Acurácia de Treinamento / Validação



Fonte: Fonte Própria

Outro aspecto importante é o gráfico que ilustra as perdas ao longo das épocas. Com 490 épocas, esse gráfico oferece insights sobre a convergência do modelo durante o treinamento. Observar a Figura 27 onde a redução das perdas ao longo do tempo é crucial para garantir que o modelo está aprendendo de maneira eficaz e não está sofrendo de overfitting ou underfitting.

Figura 27 – Perda ao longo das Épocas



Fonte: Fonte Própria

A análise dos resultados apresentado na Tabela 7 revela uma acurácia final de 81,48% para o modelo. Ao desmembrarmos essa métrica, observamos que a acurácia de treinamento atingiu 84,40%, enquanto a acurácia de teste manteve-se em 81,48%.

Tabela 7 – Resultados obtidos RNA-MLP

Fold	Acurácia	Precisão	Recall	F1-Score
1	0.5357	0.5409	0.5357	0.5204
2	0.7777	0.7777	0.7777	0.7777
3	0.6666	0.6684	0.6666	0.6666
4	0.7037	0.7263	0.7037	0.6987
5	0.8148	0.8150	0.8148	0.8137

Fonte: Autoria própria.

Esses números indicam um desempenho sólido do modelo na tarefa em questão. A acurácia de treinamento superior a 80% sugere que o modelo foi capaz de aprender bem os padrões presentes nos dados de treinamento. A acurácia de teste, próxima à acurácia de treinamento, indica que o modelo conseguiu generalizar eficientemente para novos dados, mantendo um desempenho robusto.

É importante notar que a acurácia é uma métrica geral que considera tanto instâncias positivas quanto negativas. Em cenários de classificação binária, como o apresentado, é crucial avaliar outras métricas, como precisão, *recall* e *F1-Score*, para ter uma compreensão mais completa do desempenho do modelo.

No geral, a acurácia final de 81,48% é um indicativo positivo da eficácia do modelo, mas uma análise mais aprofundada das outras métricas e a consideração do contexto específico da aplicação podem enriquecer ainda mais a avaliação do desempenho do modelo de machine learning.

No desfecho deste capítulo, a análise e apresentação dos resultados obtidos por meio de diferentes algoritmos de *machine learning* proporcionam uma visão abrangente sobre a eficácia desses métodos na detecção de falhas em sistemas industriais.

Ao iniciar com o algoritmo KNN, constatamos que a maior acurácia foi alcançada ao aplicar o método sem redução de dimensionalidade, utilizando 3 vizinhos. Isso demonstra que o método sem redução já apresentou um desempenho satisfatório em comparação com outras técnicas. Essa constatação ressalta a relevância de explorar estratégias específicas para cada algoritmo, levando em consideração fatores como a quantidade de vizinhos e o uso de técnicas de redução de dimensionalidade.

A RF, por sua vez, demonstrou uma consistência notável em sua performance, apresentando acurácias que variaram entre 66,67% e 81,48%. A análise gráfica da relação entre a quantidade de folhas e a profundidade da árvore revelou informações sobre a complexidade do modelo e existência de sobreajuste e subajuste, contribuindo para uma compreensão mais

refinada do seu comportamento.

Os resultados da MLP apresentaram um desempenho sólido, com uma acurácia final de 81,48%. A análise detalhada dos gráficos, incluindo acurácia de treinamento e validação, bem como as perdas ao longo das épocas, apresentam resultados sobre a convergência e capacidade de generalização do modelo.

O SVM revelou uma variação considerável em seu desempenho em diferentes folds, destacando a sensibilidade desse algoritmo às características específicas dos conjuntos de dados. A análise desses resultados enfatiza a necessidade de ajustes personalizados para otimizar o desempenho.

Em síntese, cada método de *machine learning* abordado neste capítulo trouxe contribuições fundamentais para a detecção de falhas em sistemas industriais. A compreensão aprofundada de suas características, aliada à interpretação dos resultados obtidos, sustenta o caminho inicial para futuras melhorias e refinamentos nos modelos, visando alcançar níveis ainda mais elevados de eficácia e precisão na detecção de falhas.

7 CONCLUSÃO

A conclusão deste estudo destaca os métodos mais promissores para a detecção de falhas em sistemas industriais, levando em consideração os resultados obtidos através de diversos algoritmos de ML. O KNN, especialmente quando aplicado sem redução de dimensionalidade e com 3 vizinhos, emergiu como uma escolha robusta, apresentando uma notável acurácia no contexto da detecção de falhas.

A RF demonstrou consistência em sua performance obtendo acurácia em 81,48%, evidenciando sua capacidade de lidar com complexidades nos dados e oferecendo uma visão da relação entre a quantidade de folhas e a profundidade da árvore, o que pode ser crucial para futuramente ajustes finos do modelo.

A MLP destacou-se com uma acurácia final de 81,48%, indicando um bom desempenho na tarefa de detecção de falhas, seria necessário uma calibração mais precisa do parâmetros, maior número de amostras. Porém a análise detalhada dos gráficos de treinamento e validação, juntamente com as perdas ao longo das épocas, proporcionou uma compreensão do comportamento do modelo.

O SVM, embora tenha apresentado uma variação considerável em seu desempenho nos diferentes grupos, ofereceu aprendizados valiosos e destaca a importância de adaptações personalizadas para otimizar seu funcionamento em contextos específicos.

A aplicação prática desses métodos no contexto industrial da dissertaçao revela um potencial significativo para aprimorar a eficiência e segurança das operações. Futuramente a utilização de sensores, como o de luminosidade, redução da porcentagem de falhas aceitas no processo, entre outras implementações para o desenvolvimento da automação, promove uma produção mais eficiente e segura. Este estudo, inicialmente ressalta a importância de investimentos contínuos na área de automação industrial.

Como perspectivas futuras, sugere-se um maior investimento na pesquisa e implementação de tecnologias de automação em diversas indústrias. A expansão do uso de sensores, a incorporação de dispositivos avançados e o desenvolvimento de modelos de *machine learning* mais especializados para contextos industriais específicos podem oferecer benefícios substanciais. Este estudo, ao apontar os métodos mais eficazes dentro do contexto aplicado, serve como um guia inicial para direcionar futuras áreas de pesquisa e desenvolvimento no campo da automação industrial e detecção de falhas.

REFERÊNCIAS

ABREU, Rafael Andrade de. Eficácia da manutenção preventiva. **Revista Ibero-Americana de Humanidades, Ciências e Educação**, v. 9, n. 3, p. 2112–2119, 2023.

AHA, David W; KIBLER, Dennis; ALBERT, Marc K. Instance-based learning algorithms. **Machine learning**, Springer, v. 6, p. 37–66, 1991.

ALPAYDIN, Ethem. **Introduction to machine learning**. Massachusetts: MIT press, 2020. 659 p.

ALVAREZ-AROS, E. L.; BERNAL-TORRES, C. A. Technological competitiveness and emerging technologies in industry 4.0 and industry 5.0. **Anais da Academia Brasileira de Ciencias**, 2021.

AMÂNCIO, ANNA CLARA DE O. C. **Um estudo sobre Máquinas de Vetores de Suporte**. 2023. 17 p. Tese (Doutorado) — Universidade Estadual de Campinas, 2023.

AMR, Adel. Future of industry 5.0 in society: human-centric solutions, challenges and prospective research areas. **Journal of cloud computing (Heidelberg, Germany)**, 2022.

ATTO, A. M.; GALICHET, S.; PASTOR, D.; MéGER, N. On joint parameterizations of linear and nonlinear functionals in neural networks. **Neural networks : the official journal of the International Neural Network Society**, 2022.

BATISTA, Gustavo Enrique de Almeida Prado *et al.* **Pré-processamento de dados em aprendizado de máquina supervisionado**. 2003. 232 p. Tese (Doutorado) — Universidade de São Paulo, 2003.

BERTOLINI, Massimo; MEZZOGORI, Davide; NERONI, Mattia; ZAMMORI, Francesco. Machine learning for industrial applications: A comprehensive literature review. **Expert Systems with Applications**, v. 175, p. 114820, 2021. ISSN 0957-4174.

BUENO, Elaine Inácio. **Utilização de Redes Neurais Artificiais na Monitoração e Detecção de Falhas em sensores do reator IEA-R1**. 2006. 111 p. Tese (Doutorado) — Instituto de Pesquisa Energéticas e Nucleares, Universidade de São Paulo, São Paulo, 2006.

CAPESTRO, Mauro; RIZZO, Cristian; KLIESTIK, Tomas; PELUSO, Alessandro M; PINO, Giovanni. Enabling digital technologies adoption in industrial districts: The key role of trust and knowledge sharing. **Technological Forecasting and Social Change**, Elsevier, v. 198, p. 123003, 2024.

CARRASCO, André Colen. **Sistema de detecção de falhas de manobras em seccionadores de alta tensão baseado em processamento digital de sinais e RNA.** Mar 2005. 99 p. Dissertação (Mestrado) — Engenharia Elétrica - Universidade Federal de Itajubá, São Paulo,SP, Mar 2005.

CHARNIAK, Eugene; GOLDMAN, Robert P. A bayesian model of plan recognition. *In*: BOARD, Editorial (Ed.). **Artificial Intelligence**. Massachusetts: Addison, 1993. v. 64, p. 53–79.

COELHO, Pedro Miguel Nogueira. **Rumo à indústria 4.0**. Julho 2016. 65 p. Dissertação (Mestrado) — FCTUC - Faculdade de Ciências e Tecnologia Universidade de Coimbra, Coimbra, Julho 2016.

CORTES, Corinna; VAPNIK, Vladimir. Support-vector networks. **Machine learning**, Springer, v. 20, p. 273–297, 1995.

DALLAGNOL; ALMEIDA, Guth Letícia de; CAVAGNOLI, Sergio. A inteligência artificial na indústria 4.0. **Universidade de Caxias do Sul**, Repositório Institucional, Belo Horizonte, fev. 2022.

DAMASCENO, Aline Moreira. **Avaliação do método SVM (Support Vector Machine) para o mapeamento multitemporal do municipio de São Gonçalo do Amarante-CE**. abr. 2021. 58 p. Dissertação (Mestrado) — UFC - Universidade Federal do Ceará, Fortaleza, abr. 2021.

FARINI, Alessandro. **functions for computing relative luminance in W3C standard**. 2023. MATLAB Central File Exchange. Disponível em: https://www.mathworks.com/matlabcentral/fileexchange/24995-functions-for-computing-relative-luminance-in-w3c-standard.

FEDULLO, T.; MORATO, A.; TRAMARIN, F.; ROVATI, L.; VITTURI, S. A comprehensive review on time sensitive networks with a special focus on its applicability to industrial smart and distributed measurement. **Sensors (Basel, Switzerland)**, 2022.

FERRERO, Carlos Andres. **Algoritmo kNN para previsão de dados temporais: funções de previsão e critérios de seleção de vizinhos próximos aplicados a variáveis ambientais em limnologia**. 2009. 129 p. Tese (Doutorado) — Universidade de São Paulo, 2009.

GAMES, Real; LTDA. **ITS PLC Edição Profissional**. 2006. Disponível em: https://realgames.co/its-plc/.

GOLDBERGER, Jacob; HINTON, Geoffrey E; ROWEIS, Sam; SALAKHUTDINOV, Russ R. Neighbourhood components analysis. **Advances in neural information processing systems**, v. 17, 2004.

GOMES, Dennis dos Santos. Inteligência artificial: conceitos e aplicações. **Revista: Olhar Científico**, v. 1, p. 234–246, ago. 2010.

GONÇALVES, André Ricardo. Aplicação da teoria de otimização convexa em problemas de classificaÇÃo. **Faculdade de Engenharia Elétrica e de Computação - UNICAMP**, ResearchGate, 2009.

GóES, Jessica Cavalcante. **Maturidade na indústria 4.0: estudo de caso em uma fabricante de eletroeletrônicos no Polo Industrial de Manaus por meio do sistema PIMM4.0**. 2024. 17 p. Dissertação (Mestrado) — UFAM - Universidade Federal do Amazonas, 2024.

HAUGELAND, John. **Artificial Intelligence: The Very Idea**. 1985. Massachusetts: The MIT Press.

INABA, Tatsuya. Performance evaluation of iot-enabled predictive maintenance. **2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)**, p. 1551–1555, 2021.

KUCHERYAVSKIY, S.; ZHILIN, S.; RODIONOVA, O.; POMERANTSEV, A. Procrustes cross-validation-a bridge between cross-validation and independent validation sets. **Analytical chemistry, 92(17), 11842–11850**, 2020.

KURZWEIL, Ray. **The Age of Spiritual Machines: When Computers Exceed Human Intelligence**. New York - U.S.A: Penguin Books, 2000. 388 p.

LEITE, Danilo Rangel Arruda; MORAES, Ronei Marcos de; LOPES, Leonardo Wanderley. Método de aprendizagem de máquina para classificação da intensidade do desvio vocal utilizando random forest. **Journal of Health Informatics**, v. 12, 2020.

LEITE, Gabriel Matos Cardoso. **Algoritmos integrados para classificação de dados com atributos categóricos**. 2019. 52 p. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2019.

LU, Yang. Industry 4.0: A survey on technologies, applications and open research issues. **Journal of Industrial Information Integration**, v. 6, p. 1–10, 2017. ISSN 2452-414X.

LYRA, Wellington da Silva; EDVAN, Cirino da Araújo; MARIO, Cesar Ugulino de Fragoso; WALLACE, Duarte Veras Germano. Classificação periódica: um exemplo didático para ensinar análise de componentes principais. **Química Nova**, Scielo Brasil, v. 33, p. 1594–1597, 2010.

MARTíNEZ-DíAZ, Margarita; SORIGUERA, Francesc. Autonomous vehicles: theoretical and practical challenges. **Transportation research procedia**, Elsevier, v. 33, p. 275–282, 2018.

MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. bulletin of mathematical biophysics. **Journal of Symbolic Logic**, Chicago, v. 5, p. 115–133, 1943.

MELLO, Alexandre Reeberg de *et al.* **Contributions to support vector machine classifier**. 2019. Tese (Doutorado) — Universidade Federal de Santa Catarina, 2019.

MENEGAZZO, Alexandre Stefanno. **Investigação de algoritmos de aprendizado de máquinas para detecção de falhas em equipamento industriais através de análise sonora**. 2021. Trabalho de Conclusão de curso - Universidade Federal do Rio Grande do Sul.

MENESES, J. Salvador; RUIZ-CHAVEZ, Z.; RODRIGUEZ, J. Garcia. Compressed knn: K-nearest neighbors with data compression. **Entropy (Basel, Switzerland)**, p. 234, 2019.

MIRANDA, Sandy Sthefany Santana de; MORETT, Amarildo José; SOUZA, Risia Kaliane Santana de. Aplicação da ferramenta fmea para análise de riscos na qualidade do processo de produção em uma empresa de tecnologia localizada no polo industrial de ilhéus. **International Journal of Scientific Management and Tourism**, v. 10, n. 2, p. e739–e739, 2024.

MURTHY, Kolluru Venkata Sreerama. **On growing better decision trees from data.** 1995. 206 p. Dissertação (Mestrado) — Johns Hopkins University, Baltimore, 1995.

NATUCCI, Gabriel Coutinho. **Diagnóstico de falhas para sistemas a eventos discretos: isolamento de componentes usando a álgebra max-plus**. jul. 2016. 118 p. Dissertação (Mestrado) — Universidade Estadual de Campinas, Campinas,SP, jul. 2016.

NGUYEN, Rang MH; BROWN, Michael S. Why you should forget luminance conversion and do something better. *In*: **Proceedings of the ieee conference on computer vision and pattern recognition**. *[S.l.: s.n.]*, 2017. p. 6750–6758.

OTANI, Mario; DIAS, Waldeir Silva; MEIRELES, Sharlene dos Santos; MEDEIROS, José Nataniel Lima; ALVES, Thiago dos Santos; SILVA, Yara Batalha. Proposta de como sair da manufatura tradicional rumo a evolução dos sistemas de fabricação inteligente, um estudo de caso. **Revista Contemporânea**, v. 4, n. 1, p. 2210–2234, 2024.

PEDREGOSA, F.; OUTHERS. Scikit-learn: Machine learning in python. **Journal of Machine Learning Research**, v. 12, p. 2825, 2011.

PEREIRA, Adriano; SIMONETTO, Eugênio de Oliveira. Indústria 4.0: conceitos e perspectivas para o brasil. **Revista da Universidade Vale do Rio Verde**, v. 16, n. 1, 2018.

PEREIRA, António Pedro Teixeira. **Implementation of the Reliability Centered Maintenance methodology to the door systems of a train carriage**. 2023. 109 p. Dissertação (Mestrado) — Universidade do Porto, 2023.

POOLE, David; MACKWORTH, Alan; GOEBEL, Randy. **Computational Intelligence:A Logical Approach**. New York: Oxford University Press, New York, 1998.

REZENDE, Ana Cláudia Barbosa; CORRêA, Henrique Pires; VIEIRA, Flávio Henrique Teles; FILHO, Gilberto Lopes; LOPES, Gustavo de Castro. Redução de dimensionalidade para reconhecimento de padrões eletromiográficos de movimentos dos dedos da mão. **Brazilian Journal of Development**, v. 7, n. 3, p. 25728–25741, 2021.

RIBEIRO, Johny Marques Borges. **Métodos de aprendizado de máquina para inventário de elementos de vias de trânsito**. 2021. 111 p. Dissertação (Mestrado) — Universidade Federal de Uberlândia, Uberlândia, 2021.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial**. 3. ed. Rio de Janeiro: Elsevier, 2013. 1188,1189 e 1194 p.

RüSSMANN, Michael; LORENZ, Markus; GERBERT, Philipp; WALDNER, Manuela; JUSTUS, Jan; ENGEL, Pascal; HARNISCH, Michael. Industry 4.0: The future of productivity and growth in manufacturing industries. **Boston consulting group**, Boston, MA, USA:, v. 9, n. 1, p. 54–89, 2015.

SANTOS, Eulanda Miranda dos. **Teoria e aplicação de support vector machines à aprendizagem e reconhecimento de objetos baseado na aparência.** 2002. 121 p. Dissertação (Mestrado) — Universidade Federal da Paraíba, Campina Grande, 2002.

SANTOS, Raphael D'Lucca Santos dos; ARAúJO, André Vinícius da Costa; PINA, Anna Júlia Sousa de; CORRêA, Brenda do Socorro da Silva; ROCHA, Marcus Pinto da Costa da; FARIAS, Valcir João da Cunha. Gerenciamento de manutenção industrial com a metodologia business intelligence. **Proceeding Series of the Brazilian Society of Computational and Applied Mathematics**, v. 10, n. 1, 2023.

SILVA, Adenilton Camilo da *et al.* **Análise discriminante linear em duas dimensões para classificação de dados químicos de segunda ordem**. 2017. 119 p. Tese (Doutorado) — Universidade Federal da Paraíba, 2017.

SILVA, Macello Ferreira Vilela. **Detecção de falhas em ativos industriais através de machine learning aplicado a análise de vibrações**. 2022. Trabalho de Conclusão de curso - Universidade Federal de Santa Catarina.

SIMEóN, Edgar Jhonny Amaya. **Aplicação de técnicas de inteligência artificial no desenvolvimento de um sistema de manutenção baseada em condição**. 2008. 193 p. Dissertação (Mestrado) — Universidade de Brasília, 2008.

TAKEKAWA, A.; KAJIURA, M.; FUKUDA, H. Role of layers and neurons in deep learning with the rectified linear unit. **Cureus**, 2021.

UDDIN, S.; HAQUE, I.; LU, H.; MONI, M. A.; GIDE, E. Comparative performance analysis of k-nearest neighbour (knn) algorithm and its different variants for disease prediction. **Scientific reports**, 2022.

VALSECCHI, C.; CONSONNI, V.; TODESCHINI, R.; ORLANDI, M. E.; GOSETTI, F.; BALLABIO, D. Parsimonious optimization of multitask neural network hyperparameters. **Molecules (Basel, Switzerland)**, 2021.

VALTONEN, L.; MäKINEN, S.J.; KIRJAVAINEN, J. Supervised machine learning in detecting patterns in competitive actions. **2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)**, p. 442–446, 2021.

VARELLA, Carlos Alberto Alves. Análise de componentes principais. **Universidade Federal Rural do Rio de Janeiro**, p. 38, 2008.

WEG. **Sensores Industriais**. 2023. WEG. Disponível em: https://static.weg.net/medias/downloadcenter/h57/h55/WEG-sensores-industriais-50029077-catalogo-pt.pdf.

WUEST, Thorsten; WEIMER, Daniel; IRGENS, Christopher; THOBEN, Klaus-Dieter. Machine learning in manufacturing: advantages, challenges, and applications. **Production & Manufacturing Research**, Taylor and Francis, v. 4, n. 1, p. 23–45, 2016.

ZAYTOON, Janan; LAFORTUNE, Stéphane. Visão geral dos métodos de diagnóstico de falhas para sistemas de eventos discretos. **Revisões Anuais em Controle**, v. 37, n. 2, p. 308–320, dez. 2013.

ZHANG, X.; SONG, Q. A multi-label learning based kernel automatic recommendation method for support vector machine. **PlosOne**, p. 30, 2015.

ZüFLE, Marwin; MOOG, Felix; LESCH, Veronika; KRUPITZER, Christian; KOUNEV, Samuel. A machine learning-based workflow for automatic detection of anomalies in machine tools. **ISA Transactions**, v. 125, p. 445–458, 2022. ISSN 0019-0578.

APÊNDICE A – CÓDIGOS PYTHON

A.1 KNN

Código do Algoritmo KNN

Fonte consultada: Scikit-Learn (PEDREGOSA; OUTHERS, 2011)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, f1_score, recall_score,
↪  precision_score, classification_report
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import NeighborhoodComponentsAnalysis

# Suponhamos que você tenha um DataFrame com nomes de atributos e classes
data = pd.read_csv('certo4t.csv')  # Substitua pelo seu conjunto de dados

# Remover as colunas especificadas
cols_to_remove = ["Vol_tank1", "Vol_tank2", "Vol_tank3"]
data = data.drop(cols_to_remove, axis=1)


# Dividir o conjunto de dados em treinamento (70%) e teste (30%)
X = data.drop(columns=['Falha'])  # Substitua 'Classe' pelo nome da sua
↪  coluna de classe
y = data['Falha']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
↪  random_state=1)

# Testar diferentes valores de k (número de vizinhos)
k_values = range(1, 11)
```

```
accuracy_values = {'Sem Redução': [], 'PCA': [], 'LDA': [], 'NCA': []}

for k in k_values:
    modelo_knn = KNeighborsClassifier(n_neighbors=k)

    # Validação cruzada para avaliar o desempenho
    accuracy_cv = cross_val_score(modelo_knn, X_train, y_train, cv=5,
    ↪  scoring='accuracy')

    accuracy_values['Sem Redução'].append(accuracy_cv.mean())

# Aplicar PCA (Redução de Dimensionalidade)
n_components_pca = min(X_train.shape[1], len(np.unique(y_train)) - 1)  #
↪  Ajustar o número de componentes principais
pca = PCA(n_components=n_components_pca)
X_pca = pca.fit_transform(X_train)

for k in k_values:
    modelo_knn = KNeighborsClassifier(n_neighbors=k)

    # Validação cruzada para avaliar o desempenho com PCA
    accuracy_cv_pca = cross_val_score(modelo_knn, X_pca, y_train, cv=5,
    ↪  scoring='accuracy')
    accuracy_values['PCA'].append(accuracy_cv_pca.mean())

# Aplicar LDA (Redução de Dimensionalidade)
n_components_lda = min(X_train.shape[1], len(np.unique(y_train)) - 1)  #
↪  Ajustar o número de componentes principais
lda = LinearDiscriminantAnalysis(n_components=n_components_lda)
X_lda = lda.fit_transform(X_train, y_train)

for k in k_values:
    modelo_knn = KNeighborsClassifier(n_neighbors=k)

    # Validação cruzada para avaliar o desempenho com LDA
    accuracy_cv_lda = cross_val_score(modelo_knn, X_lda, y_train, cv=5,
    ↪  scoring='accuracy')
    accuracy_values['LDA'].append(accuracy_cv_lda.mean())
```

```
# Aplicar NCA (Redução de Dimensionalidade)
n_components_nca = min(X_train.shape[1], len(np.unique(y_train)) - 1)  #
↪  Ajustar o número de componentes principais
nca = NeighborhoodComponentsAnalysis(n_components=n_components_nca)
X_nca = nca.fit_transform(X_train, y_train)

for k in k_values:
    modelo_knn = KNeighborsClassifier(n_neighbors=k)

    # Validação cruzada para avaliar o desempenho com NCA
    accuracy_cv_nca = cross_val_score(modelo_knn, X_nca, y_train, cv=5,
    ↪  scoring='accuracy')
    accuracy_values['NCA'].append(accuracy_cv_nca.mean())

# Plot dos gráficos de desempenho para k
plt.figure(figsize=(12, 6))

plt.subplot(121)
for method, values in accuracy_values.items():
    plt.plot(k_values, values, marker='o', label=method)

plt.title('Acurácia vs. k (Diferentes Métodos de Redução)')
plt.xlabel('k (Número de Vizinhos)')
plt.ylabel('Acurácia')
plt.xticks(k_values)
plt.legend()

plt.tight_layout()

plt.show()

results = pd.DataFrame(columns=['Método', 'Número de Vizinhos', 'Accuracy',
↪  'F1', 'Precision', 'Recall'])

for method, values in accuracy_values.items():
    for k, accuracy in zip(k_values, values):
        modelo_knn = KNeighborsClassifier(n_neighbors=k)
```

```
        modelo_knn.fit(X_train, y_train)
        y_pred = modelo_knn.predict(X_test)
        accuracy_test = accuracy_score(y_test, y_pred)
        f1_test = f1_score(y_test, y_pred, average='weighted')
        precision_test = precision_score(y_test, y_pred,
        ↪  average='weighted')
        recall_test = recall_score(y_test, y_pred, average='weighted')
        results = results.append({'Método': method, 'Número de Vizinhos':
        ↪  k, 'Accuracy': accuracy, 'F1': f1_test, 'Precision':
        ↪  precision_test, 'Recall': recall_test}, ignore_index=True)

print(results)
```

A.2 SVM

Código do Algoritmo SVM

Fonte consultada: Scikit Learn (PEDREGOSA; OUTHERS, 2011)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, StratifiedKFold
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score,
↪  precision_score, recall_score
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.decomposition import PCA
from sklearn.linear_model import SGDClassifier
from sklearn.multiclass import OneVsRestClassifier

# Carregue o conjunto de dados de teste 'certo3.csv'
data = pd.read_csv('certo4t.csv', encoding='latin1')

# Remover as colunas especificadas
cols_to_remove =
↪  ["NivelTank2","TempoTank2","TempoTank2","NivelTank3","TempoTank3","Vol_tank1",
↪  "Vol_tank2", "Vol_tank3","Luminancia","Numero_da_cor"]
```

```
data = data.drop(cols_to_remove, axis=1)

# Suponhamos que a última coluna seja a coluna de destino, ajuste isso
↪  conforme necessário
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

labelencoder = LabelEncoder()

for i in range(X.shape[1]):
    if X[:, i].dtype == 'object':
        X[:, i] = labelencoder.fit_transform(X[:, i])

# Normalização dos dados de entrada
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Divisão dos dados por validação cruzada
kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
fold = 0

# Selecionar as características mais importantes usando SelectKBest
selector = SelectKBest(score_func=f_classif, k=2)
X_selected = selector.fit_transform(X, y)

for train_index, test_index in kf.split(X_selected, y):
    X_treinamento, X_test = X_selected[train_index],
    ↪  X_selected[test_index]
    y_treinamento, y_test = y[train_index], y[test_index]

    # Crie um classificador SVM com kernel linear
    modelo_svm = SVC(kernel='linear', random_state=1)

    # Treine o modelo SVM
    modelo_svm.fit(X_treinamento, y_treinamento)

    # Realize previsões
    y_pred_test = modelo_svm.predict(X_test)
```

```
# Calcule a matriz de confusão
matriz_confusao = confusion_matrix(y_test, y_pred_test)

# Calcule as métricas de desempenho
accuracy = accuracy_score(y_test, y_pred_test)
f1 = f1_score(y_test, y_pred_test)
precision = precision_score(y_test, y_pred_test)
recall = recall_score(y_test, y_pred_test)

# Exiba as métricas de desempenho
print(f"Fold {fold+1}")
print(f"Acurácia: {accuracy}")
print(f"F1 Score: {f1}")
print(f"Precisão: {precision}")
print(f"Recall: {recall}")

# Plote a matriz de confusão com valores
plt.figure(figsize=(8, 6))
plt.imshow(matriz_confusao, interpolation='nearest',
↪  cmap=plt.cm.Blues)
plt.title(f"Matriz de Confusão (SVM - Fold {fold+1})")
plt.colorbar()
plt.xticks([0, 1], ['Negativo', 'Positivo'])
plt.yticks([0, 1], ['Negativo', 'Positivo'])

# Adicione os números aos quadrados da matriz de confusão
for i in range(matriz_confusao.shape[0]):
    for j in range(matriz_confusao.shape[1]):
        plt.text(j, i, f"{matriz_confusao[i, j]}", ha='center',
        ↪  va='center', color='black')

plt.xlabel('Previsto')
plt.ylabel('Real')
plt.show()

fold += 1
```

```
# Agora, treine o modelo SGD usando OneVsRestClassifier
sgd = SGDClassifier(max_iter=1000, tol=1e-3)
ovr = OneVsRestClassifier(sgd)
ovr.fit(X_selected, y)

# Obtenha os nomes reais das características selecionadas pelo SelectKBest
selected_feature_indices = selector.get_support(indices=True)
feature_names = data.columns[selected_feature_indices]

# Plote o modelo de predição criado pelo SGD
plt.figure(figsize=(8, 6))

# Crie um meshgrid para plotar a região de decisão
x_min, x_max = X_selected[:, 0].min() - 1, X_selected[:, 0].max() + 1
y_min, y_max = X_selected[:, 1].min() - 1, X_selected[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max,
↪  0.02))

# Calcule as previsões para a região de decisão
Z_sgd = ovr.predict(np.c_[xx.ravel(), yy.ravel()])
Z_sgd = Z_sgd.reshape(xx.shape)

# Plote a região de decisão
plt.contourf(xx, yy, Z_sgd, alpha=0.8, cmap=plt.cm.Paired)
plt.scatter(X_selected[:, 0], X_selected[:, 1], c=y, cmap=plt.cm.Paired)
plt.xlabel(feature_names[0])  # Nome real da primeira característica
plt.ylabel(feature_names[1])  # Nome real da segunda característica
plt.title('Modelo de Predição (SGD)')

plt.show()
```

A.3 RF

Código do Algoritmo RF

Fonte consultada: Scikit Learn (PEDREGOSA; OUTHERS, 2011)

```
import csv
import random
import pandas as pd
import graphviz
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import cross_val_predict, StratifiedKFold
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    classification_report,
    confusion_matrix,
)
from sklearn.metrics import roc_curve
import os

# Carregue o conjunto de dados de teste 'certo3.csv'
data = pd.read_csv('certo4t.csv', encoding='latin1')

# Remover as colunas especificadas
cols_to_remove = ["Vol_tank1", "Vol_tank2", "Vol_tank3"]
data = data.drop(cols_to_remove, axis=1)

# Suponhamos que a última coluna seja a coluna de destino, ajuste isso
↪  conforme necessário
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

labelencoder = LabelEncoder()

for i in range(X.shape[1]):
    if X[:, i].dtype == 'object':
```

```
        X[:, i] = labelencoder.fit_transform(X[:, i])

# Criar o modelo
modelo = RandomForestClassifier(random_state=1)

# Definir validação cruzada estratificada
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)

# Criar um DataFrame para armazenar os resultados
resultados_df = pd.DataFrame(columns=['Fold', 'Accuracy', 'Precision',
↪  'Recall', 'F1-Score'])

# Iniciar loop de validação cruzada
for i, (train_index, test_index) in enumerate(cv.split(X, y)):
    X_treinamento, X_test = X[train_index], X[test_index]
    y_treinamento, y_test = y[train_index], y[test_index]

    # Treinar o modelo com os dados de treinamento
    modelo.fit(X_treinamento, y_treinamento)

    # Realizar previsões nos dados de treinamento e teste
    y_pred_treinamento = modelo.predict(X_treinamento)
    y_pred_test = modelo.predict(X_test)

    # Calcula as métricas para os dados de teste
    accuracy_test = accuracy_score(y_test, y_pred_test)
    precisao_test = precision_score(y_test, y_pred_test,
    ↪  average='weighted')
    recall_test = recall_score(y_test, y_pred_test, average='weighted')
    f1_test = f1_score(y_test, y_pred_test, average='weighted')

    # Armazenar os resultados no DataFrame
    resultados_df = resultados_df.append({
        'Fold': i + 1,
        'Accuracy': accuracy_test,
        'Precision': precisao_test,
        'Recall': recall_test,
        'F1-Score': f1_test
```

```
    }, ignore_index=True)

# Exibir a tabela de resultados
print("Resultados por Fold:")
print(resultados_df)

# Plotar gráfico sobre a profundidade da floresta aleatória e o número de
↪  folhas
plt.figure(figsize=(12, 6))
sns.lineplot(x=range(len(modelo.estimators_)),
↪  y=[estimator.tree_.max_depth for estimator in modelo.estimators_],
↪  label='Profundidade da Árvore')
sns.lineplot(x=range(len(modelo.estimators_)), y=[estimator.tree_.n_leaves
↪  for estimator in modelo.estimators_], label='Número de Folhas')
plt.xlabel('Estimador na Floresta Aleatória')
plt.ylabel('Valor')
plt.title('Profundidade da Árvore e Número de Folhas por Estimador na Floresta Aleató
plt.legend()
plt.savefig('profundidade_numero_folhas.png')
plt.show()
```

A.4 RNA MLP

Código do Algoritmo RNA MLP

Fonte consultada: Scikit Learn (PEDREGOSA; OUTHERS, 2011)

```
_ Hiperparâmetros

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
from hyperopt import fmin, tpe, hp, STATUS_OK, Trials
```

```
# Carregue o conjunto de dados
data = pd.read_csv('certo4t.csv', encoding='latin1')

# Suponhamos que a última coluna seja a coluna de destino, ajuste isso
↪  conforme necessário
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

labelencoder = LabelEncoder()

for i in range(X.shape[1]):
    if X[:, i].dtype == 'object':
        X[:, i] = labelencoder.fit_transform(X[:, i])

# Dividir o conjunto de dados em treinamento (70%) e teste (30%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
↪  random_state=1)

# Codificar as saídas em formato one-hot (para classificação multiclasse)
y_train_encoded = to_categorical(y_train)
y_test_encoded = to_categorical(y_test)

# Defina uma função de avaliação para o Hyperopt
def objective(params):
    model = Sequential()
    model.add(Dense(params['neurons_layer1'], input_dim=X_train.shape[1],
    ↪  activation='relu'))
    model.add(Dense(params['neurons_layer2'], activation='relu'))
    model.add(Dense(y_train_encoded.shape[1], activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam',
    ↪  metrics=['accuracy'])

    model.fit(X_train, y_train_encoded, epochs=480, batch_size=32,
    ↪  verbose=0)

    score = model.evaluate(X_test, y_test_encoded, verbose=0)
    return {'loss': -score[1], 'status': STATUS_OK}
```

```
# Defina o espaço de busca para otimização de hiperparâmetros com o
↪  Hyperopt
space = {
    'neurons_layer1': hp.quniform('neurons_layer1', 32, 128, 1),
    'neurons_layer2': hp.quniform('neurons_layer2', 16, 64, 1),
}

trials = Trials()

# Execute a otimização usando o algoritmo TPE
best = fmin(fn=objective, space=space, algo=tpe.suggest, max_evals=30,
↪  trials=trials)

# Imprima os melhores hiperparâmetros encontrados
print("Melhores Hiperparâmetros Encontrados:")
print(best)
```

- Programação RNA

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    classification_report,
    confusion_matrix,
)
from sklearn.metrics import roc_curve, auc
from sklearn.neural_network import MLPClassifier
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
```

```
# Carregue o conjunto de dados de teste 'certo3.csv'
data = pd.read_csv('certo4t.csv', encoding='latin1')

# Remover as colunas especificadas
cols_to_remove = ["Vol_tank1", "Vol_tank2", "Vol_tank3"]
data = data.drop(cols_to_remove, axis=1)

# Suponhamos que a última coluna seja a coluna de destino, ajuste isso
↪  conforme necessário
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

labelencoder = LabelEncoder()

for i in range(X.shape[1]):
    if X[:, i].dtype == 'object':
        X[:, i] = labelencoder.fit_transform(X[:, i])

# Número de folds para validação cruzada
n_splits = 5  # Pode ajustar conforme necessário

# Configurar validação cruzada
kf = KFold(n_splits=n_splits, shuffle=True, random_state=1)

# Listas para armazenar as métricas de desempenho em cada fold
accuracy_scores = []
precision_scores = []
recall_scores = []
f1_scores = []
roc_auc_scores = []

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Codificar as saídas em formato one-hot (para classificação
    ↪  multiclasse)
    y_train_encoded = to_categorical(y_train)
```

```
y_test_encoded = to_categorical(y_test)

# Crie o modelo de RNA
model = Sequential()
model.add(Dense(94, input_dim=X_train.shape[1], activation='relu'))
↪  #58
model.add(Dense(38, activation='relu')) #58
model.add(Dense(y_train_encoded.shape[1], activation='softmax'))

# Compile o modelo
model.compile(loss='categorical_crossentropy', optimizer='adam',
↪  metrics=['accuracy'])

# Treine o modelo
history = model.fit(X_train, y_train_encoded, epochs=490,
↪  batch_size=21, validation_data=(X_test, y_test_encoded))

# Calcule e armazene as métricas de desempenho
y_pred_probs = model.predict(X_test)
y_pred = np.argmax(y_pred_probs, axis=1)

# Convertemos y_test_encoded e y_pred de volta para rótulos originais
y_test_labels = np.argmax(y_test_encoded, axis=1)

accuracy = accuracy_score(y_test_labels, y_pred)
precision = precision_score(y_test_labels, y_pred, average='weighted')
recall = recall_score(y_test_labels, y_pred, average='weighted')
f1 = f1_score(y_test_labels, y_pred, average='weighted')

accuracy_scores.append(accuracy)
precision_scores.append(precision)
recall_scores.append(recall)
f1_scores.append(f1)

# Agora, treine o modelo MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(94, 38), max_iter=490,
↪  alpha=0.01, random_state=1)
mlp.fit(X_train, y_train)
```

```
    # Calcule as probabilidades de previsão
    y_pred_probs_mlp = mlp.predict_proba(X_test)

    # Calcule o ROC-AUC para a classe positiva
    fpr, tpr, _ = roc_curve(y_test, y_pred_probs_mlp[:, 1])
    roc_auc = auc(fpr, tpr)
    roc_auc_scores.append(roc_auc)

    # # Agora, plote as métricas para este fold
    # metric_names = ['Acurácia', 'Precisão', 'Revocação', 'F1-Score',
    ↪  'ROC-AUC']
    # metrics = [accuracy, precision, recall, f1, roc_auc]

    # plt.figure(figsize=(12, 6))
    # plt.bar(range(len(metric_names)), metrics)
    # plt.xlabel('Métricas')
    # plt.ylabel('Valor')
    # plt.title(f'Desempenho do Fold {len(accuracy_scores)}')
    # plt.xticks(range(len(metric_names)), metric_names)
    # plt.show()

# Plote gráfico de quantidade de dados disponíveis
plt.figure(figsize=(12, 6))
data_counts = [len(X[test_index]) for train_index, test_index in
↪  kf.split(X)]
plt.plot(range(1, n_splits + 1), data_counts, label='Quantidade de Dados',
↪  marker='o')

plt.xlabel('Fold')
plt.ylabel('Quantidade de Dados')
plt.title('Quantidade de Dados por Fold')
plt.xticks(range(1, n_splits + 1))
plt.legend(loc='best')

plt.show()

# Calcula a quantidade de dados disponíveis em cada fold
```

```
data_counts = [len(X[test_index]) for train_index, test_index in
↪  kf.split(X)]

# Etiquetas representando cada fold
labels = [f'Fold {i+1}' for i in range(n_splits)]

# Cria um gráfico de pizza com os dados
plt.figure(figsize=(8, 8))
plt.pie(data_counts, labels=labels, autopct='%1.1f%%', startangle=140)

plt.title('Validação Cruzada - Quantidade de Dados por Fold')

plt.show()

# Acesse as métricas de treinamento
train_accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
train_loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(train_accuracy) + 1)

# Valor da largura das barras de erro
error_bar_width = 0.2

# Crie um gráfico de precisão por época com barras de erro
plt.figure(figsize=(12, 6))
plt.bar([epoch - error_bar_width for epoch in epochs], train_accuracy,
↪  width=error_bar_width, label='Acurácia de Treinamento', color='b',
↪  alpha=0.5)
plt.bar([epoch + error_bar_width for epoch in epochs], val_accuracy,
↪  width=error_bar_width, label='Acurácia de Validação', color='g',
↪  alpha=0.5)
plt.errorbar(epochs, train_accuracy, fmt='b', label='_nolegend_')
plt.errorbar(epochs, val_accuracy, fmt='g', label='_nolegend_')
plt.title('Precisão ao Longo das Épocas')
plt.xlabel('Épocas')
plt.ylabel('Precisão')
plt.legend()
```

```
plt.grid(True)
plt.show()

# Crie um gráfico de perda por época com barras de erro
plt.figure(figsize=(12, 6))
plt.bar([epoch - error_bar_width for epoch in epochs], train_loss,
↪  width=error_bar_width, label='Perda de Treinamento', color='b',
↪  alpha=0.5)
plt.bar([epoch + error_bar_width for epoch in epochs], val_loss,
↪  width=error_bar_width, label='Perda de Validação', color='g',
↪  alpha=0.5)
plt.errorbar(epochs, train_loss, fmt='b', label='_nolegend_')
plt.errorbar(epochs, val_loss, fmt='g', label='_nolegend_')
plt.title('Perda ao Longo das Épocas')
plt.xlabel('Épocas')
plt.ylabel('Perda')
plt.legend()
plt.grid(True)
plt.show()

# Acurácia final
final_accuracy = history.history['val_accuracy'][-1]
training_accuracy = history.history['accuracy'][-1]
print(f"Acurácia final: {final_accuracy:.4f}")
print(f"Acurácia de Treinamento: {training_accuracy:.4f}")
print(f"Acurácia de Teste: {final_accuracy:.4f}")

# Crie uma tabela com os resultados dos folds
results_df = pd.DataFrame({
    'Fold': range(1, n_splits + 1),
    'Acurácia': accuracy_scores,
    'Precisão': precision_scores,
    'Revocação': recall_scores,
    'F1-Score': f1_scores,
    'ROC-AUC': roc_auc_scores
})

print('Resultados de cada fold:')
```

```
print(results_df)
```

APÊNDICE B – BANCO DE DADOS

Tabela 8 – Banco de Dados

Nivel Tank 1	Tempo Tank 1	Nivel Tank 2	Tempo Tank 2	Nivel Tank 3	Tempo Tank 3	Vol. Tank 1	Vol. Tank 2	Vol. Tank3	Vol.Tank M.	Luminancia	N° da cor	Falha
238.74	2.7137	158.1	4	64.86	0.7372	0.9361	0.62	0.2543	2.5705	75.16	1.0	1.0
173.0	2.7137	255.0	4.00	47.0	0.7372	0.6784	1.0	0.1843	1.8627	26.02	1.0	0.0
94.0	2.71375	56.0	4	0.376	0.7372	0.2196	0.094	0.6823	0.9932	12.76	1.0	1.0
107.26	1.9450	255.0	3.9529	105.0	0	0.4196	1.0	0.4117	1.8313	32.76	2.0	1.0
176.08	1.9450	146.16	3.9529	14.0	0	0.6909	0.5731	0.054	1.3185	42.97	2.0	1.0
124.0	1.9450	252.0	3.9529	0	0	0.4862	0.9882	0.0	1.4744	20.65	2.0	0.0
138.0	1.5686	205.62	2.3372	146.94	3.7176	0.5411	0.8039	0.5762	1.9212	57.0	3.0	1.0
100.0	1.5686	149.0	2.3372	237.0	3.7176	0.3921	0.5843	0.9294	1.9058	14.8	3.0	0.0
58.0	1.5686	86.42	2.3372	137.46	3.7176	0.2274	0.3389	0.539	1.1053	11.9	3.0	1.0
96.0	1.5686	34.0	2.3372	106.0	3.7176	0.3764	0.1333	0.4156	0.9253	9.8	3.0	1.0
92.3	1.0196	149.1	1.6470	130.5	3.5294	0.3619	0.5847	0.5117	1.4583	35.68	4.0	1.0
65.0	1.0196	105.0	1.6470	225.0	3.5294	0.2549	0.4117	0.8823	1.548	10.51	4.0	0.0
48.0	1.0196	67.0	1.6470	110.0	3.5294	0.1882	0.2627	0.4313	0.8822	6.06	4.0	1.0
145.0	3.9215	181.76	2.007	157.32	1.7882	0.5686	0.7098	0.6169	1.8953	61.63	5.0	1.0
94.0	3.9215	74.24	2.007	66.12	1.7882	0.3686	0.2911	0.2592	0.9189	43.21	5.0	1.0
250.0	3.9215	128.0	2.007	114.0	1.7882	0.9803	0.5019	0.447	1.9292	81.5	5.0	0.0
147.9	4	10.0	0	147.9	4	0.58	0.039	0.58	1.199	42.3	6.0	1.0
255.0	4	0.0	0	255.0	4	1.0	0.0	1.0	2.0	76.71	6.0	1.0

Fonte: Elaborado pelo autor.

Breve demonstração do banco de dados que foi aplicado o estudo